

L4Re – L4 Runtime Environment

Generated by Doxygen 1.9.1

1 Overview	1
2 Introduction	3
2.1 Fiasco.OC	3
2.1.1 Communication	3
2.1.2 Kernel Objects	4
2.2 L4Re System Structure	4
2.3 L4 Runtime Environment (L4Re)	6
3 Tutorial	7
3.1 Customizations	8
4 Programming for L4Re	9
4.1 Capabilities and Naming	9
4.2 Initial Environment and Application Bootstrapping	10
4.2.1 Configuring an application before startup	11
4.2.2 Connecting clients and servers	11
4.3 Memory management - Data Spaces and the Region Map	12
4.3.1 User-level paging	12
4.3.1.1 Data spaces	12
4.3.1.2 Virtual Memory Handling	12
4.3.1.3 Memory Allocation	12
4.4 Program Input and Output	13
4.5 Initial Memory Allocator and Factory	13
4.6 Application and Server Building Blocks	13
4.6.1 Creating Additional Application Threads	13
4.6.2 Providing a Service	14
4.7 Pthread Support	14
4.8 Interface Definition Language	15
4.8.1 Parameter types for RPC	16
4.8.2 RPC Return Types	17
4.8.3 RPC Method Declaration	17
4.9 L4Re Build System	18
4.9.1 Building L4Re	18
4.9.2 Writing BID Make Files	19
4.9.3 prog.mk - Application Role	19
4.9.4 include.mk - Header File Role	22
4.9.5 test.mk - Test Application Role	23
4.10 Kernel Factory	28
4.10.1 Passing parameters for the create stream	29
5 L4Re Servers	31
5.1 Sigma0, the Root-Pager	33
5.1.1 Factory	33

5.2 Moe, the Root-Task	33
5.2.1 Moe objects	33
5.2.1.1 Factory	33
5.2.1.2 Namespace	35
5.2.1.3 Dataspace	35
5.2.1.4 Log Subsystem	35
5.2.1.5 DMA Space	35
5.2.1.6 Scheduler subsystem	35
5.2.1.7 Region Map	36
5.2.2 Command Line Options	36
5.3 Ned, the Init Process	36
5.3.1 Lua Bindings for L4Re	37
5.3.1.1 Capabilities in Lua	37
5.3.1.2 Access to L4Re::Env Capabilities	37
5.3.1.3 Constants	37
5.3.1.4 Application Startup Details	38
5.3.1.5 Access to the kernel debugger	39
5.3.1.6 Using the interactive ned prompt	39
5.3.2 Command Line Options	39
5.4 Io, the Io Server	39
5.5 l4vio_net_p2p, a virtual network point-to-point link	45
5.6 Uvmm, the virtual machine monitor	46
5.7 Mag, the GUI Multiplexer	52
5.8 Cons, the Console Multiplexer	54
6 Bootstrap, the L4 kernel bootstrapper	57
7 Deprecated List	61
8 Module Index	63
8.1 Modules	63
9 Namespace Index	67
9.1 Namespace List	67
10 Hierarchical Index	69
10.1 Class Hierarchy	69
11 Data Structure Index	79
11.1 Data Structures	79
12 File Index	93
12.1 File List	93
13 Module Documentation	105

13.1 ARM Virtual Registers (UTCB)	105
13.1.1 Detailed Description	106
13.2 Atomic Instructions	106
13.2.1 Detailed Description	108
13.2.2 Function Documentation	108
13.2.2.1 l4util_add16()	108
13.2.2.2 l4util_add16_res()	108
13.2.2.3 l4util_add32()	108
13.2.2.4 l4util_add32_res()	109
13.2.2.5 l4util_add8()	109
13.2.2.6 l4util_add8_res()	109
13.2.2.7 l4util_and16()	110
13.2.2.8 l4util_and16_res()	110
13.2.2.9 l4util_and32()	111
13.2.2.10 l4util_and32_res()	111
13.2.2.11 l4util_and8()	111
13.2.2.12 l4util_and8_res()	112
13.2.2.13 l4util_atomic_add()	112
13.2.2.14 l4util_atomic_inc()	112
13.2.2.15 l4util_cmpxchg()	113
13.2.2.16 l4util_cmpxchg16()	113
13.2.2.17 l4util_cmpxchg32()	114
13.2.2.18 l4util_cmpxchg64()	114
13.2.2.19 l4util_cmpxchg8()	115
13.2.2.20 l4util_dec16()	115
13.2.2.21 l4util_dec16_res()	116
13.2.2.22 l4util_dec32()	116
13.2.2.23 l4util_dec32_res()	116
13.2.2.24 l4util_dec8()	117
13.2.2.25 l4util_dec8_res()	117
13.2.2.26 l4util_inc16()	117
13.2.2.27 l4util_inc16_res()	117
13.2.2.28 l4util_inc32()	118
13.2.2.29 l4util_inc32_res()	118
13.2.2.30 l4util_inc8()	118
13.2.2.31 l4util_inc8_res()	119
13.2.2.32 l4util_or16()	119
13.2.2.33 l4util_or16_res()	119
13.2.2.34 l4util_or32()	120
13.2.2.35 l4util_or32_res()	120
13.2.2.36 l4util_or8()	121
13.2.2.37 l4util_or8_res()	121

13.2.2.38 l4util_sub16()	121
13.2.2.39 l4util_sub16_res()	122
13.2.2.40 l4util_sub32()	122
13.2.2.41 l4util_sub32_res()	122
13.2.2.42 l4util_sub8()	123
13.2.2.43 l4util_sub8_res()	123
13.2.2.44 l4util_xchg()	123
13.2.2.45 l4util_xchg16()	124
13.2.2.46 l4util_xchg32()	124
13.2.2.47 l4util_xchg8()	125
13.3 Auxiliary data	125
13.3.1 Detailed Description	126
13.4 Base API	126
13.4.1 Detailed Description	128
13.5 Basic Macros	128
13.5.1 Detailed Description	129
13.6 Bit Manipulation	129
13.6.1 Detailed Description	130
13.6.2 Function Documentation	130
13.6.2.1 l4util_bsf()	130
13.6.2.2 l4util_bsr()	130
13.6.2.3 l4util_btc()	131
13.6.2.4 l4util_btr()	131
13.6.2.5 l4util_bts()	132
13.6.2.6 l4util_clear_bit()	133
13.6.2.7 l4util_complement_bit()	133
13.6.2.8 l4util_find_first_set_bit()	134
13.6.2.9 l4util_find_first_zero_bit()	134
13.6.2.10 l4util_next_power2()	135
13.6.2.11 l4util_set_bit()	135
13.6.2.12 l4util_test_bit()	136
13.7 Buffer Registers (BRs)	136
13.7.1 Detailed Description	137
13.7.2 Enumeration Type Documentation	137
13.7.2.1 l4_buffer_desc_consts_t	137
13.8 C++ Exceptions	137
13.8.1 Detailed Description	138
13.9 C++ IPC Interface Definition.	138
13.9.1 Detailed Description	138
13.10 CPU related functions	138
13.11 Cache Consistency	138
13.11.1 Detailed Description	139

13.11.2 Function Documentation	139
13.11.2.1 l4_cache_clean_data()	139
13.11.2.2 l4_cache_coherent()	140
13.11.2.3 l4_cache_dma_coherent()	140
13.11.2.4 l4_cache_flush_data()	141
13.11.2.5 l4_cache_inv_data()	141
13.12 Capabilities	142
13.12.1 Detailed Description	143
13.12.2 Enumeration Type Documentation	143
13.12.2.1 l4_cap_consts_t	143
13.12.2.2 l4_default_caps_t	143
13.12.3 Function Documentation	144
13.12.3.1 l4_capability_equal()	144
13.12.3.2 l4_is_invalid_cap()	145
13.12.3.3 l4_is_valid_cap()	145
13.13 Capability allocator	146
13.13.1 Detailed Description	146
13.13.2 Function Documentation	146
13.13.2.1 l4re_util_cap_last()	147
13.14 Chunks	147
13.14.1 Detailed Description	148
13.14.2 Function Documentation	148
13.14.2.1 l4shmc_add_chunk()	148
13.14.2.2 l4shmc_chunk_capacity()	148
13.14.2.3 l4shmc_chunk_ptr()	149
13.14.2.4 l4shmc_chunk_signal()	149
13.14.2.5 l4shmc_get_chunk()	150
13.14.2.6 l4shmc_get_chunk_to()	150
13.14.2.7 l4shmc_iterate_chunk()	151
13.15 Comfortable Command Line Parsing	151
13.16 Console API	151
13.16.1 Detailed Description	152
13.17 Consumer	152
13.17.1 Detailed Description	152
13.17.2 Function Documentation	152
13.17.2.1 l4shmc_enable_signal()	152
13.17.2.2 l4shmc_wait_any()	153
13.17.2.3 l4shmc_wait_any_to()	153
13.17.2.4 l4shmc_wait_any_try()	154
13.17.2.5 l4shmc_wait_signal()	154
13.17.2.6 l4shmc_wait_signal_to()	154
13.17.2.7 l4shmc_wait_signal_try()	155

13.18 Consumer	155
13.18.1 Detailed Description	156
13.18.2 Function Documentation	156
13.18.2.1 l4shmc_chunk_consumed()	156
13.18.2.2 l4shmc_chunk_size()	156
13.18.2.3 l4shmc_enable_chunk()	157
13.18.2.4 l4shmc_is_chunk_ready()	157
13.18.2.5 l4shmc_wait_chunk()	158
13.18.2.6 l4shmc_wait_chunk_to()	158
13.18.2.7 l4shmc_wait_chunk_try()	159
13.19 DMA Space Interface	159
13.19.1 Detailed Description	160
13.19.2 Typedef Documentation	160
13.19.2.1 l4re_dma_space_t	160
13.19.3 Function Documentation	160
13.19.3.1 l4re_dma_space_associate()	160
13.19.3.2 l4re_dma_space_disassociate()	161
13.19.3.3 l4re_dma_space_map()	161
13.19.3.4 l4re_dma_space_unmap()	162
13.20 Dataspace interface	163
13.20.1 Detailed Description	164
13.20.2 Enumeration Type Documentation	164
13.20.2.1 l4re_ds_map_flags	164
13.20.3 Function Documentation	164
13.20.3.1 l4re_ds_allocate()	164
13.20.3.2 l4re_ds_clear()	165
13.20.3.3 l4re_ds_copy_in()	165
13.20.3.4 l4re_ds_flags()	165
13.20.3.5 l4re_ds_info()	166
13.20.3.6 l4re_ds_size()	166
13.21 Debug interface	166
13.21.1 Detailed Description	166
13.21.2 Function Documentation	166
13.21.2.1 l4re_debug_obj_debug()	166
13.22 Debugging API	167
13.22.1 Detailed Description	167
13.23 EDID parsing functionality	167
13.23.1 Detailed Description	168
13.23.2 Enumeration Type Documentation	168
13.23.2.1 libedid_consts	168
13.23.3 Function Documentation	168
13.23.3.1 libedid_check_header()	168

13.23.3.2 libedid_checksum()	169
13.23.3.3 libedid_dump()	169
13.23.3.4 libedid_dump_standard_timings()	169
13.23.3.5 libedid_num_ext_blocks()	170
13.23.3.6 libedid_pnp_id()	170
13.23.3.7 libedid_prefered_resolution()	170
13.23.3.8 libedid_revision()	171
13.23.3.9 libedid_version()	171
13.24 ELF binary format	172
13.24.1 Detailed Description	173
13.25 Error Handling	173
13.25.1 Detailed Description	174
13.25.2 Enumeration Type Documentation	174
13.25.2.1 l4_ipc_tcr_error_t	174
13.25.3 Function Documentation	174
13.25.3.1 l4_error()	175
13.25.3.2 l4_ipc_error()	176
13.25.3.3 l4_ipc_error_code()	177
13.25.3.4 l4_ipc_is_rcv_error()	178
13.25.3.5 l4_ipc_is_snd_error()	178
13.26 Error codes	179
13.26.1 Detailed Description	179
13.26.2 Enumeration Type Documentation	179
13.26.2.1 l4_error_code_t	179
13.27 Event API	180
13.27.1 Detailed Description	181
13.28 Event interface	181
13.28.1 Detailed Description	182
13.28.2 Function Documentation	182
13.28.2.1 l4re_event_get_axis_info()	182
13.28.2.2 l4re_event_get_buffer()	182
13.28.2.3 l4re_event_get_num_streams()	183
13.28.2.4 l4re_event_get_stream_info()	183
13.28.2.5 l4re_event_get_stream_info_for_id()	184
13.29 Exception registers	184
13.29.1 Detailed Description	185
13.29.2 Function Documentation	185
13.29.2.1 l4_utcb_exc()	185
13.29.2.2 l4_utcb_exc_is_ex_regs_exception()	186
13.29.2.3 l4_utcb_exc_is_pf()	186
13.29.2.4 l4_utcb_exc_pc()	186
13.29.2.5 l4_utcb_exc_pc_set()	187

13.30 Extended vCPU support	187
13.30.1 Detailed Description	188
13.30.2 Function Documentation	188
13.30.2.1 l4vcpu_ext_alloc()	188
13.31 Factory	188
13.31.1 Detailed Description	189
13.31.2 Function Documentation	190
13.31.2.1 l4_factory_create_factory()	190
13.31.2.2 l4_factory_create_factory_u()	191
13.31.2.3 l4_factory_create_gate()	192
13.31.2.4 l4_factory_create_gate_u()	193
13.31.2.5 l4_factory_create_irq()	194
13.31.2.6 l4_factory_create_irq_u()	195
13.31.2.7 l4_factory_create_task()	196
13.31.2.8 l4_factory_create_task_u()	197
13.31.2.9 l4_factory_create_thread()	198
13.31.2.10 l4_factory_create_thread_u()	199
13.31.2.11 l4_factory_create_vm()	200
13.31.2.12 l4_factory_create_vm_u()	201
13.32 Fiasco extensions	202
13.32.1 Detailed Description	203
13.32.2 Function Documentation	203
13.32.2.1 fiasco_gdt_get_entry_offset()	203
13.32.2.2 fiasco_gdt_set()	203
13.32.2.3 fiasco_ldt_set()	204
13.32.2.4 fiasco_tbuf_log()	204
13.32.2.5 fiasco_tbuf_log_3val()	205
13.32.2.6 fiasco_tbuf_log_binary()	206
13.33 Fiasco real time scheduling extensions	206
13.34 Fiasco-UX Virtual devices	207
13.34.1 Detailed Description	207
13.34.2 Enumeration Type Documentation	207
13.34.2.1 l4_vhw_entry_type	207
13.35 Flex pages	208
13.35.1 Detailed Description	210
13.35.2 Enumeration Type Documentation	210
13.35.2.1 anonymous enum	210
13.35.2.2 anonymous enum	210
13.35.2.3 L4_cap_fpage_rights	212
13.35.2.4 l4_fpage_cacheability_opt_t	213
13.35.2.5 l4_fpage_consts	214
13.35.2.6 L4_fpage_rights	214

13.35.2.7 L4_obj_fpage_ctl	214
13.35.3 Function Documentation	215
13.35.3.1 l4_fpage()	215
13.35.3.2 l4_fpage_all()	215
13.35.3.3 l4_fpage_contains()	216
13.35.3.4 l4_fpage_invalid()	217
13.35.3.5 l4_fpage_ioport()	217
13.35.3.6 l4_fpage_max_order()	217
13.35.3.7 l4_fpage_memaddr()	218
13.35.3.8 l4_fpage_obj()	219
13.35.3.9 l4_fpage_page()	220
13.35.3.10 l4_fpage_rights()	220
13.35.3.11 l4_fpage_set_rights()	221
13.35.3.12 l4_fpage_size()	221
13.35.3.13 l4_fpage_type()	222
13.35.3.14 l4_iofpage()	222
13.35.3.15 l4_is_fpage_writable()	223
13.35.3.16 l4_obj_fpage()	224
13.36 Functions to manipulate the local IDT	225
13.37 IA32 Port I/O API	225
13.38 IO interface	225
13.38.1 Detailed Description	226
13.38.2 Typedef Documentation	226
13.38.2.1 l4io_resource_t	226
13.38.3 Enumeration Type Documentation	227
13.38.3.1 l4io_device_types_t	227
13.38.3.2 l4io_iomem_flags_t	228
13.38.3.3 l4io_resource_types_t	228
13.38.4 Function Documentation	229
13.38.4.1 l4io_has_resource()	229
13.38.4.2 l4io_lookup_device()	229
13.38.4.3 l4io_lookup_resource()	230
13.38.4.4 l4io_release_iomem()	230
13.38.4.5 l4io_release_ioport()	230
13.38.4.6 l4io_request_iomem()	231
13.38.4.7 l4io_request_iomem_region()	232
13.38.4.8 l4io_request_ioport()	232
13.38.4.9 l4io_request_resource_iomem()	233
13.39 IPC-Gate API	233
13.39.1 Detailed Description	234
13.39.2 Function Documentation	234
13.39.2.1 l4_ipc_gate_bind_thread()	234

13.39.2.2	l4_ipc_gate_get_infos()	235
13.39.2.3	l4_rcv_ep_bind_thread()	235
13.40	IRQ handling library	236
13.40.1	Detailed Description	236
13.41	IRQs	236
13.41.1	Detailed Description	238
13.41.2	Enumeration Type Documentation	238
13.41.2.1	L4_irq_mode	238
13.41.3	Function Documentation	239
13.41.3.1	l4_irq_detach()	239
13.41.3.2	l4_irq_detach_u()	239
13.41.3.3	l4_irq_mux_chain()	240
13.41.3.4	l4_irq_mux_chain_u()	241
13.41.3.5	l4_irq_receive()	242
13.41.3.6	l4_irq_receive_u()	243
13.41.3.7	l4_irq_trigger()	244
13.41.3.8	l4_irq_trigger_u()	245
13.41.3.9	l4_irq_unmask()	246
13.41.3.10	l4_irq_unmask_u()	247
13.41.3.11	l4_irq_wait()	247
13.41.3.12	l4_irq_wait_u()	248
13.42	Initial Environment	249
13.42.1	Detailed Description	250
13.42.2	Function Documentation	250
13.42.2.1	l4re_env()	250
13.42.2.2	l4re_env_get_cap()	251
13.42.2.3	l4re_env_get_cap_e()	252
13.42.2.4	l4re_env_get_cap_l()	252
13.42.2.5	l4re_kip()	253
13.43	Integer Types	254
13.43.1	Detailed Description	255
13.44	Interface for asynchronous ISR handlers with a given IRQ capability.	255
13.44.1	Detailed Description	255
13.44.2	Function Documentation	255
13.44.2.1	l4irq_request_cap()	255
13.45	Interface for asynchronous ISR handlers.	256
13.45.1	Detailed Description	256
13.45.2	Function Documentation	257
13.45.2.1	l4irq_release()	257
13.45.2.2	l4irq_request()	257
13.46	Interface using direct functionality.	258
13.46.1	Detailed Description	258

13.46.2 Function Documentation	258
13.46.2.1 l4irq_attach_cap()	258
13.46.2.2 l4irq_attach_cap_ft()	259
13.46.2.3 l4irq_attach_thread_cap()	259
13.46.2.4 l4irq_attach_thread_cap_ft()	260
13.47 Interface using direct functionality	260
13.47.1 Detailed Description	261
13.47.2 Function Documentation	261
13.47.2.1 l4irq_attach()	261
13.47.2.2 l4irq_attach_ft()	262
13.47.2.3 l4irq_attach_thread()	262
13.47.2.4 l4irq_attach_thread_ft()	262
13.47.2.5 l4irq_detach()	263
13.47.2.6 l4irq_unmask()	263
13.47.2.7 l4irq_unmask_and_wait_any()	264
13.47.2.8 l4irq_wait()	264
13.47.2.9 l4irq_wait_any()	264
13.48 Internal Helpers	265
13.48.1 Detailed Description	265
13.49 Internal constants	265
13.50 Internal functions	266
13.51 Interrupt controller	266
13.51.1 Detailed Description	267
13.51.2 Typedef Documentation	268
13.51.2.1 l4_icu_info_t	268
13.51.3 Enumeration Type Documentation	268
13.51.3.1 L4_icu_flags	268
13.51.4 Function Documentation	268
13.51.4.1 l4_icu_bind()	268
13.51.4.2 l4_icu_bind_u()	269
13.51.4.3 l4_icu_info()	270
13.51.4.4 l4_icu_info_u()	271
13.51.4.5 l4_icu_mask()	272
13.51.4.6 l4_icu_mask_u()	272
13.51.4.7 l4_icu_msi_info()	273
13.51.4.8 l4_icu_msi_info_u()	274
13.51.4.9 l4_icu_set_mode()	275
13.51.4.10 l4_icu_set_mode_u()	276
13.51.4.11 l4_icu_unbind()	276
13.51.4.12 l4_icu_unbind_u()	277
13.51.4.13 l4_icu_unmask()	278
13.51.4.14 l4_icu_unmask_u()	279

13.52 Kernel Debugger	279
13.52.1 Detailed Description	280
13.52.2 Function Documentation	280
13.52.2.1 l4_debugger_global_id()	280
13.52.2.2 l4_debugger_kobj_to_id()	281
13.52.2.3 l4_debugger_set_object_name()	281
13.53 Kernel Interface Page	281
13.53.1 Detailed Description	282
13.54 Kernel Interface Page API	283
13.54.1 Detailed Description	283
13.55 Kernel Objects	283
13.55.1 Detailed Description	285
13.56 Kumem allocator utility	285
13.57 Kumem utilities	285
13.58 L4 IPC Opcodes	286
13.58.1 Detailed Description	286
13.58.2 Enumeration Type Documentation	286
13.58.2.1 L4_icu_opcode	286
13.58.2.2 L4_ipc_gate_ops	287
13.58.2.3 L4_platform_ctl_ops	288
13.58.2.4 L4_task_ops	288
13.58.2.5 L4_thread_ops	288
13.58.2.6 L4_vcon_ops	289
13.59 L4 VIRTIO Block Device	289
13.59.1 Detailed Description	290
13.59.2 Enumeration Type Documentation	290
13.59.2.1 L4virtio_block_operations	290
13.59.2.2 L4virtio_block_status	291
13.60 L4 VIRTIO Input Device	291
13.60.1 Detailed Description	292
13.61 L4 VIRTIO Interface	292
13.61.1 Detailed Description	293
13.62 L4 VIRTIO Transport Layer	293
13.62.1 Detailed Description	294
13.62.2 Typedef Documentation	295
13.62.2.1 l4virtio_config_queue_t	295
13.62.3 Enumeration Type Documentation	295
13.62.3.1 L4_virtio_cmd	295
13.62.3.2 L4_virtio_irq_status	295
13.62.3.3 L4_virtio_opcodes	296
13.62.3.4 L4virtio_device_ids	296
13.62.3.5 L4virtio_device_status	297

13.62.3.6 L4virtio_feature_bits	297
13.62.4 Function Documentation	298
13.62.4.1 l4virtio_config_queue()	298
13.62.4.2 l4virtio_config_queues()	298
13.62.4.3 l4virtio_device_config()	299
13.62.4.4 l4virtio_device_config_ds()	299
13.62.4.5 l4virtio_device_notification_irq()	299
13.62.4.6 l4virtio_register_ds()	300
13.62.4.7 l4virtio_register_iface()	300
13.62.4.8 l4virtio_set_status()	301
13.63 L4 Vbus functions	301
13.63.1 Detailed Description	303
13.63.2 Enumeration Type Documentation	303
13.63.2.1 L4vbus_dma_domain_assign_flags	303
13.63.3 Function Documentation	303
13.63.3.1 l4vbus_assign_dma_domain()	304
13.63.3.2 l4vbus_get_device()	304
13.63.3.3 l4vbus_get_device_by_hid()	305
13.63.3.4 l4vbus_get_hid()	306
13.63.3.5 l4vbus_get_next_device()	306
13.63.3.6 l4vbus_get_resource()	308
13.63.3.7 l4vbus_is_compatible()	309
13.63.3.8 l4vbus_release_ioport()	310
13.63.3.9 l4vbus_release_resource()	310
13.63.3.10 l4vbus_request_ioport()	311
13.63.3.11 l4vbus_request_resource()	312
13.63.3.12 l4vbus_vicu_get_cap()	312
13.64 L4 kernel object type information	313
13.64.1 Detailed Description	314
13.64.2 Function Documentation	314
13.64.2.1 kobject_typeid()	314
13.65 L4Re C Interface	314
13.65.1 Detailed Description	316
13.65.2 Function Documentation	316
13.65.2.1 l4re_inhibitor_acquire()	316
13.66 L4Re C++ Interface	317
13.66.1 Detailed Description	319
13.67 L4Re Capability API	319
13.68 L4Re ELF Auxiliary Information	319
13.69 L4Re Protocol identifiers	320
13.69.1 Detailed Description	320
13.70 L4Re Util C Interface	321

13.71 L4Re Util C++ Interface	321
13.71.1 Detailed Description	322
13.72 L4vbus GPIO functions	322
13.72.1 Detailed Description	323
13.72.2 Enumeration Type Documentation	323
13.72.2.1 L4vbus_gpio_generic_func	323
13.72.2.2 L4vbus_gpio_pull_modes	324
13.72.3 Function Documentation	324
13.72.3.1 l4vbus_gpio_config_get()	324
13.72.3.2 l4vbus_gpio_config_pad()	325
13.72.3.3 l4vbus_gpio_config_pull()	326
13.72.3.4 l4vbus_gpio_get()	326
13.72.3.5 l4vbus_gpio_multi_config_pad()	327
13.72.3.6 l4vbus_gpio_multi_get()	328
13.72.3.7 l4vbus_gpio_multi_set()	329
13.72.3.8 l4vbus_gpio_multi_setup()	329
13.72.3.9 l4vbus_gpio_set()	330
13.72.3.10 l4vbus_gpio_setup()	331
13.72.3.11 l4vbus_gpio_to_irq()	332
13.73 L4vbus PCI functions	333
13.73.1 Detailed Description	333
13.73.2 Function Documentation	333
13.73.2.1 l4vbus_pci_cfg_read()	333
13.73.2.2 l4vbus_pci_cfg_write()	334
13.73.2.3 l4vbus_pci_irq_enable()	335
13.73.2.4 l4vbus_pcidev_cfg_read()	336
13.73.2.5 l4vbus_pcidev_cfg_write()	337
13.73.2.6 l4vbus_pcidev_irq_enable()	337
13.74 Log interface	338
13.74.1 Detailed Description	339
13.74.2 Function Documentation	339
13.74.2.1 l4re_log_print()	339
13.74.2.2 l4re_log_print_srv()	339
13.74.2.3 l4re_log_printn()	340
13.74.2.4 l4re_log_printn_srv()	341
13.75 Logging interface	342
13.75.1 Detailed Description	342
13.76 Low-Level Thread Functions	342
13.77 Memory allocator	343
13.77.1 Detailed Description	343
13.77.2 Enumeration Type Documentation	343
13.77.2.1 l4re_ma_flags	344

13.77.3 Function Documentation	344
13.77.3.1 l4re_ma_alloc()	344
13.77.3.2 l4re_ma_alloc_align()	345
13.77.3.3 l4re_ma_alloc_align_srv()	346
13.77.3.4 l4re_ma_free()	347
13.77.3.5 l4re_ma_free_srv()	348
13.78 Memory descriptors (C version)	349
13.78.1 Detailed Description	350
13.78.2 Typedef Documentation	350
13.78.2.1 l4_kernel_info_mem_desc_t	350
13.78.3 Enumeration Type Documentation	350
13.78.3.1 l4_mem_info_sub_type_t	350
13.78.3.2 l4_mem_type_t	351
13.78.4 Function Documentation	351
13.78.4.1 l4_kernel_info_get_mem_desc_end()	351
13.78.4.2 l4_kernel_info_get_mem_desc_is_virtual()	352
13.78.4.3 l4_kernel_info_get_mem_desc_start()	352
13.78.4.4 l4_kernel_info_get_mem_desc_subtype()	352
13.78.4.5 l4_kernel_info_get_mem_desc_type()	353
13.78.4.6 l4_kernel_info_get_num_mem_descs()	353
13.78.4.7 l4_kernel_info_set_mem_desc()	353
13.79 Memory operations.	354
13.79.1 Detailed Description	354
13.79.2 Enumeration Type Documentation	354
13.79.2.1 L4_mem_op_widths	354
13.79.3 Function Documentation	355
13.79.3.1 l4_mem_read()	355
13.79.3.2 l4_mem_write()	356
13.80 Memory related	356
13.80.1 Detailed Description	357
13.80.2 Macro Definition Documentation	358
13.80.2.1 L4_LOG2_PAGESIZE	358
13.80.2.2 L4_LOG2_SUPERPAGESIZE	358
13.80.2.3 L4_PAGEMASK	358
13.80.2.4 L4_SUPERPAGEMASK	358
13.80.2.5 L4_SUPERPAGESIZE	359
13.80.3 Enumeration Type Documentation	359
13.80.3.1 l4_addr_consts_t	359
13.80.4 Function Documentation	359
13.80.4.1 l4_bytes_to_mwords()	359
13.80.4.2 l4_round_page()	360
13.80.4.3 l4_round_size()	360

13.80.4.4 <code>l4_trunc_page()</code>	361
13.80.4.5 <code>l4_trunc_size()</code>	362
13.81 Message Items	362
13.81.1 Detailed Description	363
13.81.2 Enumeration Type Documentation	363
13.81.2.1 <code>l4_msg_item_consts_t</code>	363
13.81.3 Function Documentation	364
13.81.3.1 <code>l4_map_control()</code>	364
13.81.3.2 <code>l4_map_obj_control()</code>	364
13.82 Message Registers (MRs)	365
13.82.1 Detailed Description	366
13.83 Message Tag	366
13.83.1 Detailed Description	368
13.83.2 Typedef Documentation	368
13.83.2.1 <code>l4_msgtag_t</code>	368
13.83.3 Enumeration Type Documentation	368
13.83.3.1 <code>l4_msgtag_flags</code>	368
13.83.3.2 <code>l4_msgtag_protocol</code>	369
13.83.3.3 <code>l4_platform_ctl_proto</code>	369
13.83.4 Function Documentation	370
13.83.4.1 <code>l4_msgtag()</code>	370
13.83.4.2 <code>l4_msgtag_flags()</code>	371
13.83.4.3 <code>l4_msgtag_has_error()</code>	371
13.83.4.4 <code>l4_msgtag_is_exception()</code>	372
13.83.4.5 <code>l4_msgtag_is_io_page_fault()</code>	373
13.83.4.6 <code>l4_msgtag_is_page_fault()</code>	373
13.83.4.7 <code>l4_msgtag_is_preemption()</code>	374
13.83.4.8 <code>l4_msgtag_is_sigma0()</code>	375
13.83.4.9 <code>l4_msgtag_is_sys_exception()</code>	375
13.83.4.10 <code>l4_msgtag_items()</code>	376
13.83.4.11 <code>l4_msgtag_label()</code>	377
13.83.4.12 <code>l4_msgtag_words()</code>	378
13.84 Name-space API	379
13.84.1 Detailed Description	379
13.85 Namespace interface	379
13.85.1 Detailed Description	380
13.85.2 Enumeration Type Documentation	380
13.85.2.1 <code>l4re_ns_register_flags</code>	380
13.85.3 Function Documentation	380
13.85.3.1 <code>l4re_ns_query_srv()</code>	380
13.85.3.2 <code>l4re_ns_query_to_srv()</code>	381
13.85.3.3 <code>l4re_ns_register_obj_srv()</code>	382

13.86 Object Invocation	383
13.86.1 Detailed Description	384
13.86.2 Enumeration Type Documentation	385
13.86.2.1 l4_syscall_flags_t	385
13.86.3 Function Documentation	386
13.86.3.1 l4_ipc()	386
13.86.3.2 l4_ipc_call()	387
13.86.3.3 l4_ipc_receive()	388
13.86.3.4 l4_ipc_reply_and_wait()	390
13.86.3.5 l4_ipc_send()	391
13.86.3.6 l4_ipc_send_and_wait()	392
13.86.3.7 l4_ipc_sleep()	393
13.86.3.8 l4_ipc_wait()	394
13.86.3.9 l4_sndpage_add()	395
13.87 Parent API	396
13.87.1 Detailed Description	396
13.88 Platform Control C API	397
13.88.1 Detailed Description	397
13.88.2 Function Documentation	397
13.88.2.1 l4_platform_ctl_cpu_disable()	397
13.88.2.2 l4_platform_ctl_cpu_enable()	398
13.88.2.3 l4_platform_ctl_system_shutdown()	398
13.88.2.4 l4_platform_ctl_system_suspend()	399
13.89 Producer	399
13.89.1 Detailed Description	400
13.89.2 Function Documentation	400
13.89.2.1 l4shmc_trigger()	400
13.90 Producer	400
13.90.1 Detailed Description	401
13.90.2 Function Documentation	401
13.90.2.1 l4shmc_chunk_ready()	401
13.90.2.2 l4shmc_chunk_ready_sig()	401
13.90.2.3 l4shmc_chunk_try_to_take()	402
13.90.2.4 l4shmc_is_chunk_clear()	402
13.91 Random number support	403
13.91.1 Detailed Description	403
13.91.2 Function Documentation	403
13.91.2.1 l4util_rand()	403
13.91.2.2 l4util_srand()	403
13.92 Realtime API	404
13.93 Region map API	404
13.93.1 Detailed Description	404

13.94 Region map interface	405
13.94.1 Detailed Description	406
13.94.2 Enumeration Type Documentation	406
13.94.2.1 l4re_rm_flags_values	406
13.94.3 Function Documentation	407
13.94.3.1 l4re_rm_attach()	407
13.94.3.2 l4re_rm_attach_srv()	408
13.94.3.3 l4re_rm_detach()	409
13.94.3.4 l4re_rm_detach_ds()	410
13.94.3.5 l4re_rm_detach_ds_unmap()	411
13.94.3.6 l4re_rm_detach_srv()	412
13.94.3.7 l4re_rm_detach_unmap()	412
13.94.3.8 l4re_rm_find()	413
13.94.3.9 l4re_rm_find_srv()	414
13.94.3.10 l4re_rm_free_area()	414
13.94.3.11 l4re_rm_free_area_srv()	415
13.94.3.12 l4re_rm_reserve_area()	415
13.94.3.13 l4re_rm_reserve_area_srv()	416
13.94.3.14 l4re_rm_show_lists()	416
13.95 Scheduler	417
13.95.1 Detailed Description	418
13.95.2 Enumeration Type Documentation	418
13.95.2.1 L4_scheduler_ops	418
13.95.3 Function Documentation	418
13.95.3.1 l4_sched_cpu_set()	418
13.95.3.2 l4_scheduler_idle_time()	419
13.95.3.3 l4_scheduler_info()	420
13.95.3.4 l4_scheduler_is_online()	420
13.95.3.5 l4_scheduler_run_thread()	421
13.96 Server-Side IPC framework	422
13.96.1 Detailed Description	423
13.96.2 Enumeration Type Documentation	423
13.96.2.1 Reply_mode	423
13.97 Shared Memory Library	423
13.97.1 Detailed Description	424
13.97.2 Function Documentation	424
13.97.2.1 l4shmc_area_overhead()	425
13.97.2.2 l4shmc_area_size()	425
13.97.2.3 l4shmc_area_size_free()	425
13.97.2.4 l4shmc_attach()	426
13.97.2.5 l4shmc_attach_to()	426
13.97.2.6 l4shmc_chunk_overhead()	427

13.97.2.7 l4shmc_connect_chunk_signal()	427
13.97.2.8 l4shmc_create()	427
13.98 Sigma0 API	428
13.98.1 Detailed Description	428
13.99 Signals	429
13.99.1 Detailed Description	429
13.99.2 Function Documentation	429
13.99.2.1 l4shmc_add_signal()	429
13.99.2.2 l4shmc_attach_signal()	430
13.99.2.3 l4shmc_attach_signal_to()	430
13.99.2.4 l4shmc_check_magic()	431
13.99.2.5 l4shmc_get_signal_to()	431
13.99.2.6 l4shmc_signal_cap()	432
13.100 Small C++ Template Library	432
13.100.1 Detailed Description	433
13.100.2 Function Documentation	433
13.100.2.1 max()	433
13.100.2.2 min()	434
13.100.2.3 operator new()	435
13.101 Task	436
13.101.1 Detailed Description	436
13.101.2 Enumeration Type Documentation	437
13.101.2.1 l4_unmap_flags_t	437
13.101.3 Function Documentation	437
13.101.3.1 l4_task_add_ku_mem()	437
13.101.3.2 l4_task_cap_equal()	438
13.101.3.3 l4_task_cap_valid()	438
13.101.3.4 l4_task_delete_obj()	439
13.101.3.5 l4_task_map()	439
13.101.3.6 l4_task_release_cap()	440
13.101.3.7 l4_task_unmap()	441
13.101.3.8 l4_task_unmap_batch()	442
13.101.3.9 l4_task_vgicc_map()	442
13.102 Thread	443
13.102.1 Detailed Description	445
13.102.2 Enumeration Type Documentation	445
13.102.2.1 L4_thread_control_flags	445
13.102.2.2 L4_thread_control_mr_indices	446
13.102.2.3 L4_thread_ex_regs_flags	446
13.102.3 Function Documentation	447
13.102.3.1 l4_thread_arm_set_tpidruro()	447
13.102.3.2 l4_thread_ex_regs()	447

13.102.3.3	l4_thread_ex_regs_ret()	448
13.102.3.4	l4_thread_ex_regs_ret_u()	449
13.102.3.5	l4_thread_ex_regs_u()	450
13.102.3.6	l4_thread_modify_sender_add()	451
13.102.3.7	l4_thread_modify_sender_commit()	452
13.102.3.8	l4_thread_modify_sender_start()	452
13.102.3.9	l4_thread_register_del_irq()	452
13.102.3.10	l4_thread_stats_time()	453
13.102.3.11	l4_thread_switch()	453
13.102.3.12	l4_thread_vcpu_control()	454
13.102.3.13	l4_thread_vcpu_control_ext()	454
13.102.3.14	l4_thread_vcpu_control_ext_u()	455
13.102.3.15	l4_thread_vcpu_control_u()	456
13.102.3.16	l4_thread_vcpu_resume_commit()	457
13.102.3.17	l4_thread_vcpu_resume_start()	458
13.102.3.18	l4_thread_yield()	458
13.103	Thread Control Registers (TCRs)	459
13.103.1	Detailed Description	459
13.104	Thread control	459
13.104.1	Detailed Description	460
13.104.2	Function Documentation	460
13.104.2.1	l4_thread_control_alien()	460
13.104.2.2	l4_thread_control_bind()	461
13.104.2.3	l4_thread_control_commit()	462
13.104.2.4	l4_thread_control_exc_handler()	463
13.104.2.5	l4_thread_control_pager()	463
13.104.2.6	l4_thread_control_start()	464
13.104.2.7	l4_thread_control_ux_host_syscall()	464
13.105	Timeouts	465
13.105.1	Detailed Description	466
13.105.2	Macro Definition Documentation	466
13.105.2.1	L4_IPC_TIMEOUT_0	466
13.105.3	Typedef Documentation	466
13.105.3.1	l4_timeout_s	467
13.105.3.2	l4_timeout_t	467
13.105.4	Function Documentation	467
13.105.4.1	l4_ipc_timeout()	467
13.105.4.2	l4_rcv_timeout()	467
13.105.4.3	l4_snd_timeout()	468
13.105.4.4	l4_timeout()	468
13.105.4.5	l4_timeout_abs()	469
13.105.4.6	l4_timeout_get()	470

13.105.4.7 l4_timeout_is_absolute()	471
13.105.4.8 l4_timeout_rel()	471
13.105.4.9 l4_timeout_rel_get()	472
13.105.4.10 l4_utcb_mr64_idx()	472
13.106 Timestamp Counter	473
13.106.1 Detailed Description	473
13.107 Utility Functions	473
13.107.1 Detailed Description	475
13.107.2 Function Documentation	475
13.107.2.1 l4_sleep()	475
13.107.2.2 l4_touch_ro()	476
13.107.2.3 l4_touch_rw()	476
13.107.2.4 l4_usleep()	477
13.107.2.5 l4util_micros2l4to()	478
13.107.2.6 l4util_splitlog2_hdl()	478
13.107.2.7 l4util_splitlog2_size()	479
13.108 VM API for SVM	480
13.108.1 Detailed Description	481
13.109 VM API for TZ	481
13.109.1 Detailed Description	481
13.110 VM API for VMX	481
13.110.1 Detailed Description	483
13.110.2 Enumeration Type Documentation	483
13.110.2.1 anonymous enum	483
13.110.2.2 L4_vm_vmx_caps_regs	484
13.110.2.3 L4_vm_vmx_dfl1_regs	484
13.110.3 Function Documentation	484
13.110.3.1 l4_vm_vmx_clear()	485
13.110.3.2 l4_vm_vmx_field_len()	485
13.110.3.3 l4_vm_vmx_field_order()	486
13.110.3.4 l4_vm_vmx_get_caps()	486
13.110.3.5 l4_vm_vmx_get_caps_default1()	487
13.110.3.6 l4_vm_vmx_get_cr2_index()	487
13.110.3.7 l4_vm_vmx_ptr_load()	488
13.110.3.8 l4_vm_vmx_read()	488
13.110.3.9 l4_vm_vmx_read_16()	489
13.110.3.10 l4_vm_vmx_read_32()	490
13.110.3.11 l4_vm_vmx_read_64()	491
13.110.3.12 l4_vm_vmx_read_nat()	491
13.110.3.13 l4_vm_vmx_write()	492
13.110.3.14 l4_vm_vmx_write_16()	493
13.110.3.15 l4_vm_vmx_write_32()	494

13.110.3.16 l4_vm_vmx_write_64()	494
13.110.3.17 l4_vm_vmx_write_nat()	495
13.111 Vbus API	495
13.111.1 Detailed Description	496
13.112 Video API	497
13.112.1 Detailed Description	498
13.112.2 Typedef Documentation	498
13.112.2.1 l4re_video_view_t	498
13.112.3 Enumeration Type Documentation	499
13.112.3.1 l4re_video_goos_info_flags_t	499
13.112.3.2 l4re_video_view_info_flags_t	499
13.112.4 Function Documentation	499
13.112.4.1 l4re_video_goos_create_buffer()	500
13.112.4.2 l4re_video_goos_create_view()	500
13.112.4.3 l4re_video_goos_delete_buffer()	500
13.112.4.4 l4re_video_goos_delete_view()	502
13.112.4.5 l4re_video_goos_get_static_buffer()	502
13.112.4.6 l4re_video_goos_get_view()	502
13.112.4.7 l4re_video_goos_info()	503
13.112.4.8 l4re_video_goos_refresh()	503
13.112.4.9 l4re_video_view_get_info()	504
13.112.4.10 l4re_video_view_refresh()	504
13.112.4.11 l4re_video_view_set_info()	505
13.112.4.12 l4re_video_view_set_viewport()	505
13.112.4.13 l4re_video_view_stack()	505
13.113 Virtual Console	506
13.113.1 Detailed Description	507
13.113.2 Enumeration Type Documentation	507
13.113.2.1 L4_vcon_i_flags	507
13.113.2.2 L4_vcon_l_flags	508
13.113.2.3 L4_vcon_o_flags	508
13.113.2.4 L4_vcon_size_consts	508
13.113.3 Function Documentation	509
13.113.3.1 l4_vcon_get_attr()	509
13.113.3.2 l4_vcon_get_attr_u()	510
13.113.3.3 l4_vcon_read()	510
13.113.3.4 l4_vcon_read_u()	511
13.113.3.5 l4_vcon_read_with_flags()	512
13.113.3.6 l4_vcon_send()	513
13.113.3.7 l4_vcon_send_u()	514
13.113.3.8 l4_vcon_set_attr()	515
13.113.3.9 l4_vcon_set_attr_u()	516

13.113.3.10 l4_vcon_write()	516
13.113.3.11 l4_vcon_write_u()	517
13.114 Virtual Machines	518
13.114.1 Detailed Description	519
13.115 Virtual Registers (UTCBS)	519
13.115.1 Detailed Description	520
13.115.2 Typedef Documentation	521
13.115.2.1 l4_utcb_t	521
13.115.3 Function Documentation	521
13.115.3.1 l4_utcb_br()	521
13.115.3.2 l4_utcb_mr()	522
13.115.3.3 l4_utcb_tcr()	522
13.116 amd64 Virtual Registers (UTCB)	523
13.116.1 Detailed Description	523
13.117 vCPU API	523
13.117.1 Detailed Description	524
13.117.2 Enumeration Type Documentation	525
13.117.2.1 L4_vcpu_state_flags	525
13.117.2.2 L4_vcpu_state_offset	525
13.117.2.3 L4_vcpu_sticky_flags	525
13.118 vCPU Support Library	526
13.118.1 Detailed Description	527
13.118.2 Function Documentation	527
13.118.2.1 l4vcpu_irq_disable()	527
13.118.2.2 l4vcpu_irq_disable_save()	528
13.118.2.3 l4vcpu_irq_enable()	528
13.118.2.4 l4vcpu_irq_restore()	529
13.118.2.5 l4vcpu_is_irq_entry()	530
13.118.2.6 l4vcpu_is_page_fault_entry()	531
13.118.2.7 l4vcpu_print_state()	532
13.118.2.8 l4vcpu_wait_for_event()	532
13.119 x86 Virtual Registers (UTCB)	533
13.119.1 Detailed Description	533
13.119.2 Enumeration Type Documentation	533
13.119.2.1 L4_utcb_consts_x86	534
14 Namespace Documentation	535
14.1 cxx Namespace Reference	535
14.1.1 Detailed Description	537
14.2 cxx::Bits Namespace Reference	537
14.2.1 Detailed Description	538
14.3 L4 Namespace Reference	538

14.3.1 Detailed Description	541
14.3.2 Enumeration Type Documentation	542
14.3.2.1 anonymous enum	542
14.3.3 Function Documentation	542
14.3.3.1 cap_cast() [1/2]	542
14.3.3.2 cap_cast() [2/2]	543
14.3.3.3 cap_dynamic_cast()	543
14.3.3.4 cap_reinterpret_cast() [1/2]	544
14.3.3.5 cap_reinterpret_cast() [2/2]	545
14.3.3.6 round_order()	545
14.3.3.7 throw_ipc_exception() [1/2]	546
14.3.3.8 throw_ipc_exception() [2/2]	547
14.3.3.9 trunc_order()	548
14.4 L4::lpc Namespace Reference	548
14.4.1 Detailed Description	550
14.4.2 Function Documentation	550
14.4.2.1 buf_cp_in()	551
14.4.2.2 buf_cp_out()	551
14.4.2.3 buf_in()	552
14.4.2.4 make_cap()	552
14.4.2.5 make_cap_full()	553
14.4.2.6 make_cap_rw()	553
14.4.2.7 make_cap_rws()	554
14.4.2.8 msg_ptr()	555
14.4.2.9 read()	555
14.4.2.10 str_cp_in()	555
14.5 L4::lpc::Msg Namespace Reference	556
14.5.1 Detailed Description	557
14.5.2 Enumeration Type Documentation	557
14.5.2.1 anonymous enum	557
14.5.3 Function Documentation	558
14.5.3.1 align_to() [1/2]	558
14.5.3.2 align_to() [2/2]	558
14.5.3.3 check_size() [1/2]	559
14.5.3.4 check_size() [2/2]	560
14.5.3.5 msg_add()	560
14.5.3.6 msg_get()	561
14.6 L4::lpc_svr Namespace Reference	561
14.6.1 Detailed Description	562
14.7 L4::Typeid Namespace Reference	562
14.7.1 Detailed Description	563
14.8 L4::Types Namespace Reference	563

14.8.1 Detailed Description	564
14.9 L4Re Namespace Reference	564
14.9.1 Detailed Description	566
14.9.2 Typedef Documentation	566
14.9.2.1 Shared_cap	566
14.9.2.2 shared_cap	566
14.9.2.3 Shared_del_cap	567
14.9.2.4 shared_del_cap	567
14.9.2.5 Unique_cap	568
14.9.2.6 unique_cap	568
14.9.2.7 Unique_del_cap	569
14.9.2.8 unique_del_cap	569
14.9.3 Function Documentation	569
14.9.3.1 chkcap()	570
14.9.3.2 chkipc()	571
14.9.3.3 chksys() [1/3]	572
14.9.3.4 chksys() [2/3]	573
14.9.3.5 chksys() [3/3]	574
14.9.3.6 make_shared_cap()	575
14.9.3.7 make_shared_del_cap()	576
14.9.3.8 make_unique_cap()	577
14.9.3.9 make_unique_del_cap()	578
14.9.3.10 throw_error()	578
14.10 L4Re::Util Namespace Reference	579
14.10.1 Detailed Description	581
14.10.2 Typedef Documentation	581
14.10.2.1 Shared_cap	582
14.10.2.2 shared_cap	583
14.10.2.3 Shared_del_cap	584
14.10.2.4 shared_del_cap	584
14.10.2.5 Unique_cap	585
14.10.2.6 unique_cap	586
14.10.2.7 Unique_del_cap	586
14.10.2.8 unique_del_cap	587
14.10.3 Function Documentation	588
14.10.3.1 kumem_alloc()	588
14.10.3.2 make_ref_cap()	588
14.10.3.3 make_ref_del_cap()	589
14.10.3.4 make_shared_cap()	589
14.10.3.5 make_shared_del_cap()	590
14.10.3.6 make_unique_cap()	591
14.10.3.7 make_unique_del_cap()	591

14.10.4 Variable Documentation	592
14.10.4.1 cap_alloc	592
14.11 L4Re::Vfs Namespace Reference	592
14.11.1 Detailed Description	593
14.12 L4vbus Namespace Reference	593
14.12.1 Detailed Description	594
14.13 L4virtio Namespace Reference	594
14.13.1 Detailed Description	594
15 Data Structure Documentation	595
15.1 cxx::Auto_ptr< T > Class Template Reference	595
15.1.1 Detailed Description	596
15.1.2 Constructor & Destructor Documentation	596
15.1.2.1 Auto_ptr() [1/2]	596
15.1.2.2 Auto_ptr() [2/2]	597
15.1.3 Member Function Documentation	597
15.1.3.1 get()	597
15.1.3.2 operator=()	597
15.1.3.3 release()	598
15.2 cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC > Class Template Reference	599
15.2.1 Detailed Description	601
15.2.2 Constructor & Destructor Documentation	601
15.2.2.1 Avl_map()	601
15.2.3 Member Function Documentation	602
15.2.3.1 insert()	602
15.2.3.2 operator[]() [1/2]	603
15.2.3.3 operator[]() [2/2]	604
15.3 cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC > Class Template Reference	604
15.3.1 Detailed Description	606
15.4 cxx::Avl_tree< Node, Get_key, Compare > Class Template Reference	607
15.4.1 Detailed Description	610
15.4.2 Member Function Documentation	610
15.4.2.1 insert()	611
15.4.2.2 remove()	611
15.5 cxx::Avl_tree_node Class Reference	612
15.5.1 Detailed Description	613
15.6 cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc > Class Template Reference	614
15.6.1 Detailed Description	615
15.6.2 Member Enumeration Documentation	616
15.6.2.1 anonymous enum	616
15.6.3 Member Function Documentation	616
15.6.3.1 alloc()	616

15.6.3.2 free()	617
15.6.3.3 free_objects()	618
15.6.3.4 total_objects()	619
15.7 cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i Struct Reference	620
15.7.1 Detailed Description	621
15.8 cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc > Class Template Reference	621
15.8.1 Detailed Description	623
15.8.2 Member Enumeration Documentation	624
15.8.2.1 anonymous enum	624
15.8.3 Member Function Documentation	624
15.8.3.1 alloc()	624
15.8.3.2 free()	625
15.8.3.3 free_objects()	625
15.8.3.4 total_objects()	626
15.9 cxx::Bitfield< T, LSB, MSB > Class Template Reference	627
15.9.1 Detailed Description	628
15.9.2 Member Typedef Documentation	628
15.9.2.1 Bits_type	628
15.9.2.2 Shift_type	629
15.9.3 Member Enumeration Documentation	629
15.9.3.1 anonymous enum	629
15.9.3.2 Masks	629
15.9.4 Member Function Documentation	629
15.9.4.1 get()	630
15.9.4.2 get_unshifted()	630
15.9.4.3 set()	631
15.9.4.4 set_dirty()	632
15.9.4.5 set_unshifted()	633
15.9.4.6 set_unshifted_dirty()	633
15.9.4.7 val()	634
15.9.4.8 val_dirty()	636
15.9.4.9 val_unshifted()	637
15.10 cxx::Bitfield< T, LSB, MSB >::Value< TT > Class Template Reference	638
15.10.1 Detailed Description	639
15.11 cxx::Bitfield< T, LSB, MSB >::Value_base< TT > Class Template Reference	639
15.11.1 Detailed Description	640
15.12 cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT > Class Template Reference	641
15.12.1 Detailed Description	642
15.13 cxx::Bitmap< BITS > Class Template Reference	642
15.13.1 Detailed Description	644
15.13.2 Member Function Documentation	645
15.13.2.1 scan_zero()	645

15.14 cxx::Bitmap_base Class Reference	646
15.14.1 Detailed Description	648
15.14.2 Member Enumeration Documentation	648
15.14.2.1 anonymous enum	648
15.14.3 Member Function Documentation	649
15.14.3.1 bit() [1/2]	649
15.14.3.2 bit() [2/2]	649
15.14.3.3 bit_index()	650
15.14.3.4 clear_bit()	651
15.14.3.5 operator[]() [1/2]	651
15.14.3.6 operator[]() [2/2]	651
15.14.3.7 scan_zero()	652
15.14.3.8 set_bit()	653
15.14.3.9 word_index()	653
15.15 cxx::Bitmap_base::Bit Class Reference	654
15.15.1 Detailed Description	654
15.16 cxx::Bitmap_base::Char< BITS > Class Template Reference	655
15.16.1 Detailed Description	655
15.17 cxx::Bitmap_base::Word< BITS > Class Template Reference	655
15.17.1 Detailed Description	656
15.18 cxx::Bits::Avl_map_get_key< KEY_TYPE > Struct Template Reference	657
15.18.1 Detailed Description	657
15.19 cxx::Bits::Avl_set_get_key< KEY_TYPE > Struct Template Reference	657
15.19.1 Detailed Description	658
15.20 cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY > Class Template Reference	658
15.20.1 Detailed Description	661
15.20.2 Member Enumeration Documentation	662
15.20.2.1 anonymous enum	662
15.20.3 Constructor & Destructor Documentation	662
15.20.3.1 Base_avl_set() [1/2]	662
15.20.3.2 Base_avl_set() [2/2]	663
15.20.4 Member Function Documentation	663
15.20.4.1 begin() [1/2]	663
15.20.4.2 begin() [2/2]	664
15.20.4.3 end() [1/2]	665
15.20.4.4 end() [2/2]	666
15.20.4.5 erase()	666
15.20.4.6 find_node()	667
15.20.4.7 insert()	668
15.20.4.8 lower_bound_node()	669
15.20.4.9 rbegin() [1/2]	669
15.20.4.10 rbegin() [2/2]	670

15.20.4.11	remove()	670
15.20.4.12	rend() [1/2]	671
15.20.4.13	rend() [2/2]	672
15.21	cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node Class Reference	672
15.21.1	Detailed Description	673
15.21.2	Member Function Documentation	673
15.21.2.1	operator*()	673
15.21.2.2	operator->()	674
15.21.2.3	valid()	674
15.22	cxx::Bits::Basic_list< POLICY > Class Template Reference	674
15.22.1	Detailed Description	677
15.22.2	Member Function Documentation	677
15.22.2.1	clear()	677
15.22.2.2	iter()	677
15.23	cxx::Bits::Bst< Node, Get_key, Compare > Class Template Reference	678
15.23.1	Detailed Description	681
15.23.2	Member Function Documentation	681
15.23.2.1	begin() [1/2]	682
15.23.2.2	begin() [2/2]	682
15.23.2.3	dir() [1/2]	683
15.23.2.4	dir() [2/2]	684
15.23.2.5	end() [1/2]	684
15.23.2.6	end() [2/2]	685
15.23.2.7	find()	685
15.23.2.8	find_node()	686
15.23.2.9	lower_bound_node()	687
15.23.2.10	rbegin() [1/2]	688
15.23.2.11	rbegin() [2/2]	689
15.23.2.12	remove_all()	689
15.23.2.13	remove_tree()	690
15.23.2.14	rend() [1/2]	691
15.23.2.15	rend() [2/2]	691
15.24	cxx::Bits::Bst_node Class Reference	692
15.24.1	Detailed Description	693
15.25	cxx::Bits::Direction Struct Reference	694
15.25.1	Detailed Description	695
15.25.2	Member Enumeration Documentation	695
15.25.2.1	Direction_e	695
15.25.3	Member Function Documentation	695
15.25.3.1	operator"!()	695
15.26	cxx::Bits::Smart_ptr_list< ITEM > Class Template Reference	696
15.26.1	Detailed Description	697

15.26.2 Member Function Documentation	697
15.26.2.1 pop_front()	697
15.27 cxx::Bits::Smart_ptr_list_item< T, STORE_T > Class Template Reference	698
15.27.1 Detailed Description	698
15.28 cxx::H_list< T, POLICY > Class Template Reference	699
15.28.1 Detailed Description	702
15.28.2 Member Function Documentation	702
15.28.2.1 erase()	702
15.28.2.2 insert()	703
15.28.2.3 insert_after()	704
15.28.2.4 insert_before()	705
15.28.2.5 iter()	705
15.28.2.6 pop_front()	706
15.28.2.7 remove()	707
15.28.2.8 replace()	708
15.29 cxx::H_list_item_t< ELEM_TYPE > Class Template Reference	708
15.29.1 Detailed Description	710
15.29.2 Constructor & Destructor Documentation	710
15.29.2.1 H_list_item_t()	710
15.29.2.2 ~H_list_item_t()	711
15.30 cxx::H_list_t< T > Struct Template Reference	711
15.30.1 Detailed Description	713
15.31 cxx::List< D, Alloc > Class Template Reference	714
15.31.1 Detailed Description	715
15.31.2 Member Function Documentation	715
15.31.2.1 operator[]() [1/2]	715
15.31.2.2 operator[]() [2/2]	715
15.32 cxx::List< D, Alloc >::Iter Class Reference	716
15.32.1 Detailed Description	716
15.33 cxx::List_alloc Class Reference	717
15.33.1 Detailed Description	717
15.33.2 Constructor & Destructor Documentation	717
15.33.2.1 List_alloc()	718
15.33.3 Member Function Documentation	718
15.33.3.1 alloc()	718
15.33.3.2 alloc_max()	718
15.33.3.3 avail()	719
15.33.3.4 free()	720
15.34 cxx::List_item Class Reference	720
15.34.1 Detailed Description	721
15.34.2 Member Function Documentation	721
15.34.2.1 push_back()	721

15.34.2.2 push_front()	722
15.34.2.3 remove()	722
15.35 cxx::List_item::Iter Class Reference	723
15.35.1 Detailed Description	725
15.36 cxx::List_item::T_iter< T, Poly > Class Template Reference	725
15.36.1 Detailed Description	727
15.37 cxx::Lt_functor< Obj > Struct Template Reference	728
15.37.1 Detailed Description	728
15.38 cxx::New_allocator< _Type > Class Template Reference	728
15.38.1 Detailed Description	729
15.39 cxx::Nothrow Class Reference	730
15.39.1 Detailed Description	730
15.40 cxx::Pair< First, Second > Struct Template Reference	730
15.40.1 Detailed Description	731
15.40.2 Constructor & Destructor Documentation	731
15.40.2.1 Pair()	731
15.41 cxx::Pair_first_compare< Cmp, Typ > Class Template Reference	732
15.41.1 Detailed Description	732
15.41.2 Constructor & Destructor Documentation	733
15.41.2.1 Pair_first_compare()	733
15.41.3 Member Function Documentation	733
15.41.3.1 operator>()	733
15.42 cxx::Ref_obj_list_item< T > Struct Template Reference	734
15.42.1 Detailed Description	735
15.43 cxx::Ref_ptr< T, CNT > Class Template Reference	735
15.43.1 Detailed Description	736
15.43.2 Constructor & Destructor Documentation	737
15.43.2.1 Ref_ptr() [1/3]	737
15.43.2.2 Ref_ptr() [2/3]	738
15.43.2.3 Ref_ptr() [3/3]	738
15.43.3 Member Function Documentation	738
15.43.3.1 get()	738
15.43.3.2 ptr()	739
15.43.3.3 release()	739
15.44 cxx::S_list< T, POLICY > Class Template Reference	739
15.44.1 Detailed Description	742
15.44.2 Member Function Documentation	742
15.44.2.1 pop_front()	742
15.45 cxx::Slab< Type, Slab_size, Max_free, Alloc > Class Template Reference	743
15.45.1 Detailed Description	745
15.45.2 Member Function Documentation	745
15.45.2.1 alloc()	745

15.45.2.2 free()	746
15.46 cxx::Slab_static< Type, Slab_size, Max_free, Alloc > Class Template Reference	746
15.46.1 Detailed Description	748
15.46.2 Member Function Documentation	750
15.46.2.1 alloc()	750
15.47 cxx::static_vector< T, IDX > Class Template Reference	751
15.47.1 Detailed Description	751
15.48 cxx::String Class Reference	752
15.48.1 Detailed Description	754
15.48.2 Constructor & Destructor Documentation	755
15.48.2.1 String()	755
15.48.3 Member Function Documentation	755
15.48.3.1 find()	755
15.48.3.2 from_dec()	756
15.48.3.3 from_hex()	757
15.48.3.4 starts_with()	758
15.49 cxx::Weak_ref< T > Class Template Reference	759
15.49.1 Detailed Description	760
15.50 cxx::Weak_ref_base Class Reference	761
15.50.1 Detailed Description	763
15.51 Elf32_Dyn Struct Reference	764
15.51.1 Detailed Description	764
15.52 Elf32_Ehdr Struct Reference	765
15.52.1 Detailed Description	766
15.52.2 Field Documentation	766
15.52.2.1 e_phnum	766
15.52.2.2 e_shnum	766
15.53 Elf32_Phdr Struct Reference	767
15.53.1 Detailed Description	767
15.54 Elf32_Shdr Struct Reference	768
15.54.1 Detailed Description	769
15.55 Elf32_Sym Struct Reference	769
15.55.1 Detailed Description	770
15.56 Elf64_Dyn Struct Reference	770
15.56.1 Detailed Description	770
15.57 Elf64_Ehdr Struct Reference	771
15.57.1 Detailed Description	772
15.57.2 Field Documentation	772
15.57.2.1 e_phnum	772
15.57.2.2 e_shnum	772
15.58 Elf64_Phdr Struct Reference	773
15.58.1 Detailed Description	773

15.59 Elf64_Shdr Struct Reference	774
15.59.1 Detailed Description	775
15.60 Elf64_Sym Struct Reference	775
15.60.1 Detailed Description	776
15.61 L4::Alloc_list Class Reference	776
15.61.1 Detailed Description	776
15.62 L4::Arm_smccc Class Reference	777
15.62.1 Detailed Description	777
15.62.2 Member Function Documentation	777
15.62.2.1 call()	777
15.63 L4::Base_exception Class Reference	778
15.63.1 Detailed Description	779
15.64 L4::Basic_registry Class Reference	780
15.64.1 Detailed Description	781
15.64.2 Member Function Documentation	781
15.64.2.1 dispatch()	781
15.64.2.2 find()	781
15.65 L4::Bounds_error Class Reference	782
15.65.1 Detailed Description	784
15.66 L4::Cap< T > Class Template Reference	785
15.66.1 Detailed Description	787
15.66.2 Constructor & Destructor Documentation	787
15.66.2.1 Cap() [1 / 4]	787
15.66.2.2 Cap() [2 / 4]	788
15.66.2.3 Cap() [3 / 4]	788
15.66.2.4 Cap() [4 / 4]	788
15.66.3 Member Function Documentation	789
15.66.3.1 copy()	789
15.66.3.2 move()	789
15.67 L4::Cap_base Class Reference	790
15.67.1 Detailed Description	792
15.67.2 Member Enumeration Documentation	792
15.67.2.1 Cap_type	792
15.67.2.2 No_init_type	793
15.67.3 Constructor & Destructor Documentation	793
15.67.3.1 Cap_base() [1 / 2]	793
15.67.3.2 Cap_base() [2 / 2]	793
15.67.4 Member Function Documentation	794
15.67.4.1 cap()	794
15.67.4.2 copy()	795
15.67.4.3 fpage()	796
15.67.4.4 is_valid()	797

15.67.4.5 move()	798
15.67.4.6 snd_base()	799
15.67.4.7 validate() [1/2]	800
15.67.4.8 validate() [2/2]	800
15.68 L4::Com_error Class Reference	801
15.68.1 Detailed Description	803
15.68.2 Constructor & Destructor Documentation	804
15.68.2.1 Com_error()	804
15.69 L4::Debugger Class Reference	804
15.69.1 Detailed Description	807
15.69.2 Member Function Documentation	807
15.69.2.1 get_object_name()	807
15.69.2.2 global_id()	808
15.69.2.3 kobj_to_id()	808
15.69.2.4 query_log_name()	809
15.69.2.5 query_log_typeid()	809
15.69.2.6 set_object_name()	810
15.69.2.7 switch_log()	810
15.70 L4::Element_already_exists Class Reference	811
15.70.1 Detailed Description	812
15.71 L4::Element_not_found Class Reference	813
15.71.1 Detailed Description	814
15.72 L4::Epiface Struct Reference	815
15.72.1 Detailed Description	816
15.72.2 Member Function Documentation	816
15.72.2.1 dispatch()	816
15.72.2.2 get_buffer_demand()	817
15.72.2.3 obj_cap()	817
15.72.2.4 server_iface()	818
15.72.2.5 set_server()	818
15.73 L4::Epiface_t< Derived, IFACE, BASE, bool > Struct Template Reference	819
15.73.1 Detailed Description	821
15.73.2 Member Function Documentation	821
15.73.2.1 dispatch()	821
15.74 L4::Epiface_t0< RPC_IFACE, BASE > Struct Template Reference	822
15.74.1 Detailed Description	823
15.74.2 Member Function Documentation	824
15.74.2.1 obj_cap()	824
15.75 L4::Exception Class Reference	824
15.75.1 Detailed Description	825
15.75.2 Member Typedef Documentation	825
15.75.2.1 Rpcs	825

15.76 L4::Exception_tracer Class Reference	825
15.76.1 Detailed Description	827
15.77 L4::Factory Class Reference	827
15.77.1 Detailed Description	830
15.77.2 Member Function Documentation	830
15.77.2.1 create() [1/2]	830
15.77.2.2 create() [2/2]	831
15.77.2.3 create_factory()	833
15.77.2.4 create_gate()	834
15.77.2.5 create_irq()	836
15.77.2.6 create_task()	836
15.77.2.7 create_thread()	837
15.77.2.8 create_vm()	838
15.78 L4::Factory::Lstr Struct Reference	839
15.78.1 Detailed Description	840
15.78.2 Constructor & Destructor Documentation	840
15.78.2.1 Lstr()	840
15.79 L4::Factory::Nil Struct Reference	840
15.79.1 Detailed Description	841
15.80 L4::Factory::S Class Reference	841
15.80.1 Detailed Description	842
15.80.2 Constructor & Destructor Documentation	842
15.80.2.1 S() [1/2]	842
15.80.2.2 S() [2/2]	843
15.80.3 Member Function Documentation	843
15.80.3.1 operator l4_msgtag_t()	843
15.80.3.2 put() [1/6]	843
15.80.3.3 put() [2/6]	844
15.80.3.4 put() [3/6]	844
15.80.3.5 put() [4/6]	845
15.80.3.6 put() [5/6]	845
15.80.3.7 put() [6/6]	845
15.81 L4::lcu Class Reference	846
15.81.1 Detailed Description	848
15.81.2 Member Function Documentation	848
15.81.2.1 bind()	848
15.81.2.2 info()	850
15.81.2.3 mask()	851
15.81.2.4 msi_info()	852
15.81.2.5 set_mode()	852
15.81.2.6 unbind()	853
15.82 L4::lcu::Info Class Reference	854

15.82.1 Detailed Description	855
15.83 L4::Invalid_capability Class Reference	855
15.83.1 Detailed Description	857
15.83.2 Constructor & Destructor Documentation	858
15.83.2.1 Invalid_capability()	858
15.83.3 Member Function Documentation	858
15.83.3.1 cap()	858
15.84 L4::io_pager Class Reference	859
15.84.1 Detailed Description	860
15.84.2 Member Function Documentation	861
15.84.2.1 io_page_fault()	861
15.85 L4::lomu Class Reference	861
15.85.1 Detailed Description	863
15.85.2 Member Function Documentation	864
15.85.2.1 bind()	864
15.85.2.2 unbind()	864
15.86 L4::IOModifier Class Reference	864
15.86.1 Detailed Description	865
15.87 L4::lpc::Array< ELEM_TYPE, LEN_TYPE > Struct Template Reference	865
15.87.1 Detailed Description	868
15.88 L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX > Struct Template Reference	868
15.88.1 Detailed Description	869
15.89 L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE > Struct Template Reference	869
15.89.1 Detailed Description	871
15.90 L4::lpc::As_value< T > Struct Template Reference	871
15.90.1 Detailed Description	872
15.91 L4::lpc::Buf_item Class Reference	872
15.91.1 Detailed Description	873
15.92 L4::lpc::Call Struct Reference	873
15.92.1 Detailed Description	874
15.93 L4::lpc::Call_t< RIGHTS > Struct Template Reference	874
15.93.1 Detailed Description	875
15.94 L4::lpc::Call_zero_send_timeout Struct Reference	875
15.94.1 Detailed Description	876
15.95 L4::lpc::Cap< T > Class Template Reference	877
15.95.1 Detailed Description	878
15.95.2 Member Enumeration Documentation	878
15.95.2.1 anonymous enum	878
15.95.3 Constructor & Destructor Documentation	878
15.95.3.1 Cap()	879
15.95.4 Member Function Documentation	879
15.95.4.1 from_ci()	879

15.96 L4::lpc::Gen_fpage< T > Class Template Reference	879
15.96.1 Detailed Description	881
15.96.2 Member Function Documentation	881
15.96.2.1 cap_received()	881
15.96.2.2 id_received()	881
15.96.2.3 is_compound()	882
15.96.2.4 local_id_received()	882
15.97 L4::lpc::In_out< T > Struct Template Reference	882
15.97.1 Detailed Description	882
15.98 L4::lpc::Iostream Class Reference	883
15.98.1 Detailed Description	886
15.98.2 Constructor & Destructor Documentation	887
15.98.2.1 Iostream()	887
15.98.3 Member Function Documentation	887
15.98.3.1 call()	887
15.98.3.2 get() [1/3]	888
15.98.3.3 get() [2/3]	888
15.98.3.4 get() [3/3]	889
15.98.3.5 put() [1/2]	889
15.98.3.6 put() [2/2]	890
15.98.3.7 reply_and_wait() [1/2]	890
15.98.3.8 reply_and_wait() [2/2]	891
15.98.3.9 reset()	891
15.99 L4::lpc::Istream Class Reference	892
15.99.1 Detailed Description	895
15.99.2 Constructor & Destructor Documentation	896
15.99.2.1 Istream()	896
15.99.3 Member Function Documentation	896
15.99.3.1 get() [1/3]	896
15.99.3.2 get() [2/3]	897
15.99.3.3 get() [3/3]	897
15.99.3.4 receive()	898
15.99.3.5 reset()	898
15.99.3.6 skip()	899
15.99.3.7 tag() [1/2]	899
15.99.3.8 tag() [2/2]	900
15.99.3.9 wait() [1/2]	900
15.99.3.10 wait() [2/2]	901
15.100 L4::lpc::Msg::Clnt_val_ops< MTYPE, DIR, CLASS > Struct Template Reference	902
15.100.1 Detailed Description	902
15.101 L4::lpc::Msg::Cls_buffer Struct Reference	904
15.101.1 Detailed Description	905

15.102 L4::lpc::Msg::Cls_data Struct Reference	905
15.102.1 Detailed Description	906
15.103 L4::lpc::Msg::Cls_item Struct Reference	906
15.103.1 Detailed Description	907
15.104 L4::lpc::Msg::Dir_in Struct Reference	907
15.104.1 Detailed Description	907
15.105 L4::lpc::Msg::Dir_out Struct Reference	908
15.105.1 Detailed Description	908
15.106 L4::lpc::Msg::Do_in_data Struct Reference	909
15.106.1 Detailed Description	909
15.107 L4::lpc::Msg::Do_in_items Struct Reference	910
15.107.1 Detailed Description	910
15.108 L4::lpc::Msg::Do_out_data Struct Reference	911
15.108.1 Detailed Description	912
15.109 L4::lpc::Msg::Do_out_items Struct Reference	912
15.109.1 Detailed Description	913
15.110 L4::lpc::Msg::Do_rcv_buffers Struct Reference	913
15.110.1 Detailed Description	914
15.111 L4::lpc::Msg::Elem< Array< A, LEN > & > Struct Template Reference	915
15.111.1 Detailed Description	915
15.112 L4::lpc::Msg::Elem< Array< A, LEN > > Struct Template Reference	916
15.112.1 Detailed Description	916
15.113 L4::lpc::Msg::Elem< Array_ref< A, LEN > & > Struct Template Reference	917
15.113.1 Detailed Description	917
15.114 L4::lpc::Msg::Is_valid_rpc_type< T > Struct Template Reference	918
15.114.1 Detailed Description	919
15.115 L4::lpc::Msg::Svr_arg_pack< IPC_TYPE > Struct Template Reference	920
15.115.1 Detailed Description	920
15.116 L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS > Struct Template Reference	920
15.116.1 Detailed Description	921
15.117 L4::lpc::Msg_ptr< T > Class Template Reference	922
15.117.1 Detailed Description	922
15.117.2 Constructor & Destructor Documentation	922
15.117.2.1 Msg_ptr()	922
15.118 L4::lpc::Opt< T > Struct Template Reference	923
15.118.1 Detailed Description	924
15.119 L4::lpc::Ostream Class Reference	924
15.119.1 Detailed Description	927
15.119.2 Member Function Documentation	927
15.119.2.1 put() [1/2]	927
15.119.2.2 put() [2/2]	928
15.119.2.3 send()	928

15.119.2.4 tag() [1/2]	929
15.119.2.5 tag() [2/2]	930
15.120 L4::lpc::Out< T > Struct Template Reference	930
15.120.1 Detailed Description	930
15.121 L4::lpc::Ret_array< T > Struct Template Reference	931
15.121.1 Detailed Description	931
15.122 L4::lpc::Send_only Struct Reference	932
15.122.1 Detailed Description	932
15.123 L4::lpc::Small_buf Class Reference	933
15.123.1 Detailed Description	933
15.123.2 Constructor & Destructor Documentation	933
15.123.2.1 Small_buf() [1/2]	933
15.123.2.2 Small_buf() [2/2]	934
15.124 L4::lpc::Snd_item Class Reference	934
15.124.1 Detailed Description	935
15.125 L4::lpc::Str_cp_in< T > Class Template Reference	935
15.125.1 Detailed Description	935
15.125.2 Constructor & Destructor Documentation	936
15.125.2.1 Str_cp_in()	936
15.126 L4::lpc::Varg Class Reference	936
15.126.1 Detailed Description	937
15.126.2 Member Function Documentation	938
15.126.2.1 data()	938
15.126.2.2 get_value()	938
15.126.2.3 is_nil()	939
15.126.2.4 is_of()	939
15.126.2.5 is_of_int()	940
15.126.2.6 length()	941
15.126.2.7 tag()	941
15.126.2.8 type()	941
15.126.2.9 value()	941
15.127 L4::lpc::Varg_list< MAX > Class Template Reference	942
15.127.1 Detailed Description	943
15.128 L4::lpc::Varg_list_ref Class Reference	944
15.128.1 Detailed Description	945
15.128.2 Constructor & Destructor Documentation	945
15.128.2.1 Varg_list_ref()	945
15.129 L4::lpc::Varg_list_ref::Iterator Class Reference	946
15.129.1 Detailed Description	946
15.130 L4::lpc::gate Class Reference	947
15.130.1 Detailed Description	949
15.130.2 Member Function Documentation	949

15.130.2.1 <code>get_infos()</code>	949
15.131 L4::lpc_svr::Br_manager_no_buffers Class Reference	950
15.131.1 Detailed Description	952
15.131.2 Member Function Documentation	952
15.131.2.1 <code>alloc_buffer_demand()</code>	952
15.132 L4::lpc_svr::Compound_reply Struct Reference	953
15.132.1 Detailed Description	954
15.133 L4::lpc_svr::Default_loop_hooks Struct Reference	954
15.133.1 Detailed Description	956
15.134 L4::lpc_svr::Default_setup_wait Struct Reference	956
15.134.1 Detailed Description	957
15.135 L4::lpc_svr::Default_timeout Struct Reference	957
15.135.1 Detailed Description	959
15.136 L4::lpc_svr::Direct_dispatch< R > Struct Template Reference	959
15.136.1 Detailed Description	960
15.137 L4::lpc_svr::Direct_dispatch< R * > Struct Template Reference	961
15.137.1 Detailed Description	961
15.138 L4::lpc_svr::Exc_dispatch< R, Exc > Struct Template Reference	962
15.138.1 Detailed Description	963
15.139 L4::lpc_svr::Ignore_errors Struct Reference	963
15.139.1 Detailed Description	965
15.140 L4::lpc_svr::Server_iface Class Reference	965
15.140.1 Detailed Description	968
15.140.2 Member Function Documentation	968
15.140.2.1 <code>add_timeout()</code>	968
15.140.2.2 <code>alloc_buffer_demand()</code>	968
15.140.2.3 <code>get_rcv_cap()</code>	969
15.140.2.4 <code>rcv_cap()</code> [1/2]	970
15.140.2.5 <code>rcv_cap()</code> [2/2]	970
15.140.2.6 <code>realloc_rcv_cap()</code>	971
15.140.2.7 <code>remove_timeout()</code>	972
15.141 L4::lpc_svr::Timed_work< HOOKS > Class Template Reference	972
15.141.1 Detailed Description	973
15.142 L4::lpc_svr::Timeout Class Reference	973
15.142.1 Detailed Description	974
15.142.2 Member Function Documentation	974
15.142.2.1 <code>expired()</code>	975
15.142.2.2 <code>timeout()</code>	975
15.143 L4::lpc_svr::Timeout_queue Class Reference	976
15.143.1 Detailed Description	976
15.143.2 Member Function Documentation	976
15.143.2.1 <code>add()</code>	977

15.143.2.2	handle_expired_timeouts()	978
15.143.2.3	next_timeout()	979
15.143.2.4	remove()	979
15.143.2.5	timeout_expired()	980
15.144	L4::lpc_srv::Timeout_queue_hooks< HOOKS, BR_MAN > Class Template Reference	981
15.144.1	Detailed Description	984
15.144.2	Member Function Documentation	984
15.144.2.1	add_timeout()	984
15.144.2.2	remove_timeout()	985
15.145	L4::lrc Class Reference	986
15.145.1	Detailed Description	989
15.145.2	Member Function Documentation	989
15.145.2.1	detach()	989
15.145.2.2	receive()	990
15.145.2.3	unmask()	991
15.145.2.4	wait()	992
15.146	L4::lrc_eoi Class Reference	992
15.146.1	Detailed Description	993
15.146.2	Member Function Documentation	994
15.146.2.1	unmask()	994
15.147	L4::lrc_handler_object Struct Reference	994
15.147.1	Detailed Description	996
15.148	L4::lrc_mux Struct Reference	997
15.148.1	Detailed Description	1000
15.148.2	Member Function Documentation	1000
15.148.2.1	chain()	1000
15.149	L4::lrcp_t< Derived, BASE, bool > Struct Template Reference	1001
15.149.1	Detailed Description	1004
15.149.2	Member Function Documentation	1004
15.149.2.1	dispatch()	1004
15.149.2.2	obj_cap()	1005
15.150	L4::Kip::Mem_desc Class Reference	1005
15.150.1	Detailed Description	1007
15.150.2	Member Enumeration Documentation	1007
15.150.2.1	Info_sub_type	1007
15.150.2.2	Mem_type	1007
15.150.3	Constructor & Destructor Documentation	1008
15.150.3.1	Mem_desc()	1008
15.150.4	Member Function Documentation	1008
15.150.4.1	all() [1/2]	1008
15.150.4.2	all() [2/2]	1009
15.150.4.3	count() [1/2]	1010

15.150.4.4 count() [2/2]	1011
15.150.4.5 end()	1011
15.150.4.6 first()	1012
15.150.4.7 is_virtual()	1013
15.150.4.8 set()	1013
15.150.4.9 size()	1014
15.150.4.10 start()	1014
15.150.4.11 sub_type()	1015
15.150.4.12 type()	1015
15.151 L4::Kobject Class Reference	1015
15.151.1 Detailed Description	1016
15.151.2 Member Function Documentation	1016
15.151.2.1 cap()	1016
15.151.2.2 dec_refcnt()	1018
15.152 L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND > Class Template Reference	1018
15.152.1 Detailed Description	1020
15.152.2 Member Typedef Documentation	1020
15.152.2.1 __iface	1021
15.152.2.2 __iface_list	1021
15.152.2.3 Class	1021
15.152.3 Member Function Documentation	1021
15.152.3.1 __check_protocols__()	1021
15.152.3.2 c()	1022
15.153 L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND > Struct Template Reference	1022
15.153.1 Detailed Description	1024
15.153.2 Member Typedef Documentation	1025
15.153.2.1 __iface	1025
15.153.2.2 __iface_list	1025
15.153.2.3 Class	1026
15.153.3 Member Function Documentation	1026
15.153.3.1 __check_protocols__()	1026
15.153.3.2 c()	1026
15.154 L4::Kobject_demand< T > Struct Template Reference	1026
15.154.1 Detailed Description	1027
15.155 L4::Kobject_t< Derived, Base, PROTO, S_DEMAND > Class Template Reference	1027
15.155.1 Detailed Description	1028
15.156 L4::Kobject_typeid< T > Struct Template Reference	1029
15.156.1 Detailed Description	1029
15.156.2 Member Typedef Documentation	1030
15.156.2.1 Demand	1030
15.156.3 Member Function Documentation	1030
15.156.3.1 demand()	1030

15.156.3.2 id()	1031
15.156.3.3 proto_dispatch()	1031
15.157 L4::Kobject_typeid< void > Struct Reference	1032
15.157.1 Detailed Description	1033
15.158 L4::Kobject_x< Derived, ARGS > Struct Template Reference	1033
15.158.1 Detailed Description	1034
15.159 L4::Meta Class Reference	1034
15.159.1 Detailed Description	1036
15.159.2 Member Function Documentation	1037
15.159.2.1 interface()	1037
15.159.2.2 num_interfaces()	1037
15.159.2.3 supports()	1037
15.160 L4::Out_of_memory Class Reference	1038
15.160.1 Detailed Description	1041
15.161 L4::Pager Class Reference	1041
15.161.1 Detailed Description	1043
15.161.2 Member Function Documentation	1044
15.161.2.1 page_fault()	1044
15.162 L4::Platform_control Class Reference	1045
15.162.1 Detailed Description	1047
15.162.2 Member Enumeration Documentation	1047
15.162.2.1 Opcode	1047
15.162.3 Member Function Documentation	1047
15.162.3.1 cpu_disable()	1047
15.162.3.2 cpu_enable()	1048
15.162.3.3 system_shutdown()	1048
15.162.3.4 system_suspend()	1048
15.163 L4::Poll_timeout_kipclock Class Reference	1049
15.163.1 Detailed Description	1049
15.163.2 Constructor & Destructor Documentation	1050
15.163.2.1 Poll_timeout_kipclock()	1050
15.163.3 Member Function Documentation	1050
15.163.3.1 set()	1050
15.163.3.2 test()	1051
15.163.3.3 timed_out()	1052
15.164 L4::Proto_t< P > Struct Template Reference	1052
15.164.1 Detailed Description	1052
15.165 L4::Rcv_endpoint Class Reference	1053
15.165.1 Detailed Description	1055
15.165.2 Member Function Documentation	1056
15.165.2.1 bind_thread()	1056
15.166 L4::Registry_iface Class Reference	1057

15.166.1 Detailed Description	1058
15.166.2 Member Function Documentation	1058
15.166.2.1 register_irq_obj()	1058
15.166.2.2 register_obj() [1/3]	1059
15.166.2.3 register_obj() [2/3]	1059
15.166.2.4 register_obj() [3/3]	1060
15.166.2.5 unregister_obj()	1061
15.167 L4::Runtime_error Class Reference	1061
15.167.1 Detailed Description	1063
15.167.2 Constructor & Destructor Documentation	1064
15.167.2.1 Runtime_error()	1064
15.167.3 Member Function Documentation	1064
15.167.3.1 err_no()	1064
15.167.3.2 extra_str()	1064
15.168 L4::Scheduler Class Reference	1065
15.168.1 Detailed Description	1067
15.168.2 Member Function Documentation	1067
15.168.2.1 idle_time()	1067
15.168.2.2 info()	1068
15.168.2.3 is_online()	1069
15.168.2.4 run_thread()	1069
15.169 L4::Semaphore Struct Reference	1070
15.169.1 Detailed Description	1073
15.169.2 Member Function Documentation	1073
15.169.2.1 down()	1073
15.169.2.2 up()	1074
15.170 L4::Server< LOOP_HOOKS > Class Template Reference	1075
15.170.1 Detailed Description	1077
15.170.2 Constructor & Destructor Documentation	1077
15.170.2.1 Server()	1077
15.170.3 Member Function Documentation	1077
15.170.3.1 internal_loop()	1077
15.170.3.2 loop()	1078
15.171 L4::Server_object Class Reference	1079
15.171.1 Detailed Description	1081
15.171.2 Member Function Documentation	1082
15.171.2.1 dispatch() [1/2]	1082
15.171.2.2 dispatch() [2/2]	1082
15.172 L4::Server_object_t< IFACE, BASE > Struct Template Reference	1084
15.172.1 Detailed Description	1086
15.172.2 Member Function Documentation	1086
15.172.2.1 dispatch_meta_request()	1086

15.172.2.2 get_buffer_demand()	1087
15.172.2.3 proto_dispatch()	1087
15.173 L4::Server_object_x< Derived, IFACE, BASE > Struct Template Reference	1088
15.173.1 Detailed Description	1091
15.174 L4::Smart_cap< T, SMART > Class Template Reference	1091
15.174.1 Detailed Description	1094
15.174.2 Constructor & Destructor Documentation	1094
15.174.2.1 Smart_cap()	1094
15.175 L4::String Class Reference	1094
15.175.1 Detailed Description	1095
15.176 L4::Task Class Reference	1095
15.176.1 Detailed Description	1098
15.176.2 Member Function Documentation	1098
15.176.2.1 add_ku_mem()	1098
15.176.2.2 cap_equal()	1099
15.176.2.3 cap_valid()	1099
15.176.2.4 delete_obj()	1100
15.176.2.5 map()	1100
15.176.2.6 release_cap()	1101
15.176.2.7 unmap()	1101
15.176.2.8 unmap_batch()	1102
15.177 L4::Thread Class Reference	1103
15.177.1 Detailed Description	1106
15.177.2 Member Function Documentation	1107
15.177.2.1 control()	1107
15.177.2.2 ex_regs() [1/2]	1107
15.177.2.3 ex_regs() [2/2]	1108
15.177.2.4 modify_senders()	1109
15.177.2.5 register_del_irq()	1110
15.177.2.6 stats_time()	1111
15.177.2.7 switch_to()	1111
15.177.2.8 vcpu_control()	1112
15.177.2.9 vcpu_control_ext()	1112
15.177.2.10 vcpu_resume_commit()	1113
15.177.2.11 vcpu_resume_start()	1114
15.178 L4::Thread::Attr Class Reference	1114
15.178.1 Detailed Description	1115
15.178.2 Constructor & Destructor Documentation	1115
15.178.2.1 Attr()	1115
15.178.3 Member Function Documentation	1116
15.178.3.1 bind()	1116
15.178.3.2 exc_handler() [1/2]	1116

15.178.3.3 exc_handler() [2/2]	1116
15.178.3.4 pager() [1/2]	1117
15.178.3.5 pager() [2/2]	1117
15.178.3.6 ux_host_syscall()	1117
15.179 L4::Thread::Modify_senders Class Reference	1118
15.179.1 Detailed Description	1118
15.179.2 Member Function Documentation	1118
15.179.2.1 add()	1119
15.180 L4::Triggerable Struct Reference	1120
15.180.1 Detailed Description	1122
15.180.2 Member Function Documentation	1122
15.180.2.1 trigger()	1123
15.181 L4::Type_info Struct Reference	1124
15.181.1 Detailed Description	1124
15.182 L4::Type_info::Demand Class Reference	1125
15.182.1 Detailed Description	1126
15.182.2 Constructor & Destructor Documentation	1126
15.182.2.1 Demand()	1127
15.182.3 Member Function Documentation	1127
15.182.3.1 no_demand()	1127
15.183 L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS > Struct Template Reference	1128
15.183.1 Detailed Description	1129
15.183.2 Member Enumeration Documentation	1130
15.183.2.1 anonymous enum	1130
15.184 L4::Type_info::Demand_union_t< D1, D2 > Struct Template Reference	1130
15.184.1 Detailed Description	1132
15.185 L4::Typeid::Detail::_Rpc< OPCODE, O, X > Struct Template Reference	1133
15.185.1 Detailed Description	1134
15.186 L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y > Struct Template Reference	1134
15.186.1 Detailed Description	1135
15.187 L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... > Struct Template Reference	1135
15.187.1 Detailed Description	1136
15.188 L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y > Struct Template Reference	1136
15.188.1 Detailed Description	1137
15.189 L4::Typeid::Detail::Rpc_end Struct Reference	1137
15.189.1 Detailed Description	1138
15.190 L4::Typeid::P_dispatch< LIST > Struct Template Reference	1138
15.190.1 Detailed Description	1138
15.191 L4::Typeid::Raw_ipc< CLASS > Struct Template Reference	1139
15.191.1 Detailed Description	1139
15.192 L4::Typeid::Rpc_nocode< OPERATION > Struct Template Reference	1139
15.192.1 Detailed Description	1141

15.193 L4::Typeid::Rpc< RPCS > Struct Template Reference	1142
15.193.1 Detailed Description	1143
15.194 L4::Typeid::Rpc_code< OPCODE_TYPE > Struct Template Reference	1144
15.194.1 Detailed Description	1144
15.195 L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS > Struct Template Reference	1145
15.195.1 Detailed Description	1146
15.196 L4::Typeid::Rpc_sys< ARG > Struct Template Reference	1147
15.196.1 Detailed Description	1148
15.197 L4::Types::Bool< V > Struct Template Reference	1149
15.197.1 Detailed Description	1150
15.198 L4::Types::False Struct Reference	1151
15.198.1 Detailed Description	1152
15.199 L4::Types::Flags< BITS_ENUM, UNDERLYING > Class Template Reference	1152
15.199.1 Detailed Description	1154
15.199.2 Member Enumeration Documentation	1155
15.199.2.1 None_type	1155
15.199.3 Constructor & Destructor Documentation	1155
15.199.3.1 Flags() [1/2]	1155
15.199.3.2 Flags() [2/2]	1156
15.199.4 Member Function Documentation	1156
15.199.4.1 clear()	1156
15.199.4.2 from_raw()	1157
15.200 L4::Types::Flags_ops_t< DT > Struct Template Reference	1157
15.200.1 Detailed Description	1158
15.201 L4::Types::Flags_t< DT, T > Struct Template Reference	1159
15.201.1 Detailed Description	1161
15.202 L4::Types::Int_for_size< SIZE, bool > Struct Template Reference	1161
15.202.1 Detailed Description	1161
15.203 L4::Types::Int_for_type< T > Struct Template Reference	1162
15.203.1 Detailed Description	1162
15.204 L4::Types::Same< A, B > Struct Template Reference	1163
15.204.1 Detailed Description	1164
15.205 L4::Types::True Struct Reference	1165
15.205.1 Detailed Description	1167
15.206 L4::Unknown_error Class Reference	1167
15.206.1 Detailed Description	1169
15.207 L4::Vcon Class Reference	1170
15.207.1 Detailed Description	1172
15.207.2 Member Function Documentation	1172
15.207.2.1 get_attr()	1172
15.207.2.2 read()	1173
15.207.2.3 read_with_flags()	1174

15.207.2.4 send()	1175
15.207.2.5 set_attr()	1175
15.207.2.6 write()	1176
15.208 L4::Vm Class Reference	1177
15.208.1 Detailed Description	1179
15.209 l4_buf_regs_t Struct Reference	1180
15.209.1 Detailed Description	1180
15.210 l4_exc_regs_t Struct Reference	1181
15.210.1 Detailed Description	1183
15.210.2 Field Documentation	1183
15.210.2.1 flags	1183
15.210.2.2 ss	1183
15.211 l4_fpage_t Union Reference	1184
15.211.1 Detailed Description	1184
15.212 l4_icu_info_t Struct Reference	1184
15.212.1 Detailed Description	1186
15.212.2 Field Documentation	1186
15.212.2.1 features	1186
15.213 l4_icu_msi_info_t Struct Reference	1186
15.213.1 Detailed Description	1187
15.214 l4_kernel_info_mem_desc_t Struct Reference	1187
15.214.1 Detailed Description	1187
15.215 l4_kernel_info_t Struct Reference	1188
15.215.1 Detailed Description	1190
15.216 l4_msg_regs_t Union Reference	1190
15.216.1 Detailed Description	1190
15.217 l4_msgtag_t Struct Reference	1191
15.217.1 Detailed Description	1192
15.217.2 Member Function Documentation	1192
15.217.2.1 flags()	1192
15.218 l4_sched_cpu_set_t Struct Reference	1193
15.218.1 Detailed Description	1193
15.218.2 Member Function Documentation	1193
15.218.2.1 granularity()	1194
15.218.2.2 offset()	1194
15.218.3 Field Documentation	1195
15.218.3.1 gran_offset	1195
15.219 l4_sched_param_t Struct Reference	1195
15.219.1 Detailed Description	1196
15.220 l4_snd_fpage_t Struct Reference	1197
15.220.1 Detailed Description	1197
15.221 l4_thread_regs_t Struct Reference	1198

15.221.1 Detailed Description	1198
15.222 l4_timeout_s Struct Reference	1199
15.222.1 Detailed Description	1199
15.223 l4_timeout_t Union Reference	1200
15.223.1 Detailed Description	1200
15.224 l4_vcon_attr_t Struct Reference	1201
15.224.1 Detailed Description	1201
15.225 l4_vcpu_ipc_regs_t Struct Reference	1202
15.225.1 Detailed Description	1202
15.226 l4_vcpu_regs_t Struct Reference	1203
15.226.1 Detailed Description	1204
15.226.2 Field Documentation	1204
15.226.2.1 ax	1205
15.226.2.2 bp	1205
15.226.2.3 bx	1205
15.226.2.4 cx	1205
15.226.2.5 di	1206
15.226.2.6 dx	1206
15.226.2.7 si	1206
15.227 l4_vcpu_state_t Struct Reference	1207
15.227.1 Detailed Description	1209
15.228 l4_vhw_descriptor Struct Reference	1209
15.228.1 Detailed Description	1210
15.229 l4_vhw_entry Struct Reference	1211
15.229.1 Detailed Description	1212
15.230 l4_vm_svm_vmcb_control_area Struct Reference	1212
15.230.1 Detailed Description	1212
15.231 l4_vm_svm_vmcb_state_save_area Struct Reference	1213
15.231.1 Detailed Description	1213
15.232 l4_vm_svm_vmcb_state_save_area_seg Struct Reference	1214
15.232.1 Detailed Description	1214
15.233 l4_vm_svm_vmcb_t Struct Reference	1215
15.233.1 Detailed Description	1216
15.234 l4_vm_tz_state Struct Reference	1216
15.234.1 Detailed Description	1216
15.235 L4Re::Cap_alloc Class Reference	1217
15.235.1 Detailed Description	1217
15.235.2 Member Function Documentation	1218
15.235.2.1 alloc() [1/2]	1218
15.235.2.2 alloc() [2/2]	1218
15.235.2.3 free()	1219
15.235.2.4 get_cap_alloc()	1220

15.236 L4Re::Console Class Reference	1221
15.236.1 Detailed Description	1222
15.237 L4Re::Dataspace Class Reference	1223
15.237.1 Detailed Description	1225
15.237.2 Member Function Documentation	1225
15.237.2.1 allocate()	1225
15.237.2.2 clear()	1226
15.237.2.3 copy_in()	1226
15.237.2.4 flags()	1228
15.237.2.5 info()	1229
15.237.2.6 map()	1229
15.237.2.7 map_region()	1230
15.237.2.8 size()	1231
15.238 L4Re::Dataspace::F Struct Reference	1231
15.238.1 Detailed Description	1232
15.238.2 Member Enumeration Documentation	1232
15.238.2.1 anonymous enum	1232
15.238.2.2 Flags	1232
15.239 L4Re::Dataspace::Stats Struct Reference	1233
15.239.1 Detailed Description	1233
15.240 L4Re::Debug_obj Class Reference	1233
15.240.1 Detailed Description	1235
15.240.2 Member Function Documentation	1236
15.240.2.1 debug()	1236
15.241 L4Re::Dma_space Class Reference	1236
15.241.1 Detailed Description	1237
15.241.2 Member Typedef Documentation	1237
15.241.2.1 Attributes	1238
15.241.3 Member Enumeration Documentation	1238
15.241.3.1 Attribute	1238
15.241.3.2 Direction	1238
15.241.3.3 Space_attrb	1239
15.241.4 Member Function Documentation	1239
15.241.4.1 associate()	1239
15.241.4.2 disassociate()	1240
15.241.4.3 map()	1240
15.241.4.4 unmap()	1241
15.242 L4Re::Env Class Reference	1241
15.242.1 Detailed Description	1243
15.242.2 Member Function Documentation	1244
15.242.2.1 env()	1244
15.242.2.2 factory() [1/2]	1245

15.242.2.3	factory() [2/2]	1245
15.242.2.4	first_free_cap() [1/2]	1245
15.242.2.5	first_free_cap() [2/2]	1245
15.242.2.6	first_free_utcb() [1/2]	1246
15.242.2.7	first_free_utcb() [2/2]	1246
15.242.2.8	get()	1246
15.242.2.9	get_cap() [1/2]	1247
15.242.2.10	get_cap() [2/2]	1248
15.242.2.11	initial_caps() [1/2]	1248
15.242.2.12	initial_caps() [2/2]	1248
15.242.2.13	log() [1/2]	1249
15.242.2.14	log() [2/2]	1249
15.242.2.15	main_thread() [1/2]	1249
15.242.2.16	main_thread() [2/2]	1250
15.242.2.17	mem_alloc() [1/2]	1250
15.242.2.18	mem_alloc() [2/2]	1250
15.242.2.19	parent() [1/2]	1251
15.242.2.20	parent() [2/2]	1251
15.242.2.21	rm() [1/2]	1251
15.242.2.22	rm() [2/2]	1252
15.242.2.23	scheduler() [1/2]	1252
15.242.2.24	scheduler() [2/2]	1252
15.242.2.25	task()	1253
15.242.2.26	utcb_area() [1/2]	1253
15.242.2.27	utcb_area() [2/2]	1253
15.243	L4Re::Event Class Reference	1254
15.243.1	Detailed Description	1257
15.243.2	Member Function Documentation	1257
15.243.2.1	get_buffer()	1257
15.244	L4Re::Event_buffer_t< PAYLOAD > Class Template Reference	1257
15.244.1	Detailed Description	1259
15.244.2	Constructor & Destructor Documentation	1259
15.244.2.1	Event_buffer_t()	1259
15.244.3	Member Function Documentation	1259
15.244.3.1	next()	1259
15.244.3.2	put()	1260
15.245	L4Re::Event_buffer_t< PAYLOAD >::Event Struct Reference	1260
15.245.1	Detailed Description	1261
15.246	L4Re::Inhibitor Class Reference	1261
15.246.1	Detailed Description	1264
15.246.2	Member Enumeration Documentation	1264
15.246.2.1	anonymous enum	1264

15.246.3 Member Function Documentation	1264
15.246.3.1 acquire()	1264
15.246.3.2 next_lock_info()	1265
15.246.3.3 release()	1266
15.247 L4Re::Log Class Reference	1266
15.247.1 Detailed Description	1269
15.247.2 Member Function Documentation	1269
15.247.2.1 print()	1269
15.247.2.2 printn()	1269
15.248 L4Re::Mem_alloc Class Reference	1269
15.248.1 Detailed Description	1272
15.248.2 Member Enumeration Documentation	1272
15.248.2.1 Mem_alloc_flags	1272
15.248.3 Member Function Documentation	1273
15.248.3.1 alloc()	1273
15.248.3.2 free()	1274
15.249 L4Re::Mmio_space Struct Reference	1274
15.249.1 Detailed Description	1277
15.249.2 Member Enumeration Documentation	1277
15.249.2.1 Access_width	1277
15.249.3 Member Function Documentation	1277
15.249.3.1 mmio_read()	1278
15.249.3.2 mmio_write()	1278
15.250 L4Re::Namespace Class Reference	1279
15.250.1 Detailed Description	1281
15.250.2 Member Enumeration Documentation	1281
15.250.2.1 Query_result_flags	1281
15.250.2.2 Register_flags	1281
15.250.3 Member Function Documentation	1282
15.250.3.1 query() [1/2]	1282
15.250.3.2 query() [2/2]	1283
15.250.3.3 register_obj()	1284
15.250.3.4 unlink()	1285
15.251 L4Re::Ned::Cmd_control Class Reference	1285
15.251.1 Detailed Description	1286
15.251.2 Member Function Documentation	1286
15.251.2.1 execute() [1/2]	1286
15.251.2.2 execute() [2/2]	1286
15.252 L4Re::Parent Class Reference	1287
15.252.1 Detailed Description	1289
15.252.2 Member Function Documentation	1290
15.252.2.1 signal()	1290

15.253 L4Re::Random Struct Reference	1290
15.253.1 Detailed Description	1293
15.253.2 Member Function Documentation	1293
15.253.2.1 get_random()	1293
15.254 L4Re::Rm Class Reference	1294
15.254.1 Detailed Description	1297
15.254.2 Member Typedef Documentation	1297
15.254.2.1 Area	1297
15.254.2.2 Region	1297
15.254.3 Member Enumeration Documentation	1297
15.254.3.1 Detach_flags	1297
15.254.3.2 Detach_result	1298
15.254.3.3 Region_flag_shifts	1298
15.254.4 Member Function Documentation	1298
15.254.4.1 attach() [1/2]	1298
15.254.4.2 attach() [2/2]	1300
15.254.4.3 detach() [1/3]	1301
15.254.4.4 detach() [2/3]	1301
15.254.4.5 detach() [3/3]	1302
15.254.4.6 find()	1302
15.254.4.7 free_area()	1303
15.254.4.8 get_areas()	1304
15.254.4.9 get_regions()	1304
15.254.4.10 reserve_area() [1/2]	1305
15.254.4.11 reserve_area() [2/2]	1306
15.255 L4Re::Rm::F Struct Reference	1307
15.255.1 Detailed Description	1307
15.255.2 Member Enumeration Documentation	1307
15.255.2.1 Attach_flags	1307
15.255.2.2 Region_flags	1308
15.256 L4Re::Rm::Range Struct Reference	1308
15.256.1 Detailed Description	1309
15.257 L4Re::Smart_cap_auto< Unmap_flags > Class Template Reference	1309
15.257.1 Detailed Description	1310
15.258 L4Re::Smart_count_cap< Unmap_flags > Class Template Reference	1310
15.258.1 Detailed Description	1311
15.259 L4Re::Util::Br_manager Class Reference	1311
15.259.1 Detailed Description	1314
15.259.2 Member Function Documentation	1314
15.259.2.1 alloc_buffer_demand()	1314
15.259.2.2 get_rcv_cap()	1315
15.259.2.3 realloc_rcv_cap()	1315

15.259.2.4 set_rcv_cap_flags()	1316
15.260 L4Re::Util::Br_manager_hooks Struct Reference	1316
15.260.1 Detailed Description	1318
15.261 L4Re::Util::Br_manager_timeout_hooks Struct Reference	1318
15.261.1 Detailed Description	1320
15.262 L4Re::Util::Cap_alloc_base Class Reference	1321
15.262.1 Detailed Description	1321
15.263 L4Re::Util::Counter< COUNTER > Struct Template Reference	1322
15.263.1 Detailed Description	1323
15.264 L4Re::Util::Counting_cap_alloc< COUNTERTYPE > Class Template Reference	1323
15.264.1 Detailed Description	1325
15.264.2 Constructor & Destructor Documentation	1325
15.264.2.1 Counting_cap_alloc()	1325
15.264.3 Member Function Documentation	1325
15.264.3.1 alloc() [1/2]	1326
15.264.3.2 alloc() [2/2]	1327
15.264.3.3 free()	1327
15.264.3.4 release()	1328
15.264.3.5 setup()	1330
15.264.3.6 take()	1331
15.265 L4Re::Util::Dataspace_svr Class Reference	1332
15.265.1 Detailed Description	1333
15.265.2 Member Function Documentation	1333
15.265.2.1 allocate()	1333
15.265.2.2 clear()	1334
15.265.2.3 copy()	1334
15.265.2.4 is_static()	1335
15.265.2.5 map()	1335
15.265.2.6 map_hook()	1335
15.265.2.7 page_shift()	1336
15.265.2.8 release()	1337
15.265.2.9 take()	1337
15.266 L4Re::Util::Event_buffer_consumer_t< PAYLOAD > Class Template Reference	1337
15.266.1 Detailed Description	1339
15.266.2 Member Function Documentation	1340
15.266.2.1 foreach_available_event()	1340
15.266.2.2 process()	1340
15.267 L4Re::Util::Event_buffer_t< PAYLOAD > Class Template Reference	1341
15.267.1 Detailed Description	1343
15.267.2 Member Function Documentation	1343
15.267.2.1 attach()	1344
15.267.2.2 buf()	1344

15.267.2.3 detach()	1344
15.268 L4Re::Util::Event_svr< SVR > Class Template Reference	1345
15.268.1 Detailed Description	1347
15.269 L4Re::Util::Event_t< PAYLOAD > Class Template Reference	1347
15.269.1 Detailed Description	1348
15.269.2 Member Enumeration Documentation	1348
15.269.2.1 Mode	1348
15.269.3 Member Function Documentation	1348
15.269.3.1 buffer()	1349
15.269.3.2 init()	1349
15.269.3.3 init_poll()	1350
15.269.3.4 irq()	1351
15.270 L4Re::Util::Item_alloc_base Class Reference	1351
15.270.1 Detailed Description	1352
15.271 L4Re::Util::Names::Name Class Reference	1352
15.271.1 Detailed Description	1353
15.272 L4Re::Util::Names::Name_space Class Reference	1353
15.272.1 Detailed Description	1354
15.272.2 Member Function Documentation	1355
15.272.2.1 alloc_dynamic_entry()	1355
15.272.2.2 copy_receive_cap()	1355
15.272.2.3 free_capability()	1355
15.272.2.4 free_dynamic_entry()	1356
15.272.2.5 free_epiface()	1356
15.272.2.6 get_epiface()	1356
15.273 L4Re::Util::Object_registry Class Reference	1357
15.273.1 Detailed Description	1359
15.273.2 Constructor & Destructor Documentation	1359
15.273.2.1 Object_registry() [1/2]	1359
15.273.2.2 Object_registry() [2/2]	1359
15.273.3 Member Function Documentation	1360
15.273.3.1 register_irq_obj()	1360
15.273.3.2 register_obj() [1/3]	1360
15.273.3.3 register_obj() [2/3]	1361
15.273.3.4 register_obj() [3/3]	1361
15.273.3.5 unregister_obj()	1362
15.274 L4Re::Util::Ref_cap< T > Struct Template Reference	1363
15.274.1 Detailed Description	1363
15.275 L4Re::Util::Ref_del_cap< T > Struct Template Reference	1364
15.275.1 Detailed Description	1364
15.276 L4Re::Util::Registry_server< LOOP_HOOKS > Class Template Reference	1365
15.276.1 Detailed Description	1368

15.276.2 Constructor & Destructor Documentation	1368
15.276.2.1 Registry_server() [1/2]	1368
15.276.2.2 Registry_server() [2/2]	1368
15.277 L4Re::Util::Smart_cap_auto< Unmap_flags > Class Template Reference	1369
15.277.1 Detailed Description	1370
15.278 L4Re::Util::Smart_count_cap< Unmap_flags > Class Template Reference	1370
15.278.1 Detailed Description	1371
15.279 L4Re::Util::Vcon_svr< SVR > Class Template Reference	1371
15.279.1 Detailed Description	1371
15.280 L4Re::Util::Video::Goos_svr Class Reference	1372
15.280.1 Detailed Description	1373
15.280.2 Member Function Documentation	1373
15.280.2.1 get_fb()	1373
15.280.2.2 init_infos()	1374
15.280.2.3 refresh()	1374
15.280.2.4 screen_info()	1374
15.280.2.5 view_info()	1375
15.281 L4Re::Vfs::Be_file Class Reference	1375
15.281.1 Detailed Description	1378
15.281.2 Member Function Documentation	1379
15.281.2.1 data_space()	1379
15.281.2.2 fstat64()	1379
15.281.2.3 unlock_all_locks()	1380
15.282 L4Re::Vfs::Be_file_system Class Reference	1380
15.282.1 Detailed Description	1381
15.282.2 Constructor & Destructor Documentation	1381
15.282.2.1 Be_file_system()	1382
15.282.2.2 ~Be_file_system()	1382
15.282.3 Member Function Documentation	1382
15.282.3.1 type()	1382
15.283 L4Re::Vfs::Directory Class Reference	1383
15.283.1 Detailed Description	1384
15.283.2 Member Function Documentation	1385
15.283.2.1 faccessat()	1385
15.283.2.2 link()	1385
15.283.2.3 mkdir()	1386
15.283.2.4 rename()	1386
15.283.2.5 rmdir()	1387
15.283.2.6 symlink()	1387
15.283.2.7 unlink()	1387
15.284 L4Re::Vfs::File Class Reference	1388
15.284.1 Detailed Description	1389

15.285 L4Re::Vfs::File_system Class Reference	1390
15.285.1 Detailed Description	1391
15.285.2 Member Function Documentation	1391
15.285.2.1 mount()	1391
15.285.2.2 type()	1392
15.286 L4Re::Vfs::Fs Class Reference	1392
15.286.1 Detailed Description	1394
15.286.2 Member Function Documentation	1395
15.286.2.1 alloc_fd()	1395
15.286.2.2 free_fd()	1395
15.286.2.3 get_file()	1396
15.286.2.4 mount()	1396
15.286.2.5 set_fd()	1396
15.287 L4Re::Vfs::Generic_file Class Reference	1397
15.287.1 Detailed Description	1399
15.287.2 Member Function Documentation	1399
15.287.2.1 fchmod()	1400
15.287.2.2 fstat64()	1400
15.287.2.3 get_status_flags()	1400
15.287.2.4 set_status_flags()	1400
15.287.2.5 unlock_all_locks()	1401
15.288 L4Re::Vfs::Mman Class Reference	1402
15.288.1 Detailed Description	1403
15.289 L4Re::Vfs::Ops Class Reference	1403
15.289.1 Detailed Description	1405
15.290 L4Re::Vfs::Regular_file Class Reference	1406
15.290.1 Detailed Description	1407
15.290.2 Member Function Documentation	1408
15.290.2.1 data_space()	1408
15.290.2.2 fdatsync()	1408
15.290.2.3 fsync()	1408
15.290.2.4 ftruncate64()	1408
15.290.2.5 get_lock()	1409
15.290.2.6 lseek64()	1409
15.290.2.7 readv()	1410
15.290.2.8 set_lock()	1410
15.290.2.9 writev()	1410
15.291 L4Re::Vfs::Special_file Class Reference	1411
15.291.1 Detailed Description	1413
15.291.2 Member Function Documentation	1413
15.291.2.1 ioctl()	1413
15.292 L4Re::Video::Color_component Class Reference	1414

15.292.1 Detailed Description	1415
15.292.2 Constructor & Destructor Documentation	1415
15.292.2.1 Color_component()	1415
15.292.3 Member Function Documentation	1415
15.292.3.1 dump()	1415
15.292.3.2 get()	1416
15.292.3.3 operator==()	1416
15.292.3.4 set()	1416
15.292.3.5 shift()	1417
15.292.3.6 size()	1417
15.293 L4Re::Video::Goos Class Reference	1418
15.293.1 Detailed Description	1420
15.293.2 Member Enumeration Documentation	1420
15.293.2.1 Flags	1420
15.293.3 Member Function Documentation	1421
15.293.3.1 create_buffer()	1421
15.293.3.2 create_view()	1421
15.293.3.3 delete_buffer()	1422
15.293.3.4 delete_view()	1422
15.293.3.5 get_static_buffer()	1422
15.293.3.6 info()	1423
15.293.3.7 view()	1423
15.294 L4Re::Video::Goos::Info Struct Reference	1424
15.294.1 Detailed Description	1425
15.295 L4Re::Video::Pixel_info Class Reference	1425
15.295.1 Detailed Description	1426
15.295.2 Constructor & Destructor Documentation	1426
15.295.2.1 Pixel_info() [1/2]	1427
15.295.2.2 Pixel_info() [2/2]	1427
15.295.3 Member Function Documentation	1427
15.295.3.1 a() [1/2]	1428
15.295.3.2 a() [2/2]	1428
15.295.3.3 b() [1/2]	1428
15.295.3.4 b() [2/2]	1428
15.295.3.5 bits_per_pixel()	1429
15.295.3.6 bytes_per_pixel() [1/2]	1429
15.295.3.7 bytes_per_pixel() [2/2]	1429
15.295.3.8 dump()	1430
15.295.3.9 g() [1/2]	1430
15.295.3.10 g() [2/2]	1431
15.295.3.11 has_alpha()	1431
15.295.3.12 operator==()	1431

15.295.3.13 r() [1/2]	1432
15.295.3.14 r() [2/2]	1432
15.296 L4Re::Video::View Class Reference	1432
15.296.1 Detailed Description	1434
15.296.2 Member Enumeration Documentation	1434
15.296.2.1 Flags	1434
15.296.2.2 V_flags	1434
15.296.3 Member Function Documentation	1435
15.296.3.1 info()	1435
15.296.3.2 refresh()	1435
15.296.3.3 set_info()	1436
15.296.3.4 set_viewport()	1436
15.296.3.5 stack()	1437
15.297 L4Re::Video::View::Info Struct Reference	1437
15.297.1 Detailed Description	1439
15.298 l4re_aux_t Struct Reference	1440
15.298.1 Detailed Description	1440
15.299 l4re_ds_stats_t Struct Reference	1441
15.299.1 Detailed Description	1441
15.300 l4re_elf_aux_mword_t Struct Reference	1441
15.300.1 Detailed Description	1442
15.301 l4re_elf_aux_t Struct Reference	1442
15.301.1 Detailed Description	1443
15.302 l4re_elf_aux_vma_t Struct Reference	1443
15.302.1 Detailed Description	1443
15.303 l4re_env_cap_entry_t Struct Reference	1444
15.303.1 Detailed Description	1444
15.303.2 Constructor & Destructor Documentation	1444
15.303.2.1 l4re_env_cap_entry_t()	1445
15.303.3 Field Documentation	1445
15.303.3.1 flags	1445
15.304 l4re_env_t Struct Reference	1446
15.304.1 Detailed Description	1447
15.305 l4re_event_t Struct Reference	1447
15.305.1 Detailed Description	1448
15.306 l4re_video_color_component_t Struct Reference	1448
15.306.1 Detailed Description	1449
15.307 l4re_video_goos_info_t Struct Reference	1449
15.307.1 Detailed Description	1450
15.308 l4re_video_pixel_info_t Struct Reference	1450
15.308.1 Detailed Description	1451
15.309 l4re_video_view_info_t Struct Reference	1451

15.309.1 Detailed Description	1453
15.310 l4re_video_view_t Struct Reference	1453
15.310.1 Detailed Description	1453
15.311 l4util_idt_desc_t Struct Reference	1454
15.311.1 Detailed Description	1454
15.312 l4util_idt_header_t Struct Reference	1455
15.312.1 Detailed Description	1455
15.313 l4util_l4mod_info Struct Reference	1456
15.313.1 Detailed Description	1456
15.313.2 Field Documentation	1456
15.313.2.1 vbe_ctrl_info	1457
15.314 l4util_l4mod_mod Struct Reference	1457
15.314.1 Detailed Description	1458
15.315 l4util_mb_addr_range_t Struct Reference	1458
15.315.1 Detailed Description	1459
15.316 l4util_mb_apm_t Struct Reference	1459
15.316.1 Detailed Description	1459
15.317 l4util_mb_drive_t Struct Reference	1460
15.317.1 Detailed Description	1460
15.317.2 Field Documentation	1461
15.317.2.1 drive_cylinders	1461
15.317.2.2 drive_mode	1461
15.317.2.3 drive_number	1461
15.318 l4util_mb_info_t Struct Reference	1462
15.318.1 Detailed Description	1463
15.319 l4util_mb_mod_t Struct Reference	1463
15.319.1 Detailed Description	1464
15.320 l4util_mb_vbe_ctrl_t Struct Reference	1464
15.320.1 Detailed Description	1465
15.321 l4util_mb_vbe_mode_t Struct Reference	1465
15.321.1 Detailed Description	1467
15.322 L4vbus::Device Class Reference	1468
15.322.1 Detailed Description	1470
15.322.2 Member Function Documentation	1470
15.322.2.1 bus_cap()	1470
15.322.2.2 dev_handle()	1471
15.322.2.3 device()	1472
15.322.2.4 device_by_hid()	1472
15.322.2.5 get_resource()	1473
15.322.2.6 is_compatible()	1474
15.322.2.7 next_device()	1475
15.322.2.8 operator!=(())	1475

15.322.2.9 operator==()	1475
15.322.3 Field Documentation	1476
15.322.3.1 _bus	1476
15.323 L4vbus::Gpio_module Class Reference	1476
15.323.1 Detailed Description	1479
15.323.2 Member Function Documentation	1479
15.323.2.1 config_pad()	1479
15.323.2.2 get()	1480
15.323.2.3 pin()	1480
15.323.2.4 set()	1481
15.323.2.5 setup()	1482
15.324 L4vbus::Gpio_module::Pin_slice Struct Reference	1482
15.324.1 Detailed Description	1483
15.325 L4vbus::Gpio_pin Class Reference	1483
15.325.1 Detailed Description	1486
15.325.2 Member Function Documentation	1486
15.325.2.1 config_get()	1486
15.325.2.2 config_pad()	1487
15.325.2.3 config_pull()	1487
15.325.2.4 get()	1488
15.325.2.5 pin()	1489
15.325.2.6 set()	1489
15.325.2.7 setup()	1489
15.325.2.8 to_irq()	1490
15.326 L4vbus::lcu Class Reference	1491
15.326.1 Detailed Description	1493
15.326.2 Member Enumeration Documentation	1493
15.326.2.1 Src_types	1493
15.327 L4vbus::Pci_dev Class Reference	1493
15.327.1 Detailed Description	1496
15.327.2 Member Function Documentation	1496
15.327.2.1 cfg_read()	1496
15.327.2.2 cfg_write()	1497
15.327.2.3 irq_enable()	1497
15.328 L4vbus::Pci_host_bridge Class Reference	1498
15.328.1 Detailed Description	1501
15.328.2 Member Function Documentation	1501
15.328.2.1 cfg_read()	1501
15.328.2.2 cfg_write()	1502
15.328.2.3 irq_enable()	1503
15.329 L4vbus::Pm< DEC > Class Template Reference	1504
15.329.1 Detailed Description	1505

15.330 L4vbus::Vbus Class Reference	1505
15.330.1 Detailed Description	1508
15.330.2 Member Function Documentation	1508
15.330.2.1 assign_dma_domain() [1/2]	1508
15.330.2.2 assign_dma_domain() [2/2]	1509
15.330.2.3 release_ioport()	1510
15.330.2.4 release_resource()	1511
15.330.2.5 request_ioport()	1512
15.330.2.6 request_resource()	1513
15.330.2.7 root()	1514
15.331 l4vbus_device_t Struct Reference	1514
15.331.1 Detailed Description	1515
15.332 l4vbus_resource_t Struct Reference	1515
15.332.1 Detailed Description	1516
15.333 L4vcpu::State Class Reference	1516
15.333.1 Detailed Description	1517
15.333.2 Constructor & Destructor Documentation	1517
15.333.2.1 State()	1517
15.333.3 Member Function Documentation	1517
15.333.3.1 add()	1517
15.333.3.2 clear()	1518
15.333.3.3 set()	1518
15.334 L4vcpu::Vcpu Class Reference	1518
15.334.1 Detailed Description	1522
15.334.2 Member Function Documentation	1522
15.334.2.1 cast() [1/2]	1522
15.334.2.2 cast() [2/2]	1522
15.334.2.3 entry_ip()	1523
15.334.2.4 entry_sp()	1523
15.334.2.5 ext_alloc()	1523
15.334.2.6 i() [1/2]	1524
15.334.2.7 i() [2/2]	1524
15.334.2.8 irq_disable_save()	1525
15.334.2.9 irq_enable()	1525
15.334.2.10 irq_restore()	1526
15.334.2.11 is_irq_entry()	1526
15.334.2.12 is_page_fault_entry()	1527
15.334.2.13 r() [1/2]	1527
15.334.2.14 r() [2/2]	1528
15.334.2.15 saved_state() [1/2]	1528
15.334.2.16 saved_state() [2/2]	1528
15.334.2.17 state() [1/2]	1529

15.334.2.18 state() [2/2]	1529
15.334.2.19 task()	1529
15.334.2.20 wait_for_event()	1529
15.335 L4virtio::Device Class Reference	1530
15.335.1 Detailed Description	1533
15.335.2 Member Function Documentation	1533
15.335.2.1 config_queue()	1533
15.335.2.2 device_config()	1534
15.335.2.3 device_notification_irq()	1535
15.335.2.4 register_ds()	1535
15.335.2.5 register_iface()	1536
15.335.2.6 set_status()	1537
15.336 L4virtio::Driver::Block_device Class Reference	1537
15.336.1 Detailed Description	1540
15.336.2 Member Function Documentation	1540
15.336.2.1 add_block()	1540
15.336.2.2 process_request()	1541
15.336.2.3 process_used_queue()	1542
15.336.2.4 send_request()	1542
15.336.2.5 setup_device()	1543
15.336.2.6 start_request()	1545
15.337 L4virtio::Driver::Block_device::Handle Class Reference	1545
15.337.1 Detailed Description	1546
15.338 L4virtio::Driver::Device Class Reference	1546
15.338.1 Detailed Description	1549
15.338.2 Member Function Documentation	1549
15.338.2.1 bind_notification_irq()	1549
15.338.2.2 config_queue()	1550
15.338.2.3 driver_acknowledge()	1551
15.338.2.4 driver_connect()	1552
15.338.2.5 max_queue_size()	1553
15.338.2.6 register_ds()	1554
15.338.2.7 send()	1555
15.338.2.8 send_and_wait()	1556
15.338.2.9 wait()	1557
15.338.2.10 wait_for_next_used()	1558
15.339 L4virtio::Driver::Virtqueue Class Reference	1559
15.339.1 Detailed Description	1562
15.339.2 Member Function Documentation	1562
15.339.2.1 alloc_descriptor()	1562
15.339.2.2 desc()	1563
15.339.2.3 enqueue_descriptor()	1564

15.339.2.4	find_next_used()	1565
15.339.2.5	free_descriptor()	1565
15.339.2.6	init_queue() [1/2]	1566
15.339.2.7	init_queue() [2/2]	1566
15.339.2.8	initialize_rings()	1567
15.340	L4virtio::Ptr< T > Class Template Reference	1568
15.340.1	Detailed Description	1570
15.340.2	Member Enumeration Documentation	1570
15.340.2.1	Invalid_type	1570
15.340.3	Member Function Documentation	1570
15.340.3.1	get()	1570
15.340.3.2	is_valid()	1571
15.341	L4virtio::Svr::Bad_descriptor Struct Reference	1572
15.341.1	Detailed Description	1573
15.341.2	Member Enumeration Documentation	1573
15.341.2.1	Error	1573
15.341.3	Constructor & Destructor Documentation	1573
15.341.3.1	Bad_descriptor()	1573
15.341.4	Member Function Documentation	1574
15.341.4.1	message()	1574
15.342	L4virtio::Svr::Block_dev_base< Ds_data > Class Template Reference	1574
15.342.1	Detailed Description	1577
15.342.2	Constructor & Destructor Documentation	1578
15.342.2.1	Block_dev_base()	1578
15.342.3	Member Function Documentation	1578
15.342.3.1	finalize_request()	1578
15.342.3.2	get_writeback()	1579
15.342.3.3	process_request()	1579
15.342.3.4	set_blk_size()	1580
15.342.3.5	set_config_wce()	1580
15.342.3.6	set_discard()	1580
15.342.3.7	set_size_max()	1582
15.342.3.8	set_topology()	1582
15.342.3.9	set_write_zeroes()	1582
15.343	L4virtio::Svr::Block_request< Ds_data > Class Template Reference	1583
15.343.1	Detailed Description	1584
15.343.2	Member Function Documentation	1584
15.343.2.1	data_size()	1584
15.343.2.2	next_block()	1585
15.344	L4virtio::Svr::Data_buffer Struct Reference	1586
15.344.1	Detailed Description	1587
15.344.2	Constructor & Destructor Documentation	1587

15.344.2.1 Data_buffer()	1587
15.344.3 Member Function Documentation	1587
15.344.3.1 copy_to()	1587
15.344.3.2 done()	1588
15.344.3.3 set()	1588
15.344.3.4 skip()	1589
15.345 L4virtio::Svr::Dev_config Class Reference	1590
15.345.1 Detailed Description	1591
15.345.2 Constructor & Destructor Documentation	1591
15.345.2.1 Dev_config() [1/2]	1591
15.345.2.2 Dev_config() [2/2]	1592
15.345.3 Member Function Documentation	1592
15.345.3.1 change_queue_config()	1593
15.345.3.2 ds()	1593
15.345.3.3 get_cmd()	1594
15.345.3.4 guest_features()	1594
15.345.3.5 hdr()	1595
15.345.3.6 negotiated_features()	1595
15.345.3.7 qconfig()	1596
15.345.3.8 reset_cmd()	1597
15.345.3.9 reset_queue()	1597
15.345.3.10 set_device_needs_reset()	1598
15.345.3.11 set_status()	1599
15.345.3.12 status()	1600
15.346 L4virtio::Svr::Dev_features Struct Reference	1601
15.346.1 Detailed Description	1602
15.347 L4virtio::Svr::Dev_status Struct Reference	1602
15.347.1 Detailed Description	1604
15.347.2 Member Function Documentation	1604
15.347.2.1 running()	1604
15.348 L4virtio::Svr::Device_t< DATA > Class Template Reference	1605
15.348.1 Detailed Description	1607
15.348.2 Member Function Documentation	1607
15.348.2.1 device_error()	1608
15.348.2.2 device_notify_irq()	1608
15.348.2.3 handle_mem_cmd_write()	1609
15.348.2.4 init_mem_info()	1609
15.348.2.5 register_driver_irq()	1610
15.348.2.6 reset_queue_config()	1610
15.348.2.7 setup_queue()	1611
15.349 L4virtio::Svr::Driver_mem_list_t< DATA > Class Template Reference	1612
15.349.1 Detailed Description	1615

15.349.2 Member Function Documentation	1615
15.349.2.1 add()	1615
15.349.2.2 find()	1616
15.349.2.3 full()	1617
15.349.2.4 init()	1617
15.349.2.5 load_desc() [1/3]	1618
15.349.2.6 load_desc() [2/3]	1619
15.349.2.7 load_desc() [3/3]	1620
15.349.2.8 remove()	1621
15.350 L4virtio::Svr::Driver_mem_region_t< DATA > Class Template Reference	1622
15.350.1 Detailed Description	1623
15.350.2 Constructor & Destructor Documentation	1624
15.350.2.1 Driver_mem_region_t()	1624
15.350.3 Member Function Documentation	1624
15.350.3.1 contains()	1625
15.350.3.2 drv_base()	1625
15.350.3.3 ds()	1626
15.350.3.4 ds_offset()	1626
15.350.3.5 empty()	1626
15.350.3.6 flags()	1626
15.350.3.7 is_writable()	1627
15.350.3.8 local()	1627
15.350.3.9 local_base()	1628
15.350.3.10 size()	1628
15.351 L4virtio::Svr::Request_processor Class Reference	1629
15.351.1 Detailed Description	1629
15.351.2 Member Function Documentation	1630
15.351.2.1 current_flags()	1630
15.351.2.2 has_more()	1630
15.351.2.3 next()	1631
15.351.2.4 start() [1/2]	1633
15.351.2.5 start() [2/2]	1635
15.352 L4virtio::Svr::Virtqueue Class Reference	1636
15.352.1 Detailed Description	1638
15.352.2 Member Function Documentation	1638
15.352.2.1 consumed()	1638
15.352.2.2 desc()	1639
15.352.2.3 desc_avail()	1639
15.352.2.4 disable_notify()	1640
15.352.2.5 enable_notify()	1641
15.352.2.6 next_avail()	1641
15.353 L4virtio::Svr::Virtqueue::Head_desc Class Reference	1642

15.353.1 Detailed Description	1642
15.353.2 Member Function Documentation	1642
15.353.2.1 desc()	1643
15.353.2.2 operator Null_ptr_check const *()	1643
15.353.2.3 valid()	1643
15.354 L4virtio::Virtqueue Class Reference	1644
15.354.1 Detailed Description	1647
15.354.2 Member Function Documentation	1647
15.354.2.1 avail_align()	1647
15.354.2.2 avail_size()	1647
15.354.2.3 desc_align()	1648
15.354.2.4 desc_size()	1649
15.354.2.5 disable()	1650
15.354.2.6 dump()	1651
15.354.2.7 get_avail_idx()	1652
15.354.2.8 get_tail_avail_idx()	1652
15.354.2.9 no_notify_guest()	1652
15.354.2.10 no_notify_host() [1/2]	1653
15.354.2.11 no_notify_host() [2/2]	1654
15.354.2.12 num()	1654
15.354.2.13 ready()	1655
15.354.2.14 setup()	1656
15.354.2.15 setup_simple()	1657
15.354.2.16 total_size() [1/2]	1658
15.354.2.17 total_size() [2/2]	1659
15.354.2.18 used_align()	1660
15.354.2.19 used_size()	1660
15.355 L4virtio::Virtqueue::Avail Class Reference	1661
15.355.1 Detailed Description	1662
15.356 L4virtio::Virtqueue::Avail::Flags Struct Reference	1662
15.356.1 Detailed Description	1663
15.356.2 Member Typedef Documentation	1663
15.356.2.1 no_irq_bfm_t	1663
15.357 L4virtio::Virtqueue::Desc Class Reference	1664
15.357.1 Detailed Description	1665
15.358 L4virtio::Virtqueue::Desc::Flags Struct Reference	1665
15.358.1 Detailed Description	1666
15.358.2 Member Typedef Documentation	1666
15.358.2.1 indirect_bfm_t	1666
15.358.2.2 next_bfm_t	1667
15.358.2.3 write_bfm_t	1667
15.359 L4virtio::Virtqueue::Used Class Reference	1667

15.359.1 Detailed Description	1668
15.360 L4virtio::Virtqueue::Used::Flags Struct Reference	1668
15.360.1 Detailed Description	1669
15.360.2 Member Typedef Documentation	1669
15.360.2.1 no_notify_bfm_t	1669
15.361 L4virtio::Virtqueue::Used_elem Struct Reference	1670
15.361.1 Detailed Description	1670
15.361.2 Constructor & Destructor Documentation	1670
15.361.2.1 Used_elem()	1670
15.362 l4virtio_block_config_t Struct Reference	1671
15.362.1 Detailed Description	1672
15.363 l4virtio_block_discard_t Struct Reference	1672
15.363.1 Detailed Description	1672
15.364 l4virtio_block_header_t Struct Reference	1673
15.364.1 Detailed Description	1673
15.365 l4virtio_config_hdr_t Struct Reference	1674
15.365.1 Detailed Description	1675
15.365.2 Field Documentation	1675
15.365.2.1 status	1675
15.366 l4virtio_config_queue_t Struct Reference	1675
15.366.1 Detailed Description	1676
15.367 l4virtio_input_absinfo_t Struct Reference	1677
15.367.1 Detailed Description	1677
15.368 l4virtio_input_config_t Struct Reference	1677
15.368.1 Detailed Description	1678
15.369 l4virtio_input_devids_t Struct Reference	1678
15.369.1 Detailed Description	1679
15.370 l4virtio_input_event_t Struct Reference	1679
15.370.1 Detailed Description	1680
16 File Documentation	1681
16.1 amd64/l4/util/apic.h File Reference	1681
16.1.1 Detailed Description	1681
16.2 apic.h	1682
16.3 x86/l4/util/apic.h File Reference	1686
16.3.1 Detailed Description	1687
16.4 apic.h	1687
16.5 amd64/l4/util/idt.h File Reference	1692
16.5.1 Detailed Description	1693
16.6 idt.h	1693
16.7 x86/l4/util/idt.h File Reference	1694
16.7.1 Detailed Description	1694

16.8 idt.h	1695
16.9 amd64/l4/util/perform.h File Reference	1695
16.9.1 Detailed Description	1696
16.10 perform.h	1696
16.11 x86/l4/util/perform.h File Reference	1701
16.11.1 Detailed Description	1702
16.12 perform.h	1702
16.13 amd64/l4/util/rdtsc.h File Reference	1707
16.13.1 Detailed Description	1709
16.13.2 Function Documentation	1709
16.13.2.1 l4_busy_wait_ns()	1709
16.13.2.2 l4_busy_wait_us()	1710
16.13.2.3 l4_calibrate_tsc()	1711
16.13.2.4 l4_get_hz()	1711
16.13.2.5 l4_ns_to_tsc()	1711
16.13.2.6 l4_rdpmc()	1712
16.13.2.7 l4_rdpmc_32()	1713
16.13.2.8 l4_rdtsc()	1713
16.13.2.9 l4_rdtsc_32()	1714
16.13.2.10 l4_tsc_init()	1714
16.13.2.11 l4_tsc_to_ns()	1715
16.13.2.12 l4_tsc_to_s_and_ns()	1715
16.13.2.13 l4_tsc_to_us()	1716
16.14 rdtsc.h	1716
16.15 x86/l4/util/rdtsc.h File Reference	1719
16.15.1 Detailed Description	1720
16.15.2 Function Documentation	1720
16.15.2.1 l4_busy_wait_ns()	1720
16.15.2.2 l4_busy_wait_us()	1721
16.15.2.3 l4_calibrate_tsc()	1722
16.15.2.4 l4_get_hz()	1722
16.15.2.5 l4_ns_to_tsc()	1723
16.15.2.6 l4_rdpmc()	1723
16.15.2.7 l4_rdpmc_32()	1723
16.15.2.8 l4_rdtsc()	1724
16.15.2.9 l4_rdtsc_32()	1724
16.15.2.10 l4_tsc_init()	1724
16.15.2.11 l4_tsc_to_ns()	1725
16.15.2.12 l4_tsc_to_s_and_ns()	1725
16.15.2.13 l4_tsc_to_us()	1726
16.16 rdtsc.h	1726
16.17 amd64/l4/util/spin.h File Reference	1729

16.17.1 Detailed Description	1729
16.18 spin.h	1730
16.19 x86/l4/util/spin.h File Reference	1730
16.19.1 Detailed Description	1731
16.20 spin.h	1731
16.21 amd64/l4/util/util.h File Reference	1731
16.21.1 Detailed Description	1732
16.21.2 Function Documentation	1732
16.21.2.1 l4_sleep()	1732
16.21.2.2 l4util_micros2l4to()	1733
16.22 util.h	1733
16.23 x86/l4/util/util.h File Reference	1734
16.23.1 Detailed Description	1735
16.23.2 Function Documentation	1735
16.23.2.1 l4_sleep()	1735
16.23.2.2 l4util_micros2l4to()	1736
16.24 util.h	1736
16.25 amd64/l4f/l4/sys/segment.h File Reference	1737
16.25.1 Detailed Description	1738
16.25.2 Function Documentation	1738
16.25.2.1 fiasco_amd64_set_fs()	1738
16.25.2.2 fiasco_amd64_set_segment_base()	1739
16.26 segment.h	1739
16.27 amd64/l4/sys/segment.h File Reference	1740
16.27.1 Detailed Description	1741
16.27.2 Enumeration Type Documentation	1741
16.27.2.1 L4_sys_segment	1741
16.27.2.2 L4_task_ldt_x86_consts	1741
16.27.3 Function Documentation	1742
16.27.3.1 fiasco_amd64_segment_info()	1742
16.27.3.2 fiasco_amd64_set_fs()	1742
16.27.3.3 fiasco_amd64_set_segment_base()	1743
16.28 segment.h	1743
16.29 x86/l4f/l4/sys/segment.h File Reference	1745
16.29.1 Detailed Description	1746
16.30 segment.h	1746
16.31 x86/l4/sys/segment.h File Reference	1746
16.31.1 Detailed Description	1747
16.31.2 Enumeration Type Documentation	1748
16.31.2.1 L4_task_ldt_x86_consts	1748
16.32 segment.h	1748
16.33 amd64/l4f/l4/util/port_io.h File Reference	1749

16.33.1 Detailed Description	1749
16.34 port_io.h	1750
16.35 amd64/l4/util/port_io.h File Reference	1750
16.35.1 Detailed Description	1750
16.36 port_io.h	1750
16.37 x86/l4f/l4/util/port_io.h File Reference	1751
16.37.1 Detailed Description	1752
16.37.2 Function Documentation	1752
16.37.2.1 l4util_ioport_map()	1752
16.38 port_io.h	1753
16.39 x86/l4/util/port_io.h File Reference	1754
16.39.1 Detailed Description	1756
16.39.2 Function Documentation	1756
16.39.2.1 l4util_in16()	1756
16.39.2.2 l4util_in32()	1756
16.39.2.3 l4util_in8()	1757
16.39.2.4 l4util_ins16()	1757
16.39.2.5 l4util_ins32()	1758
16.39.2.6 l4util_ins8()	1758
16.39.2.7 l4util_out16()	1759
16.39.2.8 l4util_out32()	1759
16.39.2.9 l4util_out8()	1759
16.39.2.10 l4util_outs16()	1760
16.39.2.11 l4util_outs32()	1760
16.39.2.12 l4util_outs8()	1761
16.40 port_io.h	1761
16.41 arm/l4/sys/linkage.h File Reference	1763
16.41.1 Detailed Description	1763
16.42 linkage.h	1763
16.43 amd64/l4/sys/linkage.h File Reference	1764
16.43.1 Detailed Description	1764
16.44 linkage.h	1764
16.45 x86/l4/sys/linkage.h File Reference	1765
16.45.1 Detailed Description	1765
16.46 linkage.h	1765
16.47 arm/l4/sys/mem_op.h File Reference	1766
16.47.1 Detailed Description	1767
16.48 mem_op.h	1767
16.49 arm/l4/sys/vm.h File Reference	1768
16.49.1 Detailed Description	1768
16.50 vm.h	1769
16.51 arm/l4/util/bitops_arch.h File Reference	1770

16.51.1 Detailed Description	1770
16.52 bitops_arch.h	1770
16.53 amd64/l4/util/bitops_arch.h File Reference	1770
16.53.1 Detailed Description	1770
16.54 bitops_arch.h	1771
16.55 x86/l4/util/bitops_arch.h File Reference	1774
16.55.1 Detailed Description	1774
16.56 bitops_arch.h	1774
16.57 arm/l4/util/cpu.h File Reference	1777
16.57.1 Detailed Description	1777
16.58 cpu.h	1778
16.59 amd64/l4/util/cpu.h File Reference	1778
16.59.1 Detailed Description	1779
16.59.2 Function Documentation	1779
16.59.2.1 l4util_cpu_capabilities()	1779
16.59.2.2 l4util_cpu_capabilities_nocheck()	1780
16.59.2.3 l4util_cpu_has_cpuid()	1780
16.60 cpu.h	1781
16.61 x86/l4/util/cpu.h File Reference	1782
16.61.1 Detailed Description	1783
16.61.2 Function Documentation	1783
16.61.2.1 l4util_cpu_capabilities()	1783
16.61.2.2 l4util_cpu_capabilities_nocheck()	1784
16.61.2.3 l4util_cpu_has_cpuid()	1784
16.62 cpu.h	1785
16.63 arm/l4/util/l4_macros.h File Reference	1786
16.63.1 Detailed Description	1786
16.64 l4_macros.h	1786
16.65 amd64/l4/util/l4_macros.h File Reference	1786
16.65.1 Detailed Description	1787
16.66 l4_macros.h	1787
16.67 l4/util/l4_macros.h File Reference	1787
16.67.1 Detailed Description	1787
16.68 l4_macros.h	1788
16.69 x86/l4/util/l4_macros.h File Reference	1788
16.69.1 Detailed Description	1788
16.70 l4_macros.h	1788
16.71 arm/l4/util/mbi_argv.h File Reference	1789
16.71.1 Detailed Description	1789
16.72 mbi_argv.h	1789
16.73 amd64/l4/util/mbi_argv.h File Reference	1790
16.73.1 Detailed Description	1790

16.74 mbi_argv.h	1790
16.75 x86/l4/util/mbi_argv.h File Reference	1791
16.75.1 Detailed Description	1792
16.76 mbi_argv.h	1792
16.77 arm/l4f/l4/sys/syscall_defs.h File Reference	1792
16.77.1 Detailed Description	1793
16.78 syscall_defs.h	1793
16.79 contrib/libio-io/l4/io/io.h File Reference	1793
16.79.1 Function Documentation	1795
16.79.1.1 l4io_get_root_device()	1795
16.79.1.2 l4io_iterate_devices()	1795
16.79.1.3 l4io_request_all_ioports()	1796
16.79.1.4 l4io_request_icu()	1796
16.80 io.h	1796
16.81 l4/cxx/alloc.h File Reference	1797
16.81.1 Detailed Description	1798
16.82 alloc.h	1798
16.83 l4/cxx/avl_map File Reference	1798
16.83.1 Detailed Description	1800
16.84 avl_map	1800
16.85 l4/cxx/avl_set File Reference	1801
16.85.1 Detailed Description	1802
16.86 avl_set	1803
16.87 l4/cxx/avl_tree File Reference	1806
16.87.1 Detailed Description	1807
16.88 avl_tree	1808
16.89 l4/cxx/basic_ostream File Reference	1812
16.89.1 Detailed Description	1812
16.90 basic_ostream	1813
16.91 l4/cxx/basic_vector.h File Reference	1816
16.91.1 Detailed Description	1816
16.92 basic_vector.h	1817
16.93 l4/cxx/bits/bst.h File Reference	1817
16.93.1 Detailed Description	1818
16.94 bst.h	1819
16.95 l4/cxx/bits/bst_base.h File Reference	1821
16.95.1 Detailed Description	1823
16.96 bst_base.h	1823
16.97 l4/cxx/bits/bst_iter.h File Reference	1824
16.97.1 Detailed Description	1825
16.98 bst_iter.h	1826
16.99 l4/cxx/exceptions File Reference	1827

16.99.1 Detailed Description	1829
16.100 exceptions	1829
16.101 l4/cxx/iostream File Reference	1831
16.101.1 Detailed Description	1832
16.102 iostream	1833
16.103 l4/cxx/ipc_helper File Reference	1833
16.103.1 Detailed Description	1834
16.104 ipc_helper	1834
16.105 l4/cxx/ipc_stream File Reference	1835
16.105.1 Detailed Description	1837
16.105.2 Function Documentation	1837
16.105.2.1 operator<<() [1/4]	1837
16.105.2.2 operator<<() [2/4]	1838
16.105.2.3 operator<<() [3/4]	1839
16.105.2.4 operator<<() [4/4]	1839
16.105.2.5 operator>>() [1/6]	1840
16.105.2.6 operator>>() [2/6]	1841
16.105.2.7 operator>>() [3/6]	1842
16.105.2.8 operator>>() [4/6]	1842
16.105.2.9 operator>>() [5/6]	1843
16.105.2.10 operator>>() [6/6]	1844
16.106 ipc_stream	1845
16.107 l4/cxx/l4iostream File Reference	1854
16.107.1 Detailed Description	1855
16.108 l4iostream	1855
16.109 l4/cxx/l4types.h File Reference	1856
16.109.1 Detailed Description	1857
16.110 l4types.h	1857
16.111 l4/cxx/main_thread File Reference	1857
16.111.1 Detailed Description	1858
16.112 main_thread	1858
16.113 l4/cxx/pair File Reference	1859
16.113.1 Detailed Description	1860
16.114 pair	1860
16.115 l4/cxx/std_exc_io File Reference	1860
16.115.1 Detailed Description	1861
16.116 std_exc_io	1861
16.117 l4/cxx/string.h File Reference	1862
16.117.1 Detailed Description	1863
16.118 string.h	1864
16.119 l4/irq/irq.h File Reference	1864
16.119.1 Detailed Description	1866

16.120 irq.h	1866
16.121 arm/I4/util/irq.h File Reference	1867
16.121.1 Detailed Description	1867
16.122 irq.h	1868
16.123 amd64/I4/util/irq.h File Reference	1868
16.123.1 Detailed Description	1869
16.123.2 Function Documentation	1869
16.123.2.1 I4util_irq_acknowledge()	1869
16.124 irq.h	1870
16.125 x86/I4/util/irq.h File Reference	1871
16.125.1 Detailed Description	1871
16.125.2 Function Documentation	1872
16.125.2.1 I4util_irq_acknowledge()	1872
16.126 irq.h	1872
16.127 I4/sys/irq.h File Reference	1873
16.127.1 Detailed Description	1875
16.128 irq.h	1875
16.129 I4/libedid/edid.h File Reference	1877
16.130 edid.h	1878
16.131 I4/re/c/dataspace.h File Reference	1879
16.131.1 Detailed Description	1880
16.132 dataspace.h	1880
16.133 I4/re/c/debug.h File Reference	1881
16.133.1 Detailed Description	1882
16.134 debug.h	1882
16.135 I4/re/c/dma_space.h File Reference	1883
16.135.1 Detailed Description	1884
16.135.2 Enumeration Type Documentation	1884
16.135.2.1 I4re_dma_space_direction	1884
16.135.2.2 I4re_dma_space_space_attris	1884
16.136 dma_space.h	1885
16.137 I4/re/c/event.h File Reference	1886
16.137.1 Detailed Description	1887
16.138 event.h	1887
16.139 I4/re/event.h File Reference	1888
16.139.1 Detailed Description	1889
16.140 event.h	1889
16.141 I4/re/c/log.h File Reference	1890
16.141.1 Detailed Description	1890
16.142 log.h	1891
16.143 I4/re/c/mem_alloc.h File Reference	1891
16.143.1 Detailed Description	1892

16.144	mem_alloc.h	1893
16.145	I4/re/c/namespace.h File Reference	1894
16.145.1	Detailed Description	1895
16.146	namespace.h	1895
16.147	I4/re/c/rm.h File Reference	1896
16.147.1	Detailed Description	1897
16.148	rm.h	1897
16.149	I4/re/c/util/cap_alloc.h File Reference	1900
16.149.1	Detailed Description	1900
16.150	cap_alloc.h	1901
16.151	I4/re/c/util/kumem_alloc.h File Reference	1901
16.151.1	Detailed Description	1902
16.151.2	Function Documentation	1902
16.151.2.1	I4re_util_kumem_alloc()	1902
16.152	kumem_alloc.h	1903
16.153	I4/re/c/util/video/goos_fb.h File Reference	1903
16.153.1	Detailed Description	1904
16.154	goos_fb.h	1904
16.155	I4/re/c/video/colors.h File Reference	1905
16.155.1	Detailed Description	1906
16.156	colors.h	1907
16.157	I4/re/c/video/goos.h File Reference	1907
16.157.1	Detailed Description	1909
16.158	goos.h	1909
16.159	I4/re/c/video/view.h File Reference	1910
16.159.1	Detailed Description	1912
16.160	view.h	1913
16.161	I4/re/cap_alloc File Reference	1914
16.161.1	Detailed Description	1915
16.162	cap_alloc	1915
16.163	I4/re/util/cap_alloc File Reference	1917
16.163.1	Detailed Description	1919
16.164	cap_alloc	1919
16.165	I4/re/dataspace File Reference	1920
16.165.1	Detailed Description	1921
16.166	dataspace	1922
16.167	I4/re/dataspace-sys.h File Reference	1923
16.167.1	Detailed Description	1924
16.168	dataspace-sys.h	1924
16.169	I4/re/debug File Reference	1924
16.169.1	Detailed Description	1925
16.170	debug	1926

16.171 l4/re/dma_space File Reference	1926
16.172 dma_space	1928
16.173 l4/re/elf_aux.h File Reference	1929
16.173.1 Detailed Description	1930
16.173.2 Macro Definition Documentation	1930
16.173.2.1 L4RE_ELF_AUX_ELEM	1930
16.173.2.2 L4RE_ELF_AUX_ELEM_T	1930
16.173.3 Enumeration Type Documentation	1931
16.173.3.1 anonymous enum	1931
16.174 elf_aux.h	1931
16.175 l4/re/env File Reference	1932
16.175.1 Detailed Description	1933
16.176 env	1934
16.177 l4/re/env.h File Reference	1935
16.177.1 Detailed Description	1936
16.177.2 Typedef Documentation	1936
16.177.2.1 l4re_env_t	1937
16.178 env.h	1937
16.179 l4/re/error_helper File Reference	1938
16.179.1 Detailed Description	1940
16.180 error_helper	1940
16.181 l4/re/util/event File Reference	1941
16.182 event	1942
16.183 l4/re/impl/dataspace_impl.h File Reference	1944
16.183.1 Detailed Description	1945
16.184 dataspace_impl.h	1945
16.185 l4/re/impl/mem_alloc_impl.h File Reference	1946
16.185.1 Detailed Description	1947
16.186 mem_alloc_impl.h	1947
16.187 l4/re/impl/namespace_impl.h File Reference	1948
16.187.1 Detailed Description	1949
16.188 namespace_impl.h	1949
16.189 l4/re/impl/rm_impl.h File Reference	1950
16.189.1 Detailed Description	1951
16.190 rm_impl.h	1951
16.191 l4/re/l4aux.h File Reference	1952
16.191.1 Detailed Description	1953
16.192 l4aux.h	1954
16.193 l4/re/log File Reference	1954
16.193.1 Detailed Description	1956
16.194 log	1956
16.195 l4/re/log-sys.h File Reference	1956

16.195.1 Detailed Description	1956
16.196 log-sys.h	1957
16.197 l4/re/mem_alloc File Reference	1957
16.197.1 Detailed Description	1958
16.198 mem_alloc	1958
16.199 l4/re/mem_alloc-sys.h File Reference	1959
16.199.1 Detailed Description	1960
16.200 mem_alloc-sys.h	1960
16.201 l4/re/namespace File Reference	1960
16.201.1 Detailed Description	1962
16.202 namespace	1962
16.203 l4/re/namespace-sys.h File Reference	1963
16.203.1 Detailed Description	1963
16.204 namespace-sys.h	1964
16.205 l4/re/parent File Reference	1964
16.205.1 Detailed Description	1966
16.206 parent	1966
16.207 l4/re/parent-sys.h File Reference	1966
16.207.1 Detailed Description	1967
16.208 parent-sys.h	1967
16.209 l4/re/protocols.h File Reference	1967
16.209.1 Detailed Description	1967
16.209.2 Enumeration Type Documentation	1968
16.209.2.1 L4re_protocols	1968
16.210 protocols.h	1968
16.211 l4/re/rm File Reference	1969
16.211.1 Detailed Description	1970
16.212 rm	1970
16.213 l4/re/rm-sys.h File Reference	1974
16.213.1 Detailed Description	1974
16.214 rm-sys.h	1975
16.215 l4/re/shared_cap File Reference	1975
16.215.1 Detailed Description	1977
16.216 shared_cap	1977
16.217 l4/re/util/shared_cap File Reference	1977
16.217.1 Detailed Description	1979
16.218 shared_cap	1979
16.219 l4/re/unique_cap File Reference	1980
16.219.1 Detailed Description	1982
16.220 unique_cap	1982
16.221 l4/re/util/unique_cap File Reference	1982
16.221.1 Detailed Description	1984

16.222 unique_cap	1984
16.223 l4/re/util/bitmap_cap_alloc File Reference	1985
16.223.1 Detailed Description	1986
16.224 bitmap_cap_alloc	1986
16.225 l4/re/util/cap File Reference	1987
16.225.1 Detailed Description	1988
16.226 cap	1989
16.227 l4/re/util/cap_alloc_impl.h File Reference	1989
16.227.1 Detailed Description	1990
16.228 cap_alloc_impl.h	1990
16.229 l4/re/util/counting_cap_alloc File Reference	1991
16.229.1 Detailed Description	1992
16.230 counting_cap_alloc	1992
16.231 l4/re/util/item_alloc File Reference	1994
16.231.1 Detailed Description	1995
16.232 item_alloc	1995
16.233 l4/re/util/kumem_alloc File Reference	1996
16.233.1 Detailed Description	1997
16.234 kumem_alloc	1997
16.235 l4/re/util/region_mapping File Reference	1998
16.235.1 Detailed Description	1999
16.236 region_mapping	1999
16.237 l4/re/video/goos-sys.h File Reference	2004
16.237.1 Detailed Description	2004
16.238 goos-sys.h	2005
16.239 l4/shmc/shmc.h File Reference	2005
16.239.1 Detailed Description	2007
16.240 shmc.h	2008
16.241 l4/sigma0/sigma0.h File Reference	2010
16.241.1 Detailed Description	2011
16.241.2 Enumeration Type Documentation	2012
16.241.2.1 l4sigma0_return_flags_t	2012
16.241.3 Function Documentation	2012
16.241.3.1 l4sigma0_debug_dump()	2012
16.241.3.2 l4sigma0_map_anypage()	2012
16.241.3.3 l4sigma0_map_errstr()	2013
16.241.3.4 l4sigma0_map_iomem()	2013
16.241.3.5 l4sigma0_map_kip()	2014
16.241.3.6 l4sigma0_map_mem()	2014
16.241.3.7 l4sigma0_new_client()	2015
16.242 sigma0.h	2015
16.243 l4/sys/__kernel_object_impl.h File Reference	2017

16.243.1 Detailed Description	2017
16.244 __kernel_object_impl.h	2017
16.245 l4/sys/__ktrace-impl.h File Reference	2018
16.245.1 Detailed Description	2019
16.246 __ktrace-impl.h	2019
16.247 l4/sys/__typeinfo.h File Reference	2020
16.247.1 Detailed Description	2023
16.248 __typeinfo.h	2023
16.249 l4/sys/__vm-arm.h File Reference	2033
16.249.1 Detailed Description	2034
16.250 __vm-arm.h	2034
16.251 l4/sys/cache.h File Reference	2035
16.251.1 Detailed Description	2036
16.252 cache.h	2036
16.253 arm/l4/sys/cache.h File Reference	2037
16.253.1 Detailed Description	2037
16.254 cache.h	2038
16.255 amd64/l4/sys/cache.h File Reference	2039
16.255.1 Detailed Description	2039
16.256 cache.h	2040
16.257 x86/l4/sys/cache.h File Reference	2040
16.257.1 Detailed Description	2041
16.258 cache.h	2041
16.259 l4/sys/capability File Reference	2042
16.259.1 Detailed Description	2043
16.259.2 Macro Definition Documentation	2043
16.259.2.1 L4_DISABLE_COPY	2043
16.260 capability	2044
16.261 l4/sys/compiler.h File Reference	2045
16.261.1 Detailed Description	2046
16.261.2 Macro Definition Documentation	2047
16.261.2.1 L4_ALWAYS_INLINE	2047
16.261.2.2 L4_HIDDEN	2047
16.261.2.3 L4_NOTHROW	2047
16.262 compiler.h	2048
16.263 l4/sys/consts.h File Reference	2049
16.263.1 Detailed Description	2051
16.264 consts.h	2052
16.265 arm/l4/sys/consts.h File Reference	2053
16.265.1 Detailed Description	2054
16.266 consts.h	2054
16.267 amd64/l4/sys/consts.h File Reference	2055

16.267.1 Detailed Description	2055
16.268 consts.h	2055
16.269 x86/I4/sys/consts.h File Reference	2055
16.269.1 Detailed Description	2056
16.270 consts.h	2056
16.271 I4/re/consts.h File Reference	2056
16.271.1 Detailed Description	2057
16.271.2 Enumeration Type Documentation	2057
16.271.2.1 anonymous enum	2058
16.272 consts.h	2058
16.273 I4/re/consts File Reference	2058
16.273.1 Detailed Description	2059
16.274 consts	2060
16.275 I4/sys/cxx/ipc_client File Reference	2060
16.275.1 Macro Definition Documentation	2062
16.275.1.1 L4_RPC_DEF	2062
16.276 ipc_client	2062
16.277 I4/sys/cxx/ipc_iface File Reference	2063
16.277.1 Detailed Description	2065
16.277.2 Macro Definition Documentation	2065
16.277.2.1 L4_INLINE_RPC	2065
16.277.2.2 L4_INLINE_RPC_NF	2066
16.277.2.3 L4_INLINE_RPC_NF_OP	2066
16.277.2.4 L4_INLINE_RPC_OP	2067
16.277.2.5 L4_RPC	2067
16.277.2.6 L4_RPC_NF	2068
16.277.2.7 L4_RPC_NF_OP	2068
16.277.2.8 L4_RPC_OP	2069
16.278 ipc_iface	2069
16.279 I4/cxx/ipc_server File Reference	2074
16.279.1 Detailed Description	2075
16.280 ipc_server	2076
16.281 I4/sys/cxx/ipc_types File Reference	2077
16.282 ipc_types	2079
16.283 I4/sys/cxx/smart_capability_1x File Reference	2085
16.284 smart_capability_1x	2086
16.285 I4/sys/cxx/types File Reference	2089
16.285.1 Macro Definition Documentation	2090
16.285.1.1 L4_TYPES_FLAGS_OPS_DEF	2090
16.286 types	2090
16.287 I4/sys/debugger File Reference	2093
16.287.1 Detailed Description	2094

16.288 debugger	2094
16.289 l4/sys/debugger.h File Reference	2095
16.289.1 Detailed Description	2096
16.289.2 Function Documentation	2096
16.289.2.1 __strcpy_maxlen()	2096
16.289.2.2 l4_debugger_get_object_name()	2097
16.289.2.3 l4_debugger_query_log_name()	2097
16.289.2.4 l4_debugger_query_log_typeid()	2098
16.289.2.5 l4_debugger_switch_log()	2098
16.290 debugger.h	2099
16.291 l4/sys/err.h File Reference	2102
16.291.1 Detailed Description	2103
16.292 err.h	2103
16.293 l4/sys/exception File Reference	2103
16.293.1 Detailed Description	2104
16.294 exception	2105
16.295 l4/sys/factory File Reference	2105
16.295.1 Detailed Description	2107
16.296 factory	2107
16.297 l4/sys/factory.h File Reference	2109
16.297.1 Detailed Description	2110
16.298 factory.h	2111
16.299 l4/sys/icu File Reference	2115
16.299.1 Detailed Description	2116
16.300 icu	2116
16.301 l4/sys/icu.h File Reference	2116
16.301.1 Detailed Description	2119
16.302 icu.h	2119
16.303 l4/sys/ipc.h File Reference	2122
16.303.1 Detailed Description	2124
16.303.2 Function Documentation	2124
16.303.2.1 l4_ipc_to_errno()	2124
16.304 ipc.h	2125
16.305 arm/l4f/l4/sys/ipc.h File Reference	2128
16.305.1 Detailed Description	2129
16.306 ipc.h	2129
16.307 x86/l4f/l4/sys/ipc.h File Reference	2129
16.307.1 Detailed Description	2130
16.308 ipc.h	2130
16.309 l4/sys/ipc_gate File Reference	2131
16.309.1 Detailed Description	2132
16.310 ipc_gate	2132

16.311 I4/sys/ipc_gate.h File Reference	2133
16.311.1 Detailed Description	2134
16.312 ipc_gate.h	2135
16.313 I4/sys/irq File Reference	2136
16.313.1 Detailed Description	2137
16.314 irq	2137
16.315 I4/sys/kernel_object.h File Reference	2139
16.315.1 Detailed Description	2140
16.316 kernel_object.h	2140
16.317 I4/sys/kip File Reference	2141
16.317.1 Detailed Description	2142
16.318 kip	2143
16.319 I4/sys/ktrace.h File Reference	2144
16.319.1 Detailed Description	2145
16.320 ktrace.h	2146
16.321 I4/sys/I4int.h File Reference	2146
16.321.1 Detailed Description	2147
16.322 I4int.h	2147
16.323 arm/I4/sys/I4int.h File Reference	2148
16.323.1 Detailed Description	2148
16.324 I4int.h	2148
16.325 amd64/I4/sys/I4int.h File Reference	2148
16.325.1 Detailed Description	2149
16.326 I4int.h	2149
16.327 x86/I4/sys/I4int.h File Reference	2149
16.327.1 Detailed Description	2150
16.328 I4int.h	2150
16.329 I4/sys/memdesc.h File Reference	2150
16.329.1 Detailed Description	2152
16.330 memdesc.h	2152
16.331 I4/sys/meta File Reference	2154
16.331.1 Detailed Description	2156
16.332 meta	2156
16.333 I4/sys/pager File Reference	2156
16.333.1 Detailed Description	2158
16.334 pager	2158
16.335 I4/sys/platform_control File Reference	2158
16.335.1 Detailed Description	2159
16.336 platform_control	2160
16.337 I4/sys/platform_control.h File Reference	2160
16.337.1 Detailed Description	2162
16.338 platform_control.h	2162

16.339 I4/sys/rcv_endpoint File Reference	2164
16.339.1 Detailed Description	2165
16.340 rcv_endpoint	2165
16.341 I4/sys/rcv_endpoint.h File Reference	2166
16.341.1 Detailed Description	2167
16.341.2 Enumeration Type Documentation	2167
16.341.2.1 L4_rcv_ep_ops	2167
16.342 rcv_endpoint.h	2167
16.343 I4/sys/scheduler File Reference	2168
16.343.1 Detailed Description	2169
16.344 scheduler	2169
16.345 I4/sys/scheduler.h File Reference	2170
16.345.1 Detailed Description	2171
16.346 scheduler.h	2172
16.347 I4/sys/semaphore File Reference	2174
16.347.1 Detailed Description	2176
16.348 semaphore	2176
16.349 I4/sys/smart_capability File Reference	2176
16.349.1 Detailed Description	2178
16.350 smart_capability	2178
16.351 I4/sys/task File Reference	2180
16.351.1 Detailed Description	2182
16.352 task	2182
16.353 I4/sys/task.h File Reference	2183
16.353.1 Detailed Description	2184
16.354 task.h	2184
16.355 I4/sys/thread File Reference	2187
16.355.1 Detailed Description	2188
16.356 thread	2189
16.357 I4/cxx/thread File Reference	2190
16.357.1 Detailed Description	2191
16.358 thread	2192
16.359 I4/sys/typeinfo_svr File Reference	2192
16.359.1 Detailed Description	2194
16.360 typeinfo_svr	2194
16.361 I4/sys/types.h File Reference	2195
16.361.1 Detailed Description	2197
16.361.2 Function Documentation	2197
16.361.2.1 I4_capability_next()	2197
16.362 types.h	2197
16.363 I4/sys/utcb.h File Reference	2200
16.363.1 Detailed Description	2202

16.364 utcb.h	2202
16.365 arm/l4/sys/utcb.h File Reference	2204
16.365.1 Detailed Description	2205
16.366 utcb.h	2206
16.367 amd64/l4/sys/utcb.h File Reference	2207
16.367.1 Detailed Description	2208
16.368 utcb.h	2208
16.369 x86/l4/sys/utcb.h File Reference	2210
16.369.1 Detailed Description	2211
16.370 utcb.h	2211
16.371 l4/sys/vcon File Reference	2212
16.371.1 Detailed Description	2214
16.372 vcon	2214
16.373 l4/sys/vcon.h File Reference	2214
16.373.1 Detailed Description	2216
16.373.2 Enumeration Type Documentation	2217
16.373.2.1 L4_vcon_read_flags	2217
16.374 vcon.h	2217
16.375 l4/sys/vhw.h File Reference	2220
16.375.1 Detailed Description	2221
16.376 vhw.h	2221
16.377 l4/sys/vm File Reference	2222
16.377.1 Detailed Description	2223
16.378 vm	2224
16.379 l4/util/assert.h File Reference	2224
16.379.1 Detailed Description	2225
16.380 assert.h	2225
16.381 l4/sys/assert.h File Reference	2226
16.381.1 Detailed Description	2228
16.381.2 Macro Definition Documentation	2228
16.381.2.1 l4_assert	2228
16.382 assert.h	2228
16.383 l4/util/atomic.h File Reference	2229
16.383.1 Detailed Description	2231
16.384 atomic.h	2232
16.385 arm/l4/sys/atomic.h File Reference	2236
16.385.1 Detailed Description	2236
16.386 atomic.h	2237
16.387 l4/cxx/atomic.h File Reference	2237
16.387.1 Detailed Description	2238
16.388 atomic.h	2238
16.389 l4/util/backtrace.h File Reference	2238

16.389.1 Detailed Description	2239
16.389.2 Function Documentation	2239
16.389.2.1 l4util_backtrace()	2240
16.390 backtrace.h	2240
16.391 l4/util/base64.h File Reference	2240
16.391.1 Detailed Description	2241
16.392 base64.h	2242
16.393 l4/util/bitops.h File Reference	2242
16.393.1 Detailed Description	2244
16.394 bitops.h	2244
16.395 l4/util/elf.h File Reference	2247
16.395.1 Detailed Description	2263
16.395.2 Macro Definition Documentation	2263
16.395.2.1 DF_1_DIRECT	2263
16.395.2.2 DF_1_DISPRELPND	2263
16.395.2.3 DF_1_GLOBAL	2264
16.395.2.4 DF_1_GROUP	2264
16.395.2.5 DF_1_INTERPOSE	2264
16.395.2.6 DF_1_NODEFLIB	2264
16.395.2.7 DF_1_NODUMP	2265
16.395.2.8 DF_1_NOOPEN	2265
16.395.2.9 DF_1_NOW	2265
16.395.2.10 DF_1_ORIGIN	2265
16.395.2.11 DF_P1_GROUPPERM	2266
16.395.2.12 DF_P1_LAZYLOAD	2266
16.395.2.13 DT_NULL	2266
16.395.2.14 EI_CLASS [1/2]	2266
16.395.2.15 EI_CLASS [2/2]	2267
16.395.2.16 EI_DATA [1/2]	2267
16.395.2.17 EI_DATA [2/2]	2267
16.395.2.18 EI_OSABI [1/2]	2267
16.395.2.19 EI_OSABI [2/2]	2268
16.395.2.20 EI_PAD [1/2]	2268
16.395.2.21 EI_PAD [2/2]	2268
16.395.2.22 EI_VERSION [1/2]	2268
16.395.2.23 EI_VERSION [2/2]	2269
16.395.2.24 ELFCLASSNONE [1/2]	2269
16.395.2.25 ELFCLASSNONE [2/2]	2269
16.395.2.26 ELFDATA2LSB [1/2]	2269
16.395.2.27 ELFDATA2LSB [2/2]	2270
16.395.2.28 ELFDATA2MSB [1/2]	2270
16.395.2.29 ELFDATA2MSB [2/2]	2270

16.395.2.30 ELFDATANONE [1/2]	2270
16.395.2.31 ELFDATANONE [2/2]	2271
16.395.2.32 ELFOSABI_AIX	2271
16.395.2.33 ELFOSABI_FREEBSD	2271
16.395.2.34 ELFOSABI_HPUX [1/2]	2271
16.395.2.35 ELFOSABI_HPUX [2/2]	2272
16.395.2.36 ELFOSABI_IRIX	2272
16.395.2.37 ELFOSABI_LINUX	2272
16.395.2.38 ELFOSABI_MODESTO	2272
16.395.2.39 ELFOSABI_NETBSD	2273
16.395.2.40 ELFOSABI_OPENBSD	2273
16.395.2.41 ELFOSABI_SOLARIS	2273
16.395.2.42 ELFOSABI_SYSV [1/2]	2273
16.395.2.43 ELFOSABI_SYSV [2/2]	2274
16.395.2.44 ELFOSABI_TRU64	2274
16.395.2.45 NT_VERSION	2274
16.395.2.46 SHF_GROUP	2274
16.395.2.47 SHF_MASKOS	2275
16.395.2.48 SHF_TLS	2275
16.395.2.49 SHT_NUM	2275
16.396 elf.h	2276
16.397 l4/util/getopt.h File Reference	2285
16.397.1 Detailed Description	2286
16.398 getopt.h	2286
16.399 l4/util/keymap.h File Reference	2287
16.399.1 Detailed Description	2288
16.400 keymap.h	2288
16.401 l4/util/kip.h File Reference	2288
16.401.1 Macro Definition Documentation	2289
16.401.1.1 l4util_kip_for_each_feature	2289
16.401.2 Function Documentation	2290
16.401.2.1 l4util_kip_kernel_abi_version()	2290
16.401.2.2 l4util_kip_kernel_has_feature()	2290
16.401.2.3 l4util_kip_kernel_is_ux()	2290
16.402 kip.h	2291
16.403 l4/sys/kip.h File Reference	2291
16.403.1 Detailed Description	2292
16.403.2 Function Documentation	2292
16.403.2.1 l4_kernel_info_version_offset()	2292
16.403.2.2 l4_kip_clock()	2293
16.403.2.3 l4_kip_clock_lw()	2294
16.403.2.4 l4_kip_version()	2295

16.403.2.5 l4_kip_version_string()	2295
16.404 kip.h	2296
16.405 l4/util/kprintf.h File Reference	2297
16.405.1 Detailed Description	2298
16.406 kprintf.h	2298
16.407 l4/util/list_alloc.h File Reference	2299
16.407.1 Detailed Description	2299
16.407.2 Function Documentation	2300
16.407.2.1 l4la_alloc()	2300
16.407.2.2 l4la_avail()	2300
16.407.2.3 l4la_dump()	2300
16.407.2.4 l4la_free()	2301
16.407.2.5 l4la_init()	2301
16.408 list_alloc.h	2301
16.409 l4/util/lock.h File Reference	2302
16.409.1 Detailed Description	2302
16.410 lock.h	2303
16.411 l4/util/mb_info.h File Reference	2303
16.411.1 Detailed Description	2305
16.411.2 Macro Definition Documentation	2306
16.411.2.1 l4util_mb_for_each_mmap_entry	2306
16.411.2.2 L4UTIL_MB_MEMORY	2306
16.411.2.3 MB_ARD_MEMORY	2306
16.411.2.4 MB_ART_MEMORY	2307
16.412 mb_info.h	2307
16.413 l4/util/parse_cmd.h File Reference	2311
16.413.1 Detailed Description	2312
16.413.2 Function Documentation	2312
16.413.2.1 parse_cmdline()	2312
16.414 parse_cmd.h	2313
16.415 l4/util/rand.h File Reference	2314
16.415.1 Detailed Description	2315
16.416 rand.h	2315
16.417 l4/util/splitlog2.h File Reference	2315
16.417.1 Detailed Description	2316
16.418 splitlog2.h	2316
16.419 l4/util/thread.h File Reference	2317
16.419.1 Detailed Description	2318
16.419.2 Macro Definition Documentation	2318
16.419.2.1 __L4UTIL_THREAD_FUNC	2319
16.420 thread.h	2319
16.421 l4/sys/thread.h File Reference	2319

16.421.1 Detailed Description	2322
16.422 thread.h	2322
16.423 arm/l4/sys/thread.h File Reference	2329
16.423.1 Detailed Description	2329
16.424 thread.h	2329
16.425 l4/vbus/vbus.h File Reference	2330
16.425.1 Detailed Description	2331
16.425.2 Enumeration Type Documentation	2331
16.425.2.1 anonymous enum	2331
16.425.2.2 l4vbus_icu_src_types	2332
16.426 vbus.h	2332
16.427 l4/vbus/vbus_interfaces.h File Reference	2333
16.427.1 Detailed Description	2335
16.427.2 Enumeration Type Documentation	2335
16.427.2.1 anonymous enum	2335
16.427.2.2 l4vbus_iface_type_t	2335
16.427.3 Function Documentation	2335
16.427.3.1 l4vbus_subinterface_supported()	2335
16.428 vbus_interfaces.h	2336
16.429 l4/vbus/vbus_types.h File Reference	2336
16.429.1 Detailed Description	2338
16.429.2 Enumeration Type Documentation	2338
16.429.2.1 l4vbus_device_flags_t	2338
16.429.2.2 l4vbus_resource_flags_t	2338
16.429.2.3 l4vbus_resource_type_t	2339
16.430 vbus_types.h	2339
17 Example Documentation	2341
17.1 hello/server/src/main.c	2341
17.2 examples/sys/ipc/ipc_example.c	2341
17.3 examples/sys/ipc/ipc.cfg	2342
17.4 examples/sys/start-with-exc/main.c	2343
17.5 examples/sys/singlestep/main.c	2344
17.6 examples/sys/aliens/main.c	2346
17.7 examples/sys/utcb-ipc/main.c	2349
17.8 examples/sys/ux-vhw/main.c	2350
17.9 examples/sys/isr/main.c	2350
17.10 examples/clntsrv/server.cc	2352
17.11 examples/clntsrv/client.cc	2352
17.12 examples/clntsrv/clntsrv.cfg	2353
17.13 examples/libs/l4re/c/ma+rm.c	2353
17.14 examples/libs/l4re/c++/mem_alloc/ma+rm.cc	2354

17.15 examples/libs/l4re/c++/shared_ds/ds_clnt.cc	2355
17.16 examples/libs/l4re/c++/shared_ds/ds_srv.cc	2356
17.17 examples/libs/l4re/c++/shared_ds/shared_ds.cfg	2358
17.18 examples/libs/l4re/streammap/server.cc	2358
17.19 examples/libs/l4re/streammap/client.cc	2359
17.20 examples/libs/l4re/streammap/streammap.cfg	2360
17.21 examples/libs/libirq/loop.c	2360
17.22 examples/libs/libirq/async_isr.c	2361
17.23 examples/sys/migrate/thread_migrate.cc	2361
17.24 examples/sys/migrate/thread_migrate.cfg	2362
17.25 tmpfs/lib/src/fs.cc	2363
17.26 examples/libs/shmc/prodcons.c	2368

Index	2371
--------------	-------------

Chapter 1

Overview

Welcome to the documentation of the L4 Runtime Environment, 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 environment 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](#) environment.

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 Fiasco.OC

The Fiasco.OC microkernel is the lowest-level piece of software running in an L4-based system. The microkernel is the only program 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](#)).

Fiasco.OC kernel 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*. Fiasco system calls 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 Fiasco.OC 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 Fiasco 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.

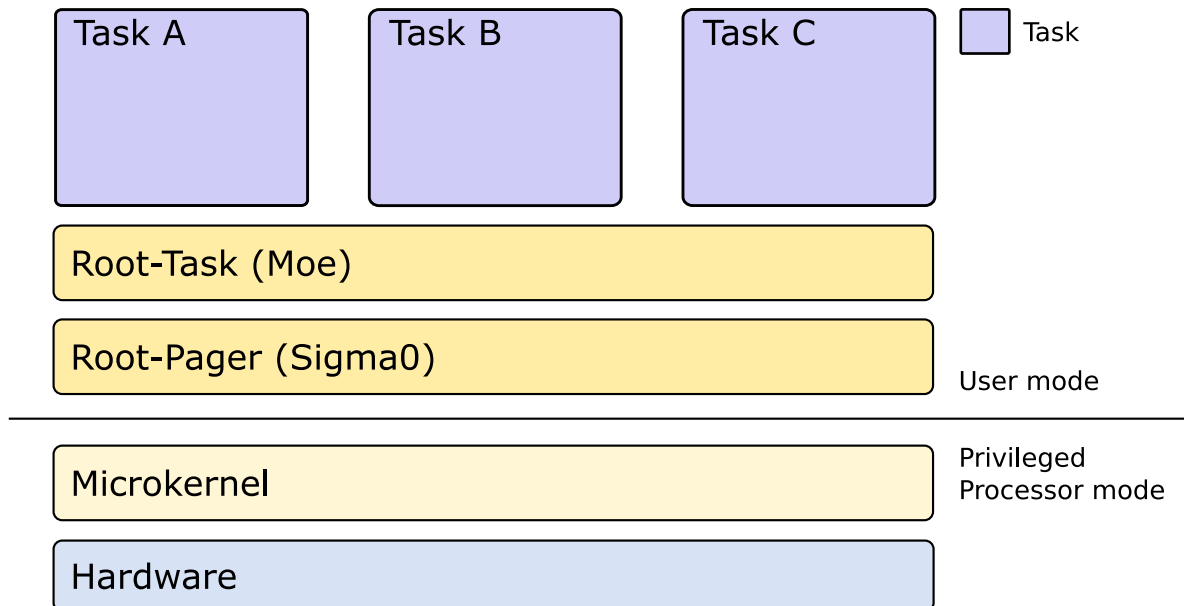


Figure 2.1 Basic Structure of an L4Re based system

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 [L4](#) 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 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 L4 Runtime Environment (L4Re)

The [L4](#) Runtime Environment ([L4Re](#)) provides a basic set of services and abstractions, which are useful to implement and run user-level applications on top of the Fiasco.OC microkernel.

[L4Re](#) consists of a set of libraries and servers. Libraries as well as server interfaces are completely object oriented. They implement prototype implementations for the classes defined by the [L4Re](#) specification.

A minimal L4Re-based application needs 3 components to be booted beforehand: the Fiasco 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 and has a working knowledge of C++.

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

- Fiasco.OC — Microkernel
- Sigma0 — Root Pager
- Moe — Root Task
- Ned — Init Process
- hello — Hello World Application

The guide assumes that you already compiled the base components and describes how to generate an ISO image, with GRUB 1 or GRUB 2 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 Fiasco.OC 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 `I4/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 grubliso E=hello MODULES_LIST=/tmp/modules.list MODULE_SEARCH_PATH=/tmp:<path_to_fiasco_builddir>
```

Or as an alternative use GRUB 2:

```
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-i386 -cdrom images/hello.iso -serial stdio
```

If you press `<ESC>` in the terminal that shows you the serial output you enter the Fiasco.OC kernel 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.

- [Capabilities and Naming](#)
- [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 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.

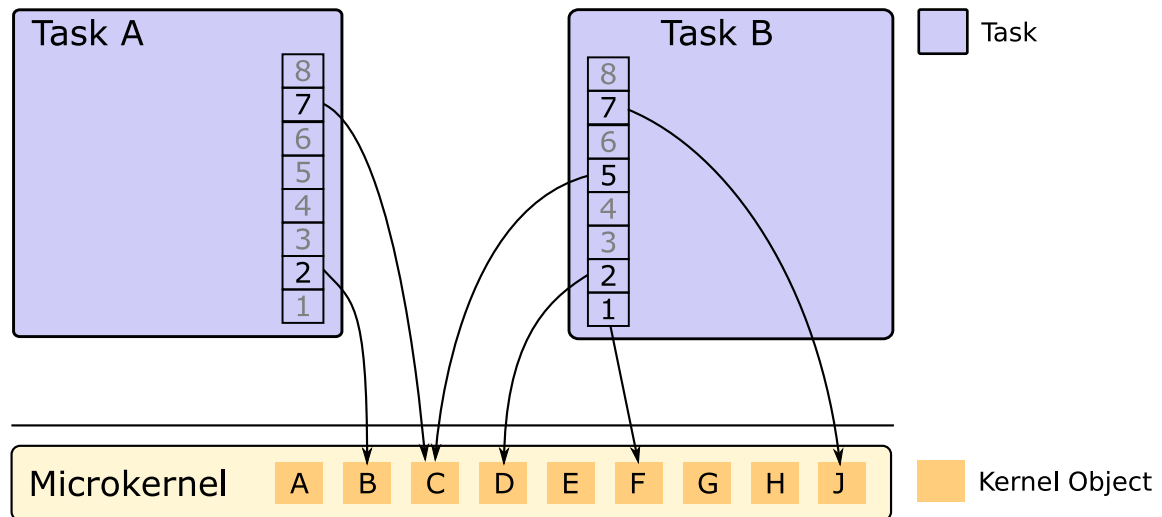


Figure 4.1 Capabilities and Local Naming in L4

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.

4.2 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.

4.2.1 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");
```

4.2.2 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.

4.3 Memory management - Data Spaces and the Region Map

4.3.1 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 grant the memory in a secure way, often referred to as *memory mapping*.

The described mechanism can be used to construct a memory 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.

4.3.1.1 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.

4.3.1.2 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.

4.3.1.3 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](#).

4.4 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 Fiasco kernel and connected either to the serial line or to the screen if available.

See also

[Virtual Console](#)

4.5 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](#)

4.6 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.

4.6.1 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.

4.6.2 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.

4.7 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.
```

4.8 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;
};
```

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.

4.8.1 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::ipc::String<>`.

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

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

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

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

Type	Direction	Client-Type	Server-Type
T	Input	T	T
T const &	Input	T const &	T const &
T const *	Input	T const *	T const &
T &	Output	T &	T &
T *	Output	T *	T &
L4::Ipc::In_out<T &>	In/Out	T &	T &
L4::Ipc::In_out<T *>	In/Out	T *	T &
L4::Ipc::Cap<T>	Input	L4::Ipc::Cap<T>	L4::Ipc::Snd_fpage
L4::Ipc::Out<L4::Cap<T> >	Output	L4::Cap<T>	L4::Ipc::Cap<T> &
L4::Ipc::Rcv_fpage	Input	L4::Ipc::Rcv_fpage	void
L4::Ipc::Small_buf	Input	L4::Ipc::Small_buf	void

Array types can be used to transmit arrays of variable length. They can either be stored in a client-provided buffer ([L4::ipc::Array](#)), copied into a server-provided buffer ([L4::ipc::Array_in_buf](#)) or directly read and written into the UTCB ([L4::ipc::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
L4::Ipc::Array<const T>	Input	L4::Ipc::Array<const T>	L4::Ipc::Array_ref<const T>
L4::Ipc::Array<const T>	Input	L4::Ipc::Array<const T>	L4::Ipc::Array_in_buf<T> const &
L4::Ipc::Array<T> &	Output	L4::Ipc::Array<T> &	L4::Ipc::Array_ref<T> &
L4::Ipc::Array_ref<T> &	Output	L4::Ipc::Array_ref<T> &	L4::Ipc::Array_ref<T> &

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>	T
<code>L4::Ipc::Opt<const T*></code>	Input	<code>L4::Ipc::Opt<const T*></code>	T
<code>L4::Ipc::Opt<T &></code>	Output	T &	<code>L4::Ipc::Opt<T> &</code>
<code>L4::Ipc::Opt<T *></code>	Output	T *	<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>

4.8.2 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.

4.8.3 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...)`.

4.9 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.

4.9.1 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.

4.9.2 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

4.9.3 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.

SRC_CC_IS_CXX11

C++ standard to use. Default is `c++0x`.

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.

4.9.4 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.

4.9.5 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_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.

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 Fiasco kernel 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.

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.

TEST_RUN_TAGS \ TEST_TAGS	tag	!tag	+tag	-tag
tag or tag=1	y		y	
unspecified		y	y	
tag=0		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` unprefixd.

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. They are run by default but can be excluded from execution by settings `TEST_RUN_TAGS` to `impl-def=0`. These tests are marked using `+impl-def`.

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

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

Fiasco kernel 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.

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

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

```
make test
```

or executing them directly, like

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

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

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

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

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

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

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

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

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

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

Running Tests in External Programs

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

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

EXTERNAL_TEST_STARTER

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

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

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

Running Tests in a Simulator

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

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

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

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

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

SIMULATOR_START

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

SIMULATOR_IMAGETYPE

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

SIMULATOR_HOST

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

SIMULATOR_PORT

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

SIMULATOR_START_SLEEPTIME

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: `taper-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/taper-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.

4.10 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 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 physical memory available on a system; ultimately, the amount of physical memory available is the strict limit for the factory to remain operational.

The kernel factory creates the following kinds of objects:

- `Dmar_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](#).

4.10.1 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 Fiasco.OC and [L4Re](#).

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 Fiasco.OC 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 of Fiasco.OC 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](#)
- [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.

5.1 Sigma0, the Root-Pager

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

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)

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,
}
```

Debugging Flags

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

```
Dbg = {
```

```

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

```

Loader Flags

Flags for configuring the loading process of an application.

```

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

```

5.3.1.4 Application Startup Details

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

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

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

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

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

- `factory` The factory used by the new process to create new kernel objects, such as threads etc. This must be a capability to an object implementing the [L4::Factory](#) protocol and defaults to the factory object provided to Ned.
- `mem` The memory allocator provided to the application and used by Ned allocates data spaces for the process. This defaults to Ned's memory allocator object (see [L4Re::Mem_alloc](#)).
- `rm_fab` The generic factory object used to allocate the region-map object for the process. (defaults to Ned's memory allocator).
- `log_fab` The generic factory to create the [L4Re::Log](#) object for the application's output (defaults to Ned's memory allocator). The `create` method of the `log_fab` object is called with `log_tag` and `log_color`, from this table, as arguments.
- `log_tag` The string used for tagging log output of this process (defaults to the program name) (see [log_fab](#)).
- `log_color` The color used for the log tag (defaults to "white").
- `scheduler` The scheduler object used for the process' threads (defaults to Ned's own scheduler).
- `caps` The table with application-specific named capabilities (default is an empty table). If the table does not contain a capability with the name 'rom', the 'rom' capability from Ned's initial caps is inserted into the table.

5.3.1.5 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")
```

5.3.1.6 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({caps = {log = L4.Env.log, 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:

```
[--cmdcap|-c CAP] [--noexit] [--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.

- Command Capability Option: **cmdcap**, **c**
Start ned in server mode. After running the initial Lua script, ned will accept Lua statements via IPC on the given capability (see [L4Re::Ned::Cmd_control](#)).
- No exit Option: **noexit**
Ned does not terminate if only Return is entered at ned's prompt.
- Execute Statement Option: **execute**, **e**
Execute the Lua statement `STATEMENT`.

5.4 Io, the Io Server

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

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

Io's configuration consists of two parts:

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

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

Hardware Description

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

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

Virtual Bus Description

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

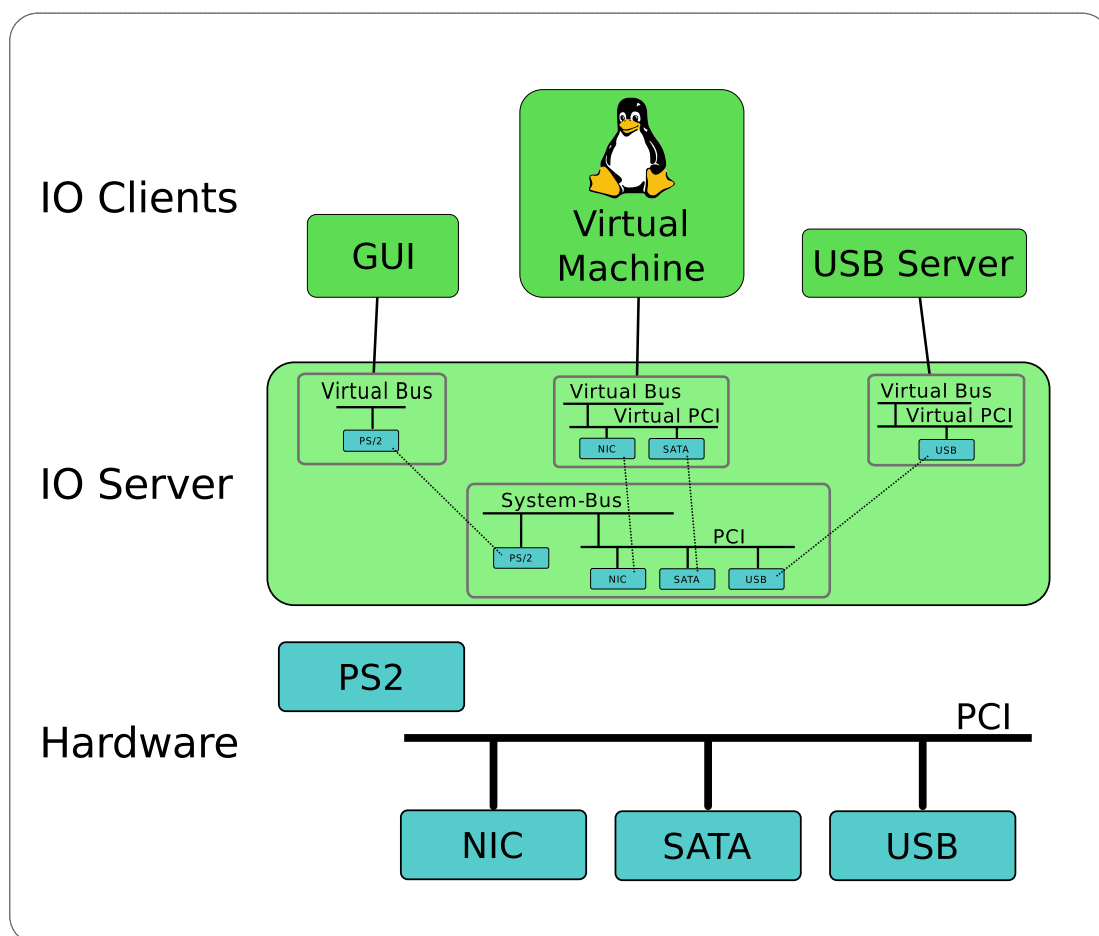


Figure 5.1 IO Service Architecture Overview

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

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

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

A very simple configuration for Io could look like this:

```
-- vim:ft=lua
-- Example configuration for io
-- Configure two platform devices to be known to io
Io.Dt.add_children(Io.system_bus(), function()
  -- create a new hardware device called "FOODEVICE"
  FOODEVICE = Io.Hw.Device(function()
    -- set the compatibility IDs for this device
    -- a client tries to match against these IDs and configures
    -- itself accordingly
    -- the list should be sorted from specific to less specific IDs
    compatible = {"dev-foo,mmio", "dev-foo"};
    -- set the 'hid' property of the device, the hid can also be used
    -- as a compatible ID when matching clients
    Property.hid = "dev-foo,Example";
    -- note: names for resources are truncated to 4 letters and a client
    -- can determine the name from the ID field of a l4vbus_resource_t
    -- add two resources 'irq0' and 'reg0' to the device
    Resource.irq0 = Io.Res.irq(17);
    Resource.reg0 = Io.Res.mmio(0x6f000000, 0x6f007fff);
  end);
  -- create a new hardware device called "BARDEVICE"
  BARDEVICE = Io.Hw.Device(function()
    -- set the compatibility IDs for this device
    -- a client tries to match against these IDs and configures
    -- itself accordingly
    -- the list should be sorted from specific to less specific IDs
    compatible = {"dev-bar,mmio", "dev-bar"};
    -- set the 'hid' property of the device, the hid can also be used
    -- as a compatible ID when matching clients
    Property.hid = "dev-bar,Example";
    -- Specify that this device is able to use direct memory access (DMA).
    -- This is needed to allow clients to gain access to DMA addresses
    -- used by this device to directly access memory.
    Property.flags = Io.Hw_device_DF_dma_supported;
    -- note: names for resources are truncated to 4 letters and a client
    -- can determine the name from the ID field of a l4vbus_resource_t
    -- add three resources 'irq0', 'irq1', and 'reg0' to the device
    Resource.irq0 = Io.Res.irq(19);
    Resource.irq1 = Io.Res.irq(20);
    Resource.reg0 = Io.Res.mmio(0x6f100000, 0x6f100fff);
  end);
end);
Io.add_vbusses
{
  -- Create a virtual bus for a client and give access to FOODEVICE
  client1 = Io.Vi.System_bus(function ()
    dev = wrap(Io.system_bus():match("FOODEVICE"));
  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("BARDEVICE"));
  end);
}
```

Each device supports a 'compatible' property. It is a list of compatibility strings. A client matches itself against one (or multiple) compatibility IDs and configures itself accordingly. All other device members are handled according to their type. If the type is a resource (Io.Res) it is added as a named resource. Note that resource names are truncated to 4 letters and are stored in the ID field of a [l4vbus_resource_t](#). If the type is a device it is added as a child device to the current one. All other types are treated as a device property which can be used to configure a device driver. Right now, device properties are internal to Io only.

Matching and Assigning PCI Devices

Assigning clients PCI devices could look like this:

```
-- This is a configuration snippet for PCI device selection
local hw = Io.system_bus();
Io.add_vbusses
{
  pciclient = Io.Vi.System_bus(function ()
    PCI = Io.Vi.PCI_bus(function ()
      pci_mm      = wrap(hw:match("PCI/CC_04"));
      pci_net     = wrap(hw:match("PCI/CC_02"));
      pci_storage = wrap(hw:match("PCI/CC_01"));
    end)
  end)
}
```

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

- PCI/CC_cc
- PCI/CC_ccss
- PCI/CC_ccssp
- PCI/VEN_vvv
- PCI/DEV_ddd
- PCI/SUBSYS_sssssss
- 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
-	-	-

Common Name	Description	PCI ID string
display	Display controller	CC_03
vga	VGA compatible controller	CC_0300
xga	XGA compatible controller	CC_0301
-	-	-
media	Multimedia controller	CC_04
mm_video	Multimedia video controller	CC_0400
mm_audio	Multimedia audio controller	CC_0401
telephony	Computer telephony device	CC_0402
audio	Audio device	CC_0403
-	-	-
bridge	Bridge	CC_06
br_host	Host bridge	CC_0600
br_isa	ISA bridge	CC_0601
br_eisa	EISA bridge	CC_0602
br_microchannel	MicroChannel bridge	CC_0603
br_pci	PCI bridge	CC_0604
br_pcmcia	PCMCIA bridge	CC_0605
br_nubus	NuBus bridge	CC_0606
br_cardbus	CardBus bridge	CC_0607
br_raceway	RACEway bridge	CC_0608
br_semi_pci	Semi-transparent PCI-to-PCI bridge	CC_0609
br_infiniband_to_pci	InfiniBand to PCI host bridge	CC_060a
-	-	-
com	Communication controller	CC_07
com_serial	Serial controller	CC_0700
com_parallel	Parallel controller	CC_0701
com_multiport_ser	Multiport serial controller	CC_0702
com_modem	Modem	CC_0703
com_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

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
```
- `-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.

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";
    l4vmm,physmap;
    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.

`l4vmm,physmap` indicates that uvmm should try to map the dataspace to its actual physical address when no IOMMU is available. If the physical address cannot be determined or an IOMMU is available, then the memory will be mapped to the addresses supplied in `regs`. It is possible to omit the `regs` property when `l4vmm,physmap` is set. In this case, uvmm will fail to start if the physical address cannot be determined.

If a `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.

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`

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**.

To have KVM support nested virtualization you need to enable it via:

```
modprobe kvm_intel nested=1
```

uvmm does currently not support AMD-V.

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.

5.7 Mag, the GUI Multiplexer

Mag is the default multiplexer for graphics hardware.

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

Command Line Options

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

Mag Sessions

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

1. Mag client session

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

2. Client framebuffer session

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

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

Session Factory Options

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

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

- Label Option: **label**

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

- Color Option: **col**

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

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

Example

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

Mag Client Session Options

These options only apply to Mag client sessions.

- Default Background Option: **default-background**

`dfl-bg, default-background` Marks this session as the default background.

Mag Client Framebuffer Session Options

These options only apply to Mag client framebuffer sessions.

- Geometry Option: **geometry**

`g=GEOMETRY, geometry=GEOMETRY` Set the session's geometry and position on the screen. GEOMETRY is provided in an X11-style format, e.g. `g=WIDTHxHEIGHT+X_OFFSET+Y_OFFSET`.

- Focus Option: **focus**

`focus` Set the focus to this session.

- Collapsed Option: **shaded**

`shaded` The window is collapsed and only the title bar is visible. The window can be expanded by clicking into the title bar with the middle mouse button. Collapsing and expanding works also independently of this option.

- Fixed Option: **fixed** `fixed` The window cannot be moved on the screen.

- Barheight Option: **barheight**

`barheight=X` Set the height of the title bar in pixels.

Example

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

5.8 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

Chapter 6

Bootstrap, the L4 kernel bootstrapper

@subpage 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>:<dev>`
Use serial port number `<dev>` at PCI card number `<card>`, e.g. use `-comport=pci:0:0` for the first 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::Factory::create_irq](#) (Cap< Irq >const &target_cap, l4_utcb_t *utcb=[l4_utcb\(\)](#))

Use [create\(\)](#) with [Cap<Irq>](#) as argument instead.

Global [L4::Factory::create_thread](#) (Cap< Thread > const &target_cap, l4_utcb_t *utcb=[l4_utcb\(\)](#)) noexcept

Use [create\(\)](#) with [Cap<Thread>](#) as argument instead.

Global [L4::Factory::create_vm](#) (Cap< Vm >const &target_cap, l4_utcb_t *utcb=[l4_utcb\(\)](#)) noexcept

Use [create\(\)](#) with [Cap<Vm>](#) as argument instead.

Class [L4::lpc_svr::Timed_work< HOOKS >](#)

Use [L4::lpc_svr::Timeout_queue_hooks](#)

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

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

Global [L4Re::Mem_alloc::free](#) (L4::Cap< Dataspace > mem) const noexcept

This function is an empty stub which remains here for backward compatibility only. Use [L4::Task<L4Re::Mem_alloc::free>::unmap\(mem, L4_FP_DELETE_OBJ\)](#) or other similar means to remove a dataspace.

Global [l4re_ma_free](#) (l4re_ds_t const mem) L4_NOTHROW

This function is deprecated. Use [l4_task_unmap\(\)](#) or similar means to remove a dataspace.

Global [l4re_ma_free_srv](#) (l4_cap_idx_t srv, l4re_ds_t const mem) L4_NOTHROW

This function is deprecated. Use [l4_task_unmap\(\)](#) or similar means to remove a dataspace.

Global [L4vbus::Vbus::release_resource](#) (l4vbus_resource_t *res) const

This function is deprecated since Q3 2019. Use [release_ioport\(\)](#) instead.

Global [L4vbus::Vbus::request_resource](#) (l4vbus_resource_t *res, int=0) const

This function is deprecated since Q3 2019. Use [request_ioport\(\)](#) instead.

Global [l4vbus_release_resource](#) (l4_cap_idx_t vbus, l4vbus_resource_t const *res)

This function is deprecated since Q3 2019. Use [l4vbus_release_ioport\(\)](#) instead.

Global [l4vbus_request_resource](#) (l4_cap_idx_t vbus, l4vbus_resource_t const *res, int flags)

This function is deprecated since Q3 2019. Use [l4vbus_request_ioport\(\)](#) instead.

Global [L4virtio::Device::register_iface](#) (L4::lpc::Cap< L4::Triggerable > guest_irq, [L4::lpc::Out](#)< L4::Cap< L4::Triggerable > > host_irq, [L4::lpc::Out](#)< L4::Cap< L4Re::Dataspace > > config_ds)

Use [device_config\(\)](#), [device_notification_irq\(\)](#) and [lcu::bind\(\)](#) instead.

Chapter 8

Module Index

8.1 Modules

Here is a list of all modules:

Base API	126
Basic Macros	128
C++ IPC Interface Definition.	138
Internal Helpers	265
Cache Consistency	138
Capabilities	142
Error codes	179
Fiasco extensions	202
Fiasco real time scheduling extensions	206
Kernel Debugger	279
Flex pages	208
Integer Types	254
Kernel Interface Page	281
Fiasco-UX Virtual devices	207
Memory descriptors (C version)	349
Kernel Objects	283
Factory	188
IPC-Gate API	233
IRQs	236
Interrupt controller	266
L4 kernel object type information	313
Platform Control C API	397
Scheduler	417
Task	436
Thread	443
Thread control	459
vCPU API	523
Virtual Console	506
Virtual Machines	518
VM API for SVM	480
VM API for TZ	481
VM API for VMX	481
Memory operations.	354
Memory related	356

Object Invocation	383
Error Handling	173
Message Items	362
Message Tag	366
Realtime API	404
Timeouts	465
Virtual Registers (UTCBS)	519
ARM Virtual Registers (UTCB)	105
Buffer Registers (BRs)	136
Message Registers (MRs)	365
Exception registers	184
Thread Control Registers (TCRs)	459
amd64 Virtual Registers (UTCB)	523
x86 Virtual Registers (UTCB)	533
EDID parsing functionality	167
IO interface	225
IRQ handling library	236
Interface for asynchronous ISR handlers.	256
Interface for asynchronous ISR handlers with a given IRQ capability.	255
Interface using direct functionality.	260
Interface using direct functionality.	258
L4 IPC Opcodes	286
L4 VIRTIO Interface	292
L4 VIRTIO Block Device	289
L4 VIRTIO Input Device	291
L4 VIRTIO Transport Layer	293
L4 Vbus functions	301
L4vbus GPIO functions	322
L4vbus PCI functions	333
L4Re C Interface	314
Capability allocator	146
DMA Space Interface	159
Dataspace interface	163
Debug interface	166
Event interface	181
Initial Environment	249
Kumem allocator utility	285
L4Re Util C Interface	321
Log interface	338
Memory allocator	343
Namespace interface	379
Region map interface	405
Video API	497
L4Re C++ Interface	317
Auxiliary data	125
C++ Exceptions	137
Console API	151
Debugging API	167
Event API	180
L4Re ELF Auxiliary Information	319
L4Re Protocol identifiers	320
L4Re Util C++ Interface	321
Kumem utilities	285
L4Re Capability API	319
Logging interface	342
Name-space API	379

Parent API	396
Region map API	404
Vbus API	495
Server-Side IPC framework	422
Shared Memory Library	423
Chunks	147
Consumer	155
Producer	400
Signals	429
Consumer	152
Producer	399
Sigma0 API	428
Internal constants	265
Small C++ Template Library	432
Utility Functions	473
Atomic Instructions	106
Bit Manipulation	129
CPU related functions	138
Comfortable Command Line Parsing	151
ELF binary format	172
Functions to manipulate the local IDT	225
IA32 Port I/O API	225
Internal functions	266
Kernel Interface Page API	283
Low-Level Thread Functions	342
Random number support	403
Timestamp Counter	473
vCPU Support Library	526
Extended vCPU support	187

Chapter 9

Namespace Index

9.1 Namespace List

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

cxx	Our C++ library	535
cxx::Bits	Internal helpers for the cxx package	537
L4	L4 low-level kernel interface	538
L4::lpc	IPC related functionality	548
L4::lpc::Msg	IPC Message related functionality	556
L4::lpc_svr	Helper classes for L4::Server instantiation	561
L4::Typeid	Definition of interface data-type helpers	562
L4::Types	L4 basic type helpers for C++	563
L4Re	L4Re C++ Interfaces	564
L4Re::Util	Documentation of the L4 Runtime Environment utility functionality in C++	579
L4Re::Vfs	Virtual file system for interfaces POSIX libc	592
L4vbus	C++ interface of the Vbus API.	593
L4virtio	L4-VIRTIO Transport C++ API	594

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 >	869
L4::lpc::Array< char const, unsigned long >	865
L4::lpc::Array_ref< ELEM_TYPE, Array_len_default >	869
L4::lpc::Array< ELEM_TYPE, LEN_TYPE >	865
cxx::Bits::Base_avl_set< ITEM_TYPE, Lt_functor< ITEM_TYPE >, New_allocator, Bits::Avl_set_get_↵ key< ITEM_TYPE > >	658
cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >	604
cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, Lt_functor< KEY_TYPE >, New_allocator, Bits::Avl_map_get_key< KEY_TYPE > >	658
cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >	599
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >	614
cxx::Base_slab< sizeof(Type), L4_PAGESIZE, 2, New_allocator >	614
cxx::Slab< Type, Slab_size, Max_free, Alloc >	743
cxx::Base_slab_static< sizeof(Type), L4_PAGESIZE, 2, New_allocator >	621
cxx::Slab_static< Type, Slab_size, Max_free, Alloc >	746
cxx::Bits::Basic_list< Bits::Basic_list_policy< T, H_list_item > >	674
cxx::H_list< T, POLICY >	699
cxx::H_list_t< Weak_ref_base >	711
cxx::Bits::Basic_list< Bits::Basic_list_policy< T, S_list_item > >	674
cxx::S_list< T, Bits::Basic_list_policy< T, S_list_item > >	739
cxx::S_list< T, POLICY >	739
L4::Types::Bool< __lface_conflict< I, I2 >::value _lface_conflict< I, LIST::type >::value >	1149
L4::Types::Bool< false >	1149
L4::Types::False	1151
L4::Types::Same< A, B >	1163
L4::Types::Bool< I1::Proto !=PROTO_EMPTY &&I1::Proto==I2::Proto >	1149
L4::Types::Bool< lface_conflict< I, L2::type >::value _Conflict< L1::type, L2::type >::value >	1149
L4::Types::Bool< true >	1149
L4::Types::True	1165
L4::lpc::Msg::ls_valid_rpc_type< A * >	918
L4::lpc::Msg::ls_valid_rpc_type< T >	918

L4::Types::Bool< Typeid::Conflict< L1::type, L2::type >::value Conflict< L1, LIST... >::value Conflict< L2, LIST... >::value >	1149
L4::Types::Bool< Typeid::Iface_conflict< L::type, L::type >::value Iface_conflict< I, LIST... >::value >	1149
cxx::Bits::Bst< Node, Get_key, Lt_functor< typename Get_key::Key_type > >	678
cxx::Avl_tree< Node, Get_key, Compare >	607
L4::lpc::Msg::Cln_val_ops< Detail::Plain< T >::type, DIR, CLASS >	902
L4Re::Util::Counter< unsigned char >	1322
cxx::Auto_ptr< T >	595
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >	614
cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >	621
cxx::Bitfield< T, LSB, MSB >	627
cxx::Bitfield< T, LSB, MSB >::Value_base< TT >	639
cxx::Bitfield< T, LSB, MSB >::Value< TT >	638
cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >	641
cxx::Bitmap_base	646
cxx::Bitmap< BITS >	642
cxx::Bitmap_base::Bit	654
cxx::Bitmap_base::Char< BITS >	655
cxx::Bitmap_base::Word< BITS >	655
cxx::Bits::Avl_map_get_key< KEY_TYPE >	657
cxx::Bits::Avl_set_get_key< KEY_TYPE >	657
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >	658
cxx::Avl_map< Region, Hdlr, cxx::Lt_functor, Alloc >	599
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node	672
cxx::Bits::Basic_list< POLICY >	674
cxx::H_list< Timeout >	699
cxx::H_list< Observer >	699
cxx::H_list< T, Bits::Basic_list_policy< T, H_list_item_t< T > > >	699
cxx::H_list_t< T >	711
cxx::H_list< cxx::Base_slab::Slab_i >	699
cxx::Bits::Bst< Node, Get_key, Compare >	678
cxx::Avl_tree< _Node, Bits::Avl_set_get_key< ITEM_TYPE >, Lt_functor< ITEM_TYPE > >	607
cxx::Avl_tree< Entry, Names_get_key >	607
cxx::Avl_tree< _Node, GET_KEY, COMPARE >	607
cxx::Avl_tree< _Node, Bits::Avl_map_get_key< KEY_TYPE >, Lt_functor< KEY_TYPE > >	607
cxx::Bits::Bst_node	692
cxx::Avl_tree_node	612
cxx::Bits::Direction	694
cxx::Bits::Smart_ptr_list< ITEM >	696
cxx::Bits::Smart_ptr_list_item< T, STORE_T >	698
cxx::Ref_obj_list_item< T >	734
cxx::H_list_item_t< ELEM_TYPE >	708
L4::lpc::svr::Timeout	973
cxx::List< D, Alloc >	714
cxx::List< D, Alloc >::Iter	716
cxx::List_alloc	717
cxx::List_item	720
cxx::List_item::Iter	723
cxx::List_item::T_iter< E >	725
cxx::List_item::T_iter< T, Poly >	725
cxx::Lt_functor< Obj >	728
cxx::New_allocator< _Type >	728
cxx::Nothrow	730
cxx::Pair< First, Second >	730
cxx::Pair_first_compare< Cmp, Typ >	732

cxx::Ref_ptr< T, CNT >	735
cxx::static_vector< T, IDX >	751
cxx::String	752
L4Re::Util::Names::Name	1352
L4virtio::Svr::Device_t< Ds_data >	1605
L4virtio::Svr::Block_dev_base< Ds_data >	1574
L4virtio::Svr::Driver_mem_list_t< Ds_data >	1612
L4virtio::Svr::Driver_mem_region_t< Ds_data >	1622
Elf32_Dyn	764
Elf32_Ehdr	765
Elf32_Phdr	767
Elf32_Shdr	768
Elf32_Sym	769
Elf64_Dyn	770
Elf64_Ehdr	771
Elf64_Phdr	773
Elf64_Shdr	774
Elf64_Sym	775
L4::Epiface_t0< IFACE, BASE >	822
L4::Epiface_t0< void, BASE >	822
L4Re::Event_buffer_t< PAYLOAD >	1257
L4Re::Util::Event_buffer_t< PAYLOAD >	1341
L4Re::Util::Event_buffer_consumer_t< PAYLOAD >	1337
L4::Types::Flags_ops_t< Flags >	1157
L4::Types::Flags_ops_t< Flags_t< DT, T > >	1157
L4::Types::Flags_t< DT, T >	1159
cxx::H_list_item_t< Weak_ref_base >	708
cxx::Weak_ref_base	761
cxx::Weak_ref< T >	759
L4::Kobject_2t< Console, Video::Goos, Event, L4::PROTO_EMPTY >	1018
L4Re::Console	1221
L4::Kobject_2t< Debug_obj_t< BASE >, BASE, Debug_obj, L4::PROTO_EMPTY >	1018
L4::Kobject_x< Iommu, Proto_t< L4_PROTO_IOMMU >, Type_info::Demand_t< 1 > >	1033
L4::Iommu	861
L4::Alloc_list	776
L4::Basic_registry	780
L4Re::Util::Object_registry	1357
L4::Cap_base	790
L4::Cap< L4::Vcon >	785
L4::Cap< L4::Factory >	785
L4::Cap< L4Re::Rm >	785
L4::Cap< L4::Irq >	785
L4::Cap< L4::Thread >	785
L4::Cap< L4Re::Namespace >	785
L4::Cap< void >	785
L4::Cap< L4::Semaphore >	785
L4::Cap< L4Re::Dataspace >	785
L4::Cap< L4virtio::Device >	785
L4::Cap< L4vbus::Vbus >	785
L4::Cap< L4Re::Video::Goos >	785
L4::Cap< T >	785
L4::Smart_cap< T, SMART >	1091
L4::Epiface	815
L4::Epiface_t0< void, Epiface >	822
L4::Irqep_t< Irq_object >	1001

L4::Irqep_t< Derived, BASE, bool >	1001
L4::Epiface_t0< IFACE, L4::Epiface >	822
L4::Epiface_t< Derived, IFACE, BASE, bool >	819
L4::Epiface_t0< RPC_IFACE, BASE >	822
L4::Epiface_t< Block_dev< Ds_data >, L4virtio::Device >	819
L4::Epiface_t< Null_handler, L4::Kobject >	819
L4::Server_object	1079
L4::Server_object_t< Kobject >	1084
L4::Irq_handler_object	994
L4::Server_object_t< IFACE, L4::Server_object >	1084
L4::Server_object_x< Derived, IFACE, BASE >	1088
L4::Server_object_t< IFACE, BASE >	1084
L4::Server_object_x< Derived, IFACE, BASE >	1088
L4::Exception_tracer	825
L4::Base_exception	778
L4::Invalid_capability	855
L4::Runtime_error	1061
L4::Bounds_error	782
L4::Com_error	801
L4::Element_already_exists	811
L4::Element_not_found	813
L4::Out_of_memory	1038
L4::Unknown_error	1167
L4::Factory::Lstr	839
L4::Factory::Nil	840
L4::Factory::S	841
L4::IOModifier	864
L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >	868
L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE >	869
L4::lpc::As_value< T >	871
L4::lpc::Buf_item	872
L4::lpc::Call	873
L4::lpc::Call_t< RIGHTS >	874
L4::lpc::Call_zero_send_timeout	875
L4::lpc::Cap< T >	877
L4::lpc::Gen_fpage< T >	879
L4::lpc::In_out< T >	882
L4::lpc::Istream	892
L4::lpc::Iostream	883
L4::lpc::Msg::Cnt_val_ops< MTYPE, DIR, CLASS >	902
L4::lpc::Msg::Cls_buffer	904
L4::lpc::Msg::Do_rcv_buffers	913
L4::lpc::Msg::Cls_data	905
L4::lpc::Msg::Do_in_data	909
L4::lpc::Msg::Do_out_data	911
L4::lpc::Msg::Cls_item	906
L4::lpc::Msg::Do_in_items	910
L4::lpc::Msg::Do_out_items	912
L4::lpc::Msg::Dir_in	907
L4::lpc::Msg::Do_in_data	909
L4::lpc::Msg::Do_in_items	910
L4::lpc::Msg::Do_rcv_buffers	913
L4::lpc::Msg::Dir_out	908
L4::lpc::Msg::Do_out_data	911
L4::lpc::Msg::Do_out_items	912

L4::lpc::Msg::Elem< Array< A, LEN > & >	915
L4::lpc::Msg::Elem< Array< A, LEN > >	916
L4::lpc::Msg::Elem< Array_ref< A, LEN > & >	917
L4::lpc::Msg::Svr_arg_pack< IPC_TYPE >	920
L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >	920
L4::lpc::Msg_ptr< T >	922
L4::lpc::Opt< T >	923
L4::lpc::Ostream	924
L4::lpc::lostream	883
L4::lpc::Out< T >	930
L4::lpc::Ret_array< T >	931
L4::lpc::Send_only	932
L4::lpc::Small_buf	933
L4::lpc::Snd_item	934
L4::lpc::Str_cp_in< T >	935
L4::lpc::Varg	936
L4::lpc::Varg_list_ref	944
L4::lpc::Varg_list< MAX >	942
L4::lpc::Varg_list_ref::iterator	946
L4::lpc_svr::Compound_reply	953
L4::lpc_svr::Default_loop_hooks	954
L4::Server< L4::lpc_svr::Default_loop_hooks >	1075
L4Re::Util::Registry_server< LOOP_HOOKS >	1365
L4::Server< LOOP_HOOKS >	1075
L4Re::Util::Br_manager_hooks	1316
L4::lpc_svr::Default_setup_wait	956
L4::lpc_svr::Default_timeout	957
L4::lpc_svr::Default_loop_hooks	954
L4Re::Util::Br_manager_hooks	1316
L4::lpc_svr::Direct_dispatch< R >	959
L4::lpc_svr::Exc_dispatch< R, Exc >	962
L4::lpc_svr::Direct_dispatch< R * >	961
L4::lpc_svr::Ignore_errors	963
L4::lpc_svr::Default_loop_hooks	954
L4Re::Util::Br_manager_hooks	1316
L4Re::Util::Br_manager_timeout_hooks	1318
L4::lpc_svr::Server_iface	965
L4::lpc_svr::Timeout_queue_hooks< Br_manager_timeout_hooks, Br_manager >	981
L4Re::Util::Br_manager_timeout_hooks	1318
L4::lpc_svr::Br_manager_no_buffers	950
L4::lpc_svr::Default_loop_hooks	954
L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >	981
L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >	981
L4Re::Util::Br_manager	1311
L4Re::Util::Br_manager_hooks	1316
L4::lpc_svr::Timed_work< HOOKS >	972
L4::lpc_svr::Timeout_queue	976
L4::Kip::Mem_desc	1005
L4::Kobject	1015
L4::Kobject_t< Namespace, L4::Kobject, L4RE_PROTO_NAMESPACE, L4::Type_info::Demand_t< 1 > >	1027
L4Re::Namespace	1279
L4::Kobject_t< Debug_obj, L4::Kobject, L4RE_PROTO_DEBUG >	1027
L4Re::Debug_obj	1233
L4::Kobject_t< Thread, Kobject, L4_PROTO_THREAD, Type_info::Demand_t< 1 > >	1027

L4::Thread	1103
L4::Kobject_t< Rcv_endpoint, Kobject, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > >	1027
L4::Rcv_endpoint	1053
L4::Kobject_2t< Irq, Triggerable, Rcv_endpoint, L4_PROTO_IRQ_SENDER >	1018
L4::Irq	986
L4::Kobject_t< lpc_gate, Rcv_endpoint, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > >	1027
L4::lpc_gate	947
L4::Kobject_t< Goos, L4::Kobject, L4RE_PROTO_GOOS >	1027
L4Re::Video::Goos	1418
L4::Kobject_t< Derived, L4::Kobject, L4::PROTO_ANY, Type_info::Demand_t<> >	1027
L4::Kobject_t< Factory, Kobject, L4_PROTO_FACTORY >	1027
L4::Factory	827
L4::Kobject_t< Mem_alloc, L4::Factory, L4::PROTO_EMPTY >	1027
L4Re::Mem_alloc	1269
L4::Kobject_t< Inhibitor, L4::Kobject, L4RE_PROTO_INHIBITOR >	1027
L4Re::Inhibitor	1261
L4::Kobject_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event >	1022
L4vbus::Vbus	1505
L4::Kobject_t< Dataspace, L4::Kobject, L4RE_PROTO_DATASPACE, L4::Type_info::Demand_t< 1 > >	1027
L4Re::Dataspace	1223
L4::Kobject_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event >	1022
L4::Kobject_t< Platform_control, Kobject, L4_PROTO_PLATFORM_CTL >	1027
L4::Platform_control	1045
L4::Kobject_t< Meta, Kobject, L4_PROTO_META >	1027
L4::Meta	1034
L4::Kobject_t< Debugger, Kobject, L4_PROTO_DEBUGGER >	1027
L4::Debugger	804
L4::Kobject_t< Parent, L4::Kobject, L4RE_PROTO_PARENT >	1027
L4Re::Parent	1287
L4::Kobject_t< Task, Kobject, L4_PROTO_TASK, Type_info::Demand_t< 2 > >	1027
L4::Task	1095
L4::Kobject_t< Vm, Task, L4_PROTO_VM >	1027
L4::Vm	1177
L4::Vm	1177
L4::Kobject_t< Mmio_space, L4::Kobject, L4RE_PROTO_MMIO_SPACE >	1027
L4Re::Mmio_space	1274
L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >	1018
L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >	1022
L4::Kobject_demand< T >	1026
L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >	1027
L4::Kobject_typeid< T >	1029
L4::Kobject_typeid< void >	1032
L4::Kobject_x< Derived, ARGS >	1033
L4::Poll_timeout_kipclock	1049
L4::Proto_t< P >	1052
L4::Registry_iface	1057
L4Re::Util::Object_registry	1357
L4::String	1094
L4::Thread::Attr	1114
L4::Thread::Modify_senders	1118
L4::Type_info	1124
L4::Type_info::Demand	1125
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 >	1128
L4::Type_info::Demand_union_t< Kobject_typeid< T1 >::Demand, Kobject_demand< T2... > >	1130

L4::Type_info::Demand_union_t< D1, D2 >	1130
L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >	1128
L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y >	1134
L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >	1135
L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y >	1136
L4::Typeid::Detail::Rpc_end	1137
L4::Typeid::Detail::_Rpc< OPCODE_TYPE, 0, RPCS... >	1133
L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS >	1145
L4::Typeid::Detail::_Rpc< l4_umword_t, 0, ARG... >	1133
L4::Typeid::Rpc_sys< ARG >	1147
L4::Typeid::Detail::_Rpc< L4::Opcode, 0, RPCS... >	1133
L4::Typeid::Rpc< RPCS >	1142
L4::Typeid::Detail::_Rpc< void, 0, OPERATION >	1133
L4::Typeid::Rpc_nocode< OPERATION >	1139
L4::Typeid::Detail::_Rpc< OPCODE, O, X >	1133
L4::Typeid::P_dispatch< LIST >	1138
L4::Typeid::Raw_ipc< CLASS >	1139
L4::Typeid::Rpc_code< OPCODE_TYPE >	1144
L4::Types::Bool< V >	1149
L4::Types::Flags< BITS_ENUM, UNDERLYING >	1152
L4::Types::Flags_ops_t< DT >	1157
L4::Types::Int_for_size< SIZE, bool >	1161
L4::Types::Int_for_type< T >	1162
l4_buf_regs_t	1180
l4_exc_regs_t	1181
l4_fpage_t	1184
l4_icu_info_t	1184
L4::Icu::Info	854
l4_icu_msi_info_t	1186
l4_kernel_info_mem_desc_t	1187
l4_kernel_info_t	1188
l4_msg_regs_t	1190
l4_msgtag_t	1191
l4_sched_cpu_set_t	1193
l4_sched_param_t	1195
l4_snd_fpage_t	1197
l4_thread_regs_t	1198
l4_timeout_s	1199
l4_timeout_t	1200
l4_vcon_attr_t	1201
l4_vcpu_ipc_regs_t	1202
l4_vcpu_regs_t	1203
l4_vcpu_state_t	1207
L4vcpu::Vcpu	1518
l4_vhw_descriptor	1209
l4_vhw_entry	1211
l4_vm_svm_vmcb_control_area	1212
l4_vm_svm_vmcb_state_save_area	1213
l4_vm_svm_vmcb_state_save_area_seg	1214
l4_vm_svm_vmcb_t	1215
l4_vm_tz_state	1216
L4Re::Cap_alloc	1217
L4Re::Dataspace::F	1231
L4Re::Dataspace::Stats	1233
L4Re::Env	1241
L4Re::Event_buffer_t< PAYLOAD >	1257
L4Re::Event_buffer_t< PAYLOAD >::Event	1260

L4Re::Rm::F	1307
L4Re::Rm::Range	1308
L4Re::Smart_cap_auto< Unmap_flags >	1309
L4Re::Smart_count_cap< Unmap_flags >	1310
L4Re::Util::Cap_alloc_base	1321
L4Re::Util::Counter< COUNTER >	1322
L4Re::Util::Counting_cap_alloc< COUNTERTYPE >	1323
L4Re::Util::Dataspace_svr	1332
L4Re::Util::Event_svr< SVR >	1345
L4Re::Util::Event_t< PAYLOAD >	1347
L4Re::Util::Item_alloc_base	1351
L4Re::Util::Names::Name_space	1353
L4Re::Util::Ref_cap< T >	1363
L4Re::Util::Ref_del_cap< T >	1364
L4Re::Util::Smart_cap_auto< Unmap_flags >	1369
L4Re::Util::Smart_count_cap< Unmap_flags >	1370
L4Re::Util::Vcon_svr< SVR >	1371
L4Re::Util::Video::Goos_svr	1372
L4Re::Vfs::Directory	1383
L4Re::Vfs::File	1388
L4Re::Vfs::Be_file	1375
L4Re::Vfs::File_system	1390
L4Re::Vfs::Be_file_system	1380
L4Re::Vfs::Fs	1392
L4Re::Vfs::Ops	1403
L4Re::Vfs::Generic_file	1397
L4Re::Vfs::File	1388
L4Re::Vfs::Mman	1402
L4Re::Vfs::Ops	1403
L4Re::Vfs::Regular_file	1406
L4Re::Vfs::File	1388
L4Re::Vfs::Special_file	1411
L4Re::Vfs::File	1388
L4Re::Video::Color_component	1414
L4Re::Video::Goos::Info	1424
L4Re::Video::Pixel_info	1425
L4Re::Video::View	1432
L4Re::Video::View::Info	1437
l4re_aux_t	1440
l4re_ds_stats_t	1441
l4re_elf_aux_mword_t	1441
l4re_elf_aux_t	1442
l4re_elf_aux_vma_t	1443
l4re_env_cap_entry_t	1444
l4re_env_t	1446
l4re_event_t	1447
l4re_video_color_component_t	1448
l4re_video_goos_info_t	1449
l4re_video_pixel_info_t	1450
l4re_video_view_info_t	1451
l4re_video_view_t	1453
l4util_idt_desc_t	1454
l4util_idt_header_t	1455
l4util_l4mod_info	1456
l4util_l4mod_mod	1457
l4util_mb_addr_range_t	1458

l4util_mb_apm_t	1459
l4util_mb_drive_t	1460
l4util_mb_info_t	1462
l4util_mb_mod_t	1463
l4util_mb_vbe_ctrl_t	1464
l4util_mb_vbe_mode_t	1465
L4vbus::Gpio_module::Pin_slice	1482
L4vbus::Pm< DEC >	1504
l4vbus_device_t	1514
l4vbus_resource_t	1515
L4vcpu::State	1516
L4virtio::Driver::Block_device::Handle	1545
L4virtio::Driver::Device	1546
L4virtio::Driver::Block_device	1537
L4virtio::Ptr< T >	1568
L4virtio::Svr::Bad_descriptor	1572
L4virtio::Svr::Block_request< Ds_data >	1583
L4virtio::Svr::Data_buffer	1586
L4virtio::Svr::Dev_config	1590
L4virtio::Svr::Dev_features	1601
L4virtio::Svr::Dev_status	1602
L4virtio::Svr::Device_t< DATA >	1605
L4virtio::Svr::Driver_mem_list_t< DATA >	1612
L4virtio::Svr::Driver_mem_region_t< DATA >	1622
L4virtio::Svr::Request_processor	1629
L4virtio::Svr::Virtqueue::Head_desc	1642
L4virtio::Virtqueue	1644
L4virtio::Driver::Virtqueue	1559
L4virtio::Svr::Virtqueue	1636
L4virtio::Virtqueue::Avail	1661
L4virtio::Virtqueue::Avail::Flags	1662
L4virtio::Virtqueue::Desc	1664
L4virtio::Virtqueue::Desc::Flags	1665
L4virtio::Virtqueue::Used	1667
L4virtio::Virtqueue::Used::Flags	1668
L4virtio::Virtqueue::Used_elem	1670
l4virtio_block_config_t	1671
l4virtio_block_discard_t	1672
l4virtio_block_header_t	1673
l4virtio_config_hdr_t	1674
l4virtio_config_queue_t	1675
l4virtio_input_absinfo_t	1677
l4virtio_input_config_t	1677
l4virtio_input_devids_t	1678
l4virtio_input_event_t	1679
cxx::New_allocator< _Node >	728
cxx::New_allocator< Node >	728
cxx::New_allocator< Slab_i >	728
L4vbus::Pm< Device >	1504
L4vbus::Device	1468
L4vbus::Gpio_module	1476
L4vbus::Gpio_pin	1483
L4vbus::Icu	1491
L4vbus::Pci_dev	1493
L4vbus::Pci_host_bridge	1498
L4virtio::Ptr< void >	1568
cxx::Ref_ptr< L4Re::Vfs::File >	735

cxx::Ref_ptr< Mount_tree >	735
L4::lpc::Msg::Svr_val_ops< Array_ref< A, LEN >, Dir_in, CLASS >	920
L4::lpc::Msg::Svr_val_ops< Array_ref< A, LEN >, Dir_out, CLASS >	920
L4::lpc::Msg::Svr_val_ops< L4::lpc::Snd_fpage, Dir_in, CLASS >	920
L4::lpc::Msg::Svr_val_ops< typename ELEM::svr_type, typename Direction< T >::type, typename Class< typename Detail::_Plain< T >::type >::type >	920
cxx::Bitmap_base::Word< Bits >	655
cxx::Bitmap_base::Word< Size >	655

Chapter 11

Data Structure Index

11.1 Data Structures

Here are the data structures with brief descriptions:

cxx::Auto_ptr< T >	
Smart pointer with automatic deletion	595
cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >	
AVL tree based associative container	599
cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >	
AVL set for simple compareable items	604
cxx::Avl_tree< Node, Get_key, Compare >	
A generic AVL tree	607
cxx::Avl_tree_node	
Node of an AVL tree	612
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >	
Basic slab allocator	614
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i	
Type of a slab	620
cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >	
Merged slab allocator (allocators for objects of the same size are merged together)	621
cxx::Bitfield< T, LSB, MSB >	
Definition for a member (part) of a bit field	627
cxx::Bitfield< T, LSB, MSB >::Value< TT >	
Internal helper type	638
cxx::Bitfield< T, LSB, MSB >::Value_base< TT >	
Internal helper type	639
cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >	
Internal helper type	641
cxx::Bitmap< BITS >	
A static bit map	642
cxx::Bitmap_base	
Basic bitmap abstraction	646
cxx::Bitmap_base::Bit	
A writeable bit in a bitmap	654
cxx::Bitmap_base::Char< BITS >	
Helper abstraction for a byte contained in the bitmap	655
cxx::Bitmap_base::Word< BITS >	
Helper abstraction for a word contained in the bitmap	655
cxx::Bits::Avl_map_get_key< KEY_TYPE >	
Key-getter for Avl_map	657

cxx::Bits::Avl_set_get_key< KEY_TYPE >	
Internal, key-getter for Avl_set nodes	657
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >	
Internal: AVL set with internally managed nodes	658
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node	
A smart pointer to a tree item	672
cxx::Bits::Basic_list< POLICY >	
Internal: Common functions for all head-based list implementations	674
cxx::Bits::Bst< Node, Get_key, Compare >	
Basic binary search tree (BST)	678
cxx::Bits::Bst_node	
Basic type of a node in a binary search tree (BST)	692
cxx::Bits::Direction	
The direction to go in a binary search tree	694
cxx::Bits::Smart_ptr_list< ITEM >	
List of smart-pointer-managed objects	696
cxx::Bits::Smart_ptr_list_item< T, STORE_T >	
List item for an arbitrary item in a Smart_ptr_list	698
cxx::H_list< T, POLICY >	
General double-linked list of unspecified cxx::H_list_item elements	699
cxx::H_list_item_t< ELEM_TYPE >	
Basic element type for a double-linked H_list	708
cxx::H_list_t< T >	
Double-linked list of typed H_list_item_t elements	711
cxx::List< D, Alloc >	
Doubly linked list, with internal allocation	714
cxx::List< D, Alloc >::Iter	
Iterator	716
cxx::List_alloc	
Standard list-based allocator	717
cxx::List_item	
Basic list item	720
cxx::List_item::Iter	
Iterator for a list of ListItem -s	723
cxx::List_item::T_iter< T, Poly >	
Iterator for derived classes from ListItem	725
cxx::Lt_func< Obj >	
Generic comparator class that defaults to the less-than operator	728
cxx::New_allocator< _Type >	
Standard allocator based on <code>operator new ()</code>	728
cxx::Nothrow	
Helper type to distinguish the <code>operator new</code> version that does not throw exceptions	730
cxx::Pair< First, Second >	
Pair of two values	730
cxx::Pair_first_compare< Cmp, Typ >	
Comparison functor for Pair	732
cxx::Ref_obj_list_item< T >	
Item for list linked via cxx::Ref_ptr with default reference counting	734
cxx::Ref_ptr< T, CNT >	
A reference-counting pointer with automatic cleanup	735
cxx::S_list< T, POLICY >	
Simple single-linked list	739
cxx::Slab< Type, Slab_size, Max_free, Alloc >	
Slab allocator for object of type <code>Type</code>	743
cxx::Slab_static< Type, Slab_size, Max_free, Alloc >	
Merged slab allocator (allocators for objects of the same size are merged together)	746
cxx::static_vector< T, IDX >	
Simple encapsulation for a dynamically allocated array	751

cxx::String	Allocation free string class with explicit length field	752
cxx::Weak_ref< T >	Typed weak reference to an object of type T	759
cxx::Weak_ref_base	Generic (base) weak reference to some object	761
Elf32_Dyn	ELF32 dynamic entry	764
Elf32_Ehdr	ELF32 header	765
Elf32_Phdr	ELF32 program header	767
Elf32_Shdr	ELF32 section header - figure 1-9, page 1-9	768
Elf32_Sym	ELF32 symbol table entry	769
Elf64_Dyn	ELF64 dynamic entry	770
Elf64_Ehdr	ELF64 header	771
Elf64_Phdr	ELF64 program header	773
Elf64_Shdr	ELF64 section header	774
Elf64_Sym	ELF64 symbol table entry	775
L4::Alloc_list	A simple list-based allocator	776
L4::Arm_smccc	Wrapper for function calls that follow the ARM SMC/HVC calling convention	777
L4::Base_exception	Base class for all exceptions, thrown by the L4Re framework	778
L4::Basic_registry	This registry returns the corresponding server object based on the label of an lpc_gate	780
L4::Bounds_error	Access out of bounds	782
L4::Cap< T >	C++ interface for capabilities	785
L4::Cap_base	Base class for all kinds of capabilities	790
L4::Com_error	Error conditions during IPC	801
L4::Debugger	C++ kernel debugger API	804
L4::Element_already_exists	Exception for duplicate element insertions	811
L4::Element_not_found	Exception for a failed lookup (element not found)	813
L4::Epiface	Base class for interface implementations	815
L4::Epiface_t< Derived, IFACE, BASE, bool >	Epiface implementation for Kobject-based interface implementations	819
L4::Epiface_t0< RPC_IFACE, BASE >	Epiface mixin for generic Kobject-based interfaces	822
L4::Exception	Exception interface	824
L4::Exception_tracer	Back-trace support for exceptions	825

L4::Factory	
C++ Factory interface	827
L4::Factory::Lstr	
Special type to add a pascal string into the factory create stream	839
L4::Factory::Nil	
Special type to add a void argument into the factory create stream	840
L4::Factory::S	
Stream class for the create() argument stream	841
L4::lcu	
C++ lcu interface	846
L4::lcu::Info	
This class encapsulates information about an ICU	854
L4::Invalid_capability	
Indicates that an invalid object was invoked	855
L4::io_pager	
io_pager interface	859
L4::lomu	
Interface for IO-MMUs used for DMA remapping	861
L4::IOModifier	
Modifier class for the IO stream	864
L4::lpc::Array< ELEM_TYPE, LEN_TYPE >	
Array data type for dynamically sized arrays in RPCs	865
L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >	
Server-side copy in buffer for Array	868
L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE >	
Array reference data type for arrays located in the message	869
L4::lpc::As_value< T >	
Pass the argument as plain data value	871
L4::lpc::Buf_item	
RPC warpper for a receive item	872
L4::lpc::Call	
RPC attribute for a standard RPC call	873
L4::lpc::Call_t< RIGHTS >	
RPC attribute for an RPC call with required rights	874
L4::lpc::Call_zero_send_timeout	
RPC attribute for an RPC call, with zero send timeout	875
L4::lpc::Cap< T >	
Capability type for RPC interfaces (see L4::Cap<T>)	877
L4::lpc::Gen_fpage< T >	
Generic RPC wrapper for L4 flex-pages	879
L4::lpc::In_out< T >	
Mark an argument as in-out argument	882
L4::lpc::lostream	
Input/Output stream for IPC [un]marshalling	883
L4::lpc::lstream	
Input stream for IPC unmarshalling	892
L4::lpc::Msg::Clnt_val_ops< MTYPE, DIR, CLASS >	
Defines client-side handling of 'MTYPE' as RPC argument	902
L4::lpc::Msg::Cls_buffer	
Marker type for receive buffer values	904
L4::lpc::Msg::Cls_data	
Marker type for data values	905
L4::lpc::Msg::Cls_item	
Marker type for item values	906
L4::lpc::Msg::Dir_in	
Marker type for input values	907
L4::lpc::Msg::Dir_out	
Marker type for output values	908

L4::lpc::Msg::Do_in_data	
Marker for Input data	909
L4::lpc::Msg::Do_in_items	
Marker for Input items	910
L4::lpc::Msg::Do_out_data	
Marker for Output data	911
L4::lpc::Msg::Do_out_items	
Marker for Output items	912
L4::lpc::Msg::Do_rcv_buffers	
Marker for receive buffers	913
L4::lpc::Msg::Elem< Array< A, LEN > & >	
Array as output argument	915
L4::lpc::Msg::Elem< Array< A, LEN > >	
Array as input arguments	916
L4::lpc::Msg::Elem< Array_ref< A, LEN > & >	
Array_ref as output argument	917
L4::lpc::Msg::Is_valid_rpc_type< T >	
Type trait defining a valid RPC parameter type	918
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	920
L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >	
Defines server-side handling for MTYPE server arguments	920
L4::lpc::Msg_ptr< T >	
Pointer to an element of type T in an lpc::Istream	922
L4::lpc::Opt< T >	
Attribute for defining an optional RPC argument	923
L4::lpc::Ostream	
Output stream for IPC marshalling	924
L4::lpc::Out< T >	
Mark an argument as a output value in an RPC signature	930
L4::lpc::Ret_array< T >	
Dynamically sized output array of type T	931
L4::lpc::Send_only	
RPC attribute for a send-only RPC	932
L4::lpc::Small_buf	
A receive item for receiving a single capability	933
L4::lpc::Snd_item	
RPC wrapper for a send item	934
L4::lpc::Str_cp_in< T >	
Abstraction for extracting a zero-terminated string from an lpc::Istream	935
L4::lpc::Varg	
Variably sized RPC argument	936
L4::lpc::Varg_list< MAX >	
Self-contained list of variable-sized RPC parameters	942
L4::lpc::Varg_list_ref	
List of variable-sized RPC parameters as received by the server	944
L4::lpc::Varg_list_ref::Iterator	
Iterator for Valists	946
L4::lpc_gate	
The C++ IPC gate interface	947
L4::lpc_svr::Br_manager_no_buffers	
Empty implementation of Server_iface	950
L4::lpc_svr::Compound_reply	
Mix in for LOOP_HOOKS to always use compound reply and wait	953
L4::lpc_svr::Default_loop_hooks	
Default LOOP_HOOKS	954

L4::lpc_svr::Default_setup_wait	
Mix in for LOOP_HOOKS for setup_wait no op	956
L4::lpc_svr::Default_timeout	
Mix in for LOOP_HOOKS to use a 0 send and a infinite receive timeout	957
L4::lpc_svr::Direct_dispatch< R >	
Direct dispatch helper, for forwarding dispatch calls to a registry <i>R</i>	959
L4::lpc_svr::Direct_dispatch< R * >	
Direct dispatch helper, for forwarding dispatch calls via a pointer to a registry <i>R</i>	961
L4::lpc_svr::Exc_dispatch< R, Exc >	
Dispatch helper wrapping try {} catch {} around the dispatch call	962
L4::lpc_svr::Ignore_errors	
Mix in for LOOP_HOOKS to ignore IPC errors	963
L4::lpc_svr::Server_iface	
Interface for server-loop related functions	965
L4::lpc_svr::Timed_work< HOOKS >	
DEPRECATED	972
L4::lpc_svr::Timeout	
Callback interface for Timeout_queue	973
L4::lpc_svr::Timeout_queue	
Timeout queue to be used in L4re server loop	976
L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >	
Loop hooks mixin for integrating a timeout queue into the server loop	981
L4::lrq	
C++ lrq interface	986
L4::lrq_eoi	
Interface for sending an acknowledge message to an object	992
L4::lrq_handler_object	
Server object base class for handling IRQ messages	994
L4::lrq_mux	
IRQ multiplexer for shared IRQs	997
L4::lrqep_t< Derived, BASE, bool >	
Epiface implementation for interrupt handlers	1001
L4::Kip::Mem_desc	
Memory descriptors stored in the kernel interface page	1005
L4::Kobject	
Base class for all kinds of kernel objects and remote objects, referenced by capabilities	1015
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)	1018
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)	1022
L4::Kobject_demand< T >	
Get the combined server-side resource requirements for all type <i>T</i>	1026
L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >	
Helper class to create an L4Re interface class that is derived from a single base class	1027
L4::Kobject_typeid< T >	
Meta object for handling access to type information of Kobjects	1029
L4::Kobject_typeid< void >	
Minimalistic ID for <code>void</code> interface	1032
L4::Kobject_x< Derived, ARGS >	
Generic Kobject inheritance template	1033
L4::Meta	
Meta interface that shall be implemented by each L4Re object and gives access to the dynamic type information for L4Re objects	1034
L4::Out_of_memory	
Exception signalling insufficient memory	1038

L4::Pager	
Pager interface including the lo_pager interface	1041
L4::Platform_control	
L4 C++ interface for controlling platform-wide properties	1045
L4::Poll_timeout_kipclock	
A polling timeout based on the L4Re clock	1049
L4::Proto_t< P >	
Data type for defining protocol numbers	1052
L4::Rcv_endpoint	
Interface for kernel objects that allow to receive IPC from them	1053
L4::Registry_iface	
Abstract interface for object registries	1057
L4::Runtime_error	
Exception for an abstract runtime error	1061
L4::Scheduler	
C++ interface of the Scheduler kernel object	1065
L4::Semaphore	
Kernel-provided semaphore object	1070
L4::Server< LOOP_HOOKS >	
Basic server loop for handling client requests	1075
L4::Server_object	
Abstract server object to be used with L4::Server and L4::Basic_registry	1079
L4::Server_object_t< IFACE, BASE >	
Base class (template) for server implementing server objects	1084
L4::Server_object_x< Derived, IFACE, BASE >	
Helper class to implement p_dispatch based server objects	1088
L4::Smart_cap< T, SMART >	
Smart capability class	1091
L4::String	
A null-terminated string container class	1094
L4::Task	
C++ interface of the Task kernel object	1095
L4::Thread	
C++ L4 kernel thread interface	1103
L4::Thread::Attr	
Thread attributes used for control_commit()	1114
L4::Thread::Modify_senders	
Wrapper class for modifying senders	1118
L4::Triggerable	
Interface that allows an object to be triggered by some source	1120
L4::Type_info	
Dynamic Type Information for L4Re Interfaces	1124
L4::Type_info::Demand	
Data type for expressing the needed receive buffers at the server-side of an interface	1125
L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >	
Template type statically describing demand of receive buffers	1128
L4::Type_info::Demand_union_t< D1, D2 >	
Template type statically describing the combination of two Demand object	1130
L4::Typeid::Detail::_Rpc< OPCODE, O, X >	
Empty list of RPCs	1133
L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y >	
Find the given RPC in the list	1134
L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >	
Non-empty list of RPCs	1135
L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y >	
Find the given RPC in the list	1136
L4::Typeid::Detail::Rpc_end	
Internal end-of-list marker	1137

L4::Typeid::P_dispatch< LIST >	Use for protocol based dispatch stage	1138
L4::Typeid::Raw_ipc< CLASS >	RPCs list for passing raw incoming IPC to the server object	1139
L4::Typeid::Rpc_nocode< OPERATION >	List of RPCs of an interface using a single operation without an opcode	1139
L4::Typeid::Rpcs< RPCS >	Standard list of RPCs of an interface	1142
L4::Typeid::Rpcs_code< OPCODE_TYPE >	List of RPCs of an interface using a special opcode type	1144
L4::Typeid::Rpcs_code< OPCODE_TYPE >::F< RPCS >	1145
L4::Typeid::Rpcs_sys< ARG >	List of RPCs typically used for kernel interfaces	1147
L4::Types::Bool< V >	Boolean meta type	1149
L4::Types::False	False meta value	1151
L4::Types::Flags< BITS_ENUM, UNDERLYING >	Template for defining typical Flags bitmaps	1152
L4::Types::Flags_ops_t< DT >	Mixin class to define a set of friend bitwise operators on DT	1157
L4::Types::Flags_t< DT, T >	Template type to define a flags type with bitwise operations	1159
L4::Types::Int_for_size< SIZE, bool >	Metafunction to get an unsigned integral type for the given size	1161
L4::Types::Int_for_type< T >	Metafunction to get an integral type of the same size as T	1162
L4::Types::Same< A, B >	Compare two data types for equality	1163
L4::Types::True	True meta value	1165
L4::Unknown_error	Exception for an unknown condition	1167
L4::Vcon	C++ L4 Vcon interface	1170
L4::Vm	Virtual machine host address space	1177
l4_buf_regs_t	Encapsulation of the buffer-registers block in the UTCB	1180
l4_exc_regs_t	UTCB structure for exceptions	1181
l4_fpage_t	L4 flexpage type	1184
l4_icu_info_t	Info structure for an ICU	1184
l4_icu_msi_info_t	Info to use for a specific MSI	1186
l4_kernel_info_mem_desc_t	Memory descriptor data structure	1187
l4_kernel_info_t	L4 Kernel Interface Page	1188
l4_msg_regs_t	Encapsulation of the message-register block in the UTCB	1190
l4_msgtag_t	Message tag data structure	1191
l4_sched_cpu_set_t	CPU sets	1193

l4_sched_param_t	Scheduler parameter set	1195
l4_snd_fpage_t	Send-flex-page types	1197
l4_thread_regs_t	Encapsulation of the thread-control-register block of the UTCB	1198
l4_timeout_s	Basic timeout specification	1199
l4_timeout_t	Timeout pair	1200
l4_vcon_attr_t	Vcon attribute structure	1201
l4_vcpu_ipc_regs_t	VCPU message registers	1202
l4_vcpu_regs_t	VCPU registers	1203
l4_vcpu_state_t	State of a vCPU	1207
l4_vhw_descriptor	Virtual hardware devices description	1209
l4_vhw_entry	Description of a device	1211
l4_vm_svm_vmcb_control_area	VMCB structure for SVM VMs	1212
l4_vm_svm_vmcb_state_save_area	State save area structure for SVM VMs	1213
l4_vm_svm_vmcb_state_save_area_seg	State save area segment selector struct	1214
l4_vm_svm_vmcb_t	Control structure for SVM VMs	1215
l4_vm_tz_state	State structure for TrustZone VMs	1216
L4Re::Cap_alloc	Capability allocator interface	1217
L4Re::Console	Console class	1221
L4Re::Dataspace	Interface for memory-like objects	1223
L4Re::Dataspace::F	Dataspace flags definitions	1231
L4Re::Dataspace::Stats	Information about the dataspace	1233
L4Re::Debug_obj	Debug interface	1233
L4Re::Dma_space	DMA Address Space	1236
L4Re::Env	C++ interface of the initial environment that is provided to an L4 task	1241
L4Re::Event	Event class	1254
L4Re::Event_buffer_t< PAYLOAD >	Event buffer class	1257
L4Re::Event_buffer_t< PAYLOAD >::Event	Event structure used in buffer	1260
L4Re::Inhibitor	Set of inhibitor locks, which inhibit specific actions when held	1261
L4Re::Log	Log interface class	1266

L4Re::Mem_alloc	
Memory allocation interface	1269
L4Re::Mmio_space	
Interface for memory-like address space accessible via IPC	1274
L4Re::Namespace	
Name-space interface	1279
L4Re::Ned::Cmd_control	
Direct control interface for Ned	1285
L4Re::Parent	
Parent interface	1287
L4Re::Random	
Low-bandwidth interface for random number generators	1290
L4Re::Rm	
Region map	1294
L4Re::Rm::F	
Rm flags definitions	1307
L4Re::Rm::Range	
A range of virtual addresses	1308
L4Re::Smart_cap_auto< Unmap_flags >	
Helper for Unique_cap and Unique_del_cap	1309
L4Re::Smart_count_cap< Unmap_flags >	
Helper for Ref_cap and Ref_del_cap	1310
L4Re::Util::Br_manager	
Buffer-register (BR) manager for L4::Server	1311
L4Re::Util::Br_manager_hooks	
Predefined server-loop hooks for a server loop using the Br_manager	1316
L4Re::Util::Br_manager_timeout_hooks	
Predefined server-loop hooks for a server with using the Br_manager and a timeout queue	1318
L4Re::Util::Cap_alloc_base	
Capability allocator	1321
L4Re::Util::Counter< COUNTER >	
Counter for Counting_cap_alloc with variable data width	1322
L4Re::Util::Counting_cap_alloc< COUNTERTYPE >	
Internal reference-counting cap allocator	1323
L4Re::Util::Dataspace_svr	
Dataspace server class	1332
L4Re::Util::Event_buffer_consumer_t< PAYLOAD >	
An event buffer consumer	1337
L4Re::Util::Event_buffer_t< PAYLOAD >	
Event_buffer utility class	1341
L4Re::Util::Event_svr< SVR >	
Convenience wrapper for implementing an event server	1345
L4Re::Util::Event_t< PAYLOAD >	
Convenience wrapper for getting access to an event object	1347
L4Re::Util::Item_alloc_base	
Item allocator	1351
L4Re::Util::Names::Name	
Name class	1352
L4Re::Util::Names::Name_space	
Abstract server-side implementation of the L4::Namespace interface	1353
L4Re::Util::Object_registry	
A registry that manages server objects and their attached IPC gates for a single server loop for a specific thread	1357
L4Re::Util::Ref_cap< T >	
Automatic capability that implements automatic free and unmap of the capability selector	1363
L4Re::Util::Ref_del_cap< T >	
Automatic capability that implements automatic free and unmap+delete of the capability selector	1364

L4Re::Util::Registry_server< LOOP_HOOKS >	
A server loop object which has a Object_registry included	1365
L4Re::Util::Smart_cap_auto< Unmap_flags >	
Helper for Unique_cap and Unique_del_cap	1369
L4Re::Util::Smart_count_cap< Unmap_flags >	
Helper for Ref_cap and Ref_del_cap	1370
L4Re::Util::Vcon_svr< SVR >	
Console server template class	1371
L4Re::Util::Video::Goos_svr	
Goos server class	1372
L4Re::Vfs::Be_file	
Boiler plate class for implementing an open file for L4Re::Vfs	1375
L4Re::Vfs::Be_file_system	
Boilerplate class for implementing a L4Re::Vfs::File_system	1380
L4Re::Vfs::Directory	
Interface for a POSIX file that is a directory	1383
L4Re::Vfs::File	
The basic interface for an open POSIX file	1388
L4Re::Vfs::File_system	
Basic interface for an L4Re::Vfs file system	1390
L4Re::Vfs::Fs	
POSIX File-system related functionality	1392
L4Re::Vfs::Generic_file	
The common interface for an open POSIX file	1397
L4Re::Vfs::Mman	
Interface for the POSIX memory management	1402
L4Re::Vfs::Ops	
Interface for the POSIX backends for an application	1403
L4Re::Vfs::Regular_file	
Interface for a POSIX file that provides regular file semantics	1406
L4Re::Vfs::Special_file	
Interface for a POSIX file that provides special file semantics	1411
L4Re::Video::Color_component	
A color component	1414
L4Re::Video::Goos	
A goos	1418
L4Re::Video::Goos::Info	
Information structure of a goos	1424
L4Re::Video::Pixel_info	
Pixel information	1425
L4Re::Video::View	
View	1432
L4Re::Video::View::Info	
Information structure of a view	1437
l4re_aux_t	
Auxiliary descriptor	1440
l4re_ds_stats_t	
Information about the data space	1441
l4re_elf_aux_mword_t	
Auxiliary vector element for a single unsigned data word	1441
l4re_elf_aux_t	
Generic header for each auxiliary vector element	1442
l4re_elf_aux_vma_t	
Auxiliary vector element for a reserved virtual memory area	1443
l4re_env_cap_entry_t	
Entry in the L4Re environment array for the named initial objects	1444
l4re_env_t	
Initial environment data structure	1446

l4re_event_t	Event structure used in buffer	1447
l4re_video_color_component_t	Color component structure	1448
l4re_video_goos_info_t	Goos information structure	1449
l4re_video_pixel_info_t	Pixel_info structure	1450
l4re_video_view_info_t	View information structure	1451
l4re_video_view_t	C representation of a goos view	1453
l4util_idt_desc_t	IDT entry	1454
l4util_idt_header_t	Header of an IDT table	1455
l4util_l4mod_info	Base module structure	1456
l4util_l4mod_mod	A single module	1457
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	1458
l4util_mb_apm_t	APM BIOS info	1459
l4util_mb_drive_t	Drive Info structure	1460
l4util_mb_info_t	MultiBoot Info description	1462
l4util_mb_mod_t	The structure type "mod_list" is used by the multiboot_info structure	1463
l4util_mb_vbe_ctrl_t	VBE controller information	1464
l4util_mb_vbe_mode_t	VBE mode information	1465
L4vbus::Device	Device on a L4vbus::Vbus	1468
L4vbus::Gpio_module	A Gpio_module groups multiple GPIO pins together	1476
L4vbus::Gpio_module::Pin_slice	A slice of the pins provided by this module	1482
L4vbus::Gpio_pin	A GPIO pin	1483
L4vbus::Icu	Vbus Interrupt controller API	1491
L4vbus::Pci_dev	A PCI device	1493
L4vbus::Pci_host_bridge	A Pci host bridge	1498
L4vbus::Pm< DEC >	Power-management API mixin	1504
L4vbus::Vbus	The virtual bus (Vbus) interface	1505
l4vbus_device_t	Detailed information about a vbus device	1514
l4vbus_resource_t	Description of a single vbus resource	1515

L4vcpu::State	
C++ implementation of state word in the vCPU area	1516
L4vcpu::Vcpu	
C++ implementation of the vCPU save state area	1518
L4virtio::Device	
IPC interface for virtio over L4 IPC	1530
L4virtio::Driver::Block_device	
Simple class for accessing a virtio block device synchronously	1537
L4virtio::Driver::Block_device::Handle	
Handle to an ongoing request	1545
L4virtio::Driver::Device	
Client-side implementation for a general virtio device	1546
L4virtio::Driver::Virtqueue	
Driver-side implementation of a Virtqueue	1559
L4virtio::Ptr< T >	
Pointer used in virtio descriptors	1568
L4virtio::Svr::Bad_descriptor	
Exception used by Queue to indicate descriptor errors	1572
L4virtio::Svr::Block_dev_base< Ds_data >	
Base class for virtio block devices	1574
L4virtio::Svr::Block_request< Ds_data >	
A request to read or write data	1583
L4virtio::Svr::Data_buffer	
Abstract data buffer	1586
L4virtio::Svr::Dev_config	
Abstraction for L4-Virtio device config memory	1590
L4virtio::Svr::Dev_features	
Type for device feature bitmap	1601
L4virtio::Svr::Dev_status	
Type of the device status register	1602
L4virtio::Svr::Device_t< DATA >	
Server-side L4-VIRTIO device stub	1605
L4virtio::Svr::Driver_mem_list_t< DATA >	
List of driver memory regions assigned to a single L4-VIRTIO transport instance	1612
L4virtio::Svr::Driver_mem_region_t< DATA >	
Region of driver memory, that shall be managed locally	1622
L4virtio::Svr::Request_processor	
Encapsulate the state for processing a VIRTIO request	1629
L4virtio::Svr::Virtqueue	
Virtqueue implementation for the device	1636
L4virtio::Svr::Virtqueue::Head_desc	
VIRTIO request, essentially a descriptor from the available ring	1642
L4virtio::Virtqueue	
Low-level Virtqueue	1644
L4virtio::Virtqueue::Avail	
Type of available ring, this is read-only for the host	1661
L4virtio::Virtqueue::Avail::Flags	
Flags of the available ring	1662
L4virtio::Virtqueue::Desc	
Descriptor in the descriptor table	1664
L4virtio::Virtqueue::Desc::Flags	
Type for descriptor flags	1665
L4virtio::Virtqueue::Used	
Used ring	1667
L4virtio::Virtqueue::Used::Flags	
Flags for the used ring	1668
L4virtio::Virtqueue::Used_elem	
Type of an element of the used ring	1670

l4virtio_block_config_t	Device configuration for block devices	1671
l4virtio_block_discard_t	Structure used for the write zeroes and discard commands	1672
l4virtio_block_header_t	Header structure of a request for a block device	1673
l4virtio_config_hdr_t	L4-VIRTIO config header, provided in shared data space	1674
l4virtio_config_queue_t	Queue configuration entry	1675
l4virtio_input_absinfo_t	Information about the absolute axis in the underlying evdev implementation	1677
l4virtio_input_config_t	Device configuration for input devices	1677
l4virtio_input_devids_t	Device ID information for the device	1678
l4virtio_input_event_t	Single event in event or status queue	1679

Chapter 12

File Index

12.1 File List

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

pkg/l4re-core/ned/lib/include/ cmd_control	??
pkg/l4re-core/ned/lib/include/ Makefile	??
amd64/l4/sys/ __kip-arch.h	??
amd64/l4/sys/ __vcpu-arch.h	??
amd64/l4/sys/ cache.h	
Cache functions	2039
amd64/l4/sys/ consts.h	
Common L4 constants, amd64 version	2055
amd64/l4/sys/ ktrace_events.h	??
amd64/l4/sys/ l4int.h	
Fixed sized integer types, amd64 version	2148
amd64/l4/sys/ linkage.h	
Linkage	1764
amd64/l4/sys/ segment.h	
Segment handling	1740
amd64/l4/sys/ utcb.h	
UTCB definitions for amd64	2207
amd64/l4/sys/ vm.h	??
amd64/l4/util/ apic.h	
APIC for AMD64	1681
amd64/l4/util/ bitops_arch.h	
Amd64 bit manipulation functions	1770
amd64/l4/util/ cpu.h	
CPU related functions	1778
amd64/l4/util/ idt.h	
IDT related functions	1692
amd64/l4/util/ irq.h	
Some PIC and hardware interrupt related functions	1868
amd64/l4/util/ l4_macros.h	
Main function	1786
amd64/l4/util/ mbi_argv.h	
Command line handling	1790
amd64/l4/util/ perform.h	
Performance Monitoring using P5/P6 Measurement Counters	1695
amd64/l4/util/ port_io.h	
Port I/O functions	1750

amd64/l4/util/ rdtsc.h	
Time stamp counter related functions	1707
amd64/l4/util/ spin.h	
Spinning for amd64	1729
amd64/l4/util/ util.h	
Utilities, amd64 version	1731
amd64/l4f/l4/sys/ ipc.h	??
amd64/l4f/l4/sys/ segment.h	
L4f specific fs/gs manipulation	1737
amd64/l4f/l4/util/ port_io.h	
Port I/O functions	1749
arm/l4/sys/ __kip-arch.h	??
arm/l4/sys/ __vcpu-arch.h	??
arm/l4/sys/ atomic.h	
Atomic memory modifications	2236
arm/l4/sys/ cache.h	
Cache functions	2037
arm/l4/sys/ consts.h	
Common L4 constants, arm version	2053
arm/l4/sys/ ktrace_events.h	??
arm/l4/sys/ l4int.h	
Fixed sized integer types, arm version	2148
arm/l4/sys/ linkage.h	
Linkage	1763
arm/l4/sys/ mem_op.h	
Memory access functions (ARM specific)	1766
arm/l4/sys/ task.h	??
arm/l4/sys/ thread.h	
ARM-specific thread related definitions	2329
arm/l4/sys/ utcb.h	
UTCB definitions for ARM	2204
arm/l4/sys/ vm	??
arm/l4/sys/ vm.h	
ARM virtualization interface	1768
arm/l4/util/ bitops_arch.h	
ARM specific implementation of bitops functions	1770
arm/l4/util/ cpu.h	
CPU related functions	1777
arm/l4/util/ irq.h	
ARM specific implementation of irq functions	1867
arm/l4/util/ l4_macros.h	
Main function	1786
arm/l4/util/ mbi_argv.h	
Multiboot	1789
arm/l4f/l4/sys/ ipc.h	
L4 IPC System Calls, ARM	2128
arm/l4f/l4/sys/ syscall_defs.h	
Syscall entry definitions	1792
contrib/libio-io/l4/io/ io.h	1793
contrib/libio-io/l4/io/ types.h	??
l4/cxx/ alloc.h	
Alloc list	1797
l4/cxx/ arith	??
l4/cxx/ atomic.h	
Atomic template	2237
l4/cxx/ auto_ptr	??
l4/cxx/ avl_map	
AVL map	1798

l4/cxx/ avl_set	
AVL set	1801
l4/cxx/ avl_tree	
AVL tree	1806
l4/cxx/ basic_ostream	
Basic IO stream	1812
l4/cxx/ basic_vector.h	
Basic vector	1816
l4/cxx/ bitfield	??
l4/cxx/ bitmap	??
l4/cxx/ dlist	??
l4/cxx/ exceptions	
Base exceptions	1827
l4/cxx/ hlist	??
l4/cxx/ iostream	
IO Stream	1831
l4/cxx/ ipc_helper	
IPC helper	1833
l4/cxx/ ipc_server	
IPC server loop	2074
l4/cxx/ ipc_stream	
IPC stream	1835
l4/cxx/ ipc_timeout_queue	??
l4/cxx/ l4iostream	
L4 IO stream	1854
l4/cxx/ l4types.h	
L4 Types	1856
l4/cxx/ list	??
l4/cxx/ list_alloc	??
l4/cxx/ main_thread	
Main thread	1857
l4/cxx/ minmax	??
l4/cxx/ observer	??
l4/cxx/ pair	
Pair implementation	1859
l4/cxx/ ref_ptr	??
l4/cxx/ ref_ptr_list	??
l4/cxx/ slab_alloc	??
l4/cxx/ slist	??
l4/cxx/ static_container	??
l4/cxx/ static_vector	??
l4/cxx/ std_alloc	??
l4/cxx/ std_exc_io	
Base exceptions std stream operator	1860
l4/cxx/ std_ops	??
l4/cxx/ string	??
l4/cxx/ string.h	
String	1862
l4/cxx/ thread	
Thread implementation	2190
l4/cxx/ type_list	??
l4/cxx/ type_traits	??
l4/cxx/ unique_ptr	??
l4/cxx/ unique_ptr_list	??
l4/cxx/ utils	??
l4/cxx/ weak_ref	??
l4/cxx/bits/ bst.h	
AVL tree	1817

l4/cxx/bits/bst_base.h	
AVL tree	1821
l4/cxx/bits/bst_iter.h	
AVL tree	1824
l4/cxx/bits/list_basics.h	??
l4/cxx/bits/smart_ptr_list.h	??
l4/cxx/bits/type_traits.h	??
l4/irq/irq.h	
IRQ handling routines	1864
l4/l4re_vfs/backend	??
l4/l4re_vfs/vfs.h	??
l4/l4re_vfs/impl/default_ops_impl.h	??
l4/l4re_vfs/impl/fd_store.h	??
l4/l4re_vfs/impl/fd_store_impl.h	??
l4/l4re_vfs/impl/ns_fs.h	??
l4/l4re_vfs/impl/ns_fs_impl.h	??
l4/l4re_vfs/impl/ro_file.h	??
l4/l4re_vfs/impl/ro_file_impl.h	??
l4/l4re_vfs/impl/vcon_stream.h	??
l4/l4re_vfs/impl/vcon_stream_impl.h	??
l4/l4re_vfs/impl/vfs_impl.h	??
l4/l4virtio/l4virtio	??
l4/l4virtio/virtio.h	??
l4/l4virtio/virtio_block.h	??
l4/l4virtio/virtio_input.h	??
l4/l4virtio/virtqueue	??
l4/l4virtio/client/virtio-block	??
l4/l4virtio/server/l4virtio	??
l4/l4virtio/server/virtio	??
l4/l4virtio/server/virtio-block	??
l4/libedid/edid.h	1877
l4/re/cap_alloc	
Abstract capability-allocator interface	1914
l4/re/console	??
l4/re/consts	
Constants	2058
l4/re/consts.h	
Constants	2056
l4/re/dataspace	
Dataspace interface	1920
l4/re/dataspace-sys.h	
Dataspace protocol definition	1923
l4/re/debug	
Debug interface	1924
l4/re/dma_space	1926
l4/re/elf_aux.h	
Auxiliary information for binaries	1929
l4/re/env	
Environment interface	1932
l4/re/env.h	
Environment interface	1935
l4/re/error_helper	
Error helper	1938
l4/re/event	??
l4/re/event-sys.h	??
l4/re/event.h	
Events	1888
l4/re/event_enums.h	??

l4/re/inhibitor	??
l4/re/inhibitor-sys.h	??
l4/re/l4aux.h	
Auxiliary definitions	1952
l4/re/log	
Log interface	1954
l4/re/log-sys.h	
Log protocol definition	1956
l4/re/mem_alloc	
Memory allocator interface	1957
l4/re/mem_alloc-sys.h	
Memory allocator protocol definitions	1959
l4/re/mmio_space	??
l4/re/namespace	
Namespace interface	1960
l4/re/namespace-sys.h	
Namespace protocol definitions	1963
l4/re/parent	
Parent interface	1964
l4/re/parent-sys.h	
Parent protocol definition	1966
l4/re/protocols.h	
L4Re Protocol Constants (C version)	1967
l4/re/random	??
l4/re/rm	
Region mapper interface	1969
l4/re/rm-sys.h	
Region mapper protocol definitions	1974
l4/re/shared_cap	
Shared_cap / Shared_del_cap	1975
l4/re/unique_cap	
Unique_cap / Unique_del_cap	1980
l4/re/c/dataspace.h	
Data space C interface	1879
l4/re/c/debug.h	
Debug C interface	1881
l4/re/c/dma_space.h	
DMA space C interface	1883
l4/re/c/event.h	
Event C interface	1886
l4/re/c/event_buffer.h	??
l4/re/c/inhibitor.h	??
l4/re/c/log.h	
Log C interface	1890
l4/re/c/mem_alloc.h	
Memory allocator C interface	1891
l4/re/c/namespace.h	
Namespace functions, C interface	1894
l4/re/c/rm.h	
Region map interface, C interface	1896
l4/re/c/util/cap_alloc.h	
Capability allocator C interface	1900
l4/re/c/util/kumem_alloc.h	
Kumem allocator utility C interface	1901
l4/re/c/util/video/goos_fb.h	
Framebuffer utility functionality	1903
l4/re/c/video/colors.h	1905
l4/re/c/video/goos.h	1907

l4/re/c/video/view.h	1910
l4/re/impl/dataspace_impl.h	
Dataspace client stub implementation	1944
l4/re/impl/mem_alloc_impl.h	
Memory allocator client stub implementation	1946
l4/re/impl/namespace_impl.h	
Namespace client stub implementation	1948
l4/re/impl/rm_impl.h	
Region map client stub implementation	1950
l4/re/util/bitmap_cap_alloc	
Bitmap capability allocator	1985
l4/re/util/br_manager	??
l4/re/util/cap	
Capability utility functions	1987
l4/re/util/cap_alloc	
Capability allocator	1917
l4/re/util/cap_alloc_impl.h	
Capability allocator implementation	1989
l4/re/util/counting_cap_alloc	
Reference-counting capability allocator	1991
l4/re/util/dataspace_svr	??
l4/re/util/debug	??
l4/re/util/env_ns	??
l4/re/util/event	1941
l4/re/util/event_buffer	??
l4/re/util/event_svr	??
l4/re/util/icu_svr	??
l4/re/util/item_alloc	
Item allocator	1994
l4/re/util/kumem_alloc	
Kumem allocator helper	1996
l4/re/util/meta	??
l4/re/util/name_space_svr	??
l4/re/util/object_registry	??
l4/re/util/poll_timeout_kipclock	??
l4/re/util/region_mapping	
Region handling	1998
l4/re/util/region_mapping_svr_2	??
l4/re/util/shared_cap	
Shared_cap / Shared_del_cap	1977
l4/re/util/unique_cap	
Unique_cap / Unique_del_cap	1982
l4/re/util/vcon_svr	??
l4/re/util/video/goos_fb	??
l4/re/util/video/goos_svr	??
l4/re/video/colors	??
l4/re/video/goos	??
l4/re/video/goos-sys.h	
Goos protocol definition	2004
l4/re/video/view	??
l4/shmc/shmc.h	
Shared memory library header file	2005
l4/sigma0/sigma0.h	
Sigma0 interface	2010
l4/sys/__kernel_object_impl.h	
Low-level kernel debugger functions	2017
l4/sys/__kip-32bit.h	??
l4/sys/__kip-64bit.h	??

l4/sys/___ktrace-impl.h	
L4 kernel event tracing	2018
l4/sys/___l4_fpage.h	??
l4/sys/___task-arm.h	??
l4/sys/___timeout.h	??
l4/sys/___typeinfo.h	
Type information handling	2020
l4/sys/___vcpu-arm.h	??
l4/sys/___vm-arm.h	
Virtualization interface	2033
l4/sys/___vm-svm.h	??
l4/sys/___vm-vmx.h	??
l4/sys/arm_smccc	??
l4/sys/arm_smccc.h	??
l4/sys/assert.h	
Low-level assert implementation	2226
l4/sys/cache.h	
Cache-consistency functions	2035
l4/sys/capability	
L4::Cap related definitions	2042
l4/sys/compiler.h	
L4 compiler related defines	2045
l4/sys/consts.h	
Common constants	2049
l4/sys/debugger	
The debugger interface specifies common debugging related definitions	2093
l4/sys/debugger.h	
Debugger related definitions	2095
l4/sys/err.h	
Error codes	2102
l4/sys/exception	
Exception C++ interface	2103
l4/sys/factory	
Common factory related definitions	2105
l4/sys/factory.h	
Common factory related definitions	2109
l4/sys/icu	
Interrupt controller	2115
l4/sys/icu.h	
Interrupt controller	2116
l4/sys/iommu	??
l4/sys/ipc.h	
Common IPC interface	2122
l4/sys/ipc_gate	
The C++ IPC gate interface	2131
l4/sys/ipc_gate.h	
The C IPC gate interface	2133
l4/sys/irq	
C++ Irq interface	2136
l4/sys/irq.h	
C Irq interface	1873
l4/sys/kdebug.h	??
l4/sys/kernel_object.h	
Kernel object system calls	2139
l4/sys/kip	2141
l4/sys/kip.h	
Kernel Info Page access functions	2291
l4/sys/kobject	??

l4/sys/ktrace.h	
L4 kernel event tracing	2144
l4/sys/l4int.h	
Fixed sized integer types, generic version	2146
l4/sys/memdesc.h	
Memory description functions	2150
l4/sys/meta	
Meta interface for getting dynamic type information about objects behind capabilities	2154
l4/sys/pager	
Pager and lo_pager C++ interface	2156
l4/sys/platform_control	
Platform control object	2158
l4/sys/platform_control.h	
Platform control object	2160
l4/sys/rcv_endpoint	
The C++ Receive endpoint interface	2164
l4/sys/rcv_endpoint.h	
Receive endpoint C interface	2166
l4/sys/scheduler	
Scheduler object functions	2168
l4/sys/scheduler.h	
Scheduler object functions	2170
l4/sys/semaphore	
Semaphore class definition	2174
l4/sys/semaphore.h	??
l4/sys/smart_capability	
L4::Capability class	2176
l4/sys/task	
Common task related definitions	2180
l4/sys/task.h	
Common task related definitions	2183
l4/sys/thread	
Common thread related definitions	2187
l4/sys/thread.h	
Common thread related definitions	2319
l4/sys/typeinfo_svr	
Type information server template	2192
l4/sys/types.h	
Common L4 ABI Data Types	2195
l4/sys/utcb.h	
UTCB definitions	2200
l4/sys/vcon	
Virtual console interface	2212
l4/sys/vcon.h	
Virtual console interface	2214
l4/sys/vcpu.h	??
l4/sys/vhw.h	
Descriptors for virtual hardware (under UX)	2220
l4/sys/vm	
Virtualization interface	2222
l4/sys/cxx/capability.h	??
l4/sys/cxx/consts	??
l4/sys/cxx/ipc_array	??
l4/sys/cxx/ipc_basics	??
l4/sys/cxx/ipc_client	2060
l4/sys/cxx/ipc_epiface	??
l4/sys/cxx/ipc_iface	
Interface Definition Language	2063

l4/sys/cxx/ipc_legacy	??
l4/sys/cxx/ipc_ret_array	??
l4/sys/cxx/ipc_server	??
l4/sys/cxx/ipc_server_loop	??
l4/sys/cxx/ipc_string	??
l4/sys/cxx/ipc_types	2077
l4/sys/cxx/ipc_varg	??
l4/sys/cxx/smart_capability_1x	2085
l4/sys/cxx/types	2089
l4/util/assert.h	
Some useful assert-style macros	2224
l4/util/atomic.h	
Atomic operations header and generic implementations	2229
l4/util/backtrace.h	
Backtrace	2238
l4/util/base64.h	
Base 64 encoding and decoding functions adapted from Bob Trower 08/04/01	2240
l4/util/bitops.h	
Bit manipulation functions	2242
l4/util/elf.h	
ELF definition	2247
l4/util/getopt.h	
Getopt	2285
l4/util/keymap.h	
Event to ASCII key mapping	2287
l4/util/kip.h	2288
l4/util/kprintf.h	
Printf using the kernel debugger	2297
l4/util/l4_macros.h	
Some useful generic macros, L4f version	1787
l4/util/l4mod.h	??
l4/util/list_alloc.h	
Simple list-based allocator	2299
l4/util/lulc.h	??
l4/util/lock.h	
Simple lock implementation	2302
l4/util/mb_info.h	
Multiboot info structure as defined by GRUB	2303
l4/util/parse_cmd.h	
Comfortable command-line parsing	2311
l4/util/rand.h	
Simple Pseudo-Random Number Generator	2314
l4/util/splitlog2.h	
Split a range in log2 aligned and size-aligned chunks	2315
l4/util/thread.h	
Low-level Thread Functions	2317
l4/util/util.h	??
l4/vbus/vbus	??
l4/vbus/vbus.h	
Description of the vbus C API	2330
l4/vbus/vbus_generic	??
l4/vbus/vbus_gpio	??
l4/vbus/vbus_gpio-ops.h	??
l4/vbus/vbus_gpio.h	??
l4/vbus/vbus_i2c.h	??
l4/vbus/vbus_inhibitor.h	??

l4/vbus/vbus_interfaces.h	
This header contains the definition of VBUS sub-interfaces and convenience functions to work with the interface IDs	2333
l4/vbus/vbus_mcspi.h	??
l4/vbus/vbus_pci	??
l4/vbus/vbus_pci-ops.h	??
l4/vbus/vbus_pci.h	??
l4/vbus/vbus_pm-ops.h	??
l4/vbus/vbus_pm.h	??
l4/vbus/vbus_types.h	
This header file contains descriptions of vbus related data types and constants	2336
l4/vbus/vdevice-ops.h	??
l4/vcpu/vcpu	??
l4/vcpu/vcpu.h	??
x86/l4/sys/__kip-arch.h	??
x86/l4/sys/__vcpu-arch.h	??
x86/l4/sys/cache.h	
Cache functions	2040
x86/l4/sys/consts.h	
Common L4 constants, x86 version	2055
x86/l4/sys/ipc-invoke.h	??
x86/l4/sys/ktrace_events.h	??
x86/l4/sys/l4int.h	
Fixed sized integer types, x86 version	2149
x86/l4/sys/linkage.h	
Linkage	1765
x86/l4/sys/segment.h	
Segment handling	1746
x86/l4/sys/utcb.h	
UTCB definitions for X86	2210
x86/l4/sys/vm.h	??
x86/l4/util/apic.h	
APIC for X86	1686
x86/l4/util/bitops_arch.h	
X86 bit manipulation functions	1774
x86/l4/util/cpu.h	
CPU related functions	1782
x86/l4/util/idt.h	
IDT related functions	1694
x86/l4/util/irq.h	
Some PIC and hardware interrupt related functions	1871
x86/l4/util/l4_macros.h	
Main function	1788
x86/l4/util/mbi_argv.h	
Command line handling	1791
x86/l4/util/perform.h	
Performance Monitoring using P5/P6 Measurement Counters	1701
x86/l4/util/port_io.h	
X86 port I/O	1754
x86/l4/util/rdtsc.h	
Time stamp counter related functions	1719
x86/l4/util/spin.h	
Spinning for x86	1730
x86/l4/util/util.h	
Utilities for x86	1734
x86/l4f/l4/sys/ipc-l42-gcc3-nopic.h	??
x86/l4f/l4/sys/ipc.h	
L4 IPC System Calls, x86	2129

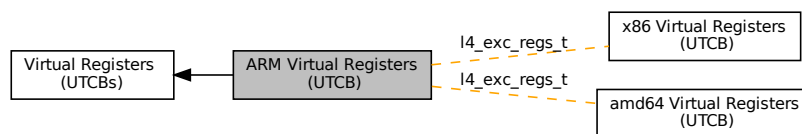
x86/l4f/l4/sys/ segment.h	
L4f specific segment manipulation	1745
x86/l4f/l4/util/ port_io.h	
Port I/O functions	1751

Chapter 13

Module Documentation

13.1 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.

13.1.1 Detailed Description

13.2 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)

13.2.1 Detailed Description

13.2.2 Function Documentation

13.2.2.1 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](#).

13.2.2.2 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](#).

13.2.2.3 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](#).

13.2.2.4 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](#).

13.2.2.5 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](#).

13.2.2.6 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](#).

13.2.2.7 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](#).

13.2.2.8 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](#).

13.2.2.9 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](#).

13.2.2.10 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](#).

13.2.2.11 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](#).

13.2.2.12 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](#).

13.2.2.13 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](#).

13.2.2.14 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](#).

13.2.2.15 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

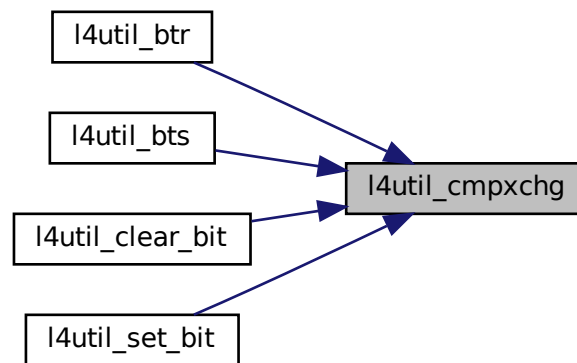
0 if comparison failed, 1 otherwise

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:



13.2.2.16 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](#).

13.2.2.17 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](#).

13.2.2.18 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](#).

13.2.2.19 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](#).

13.2.2.20 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](#).

13.2.2.21 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](#).

13.2.2.22 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](#).

13.2.2.23 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](#).

13.2.2.24 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](#).

13.2.2.25 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](#).

13.2.2.26 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](#).

13.2.2.27 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](#).

13.2.2.28 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](#).

13.2.2.29 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](#).

13.2.2.30 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](#).

13.2.2.31 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](#).

13.2.2.32 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](#).

13.2.2.33 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](#).

13.2.2.34 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](#).

13.2.2.35 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](#).

13.2.2.36 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](#).

13.2.2.37 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](#).

13.2.2.38 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](#).

13.2.2.39 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](#).

13.2.2.40 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](#).

13.2.2.41 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](#).

13.2.2.42 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](#).

13.2.2.43 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](#).

13.2.2.44 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](#).

13.2.2.45 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](#).

13.2.2.46 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](#).

13.2.2.47 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](#).

13.3 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

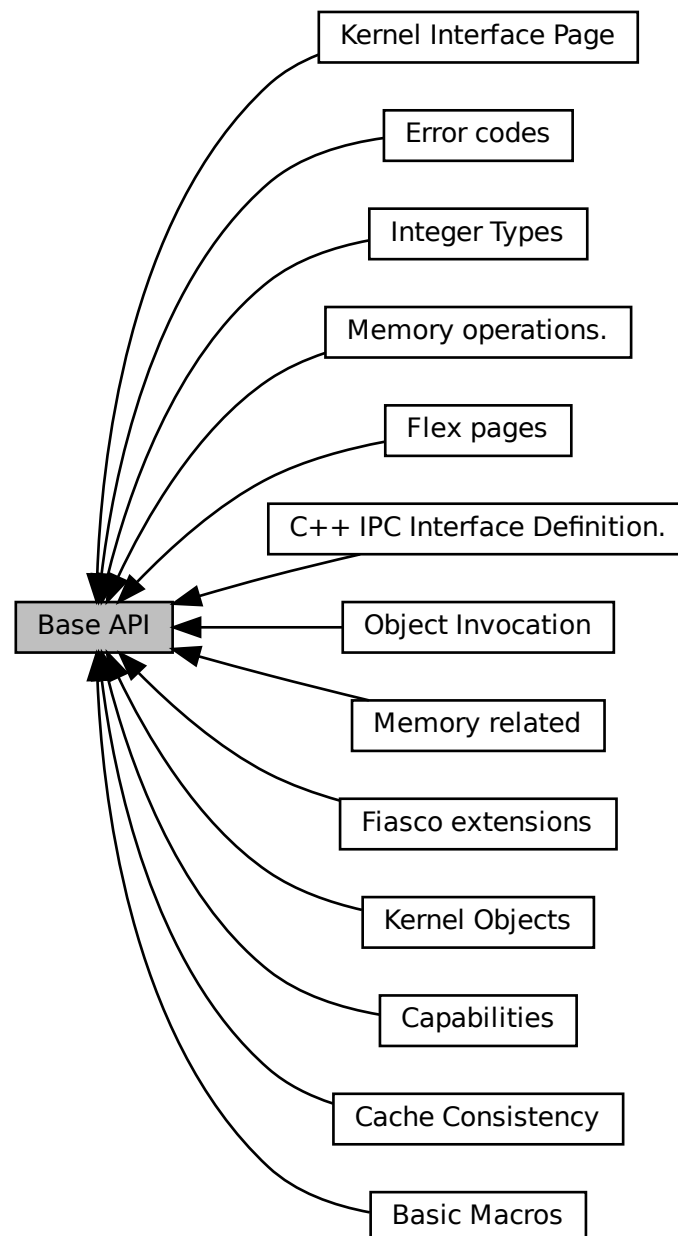
- enum [l4re_aux_ldr_flags_t](#)
Flags for program loading.

13.3.1 Detailed Description

13.4 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](#)
Kernel debugger 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](#)

- file [memdesc.h](#)
Kernel Info Page access functions.
- file [types.h](#)
Memory description functions.
- file [vhw.h](#)
Common L4 ABI Data Types.
- file [consts.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.

13.4.1 Detailed Description

Interfaces for all kinds of base functionality.

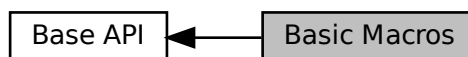
Some notes on Inter Process Communication (IPC)

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.

13.5 Basic Macros

L4 standard macros for header files, function definitions, and public APIs etc.

Collaboration diagram for Basic Macros:



Macros

- `#define L4_CV`
Define calling convention.
- `#define L4_CV`
Define calling convention.
- `#define L4_CV __attribute__((regparm(0)))`
Define calling convention.

13.5.1 Detailed Description

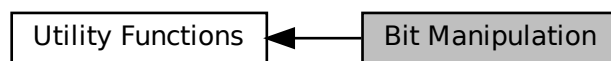
[L4](#) standard macros for header files, function definitions, and public APIs etc.

Include File

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

13.6 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.

13.6.1 Detailed Description

13.6.2 Function Documentation

13.6.2.1 l4util_bsf()

```
int l4util_bsf (  
    l4_umword_t word ) [inline]
```

Bit scan forward.

Parameters

<i>word</i>	value (machine size)
-------------	----------------------

Returns

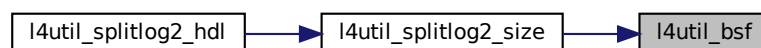
index of least significant bit set in word, -1 if no bit is set (word == 0)

"bit scan forward", find least significant bit set in word.

Definition at line 316 of file [bitops.h](#).

Referenced by [l4util_splitlog2_size\(\)](#).

Here is the caller graph for this function:



13.6.2.2 l4util_bsr()

```
int l4util_bsr (  
    l4_umword_t word ) [inline]
```

Bit scan reverse.

Parameters

<i>word</i>	value (machine size)
-------------	----------------------

Returns

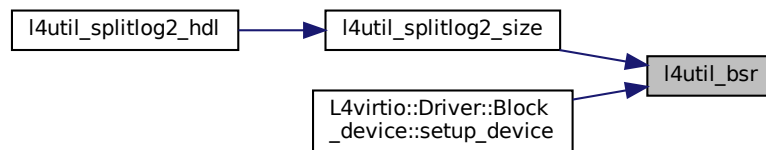
index of most significant set bit in word, -1 if no bit is set (word == 0)

"bit scan reverse", find most significant set bit in word (-> LOG2(word))

Definition at line 299 of file [bitops.h](#).

Referenced by [l4util_splitlog2_size\(\)](#), and [L4virtio::Driver::Block_device::setup_device\(\)](#).

Here is the caller graph for this function:

**13.6.2.3 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 394 of file [bitops.h](#).

13.6.2.4 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 278 of file [bitops.h](#).

References [l4util_cmpxchg\(\)](#).

Here is the call graph for this function:

**13.6.2.5 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

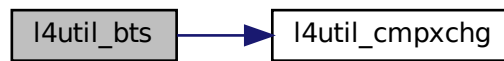
Old value of bit *b*.

Set the *b* bit of *dest* to 1 and return the old value.

Definition at line 256 of file [bitops.h](#).

References [l4util_cmpxchg\(\)](#).

Here is the call graph for this function:



13.6.2.6 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 226 of file [bitops.h](#).

References [l4util_cmpxchg\(\)](#).

Here is the call graph for this function:



13.6.2.7 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 359 of file [bitops.h](#).

13.6.2.8 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 32!)

Returns

number of the first set bit, \geq size if no bit is set

Definition at line 400 of file [bitops.h](#).

13.6.2.9 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 32!)

Returns

number of the first zero bit, \geq size if no bit is set

Definition at line 333 of file [bitops.h](#).

13.6.2.10 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 373 of file [bitops.h](#).

13.6.2.11 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 207 of file [bitops.h](#).

References [l4util_cmpxchg\(\)](#).

Here is the call graph for this function:



13.6.2.12 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

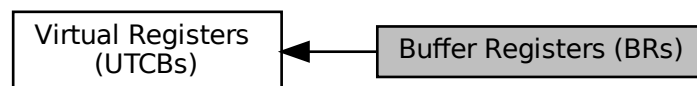
Returns

Value of bit *b*.

Definition at line 244 of file [bitops.h](#).

13.7 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.

13.7.1 Detailed Description

13.7.2 Enumeration Type Documentation

13.7.2.1 l4_buffer_desc_consts_t

enum [l4_buffer_desc_consts_t](#)

Constants for buffer descriptors.

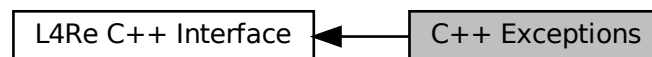
Enumerator

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 [219](#) of file [consts.h](#).

13.8 C++ Exceptions

Collaboration diagram for C++ Exceptions:



Files

- file [exceptions](#)
Base exceptions.
- file [std_exc_io](#)
Base exceptions std stream operator.

13.8.1 Detailed Description

13.9 C++ IPC Interface Definition.

APIs for defining IPC interfaces using C++ as language.

Collaboration diagram for C++ IPC Interface Definition.:



Modules

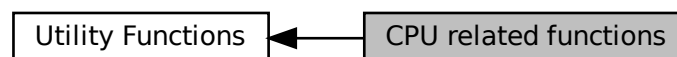
- [Internal Helpers](#)

13.9.1 Detailed Description

APIs for defining IPC interfaces using C++ as language.

13.10 CPU related functions

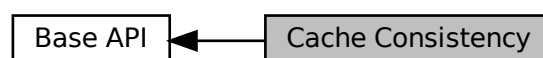
Collaboration diagram for CPU related functions:



13.11 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.
- int [l4_cache_flush_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache flush a range.
- int [l4_cache_inv_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache invalidate a range.
- int [l4_cache_coherent](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Make memory coherent between I-cache and D-cache.
- int [l4_cache_dma_coherent](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Make memory coherent for use with external memory.
- int [l4_cache_dma_coherent_full](#) (void) [L4_NOTHROW](#)
Make memory coherent for use with external memory.

13.11.1 Detailed Description

Various functions for cache consistency.

Include File

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

13.11.2 Function Documentation

13.11.2.1 l4_cache_clean_data()

```
int l4_cache_clean_data (
    unsigned long start,
    unsigned long end ) [inline]
```

Cache clean a range in D-cache.

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.

Examples

[examples/libs/l4re/c++/shared_ds/ds_clnt.cc](#).

Definition at line 81 of file [cache.h](#).

13.11.2.2 l4_cache_coherent()

```
int l4_cache_coherent (
    unsigned long start,
    unsigned long end ) [inline]
```

Make memory coherent between I-cache and D-cache.

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

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

13.11.2.3 l4_cache_dma_coherent()

```
int l4_cache_dma_coherent (
    unsigned long start,
    unsigned long end ) [inline]
```

Make memory coherent for use with external memory.

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

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

13.11.2.4 l4_cache_flush_data()

```
int l4_cache_flush_data (
    unsigned long start,
    unsigned long end ) [inline]
```

Cache flush a range.

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](#).

13.11.2.5 l4_cache_inv_data()

```
int l4_cache_inv_data (
    unsigned long start,
    unsigned long end ) [inline]
```

Cache invalidate a range.

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

Invalidates all cache entries in the range but does not necessarily write back dirty cache lines.

Note

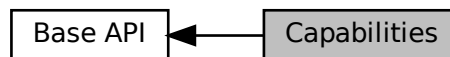
Implementations may choose to write back dirty lines nonetheless if this is more efficient.

Definition at line 97 of file [cache.h](#).

13.12 Capabilities

C interface for capabilities.

Collaboration diagram for Capabilities:



Typedefs

- typedef unsigned long [l4_cap_idx_t](#)
L4 Capability selector Type.

Enumerations

- enum [l4_cap_consts_t](#) { [L4_CAP_SHIFT](#) , [L4_CAP_SIZE](#) , [L4_CAP_OFFSET](#) = 1UL << [L4_CAP_SHIFT](#) , [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 two capability selectors are equal.

13.12.1 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.

13.12.2 Enumeration Type Documentation

13.12.2.1 l4_cap_consts_t

```
enum l4_cap_consts_t
```

Constants related to capability selectors.

Enumerator

L4_CAP_SHIFT	Capability index shift.
L4_CAP_SIZE	Offset of two consecutive capability selectors.
L4_CAP_MASK	Mask to get only the relevant bits of an l4_cap_idx_t.
L4_INVALID_CAP	Invalid capability selector.

Definition at line 128 of file [consts.h](#).

13.12.2.2 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 240 of file [consts.h](#).

13.12.3 Function Documentation

13.12.3.1 l4_capability_equal()

```
unsigned l4_capability_equal (
    l4_cap_idx_t c1,
    l4_cap_idx_t c2 ) [inline]
```

Test if two capability selectors are equal.

Parameters

<i>c1</i>	Capability
<i>c2</i>	Capability

Return values

0	The given capabilities are not in the same slot.
1	The given capabilities are in the same slot.

Definition at line 400 of file [types.h](#).

References [L4_CAP_SHIFT](#).

13.12.3.2 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 392 of file [types.h](#).

13.12.3.3 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

<i>c</i>	Capability selector
----------	---------------------

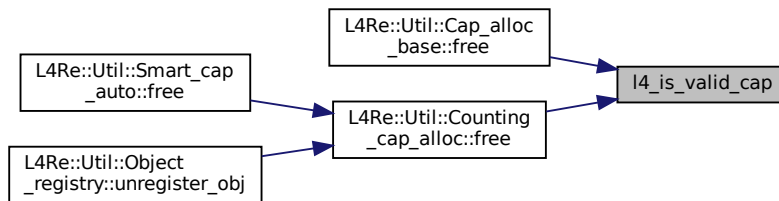
Return values

0	The capability selector is not valid.
>0	The capability selector is valid.

Definition at line 396 of file [types.h](#).

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

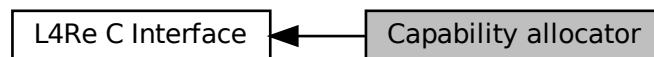
Here is the caller graph for this function:



13.13 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.

13.13.1 Detailed Description

Capability allocator C interface.

13.13.2 Function Documentation

13.13.2.1 l4re_util_cap_last()

```
long l4re_util_cap_last (
    void )
```

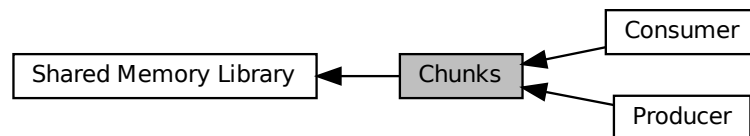
Return last capability index the allocator can return.

Returns

last/biggest capability index the allocator can return

13.14 Chunks

Collaboration diagram for Chunks:



Modules

- [Consumer](#)
- [Producer](#)

Functions

- long [l4shmc_add_chunk](#) (l4shmc_area_t *shmarea, const char *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, const char *chunk_name, l4shmc_chunk_t *chunk)
Get chunk out of shared memory area.
- long [l4shmc_get_chunk_to](#) (l4shmc_area_t *shmarea, const char *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 *shmarea, const char **chunk_name, long offs)
Iterate over names of all existing chunks.
- void * [l4shmc_chunk_ptr](#) (l4shmc_chunk_t *chunk)
Get data pointer to chunk.
- long [l4shmc_chunk_capacity](#) (l4shmc_chunk_t *chunk)
Get capacity of a chunk.
- l4shmc_signal_t * [l4shmc_chunk_signal](#) (l4shmc_chunk_t *chunk)
Get the signal of a chunk.

13.14.1 Detailed Description

13.14.2 Function Documentation

13.14.2.1 l4shmc_add_chunk()

```
long l4shmc_add_chunk (
    l4shmc_area_t * shmarea,
    const char * 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](#).

13.14.2.2 l4shmc_chunk_capacity()

```
long l4shmc_chunk_capacity (
    l4shmc_chunk_t * chunk ) [inline]
```

Get capacity of a chunk.

Parameters

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

Return values

0	Success.
---	----------

Return values

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

13.14.2.3 l4shmc_chunk_ptr()

```
void* l4shmc_chunk_ptr (  
    l4shmc_chunk_t * chunk ) [inline]
```

Get data pointer to chunk.

Parameters

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

Return values

0	Success.
<0	Error.

Examples

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

13.14.2.4 l4shmc_chunk_signal()

```
l4shmc_signal_t* l4shmc_chunk_signal (  
    l4shmc_chunk_t * chunk ) [inline]
```

Get the signal of a chunk.

Parameters

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

Return values

0	No signal has been registered with this chunk.
>0	Pointer to signal otherwise.

13.14.2.5 l4shmc_get_chunk()

```
long l4shmc_get_chunk (
    l4shmc_area_t * shmarea,
    const char * chunk_name,
    l4shmc_chunk_t * chunk ) [inline]
```

Get chunk out of shared memory area.

Parameters

	<i>shmarea</i>	Shared memory area.
	<i>chunk_name</i>	Name of the chunk.
out	<i>chunk</i>	Chunk data structure to fill.

Return values

0	Success.
<0	Error.

Examples

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

13.14.2.6 l4shmc_get_chunk_to()

```
long l4shmc_get_chunk_to (
    l4shmc_area_t * shmarea,
    const char * chunk_name,
    l4_umword_t timeout_ms,
    l4shmc_chunk_t * chunk )
```

Get chunk out of shared memory area, with timeout.

Parameters

	<i>shmarea</i>	Shared memory area.
	<i>chunk_name</i>	Name of the chunk.
	<i>timeout_ms</i>	Timeout in milliseconds to wait for the chunk to appear in the shared memory area.
out	<i>chunk</i>	Chunk data structure to fill.

Return values

0	Success.
<0	Error.

13.14.2.7 l4shmc_iterate_chunk()

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

Iterate over names of all existing chunks.

Parameters

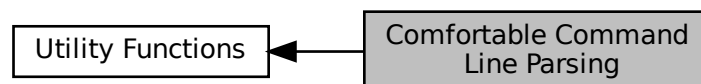
<i>shmarea</i>	Shared memory area.
<i>chunk_name</i>	Where the name of the current chunk will be stored
<i>offs</i>	0 to start iteration, return value of previous call to l4shmc_iterate_chunk() to get next chunk

Return values

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

13.15 Comfortable Command Line Parsing

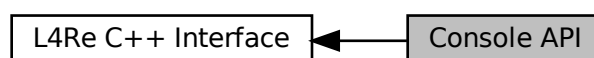
Collaboration diagram for Comfortable Command Line Parsing:



13.16 Console API

[Console](#) interface.

Collaboration diagram for Console API:



Data Structures

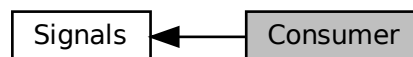
- class [L4Re::Console](#)
Console class.

13.16.1 Detailed Description

[Console](#) interface.

13.17 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.

13.17.1 Detailed Description

13.17.2 Function Documentation

13.17.2.1 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.

13.17.2.2 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.

13.17.2.3 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.

13.17.2.4 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.

13.17.2.5 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](#).

13.17.2.6 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.

13.17.2.7 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.
---------------	------------------

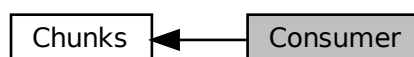
Return values

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.

13.18 Consumer

Collaboration diagram for Consumer:



Functions

- 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 *chunk)
Check whether data is available.
- long [l4shmc_chunk_size](#) (l4shmc_chunk_t *chunk)
Get current size of a chunk.

13.18.1 Detailed Description

13.18.2 Function Documentation

13.18.2.1 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](#).

13.18.2.2 l4shmc_chunk_size()

```
long l4shmc_chunk_size (
    l4shmc_chunk_t * chunk ) [inline]
```

Get current size of a chunk.

Parameters

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

Return values

0	Success.
<0	Error.

Examples

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

13.18.2.3 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.

13.18.2.4 l4shmc_is_chunk_ready()

```
long l4shmc_is_chunk_ready (
    l4shmc_chunk_t * chunk ) [inline]
```

Check whether data is available.

Parameters

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

Return values

0	Success.
<0	Error.

13.18.2.5 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

0	Success.
<0	Error.

Examples

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

13.18.2.6 l4shmc_wait_chunk_to()

```
long l4shmc_wait_chunk_to (
    l4shmc_chunk_t * chunk,
    l4_timeout_t timeout )
```

Check whether a specific chunk has an event pending, with timeout.

Parameters

<i>chunk</i>	Chunk to check.
<i>timeout</i>	Timeout.

Return values

0	Success.
<0	Error.

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

13.18.2.7 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

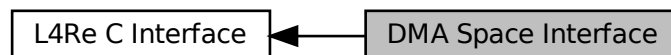
0	Success.
<0	Error.

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.

13.19 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 DMA task for a device to this Dma_space.
- long [l4re_dma_space_disassociate](#) ([l4re_dma_space_t](#) dma)
Disassociate the DMA task from this Dma_space.

13.19.1 Detailed Description

DMA Space C interface.

13.19.2 Typedef Documentation

13.19.2.1 [l4re_dma_space_t](#)

```
typedef l4\_cap\_idx\_t l4re\_dma\_space\_t
```

DMA space capability type.

DMA Address Space. A [Dma_space](#) represents the DMA address space of a device. Whenever a device needs direct access to parts of an [L4Re::Dataspace](#), that part of the data space must be mapped to the DMA address space that is assigned to that device. After the DMA accesses to the memory are finished the memory must be unmapped from the device's DMA address space.

Mapping to a 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](#).

13.19.3 Function Documentation

13.19.3.1 [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 DMA task for a device to this [Dma_space](#).

Parameters

<i>dma</i>	DMA space capability
------------	----------------------

Precondition

requires capability rights: {RW}

Parameters

in	<i>dma_task</i>	The DMA task used for the device that shall be associated with this DMA space. 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 .

13.19.3.2 l4re_dma_space_disassociate()

```
long l4re_dma_space_disassociate (
    l4re_dma_space_t dma )
```

Disassociate the DMA task from this *Dma_space*.

Parameters

<i>dma</i>	DMA space capability
------------	----------------------

Precondition

requires capability rights: {RW}

13.19.3.3 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
------------	----------------------

Precondition

requires capability rights: {R}

Parameters

in	<i>src</i>	Source data space (that describes the memory). Caller needs write rights 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.

Returns

0 in the case of success, a negative error code otherwise.

13.19.3.4 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
------------	----------------------

Precondition

requires capability rights: {R}

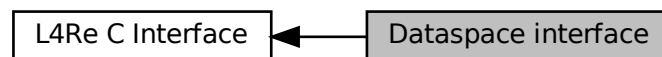
Parameters

<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.

13.20 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.
- typedef [l4_cap_idx_t](#) [l4re_namespace_t](#)
Namespace 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](#)
- long [l4re_ds_allocate](#) ([l4re_ds_t](#) ds, [l4re_ds_offset_t](#) offset, [l4re_ds_size_t](#) size) [L4_NOTHROW](#)
- 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](#)
- [l4re_ds_size_t](#) [l4re_ds_size](#) ([l4re_ds_t](#) ds) [L4_NOTHROW](#)
- [l4re_ds_flags_t](#) [l4re_ds_flags](#) ([l4re_ds_t](#) ds) [L4_NOTHROW](#)
- int [l4re_ds_info](#) ([l4re_ds_t](#) ds, [l4re_ds_stats_t](#) *stats) [L4_NOTHROW](#)

13.20.1 Detailed Description

Dataspace C interface.

13.20.2 Enumeration Type Documentation

13.20.2.1 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](#).

13.20.3 Function Documentation

13.20.3.1 l4re_ds_allocate()

```
long l4re_ds_allocate (
    l4re_ds_t ds,
    l4re_ds_offset_t offset,
    l4re_ds_size_t size )
```

Returns

0 on success, <0 on errors

See also

[L4Re::Dataspace::allocate](#)

13.20.3.2 l4re_ds_clear()

```
long l4re_ds_clear (
    l4re_ds_t ds,
    l4re_ds_offset_t offset,
    l4re_ds_size_t size )
```

Returns

0 on success, <0 on errors

See also

[L4Re::Dataspace::clear](#)

13.20.3.3 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 )
```

Returns

0 on success, <0 on errors

See also

[L4Re::Dataspace::copy_in](#)

13.20.3.4 l4re_ds_flags()

```
l4re_ds_flags_t l4re_ds_flags (
    l4re_ds_t ds )
```

See also

[L4Re::Dataspace::flags](#)

13.20.3.5 l4re_ds_info()

```
int l4re_ds_info (
    l4re_ds_t ds,
    l4re_ds_stats_t * stats )
```

See also

[L4Re::Dataspace::info](#)

13.20.3.6 l4re_ds_size()

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

Returns

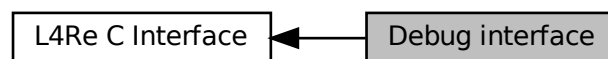
Size of the dataspace in bytes.

See also

[L4Re::Dataspace::size](#)

13.21 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.

13.21.1 Detailed Description

13.21.2 Function Documentation

13.21.2.1 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.

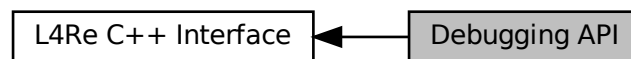
See also

[L4Re::Debug_obj::debug](#)

13.22 Debugging API

Debugging Interface.

Collaboration diagram for Debugging API:



Data Structures

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

13.22.1 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.

13.23 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.

13.23.1 Detailed Description

13.23.2 Enumeration Type Documentation

13.23.2.1 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](#).

13.23.3 Function Documentation

13.23.3.1 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

13.23.3.2 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

13.23.3.3 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
-------------	---------------------------------

13.23.3.4 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

13.23.3.5 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

13.23.3.6 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
-------------	---------------------------------

Return values

<i>id</i>	Return the PnP id. Must point to 4 bytes.
-----------	---

13.23.3.7 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
-------------	---------------------------------

Return values

<i>w</i>	X resolution of preferred video mode in pixels.
<i>h</i>	Y resolution of preferred video mode in pixels.

13.23.3.8 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

13.23.3.9 libedid_version()

```
unsigned libedid_version (  
    const unsigned char * edid )
```

Returns the EDID version number.

Parameters

<i>edid</i>	Pointer to a 128byte EDID block
-------------	---------------------------------

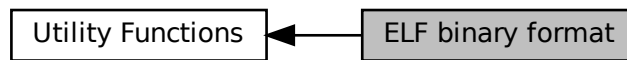
Returns

Version number

13.24 ELF binary format

Functions and types related to ELF binaries.

Collaboration diagram for ELF binary format:



Files

- file [elf.h](#)
ELF definition.

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

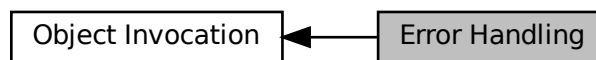
13.24.1 Detailed Description

Functions and types related to ELF binaries.

13.25 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 error code for an object invocation.
- `long l4_error (l4_msgtag_t tag) L4_NOTHROW`
Return error code of a system call return message tag or the tag label.
- `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.

13.25.1 Detailed Description

Error handling for [L4](#) object invocation.

Include File

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

13.25.2 Enumeration Type Documentation

13.25.2.1 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

L4_IPC_ERROR_MASK	Mask for error bits.
L4_IPC_SND_ERR_MASK	Send error mask.
L4_IPC_ENOT_EXISTENT	Non-existing destination or source.
L4_IPC_RETIMEOUT	Timeout during receive operation.
L4_IPC_SETIMEOUT	Timeout during send operation.
L4_IPC_RECANCELED	Receive operation canceled.
L4_IPC_SECANCELED	Send operation canceled.
L4_IPC_REMAPFAILED	Map flexpage failed in receive operation.
L4_IPC_SEMAPFAILED	Map flexpage failed in send operation.
L4_IPC_RESNDPFTO	Send-pagefault timeout in receive operation.
L4_IPC_SESNDPFTO	Send-pagefault timeout in send operation.
L4_IPC_RERCVPFTO	Receive-pagefault timeout in receive operation.
L4_IPC_SERCVPFTO	Receive-pagefault timeout in send operation.
L4_IPC_REABORTED	Receive operation aborted.
L4_IPC_SEABORTED	Send operation aborted.
L4_IPC_REMSGCUT	Cut receive message, due to message buffer is too small.
L4_IPC_SEMSGCUT	Cut send message. due to message buffer is too small,

Definition at line [75](#) of file [ipc.h](#).

13.25.3 Function Documentation

13.25.3.1 l4_error()

```
long l4_error (
    l4_msgtag_t tag ) [inline]
```

Return error code of a system call return message tag or the tag label.

Parameters

<i>tag</i>	System call return message type.
------------	----------------------------------

Returns

In case of IPC error a negative error code in the range of L4_EIPC_LO to L4_EIPC_HI, otherwise the tag label.

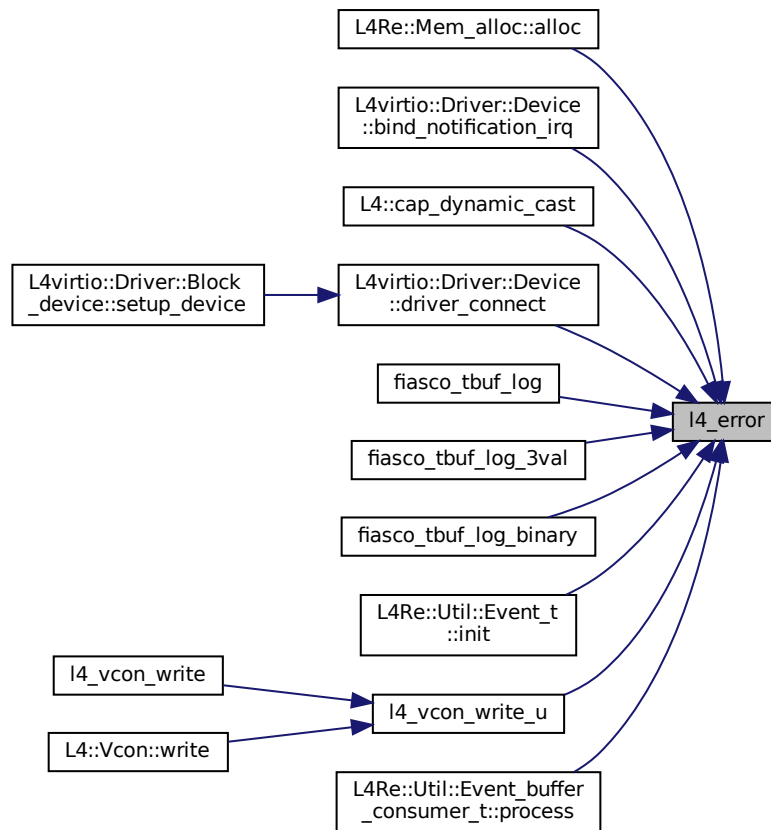
Examples

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

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 caller graph for this function:



13.25.3.2 l4_ipc_error()

```
l4_umword_t l4_ipc_error (
    l4_msgtag_t tag,
    l4_utcb_t * utcb ) [inline]
```

Get the error code for an object invocation.

Parameters

<i>tag</i>	Return value of the invocation.
<i>utcb</i>	UTCB that was used for the invocation.

Returns

0 if no error condition is set, error code otherwise (see [l4_ipc_tcr_error_t](#)).

Examples

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

Definition at line 518 of file [ipc.h](#).

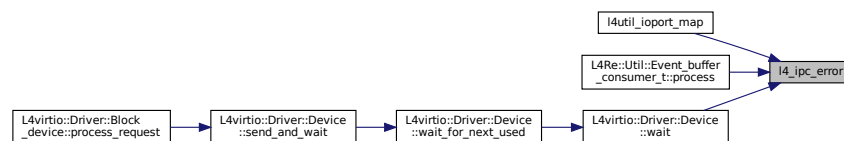
References [l4_msgtag_has_error\(\)](#).

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:

**13.25.3.3 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

<code>utcb</code>	UTCB to check.
-------------------	----------------

Returns

Error condition of type `l4_ipc_tcr_error_t`.

Definition at line 547 of file `ipc.h`.

13.25.3.4 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 544 of file `ipc.h`.

13.25.3.5 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 541 of file [ipc.h](#).

13.26 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](#).

13.26.1 Detailed Description

Common error codes.

Include File

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

13.26.2 Enumeration Type Documentation

13.26.2.1 [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

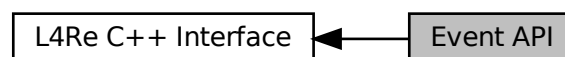
L4_EOK	Ok.
L4_EPERM	No permission.
L4_ENOENT	No such entity.
L4_EIO	I/O error.
L4_ENXIO	No such device or address.
L4_E2BIG	Argument value too big.
L4_EAGAIN	Try again.
L4_ENOMEM	No memory.
L4_EACCESS	Permission denied.
L4_EFAULT	Invalid memory address.
L4_EBUSY	Object currently busy, try later.
L4_EEXIST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

13.27 Event API

[Event API](#).

Collaboration diagram for Event API:



Data Structures

- class [L4Re::Event](#)
Event class.
- class [L4Re::Event_buffer_t< PAYLOAD >](#)
Event buffer class.

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

13.28 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.

13.28.1 Detailed Description

Event C interface.

13.28.2 Function Documentation

13.28.2.1 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

13.28.2.2 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](#)

13.28.2.3 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](#)

13.28.2.4 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.

Return values

<i>info</i>	Information buffer.
-------------	---------------------

Returns

0 for success, <0 on error

See also

L4Re::Event::get_stream_info

13.28.2.5 l4re_event_get_stream_info_for_id()

```
long l4re_event_get_stream_info_for_id (
    const l4_cap_idx_t server,
    l4_umword_t stream_id,
    l4re_event_stream_info_t * info )
```

Get info for a stream given a stream id.

Parameters

<i>server</i>	Server to talk to.
<i>stream↔ _id</i>	Stream ID.

Return values

<i>info</i>	Information buffer.
-------------	---------------------

Returns

0 for success, <0 on error

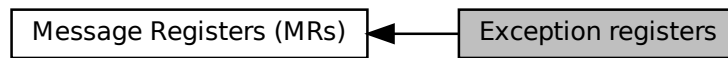
See also

L4Re::Event::get_stream_info_for_id

13.29 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 UTCTB (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 UTCTB 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()`.

13.29.1 Detailed Description

Overly definition of the MRs for exception messages.

13.29.2 Function Documentation

13.29.2.1 `l4_utcb_exc()`

```
l4_exc_regs_t * l4_utcb_exc (
    void ) [inline]
```

Get the message-register block of a UTCTB (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`.

13.29.2.2 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

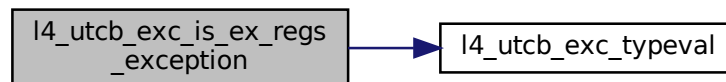
0	Exception was not triggered through ex_regs.
!=0	Exception was triggered through ex_regs.

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 115 of file [utcb.h](#).

References [l4_utcb_exc_typeval\(\)](#).

Here is the call graph for this function:



13.29.2.3 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 105 of file [utcb.h](#).

13.29.2.4 l4_utcb_exc_pc()

```
l4_umword_t l4_utcb_exc_pc (
    l4_exc_regs_t const * u ) [inline]
```

Access function to get the program counter of the exception state.

Parameters

<i>u</i>	UTCB
----------	------

Returns

The program counter register out of the exception state.

Examples

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

Definition at line 90 of file [utcb.h](#).

13.29.2.5 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

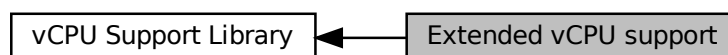
<i>u</i>	UTCB
<i>pc</i>	The program counter to set.

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

13.30 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.

13.30.1 Detailed Description

extended vCPU handling functionality.

13.30.2 Function Documentation

13.30.2.1 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.

Return values

<i>vcpu</i>	Allocated vcpu-state area.
<i>ext_state</i>	Allocated extended vcpu-state area.

Parameters

<i>task</i>	Task to use for allocation.
<i>regmgr</i>	Region manager to use for allocation.

Returns

0 for success, error code otherwise

13.31 Factory

C factory interface to create kernel objects.

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_task_u](#) ([l4_cap_idx_t](#) factory, [l4_cap_idx_t](#) target_cap, [l4_fpage_t](#) utcb_area, [l4_utcb_t](#) *utcb) [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_thread_u](#) ([l4_cap_idx_t](#) factory, [l4_cap_idx_t](#) target_cap, [l4_utcb_t](#) *utcb) [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_factory_u](#) ([l4_cap_idx_t](#) factory, [l4_cap_idx_t](#) target_cap, unsigned long limit, [l4_utcb_t](#) *utcb) [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_gate_u](#) ([l4_cap_idx_t](#) factory, [l4_cap_idx_t](#) target_cap, [l4_cap_idx_t](#) thread_cap, [l4_umword_t](#) label, [l4_utcb_t](#) *utcb) [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_irq_u](#) ([l4_cap_idx_t](#) factory, [l4_cap_idx_t](#) target_cap, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Create a new IRQ.
- [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_vm_u](#) ([l4_cap_idx_t](#) factory, [l4_cap_idx_t](#) target_cap, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Create a new virtual machine.

13.31.1 Detailed Description

C factory interface to create kernel objects.

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](#).

13.31.2 Function Documentation**13.31.2.1 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

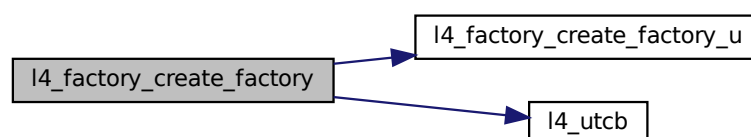
Note

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

Definition at line 373 of file [factory.h](#).

References [l4_factory_create_factory_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.31.2.2 l4_factory_create_factory_u()

```
l4_msgtag_t l4_factory_create_factory_u (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    unsigned long limit,
    l4_utcb_t * utcb ) [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.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

Note

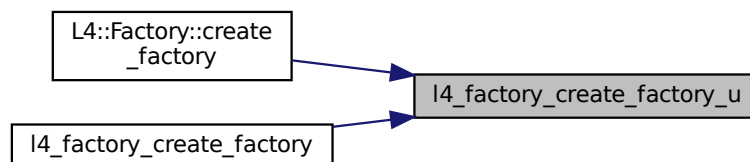
The `limit` (quota) of the new factory is subtracted from the limit of the factory invoked on its creation.

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 307 of file [factory.h](#).

Referenced by [L4::Factory::create_factory\(\)](#), and [l4_factory_create_factory\(\)](#).

Here is the caller graph for this function:



13.31.2.3 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 the thread to bind the gate to. Use L4_INVALID_CAP to create an unbound IPC gate.
	<i>label</i>	Optional label of the gate (is used if <i>thread_cap</i> is valid).

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_ENOENT</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>thread_cap</i> .

An unbound IPC gate can be bound to a thread using [l4_ipc_gate_bind_thread](#).

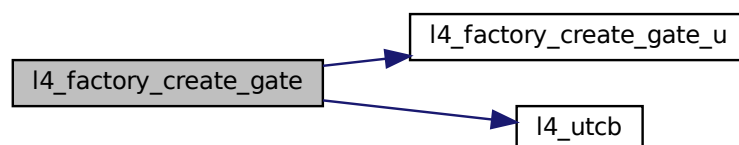
See also

[IPC-Gate API](#)

Definition at line 381 of file [factory.h](#).

References [l4_factory_create_gate_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.31.2.4 l4_factory_create_gate_u()

```
l4_msgtag_t l4_factory_create_gate_u (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    l4_cap_idx_t thread_cap,
    l4_umword_t label,
    l4_utcb_t * utcb ) [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 the thread to bind the gate to. Use L4_INVALID_CAP to create an unbound IPC gate.
	<i>label</i>	Optional label of the gate (is used if <i>thread_cap</i> is valid).
	<i>utcb</i>	The UTCB to use for the operation.

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 <code>lpc_gate</code> object.
<i>-L4_ENOENT</i>	<code>thread_cap</code> is void or points to something that is not a thread.
<i>-L4_EPERM</i>	No L4_CAP_FPAGE_S rights on <code>thread_cap</code> .

An unbound IPC gate can be bound to a thread using [L4::lpc_gate::bind_thread\(\)](#).

Note

This method is only guaranteed to work with the [Kernel Factory](#).

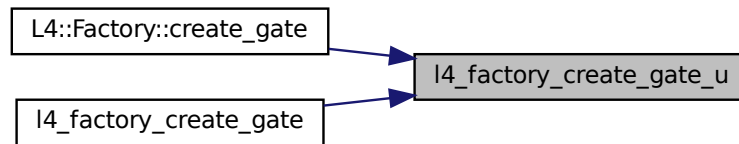
See also

[L4::lpc_gate](#)

Definition at line 318 of file `factory.h`.

Referenced by [L4::Factory::create_gate\(\)](#), and [l4_factory_create_gate\(\)](#).

Here is the caller graph for this function:



13.31.2.5 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.

Returns

Syscall return tag

See also

[IRQs](#)

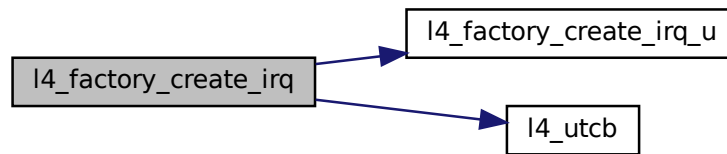
Examples

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

Definition at line [389](#) of file [factory.h](#).

References [l4_factory_create_irq_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.31.2.6 l4_factory_create_irq_u()

```
l4_msgtag_t l4_factory_create_irq_u (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    l4_utcb_t * utcb ) [inline]
```

Create a new IRQ.

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.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

Deprecated Use `create()` with `Cap<Irq>` as argument instead.

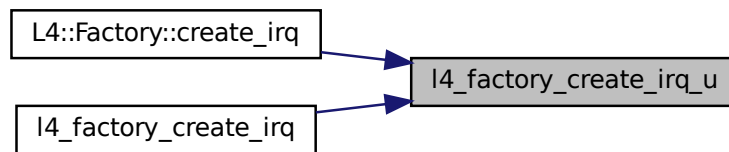
See also

[L4::Irq](#)

Definition at line 340 of file [factory.h](#).

Referenced by [L4::Factory::create_irq\(\)](#), and [l4_factory_create_irq\(\)](#).

Here is the caller graph for this function:



13.31.2.7 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.

Note

The size of the UTCB area specifies indirectly the number of UTCBs available for this task. Refer to [L4::Task::add_ku_mem](#) / [l4_task_add_ku_mem\(\)](#) for adding more of this type of memory.

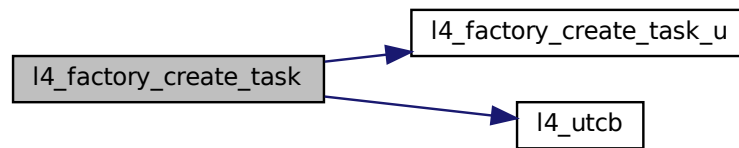
See also

[Task](#)

Definition at line 359 of file [factory.h](#).

References [l4_factory_create_task_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.31.2.8 l4_factory_create_task_u()

```

l4_msgtag_t l4_factory_create_task_u (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    l4_fpage_t utcb_area,
    l4_utcb_t * utcb ) [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 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>	The UTCB to use for the operation.

Returns

Syscall return tag

Note

The size of the UTCB area specifies indirectly the number of UTCBs available for this task. Refer to [L4::Task::add_ku_mem / l4_task_add_ku_mem\(\)](#) for adding more of this type of memory.

This method is only guaranteed to work with the [Kernel Factory](#).

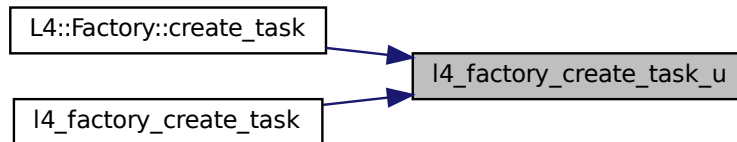
See also

[L4::Task](#)

Definition at line 289 of file [factory.h](#).

Referenced by [L4::Factory::create_task\(\)](#), and [l4_factory_create_task\(\)](#).

Here is the caller graph for this function:



13.31.2.9 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

See also

[Thread](#)

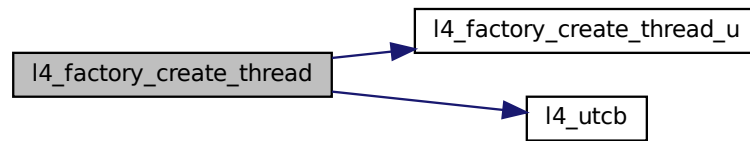
Examples

[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 366 of file [factory.h](#).

References [l4_factory_create_thread_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.31.2.10 l4_factory_create_thread_u()

```
l4_msgtag_t l4_factory_create_thread_u (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    l4_utcb_t * utcb ) [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.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

Deprecated Use `create()` with `Cap<Thread>` as argument instead.

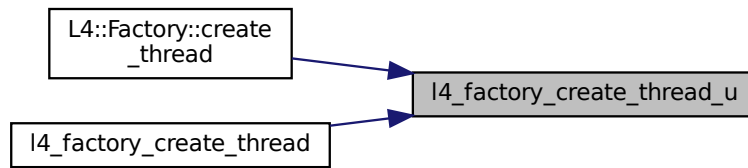
See also

[L4::Thread](#)

Definition at line 300 of file [factory.h](#).

Referenced by [L4::Factory::create_thread\(\)](#), and [l4_factory_create_thread\(\)](#).

Here is the caller graph for this function:



13.31.2.11 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

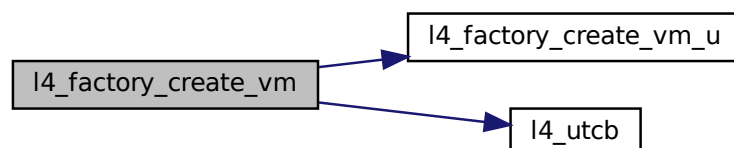
See also

[Virtual Machines](#)

Definition at line 396 of file [factory.h](#).

References [l4_factory_create_vm_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.31.2.12 l4_factory_create_vm_u()

```
l4_msgtag_t l4_factory_create_vm_u (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    l4_utcb_t * utcb ) [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.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

Deprecated Use `create()` with `Cap<Vm>` as argument instead.

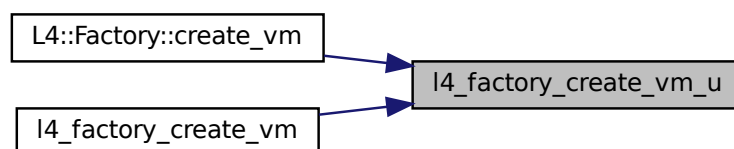
See also

[L4::Vm](#)

Definition at line 347 of file [factory.h](#).

Referenced by [L4::Factory::create_vm\(\)](#), and [l4_factory_create_vm\(\)](#).

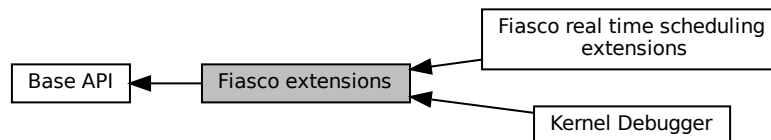
Here is the caller graph for this function:



13.32 Fiasco extensions

Kernel debugger 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.

Files

- file [segment.h](#)
l4f specific fs/gs manipulation
- file [segment.h](#)
l4f specific segment manipulation

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.
- 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.

13.32.1 Detailed Description

Kernel debugger extensions of the Fiasco L4 implementation.

13.32.2 Function Documentation

13.32.2.1 `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 of the caller.

Definition at line 167 of file `segment.h`.

13.32.2.2 `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 3 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: 0-2).
<i>utcb</i>	UTCB of the caller.

Returns

System call error

Definition at line 52 of file [segment.h](#).

13.32.2.3 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 of the caller.

Definition at line 154 of file [segment.h](#).

References [L4_EINVAL](#), and [L4_TASK_LDT_X86_MAX_ENTRIES](#).

13.32.2.4 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 [l4_error\(\)](#).

Here is the call graph for this function:



13.32.2.5 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

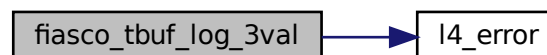
Returns

Pointer to trace-buffer entry

Definition at line 42 of file [__ktrace-impl.h](#).

References [l4_error\(\)](#).

Here is the call graph for this function:



13.32.2.6 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
-------------	-------------

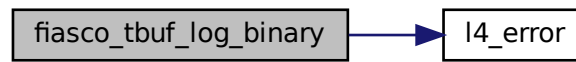
Returns

Pointer to trace-buffer entry

Definition at line 65 of file [__ktrace-impl.h](#).

References [l4_error\(\)](#).

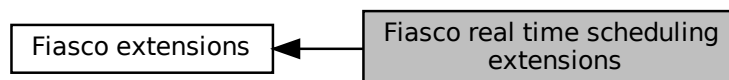
Here is the call graph for this function:



13.33 Fiasco real time scheduling extensions

Real time scheduling extension for the Fiasco [L4](#) implementation.

Collaboration diagram for Fiasco real time scheduling extensions:

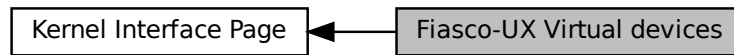


Real time scheduling extension for the Fiasco [L4](#) implementation.

13.34 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.

13.34.1 Detailed Description

Virtual hardware devices, provided by Fiasco-UX.

Include File

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

13.34.2 Enumeration Type Documentation

13.34.2.1 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](#).

13.35 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 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](#) }
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_R` | `L4_CAP_FPAGE_W` | `L4_CAP_FPAGE_S` ,
`L4_CAP_FPAGE_RWSD` = `L4_CAP_FPAGE_RWS` | `L4_CAP_FPAGE_D` , `L4_CAP_FPAGE_RWD` = `L4_CAP_FPAGE_R` | `L4_CAP_FPAGE_W` | `L4_CAP_FPAGE_D` , `L4_CAP_FPAGE_RSD` = `L4_CAP_FPAGE_RS` | `L4_CAP_FPAGE_D` }
Cap-flex-page rights.
- enum `L4_fpage_type`
Flex-page type.
- enum `L4_fpage_control`
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 f)` `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.

13.35.1 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.

13.35.2 Enumeration Type Documentation

13.35.2.1 anonymous enum

anonymous enum

Constants for flexpages.

Enumerator

L4_WHOLE_ADDRESS_SPACE	Whole address space size.
--	---------------------------

Definition at line 91 of file [__l4_fpage.h](#).

13.35.2.2 anonymous enum

anonymous enum

Special constants for IO flex pages.

Enumerator

L4_WHOLE_IOADDRESS_SPACE	Whole I/O address space size.
L4_IOPORT_MAX	Maximum I/O port address.

Definition at line 283 of file [__l4_fpage.h](#).

13.35.2.3 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. The kernel masks this right with the 'S' right of the IPC gate over which the capability is mapped. That means that the receiver capability will only have the 'S' right set if both the flex-page and the IPC gate have the 'S' bit set.
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 unmap() 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

Enumerator

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 136 of file [__l4_fpage.h](#).

13.35.2.4 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 264 of file [__l4_fpage.h](#).

13.35.2.5 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.

Definition at line 57 of file [__l4_fpage.h](#).

13.35.2.6 L4_fpage_rights

enum [L4_fpage_rights](#)

Flex-page rights.

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 109 of file [__l4_fpage.h](#).

13.35.2.7 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).
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 242 of file [__l4_fpage.h](#).

13.35.3 Function Documentation

13.35.3.1 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 address 0)
<i>rights</i>	Access rights, see L4_fpage_rights

Returns

Memory flex page

Definition at line 635 of file [__l4_fpage.h](#).

13.35.3.2 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.

Definition at line 655 of file [__l4_fpage.h](#).

References [L4_WHOLE_ADDRESS_SPACE](#).

13.35.3.3 l4_fpage_contains()

```
int l4_fpage_contains (
    l4_fpage_t fpage,
    l4_addr_t addr,
    unsigned size ) [inline]
```

Test whether a given range is completely within an fpage.

Parameters

<i>fpage</i>	Flex page
<i>addr</i>	Address
<i>size</i>	Size of range in log2.

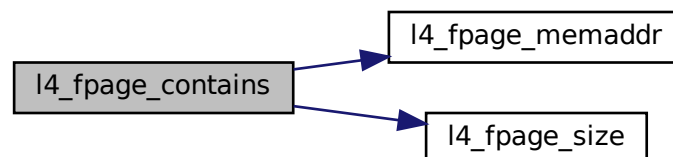
Return values

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

Definition at line 687 of file [__l4_fpage.h](#).

References [l4_fpage_memaddr\(\)](#), and [l4_fpage_size\(\)](#).

Here is the call graph for this function:



13.35.3.4 l4_fpage_invalid()

```
l4_fpage_t l4_fpage_invalid (
    void ) [inline]
```

Get an invalid flex page.

Returns

Special *invalid* flex page.

Definition at line 661 of file `__l4_fpage.h`.

13.35.3.5 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 591 of file `__l4_fpage.h`.

13.35.3.6 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.

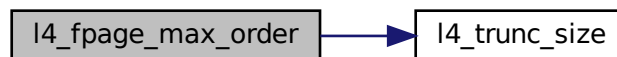
Note

The start address of the flex-page can be determined with `l4_trunc_size(addr, returnvalue)`

Definition at line 695 of file `__l4_fpage.h`.

References `l4_trunc_size()`.

Here is the call graph for this function:

**13.35.3.7 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 597 of file `__l4_fpage.h`.

Referenced by `l4_fpage_contains()`.

Here is the caller graph for this function:

**13.35.3.8 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 603 of file `__l4_fpage.h`.

13.35.3.9 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 585 of file [__l4_fpage.h](#).

13.35.3.10 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 567 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:



13.35.3.11 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 626 of file `__l4_fpage.h`.

References `l4_fpage_t::raw`.

13.35.3.12 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.

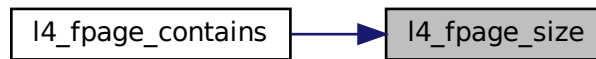
See also

[l4_fpage_memaddr\(\)](#), [l4_fpage_obj\(\)](#), [l4_fpage_ioport\(\)](#)

Definition at line 579 of file [__l4_fpage.h](#).

Referenced by [l4_fpage_contains\(\)](#).

Here is the caller graph for this function:

**13.35.3.13 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 573 of file [__l4_fpage.h](#).

13.35.3.14 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

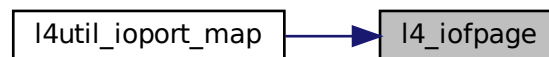
I/O flex page

Definition at line 641 of file [__l4_fpage.h](#).

References [L4_FPAGE_ADDR_SHIFT](#).

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

Here is the caller graph for this function:



13.35.3.15 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 668 of file [__l4_fpage.h](#).

References [l4_fpage_rights\(\)](#), and [L4_FPAGE_W](#).

Here is the call graph for this function:



13.35.3.16 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.

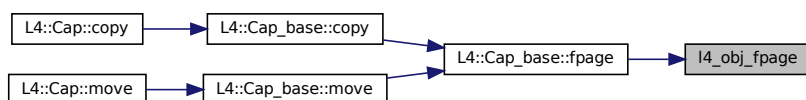
Note

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

Definition at line 647 of file [__l4_fpage.h](#).

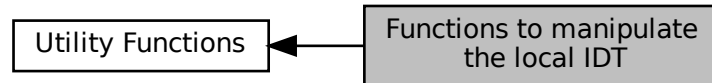
Referenced by [L4::Cap_base::fpage\(\)](#).

Here is the caller graph for this function:



13.36 Functions to manipulate the local IDT

Collaboration diagram for Functions to manipulate the local IDT:



13.37 IA32 Port I/O API

Collaboration diagram for IA32 Port I/O API:



13.38 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.

13.38.1 Detailed Description

13.38.2 Typedef Documentation

13.38.2.1 `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`.

13.38.3 Enumeration Type Documentation

13.38.3.1 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](#).

13.38.3.2 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](#).

13.38.3.3 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](#).

13.38.4 Function Documentation

13.38.4.1 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.

13.38.4.2 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
----------------	----------------

Return values

<i>dev_handle</i>	Device handle for found device, can be NULL.
<i>dev</i>	Device information, filled by the function, can be NULL.
<i>res_handle</i>	Resource handle, can be NULL.

Returns

0 on success, error code otherwise

13.38.4.3 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)
<i>reshandle</i>	Resource handle, start with handle returned by device functions.

Return values

<i>reshandle</i>	Next resource handle.
<i>res</i>	Device descriptor

Returns

0 on success, error code otherwise, esp. -L4_ENOENT if no more resources found

13.38.4.4 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

13.38.4.5 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

13.38.4.6 `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.

13.38.4.7 `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

<i>0</i>	Success.
<i>-L4_ENOENT</i>	No area could be found to map the IO memory region.
<i>-L4_EPERM</i>	Operation not allowed.
<i>-L4_EINVAL</i>	Invalid value.
<i>-L4_EADDRNOTAVAIL</i>	The requested virtual address is not available.
<i>-L4_ENOMEM</i>	The requested IO memory region could not be allocated.
<i><0</i>	IPC errors.

Note

This function uses [L4Re](#) functionality to reserve a part of the virtual address space of the caller.

13.38.4.8 `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

13.38.4.9 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.

13.39 IPC-Gate API

Secure communication object.

Collaboration diagram for IPC-Gate API:

**Functions**

- `l4_msgtag_t l4_ipc_gate_bind_thread (l4_cap_idx_t gate, l4_cap_idx_t thread, l4_umword_t label)`
Bind the IPC gate to a thread.
- `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 gate to a thread.

13.39.1 Detailed Description

Secure communication object.

IPC-Gate objects provide a means to establish secure communication channels to [L4 Threads \(Thread\)](#). An IPC-Gate object can be created using a [Factory \(l4_factory_create_gate\(\)\)](#) and get assigned a specific [L4 thread](#) and a *label* as protected payload. The *label* has the size of one machine word and can only be seen by the Task running the thread that is assigned of the IPC-gate. The *label* is received as part of the IPC message. The *label* can thus be used to securely identify the IPC-gate that was used to send a message.

An IPC-gate is usually used to represent an user-level object and may be the address of the data structure for the object in the server task.

With client privileges an IPC-gate does not provide any direct API and thus an IPC-gate kernel object cannot be modified by invocations. Each invocation of an IPC-gate kernel object is translated into an IPC message to the assigned thread. To invoke a method of the IPC-gate, it has to be mapped with the [L4_FPAGE_C_IPCGATE_SVR](#) right.

Include File

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

For the C++ interface refer to the [L4::ipc_gate](#) documentation.

See also

[Object Invocation](#)

13.39.2 Function Documentation

13.39.2.1 l4_ipc_gate_bind_thread()

```
l4_msgtag_t l4_ipc_gate_bind_thread (
    l4_cap_idx_t gate,
    l4_cap_idx_t thread,
    l4_umword_t label ) [inline]
```

Bind the IPC gate to a thread.

Parameters

<i>gate</i>	The IPC gate object.
<i>thread</i>	The thread object that shall be bound to <i>gate</i> .
<i>label</i>	Label to assign to <i>gate</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>	<code>thread</code> is not a thread object or other arguments were malformed.
<i>-L4_EPERM</i>	No L4_CAP_FPAGE_S rights on <code>gate</code> or <code>thread</code> .

Definition at line 130 of file [ipc_gate.h](#).

13.39.2.2 `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.

Definition at line 137 of file [ipc_gate.h](#).

13.39.2.3 `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 gate to a thread.

Parameters

<i>ep</i>	The IPC receive endpoint object.
<i>thread</i>	The thread object that shall be bound to <code>ep</code> .
<i>label</i>	Label to assign to <code>ep</code> . 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>thread</code> is not a thread object or other arguments were malformed.
<code>-L4_EPERM</code>	No <code>L4_CAP_FPAGE_S</code> rights on <code>ep</code> or <code>thread</code> .

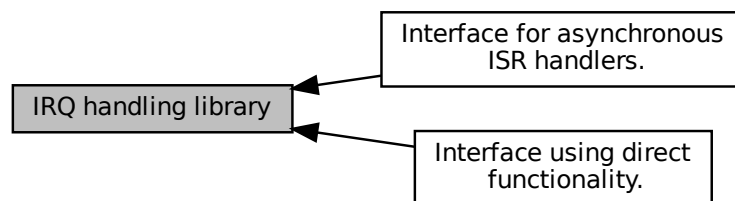
Examples

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

Definition at line 80 of file [rcv_endpoint.h](#).

13.40 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.](#)

13.40.1 Detailed Description

13.41 IRQs

C IRQ 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.
- `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.

13.41.1 Detailed Description

C IRQ 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)

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.

13.41.2 Enumeration Type Documentation

13.41.2.1 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 67 of file [icu.h](#).

13.41.3 Function Documentation

13.41.3.1 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

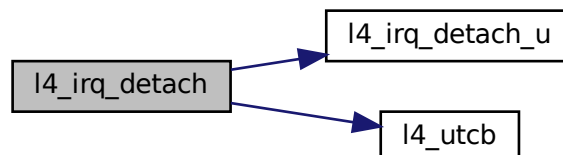
Examples

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

Definition at line 287 of file [irq.h](#).

References [l4_irq_detach_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.41.3.2 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, usually the UTCB of the calling thread.

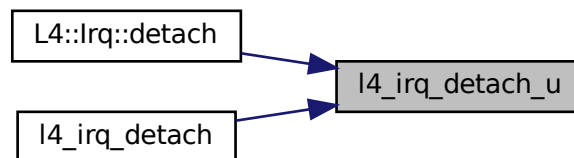
Returns

Syscall return tag

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

Referenced by [L4::Irq::detach\(\)](#), and [l4_irq_detach\(\)](#).

Here is the caller graph for this function:

**13.41.3.3 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.

Parameters

<i>irq</i>	The master IRQ object.
<i>slave</i>	The slave that shall be attached to the master.

Returns

Syscall return tag

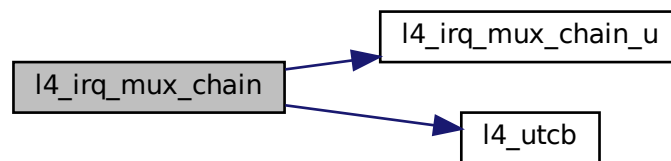
The chaining feature of IRQ objects allows to deal with shared IRQs. For chaining IRQs there must be a master IRQ object, bound to the real IRQ source. Note, the master IRQ must not have a thread bound to it.

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 281 of file [irq.h](#).

References [l4_irq_mux_chain_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.41.3.4 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, usually the UTCB of the calling thread.

Returns

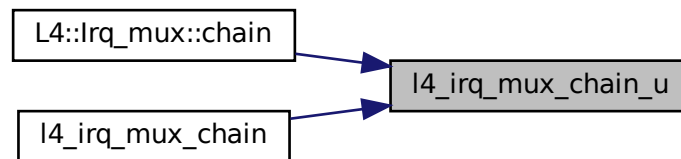
Syscall return tag

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 230 of file [irq.h](#).

Referenced by [L4::l4_irq_mux::chain\(\)](#), and [l4_irq_mux_chain\(\)](#).

Here is the caller graph for this function:



13.41.3.5 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

Examples

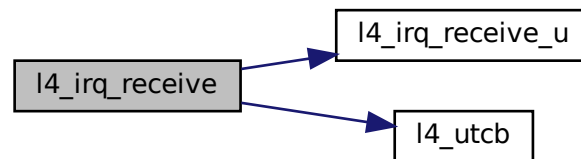
[examples/sys/isr/main.c](#).

Definition at line 299 of file [irq.h](#).

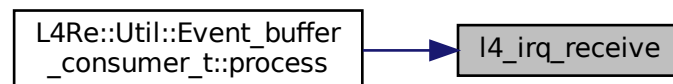
References [l4_irq_receive_u\(\)](#), and [l4_utcb\(\)](#).

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:



13.41.3.6 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, usually the UTCB of the calling thread.

Returns

Syscall return tag

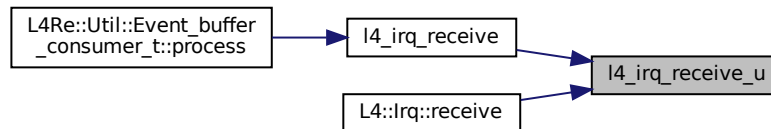
Note

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

Definition at line 257 of file [irq.h](#).

Referenced by [l4_irq_receive\(\)](#), and [L4::Irq::receive\(\)](#).

Here is the caller graph for this function:

**13.41.3.7 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.

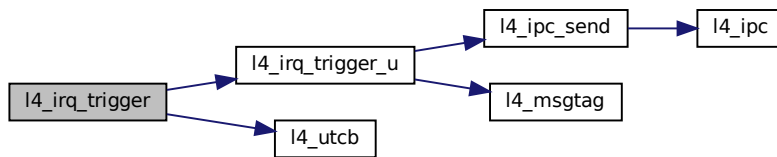
Note that this function is a send only operation, i.e. there is no return value except for a failed send operation. Especially [l4_error\(\)](#) will return an error value from the message tag which still contains the IRQ protocol used for the send operation.

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

Definition at line 293 of file [irq.h](#).

References [l4_irq_trigger_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.41.3.8 l4_irq_trigger_u()

```

l4_msgtag_t l4_irq_trigger_u (
    l4_cap_idx_t irq,
    l4_utcb_t * utcb ) [inline]
  
```

Trigger.

Parameters

<i>irq</i>	The IRQ object that shall be triggered.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag for a send-only operation, use [l4_ipc_error\(\)](#) to check for errors (**do not** use [l4_error\(\)](#)).

Note

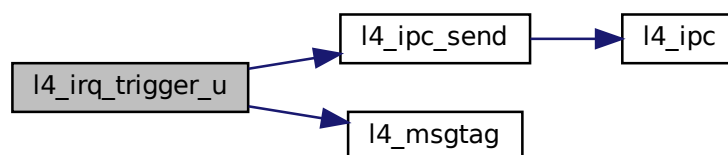
This function is a send-only operation, this means there is no return value except for a failed send operation. Use [l4_ipc_error\(\)](#) to check for errors, **do not** use [l4_error\(\)](#), because [l4_error\(\)](#) will always return an error.

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

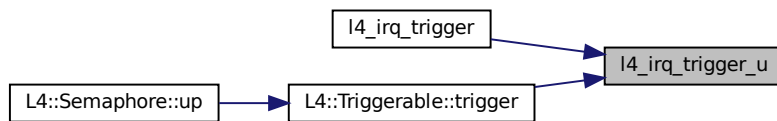
References [l4_ipc_send\(\)](#), [l4_msgtag\(\)](#), and [L4_PROTO_IRQ](#).

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:



13.41.3.9 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

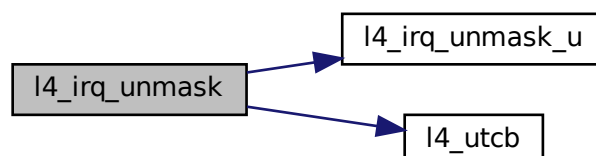
Note

`l4_irq_wait()` and `l4_irq_receive()` are doing the unmask themselves.

Definition at line 312 of file `irq.h`.

References `l4_irq_unmask_u()`, and `l4_utcb()`.

Here is the call graph for this function:



13.41.3.10 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, usually the UTCB of the calling thread.

Returns

Syscall return tag for a send-only operation, use [l4_ipc_error\(\)](#) to check for errors (**do not** use [l4_error\(\)](#)).

Note

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 errors, **do not** use [l4_error\(\)](#), because [l4_error\(\)](#) will always return an 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 273 of file [irq.h](#).

Referenced by [l4_irq_unmask\(\)](#).

Here is the caller graph for this function:



13.41.3.11 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.

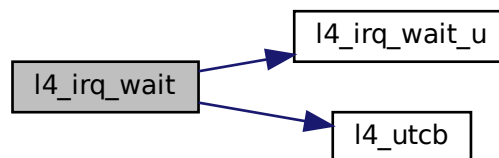
Returns

Syscall return tag

Definition at line 305 of file [irq.h](#).

References [l4_irq_wait_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.41.3.12 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, usually the UTCB of the calling thread.

Returns

Syscall return tag

Definition at line 264 of file [irq.h](#).

Referenced by [l4_irq_wait\(\)](#).

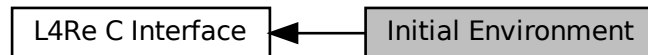
Here is the caller graph for this function:



13.42 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 * 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.

13.42.1 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](#).

13.42.2 Function Documentation

13.42.2.1 l4re_env()

```
l4re\_env\_t * l4re_env (
    void ) [inline]
```

Get [L4Re](#) initial environment.

Returns

Pointer to [L4Re](#) initial environment.

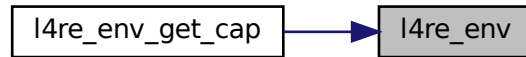
Examples

[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 185 of file [env.h](#).

Referenced by [l4re_env_get_cap\(\)](#).

Here is the caller graph for this function:



13.42.2.2 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\(\)](#)).

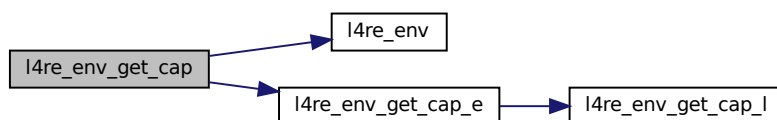
Examples

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

Definition at line 227 of file [env.h](#).

References [l4re_env\(\)](#), and [l4re_env_get_cap_e\(\)](#).

Here is the call graph for this function:



13.42.2.3 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 214 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:



13.42.2.4 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

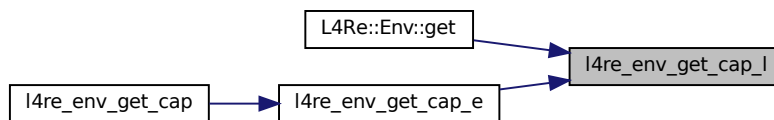
A pointer to an `l4re_env_cap_entry_t` if the object exists or NULL if not.

Definition at line 196 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:



13.42.2.5 l4re_kip()

```
l4_kernel_info_t * l4re_kip (
    void ) [inline]
```

Get Kernel Info Page.

Returns

Pointer to Kernel Info Page (KIP) structure.

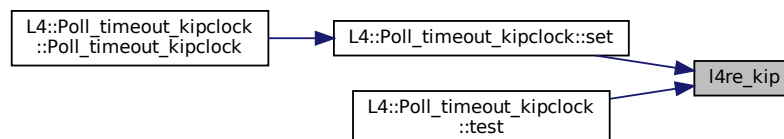
Examples

`examples/sys/aliens/main.c`, and `examples/sys/ux-vhw/main.c`.

Definition at line 189 of file `env.h`.

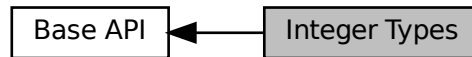
Referenced by `L4::Poll_timeout_kipclock::set()`, and `L4::Poll_timeout_kipclock::test()`.

Here is the caller graph for this function:



13.43 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.

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.

13.43.1 Detailed Description

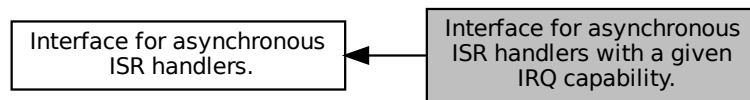
Include File

```
#include <l4/sys/l4int.h>
```

13.44 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.

13.44.1 Detailed Description

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

13.44.2 Function Documentation

13.44.2.1 `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](#)

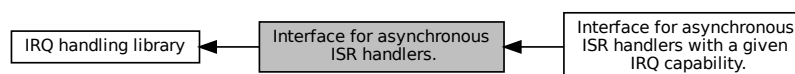
Returns

Pointer to `l4irq_t` structure, 0 on error

13.45 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.

13.45.1 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.

13.45.2 Function Documentation

13.45.2.1 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](#).

13.45.2.2 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 isr_handler
<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

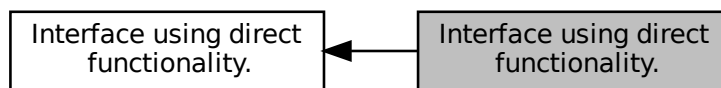
Pointer to `l4irq_t` structure, 0 on error

Examples

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

13.46 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.

13.46.1 Detailed Description

13.46.2 Function Documentation

13.46.2.1 `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.

13.46.2.2 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.

13.46.2.3 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.

13.46.2.4 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](#)

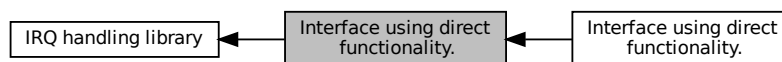
Returns

Pointer to `l4irq_t` structure, 0 on error

The pointer to the IRQ structure is used as a label in the IRQ object.

13.47 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.

13.47.1 Detailed Description

13.47.2 Function Documentation

13.47.2.1 `l4irq_attach()`

```
l4irq_t* l4irq_attach (
    int irqnum )
```

Attach/connect to IRQ.

Parameters

<code>irqnum</code>	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](#).

13.47.2.2 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.

13.47.2.3 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.

13.47.2.4 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.

13.47.2.5 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

13.47.2.6 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`.

13.47.2.7 `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

<code>unmask_irq</code>	IRQ data structure for unmask.
-------------------------	--------------------------------

Return values

<code>ret_irq</code>	Received interrupt.
----------------------	---------------------

Returns

0 on success, != 0 on error

13.47.2.8 `l4irq_wait()`

```
long l4irq_wait (
    l4irq_t * irq )
```

Wait for specified IRQ.

Parameters

<code>irq</code>	IRQ data structure
------------------	--------------------

Returns

0 on success, != 0 on error

Examples

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

13.47.2.9 `l4irq_wait_any()`

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

Wait for any attached IRQ.

Return values

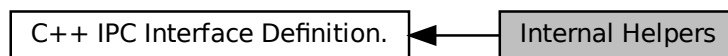
<i>irq</i>	Received interrupt.
------------	---------------------

Returns

0 on success, != 0 on error

13.48 Internal Helpers

Collaboration diagram for Internal Helpers:



Data Structures

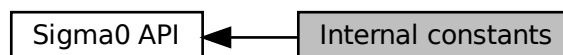
- struct `L4::Types::Bool< V >`
Boolean meta type.
- struct `L4::Types::False`
False meta value.
- struct `L4::Types::True`
True meta value.
- struct `L4::Types::Same< A, B >`
Compare two data types for equality.

13.48.1 Detailed Description

13.49 Internal constants

Internal sigma0 definitions.

Collaboration diagram for Internal constants:



Internal sigma0 definitions.

13.50 Internal functions

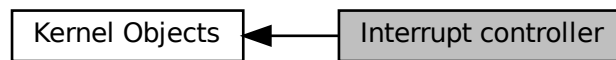
Collaboration diagram for Internal functions:



13.51 Interrupt controller

The C Icu 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 capabilities of the ICU.
- [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 capabilities of the ICU.
- [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](#)
Acknowledge 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.

13.51.1 Detailed Description

The C Icu interface.

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 bind the IRQ capability to a thread
3. [l4_icu_bind\(\)](#)
4. [l4_icu_unmask\(\)](#) to receive the first IRQ

Include File

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

13.51.2 Typedef Documentation

13.51.2.1 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\(\)](#).

13.51.3 Enumeration Type Documentation

13.51.3.1 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 50 of file [icu.h](#).

13.51.4 Function Documentation

13.51.4.1 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\(\)](#).

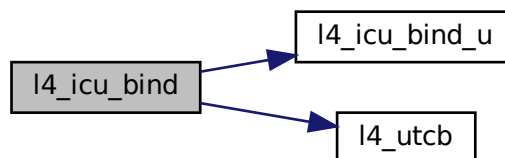
Examples

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

Definition at line 473 of file [icu.h](#).

References [l4_icu_bind_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.51.4.2 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, usually the UTCB of the calling thread.

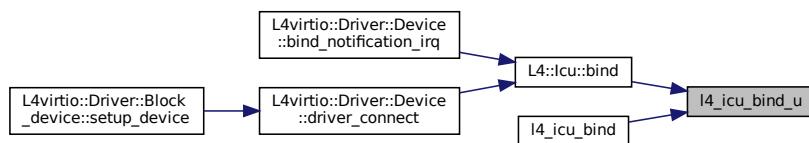
Returns

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](#).

Definition at line 373 of file [icu.h](#).

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

Here is the caller graph for this function:



13.51.4.3 l4_icu_info()

```

l4_msgtag_t l4_icu_info (
    l4_cap_idx_t icu,
    l4_icu_info_t * info ) [inline]

```

Get information about the capabilities of the ICU.

Parameters

	<i>icu</i>	The ICU object from which information shall be retrieved.
out	<i>info</i>	Pointer to an info structure to be filled with information. The memory for this structure has to be allocated by the caller.

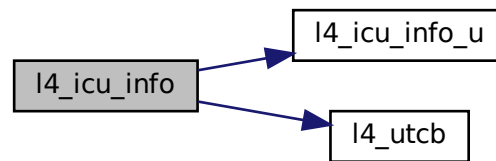
Returns

Syscall return tag

Definition at line 481 of file [icu.h](#).

References [l4_icu_info_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.51.4.4 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 capabilities of the ICU.

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, usually the UTCB of the calling thread.

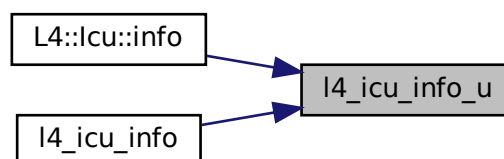
Returns

Syscall return tag

Definition at line 397 of file [icu.h](#).

Referenced by [L4::Icu::info\(\)](#), and [l4_icu_info\(\)](#).

Here is the caller graph for this function:



13.51.4.5 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 'irqnum' shall be masked.
<i>irqnum</i>	IRQ line at the ICU.
<i>label</i>	If non-NULL the function also waits for the next message.
<i>to</i>	Timeout for message to ICU, if unsure use L4_IPC_NEVER.

Returns

Syscall return tag

Definition at line 495 of file [icu.h](#).

13.51.4.6 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 'irqnum' 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.
<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, usually the UTCB of the calling thread.

Returns

Syscall return tag

Definition at line 460 of file [icu.h](#).

Referenced by [L4::Icu::mask\(\)](#).

Here is the caller graph for this function:

**13.51.4.7 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 l4_icu_msi_info_t structure receiving the address and the data value to trigger this MSI.

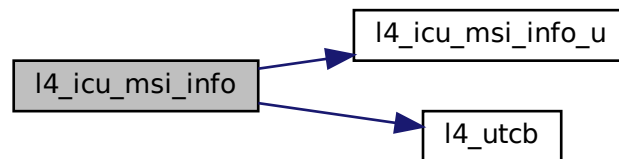
Returns

Syscall return tag

Definition at line 485 of file [icu.h](#).

References [l4_icu_msi_info_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.51.4.8 `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, usually the UTCB of the calling thread.
	<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 411 of file [icu.h](#).

Referenced by [l4_icu_msi_info\(\)](#).

Here is the caller graph for this function:



13.51.4.9 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 .

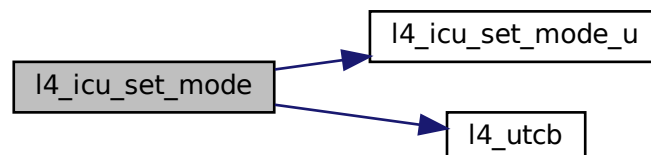
Returns

Syscall return tag

Definition at line [500](#) of file [icu.h](#).

References [l4_icu_set_mode_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.51.4.10 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, usually the UTCB of the calling thread.

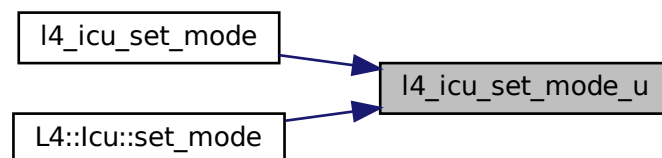
Returns

Syscall return tag

Definition at line [434](#) of file [icu.h](#).

Referenced by [l4_icu_set_mode\(\)](#), and [L4::Icu::set_mode\(\)](#).

Here is the caller graph for this function:



13.51.4.11 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.

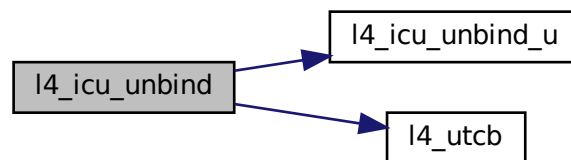
Returns

Syscall return tag

Definition at line 477 of file [icu.h](#).

References [l4_icu_unbind_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

13.51.4.12 `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, usually the UTCB of the calling thread.

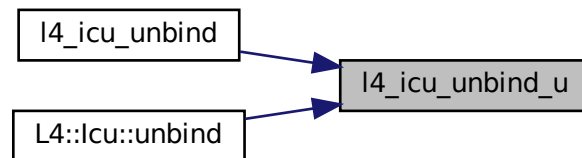
Returns

Syscall return tag

Definition at line 385 of file [icu.h](#).

Referenced by [l4_icu_unbind\(\)](#), and [L4::Icu::unbind\(\)](#).

Here is the caller graph for this function:

**13.51.4.13 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 waits for the next message.
<i>to</i>	Timeout for message to ICU, if unsure use L4_IPC_NEVER.

Returns

Syscall return tag, the error values therein are undefined because [l4_icu_unmask\(\)](#) is a sender-only IPC.

Definition at line 490 of file [icu.h](#).

13.51.4.14 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]
```

Acknowledge the given interrupt line.

Parameters

	<i>icu</i>	The ICU object where the IRQ line shall be unmasked.
	<i>irqnum</i>	The interrupt line that shall be acknowledged.
out	<i>label</i>	If NULL this is a send-only unmask, if not NULL then 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, usually the UTCB of the calling thread.

Returns

Syscall return tag.

Note

If *label* is NULL this function is a send-only operation and there is no return value except for a failed send operation. In this case use [l4_ipc_error\(\)](#) to check for errors, **do not** use [l4_error\(\)](#), because [l4_error\(\)](#) will always return an error.

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

13.52 Kernel Debugger

Kernel debugger related functionality.

Collaboration diagram for 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.
- 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.

13.52.1 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>
```

13.52.2 Function Documentation

13.52.2.1 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

$\sim 0UL$	Capability is not valid.
≥ 0	Global debugger id.

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

Definition at line [346](#) of file [debugger.h](#).

13.52.2.2 l4_debugger_kobj_to_id()

```
unsigned long l4_debugger_kobj_to_id (
    l4_cap_idx_t cap,
    l4_addr_t kobjp ) [inline]
```

Get the globally unique ID of the object behind the kobject pointer.

Parameters

<i>cap</i>	Capability
<i>kobjp</i>	Kobject pointer

Return values

$\sim 0UL$	The capability or the kobject pointer are invalid.
≥ 0	The globally unique id.

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

Definition at line 352 of file [debugger.h](#).

13.52.2.3 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

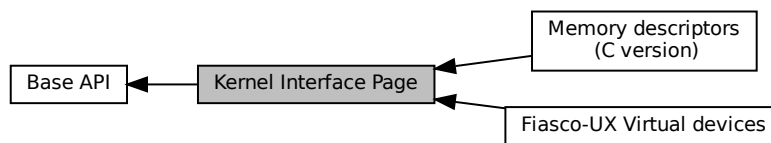
[examples/sys/aliens/main.c](#).

Definition at line 339 of file [debugger.h](#).

13.53 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.

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.

13.53.1 Detailed Description

Kernel Interface Page.

C interface for the Kernel Interface Page:

C++ interface for the Kernel Interface Page:

Include File

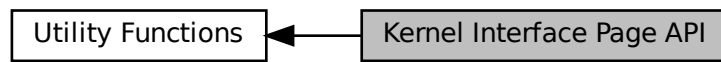
```
#include <l4/sys/kip>
```

Include File

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

13.54 Kernel Interface Page API

Collaboration diagram for Kernel Interface Page API:



Files

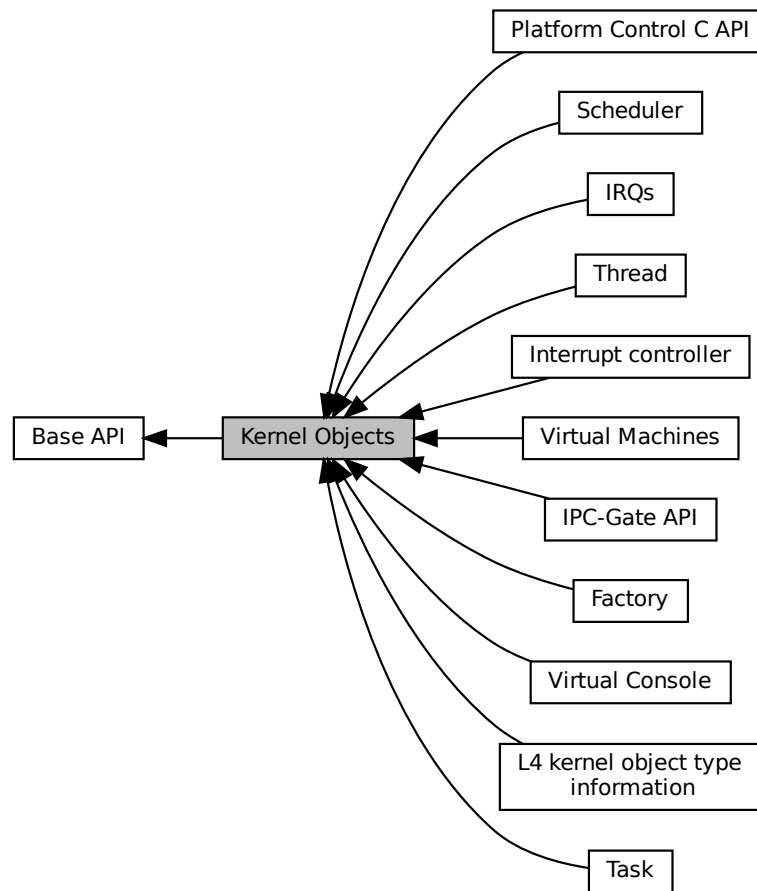
- file [kip.h](#)

13.54.1 Detailed Description

13.55 Kernel Objects

API of kernel objects.

Collaboration diagram for Kernel Objects:



Modules

- [Factory](#)
C factory interface to create kernel objects.
- [IPC-Gate API](#)
Secure communication object.
- [IRQs](#)
C IRQ interface.
- [Interrupt controller](#)
The C Icu 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.
- [Scheduler](#)
C interface of the Scheduler kernel object.
- [Task](#)

C interface of the Task kernel object.

- [Thread](#)

Thread object.

- [Virtual Console](#)

Virtual console for simple character based input and output.

- [Virtual Machines](#)

Virtual Machine API.

Data Structures

- class [L4::Vm](#)

Virtual machine host address space.

- class [L4::Kobject](#)

Base class for all kinds of kernel objects and remote objects, referenced by capabilities.

13.55.1 Detailed Description

API of kernel objects.

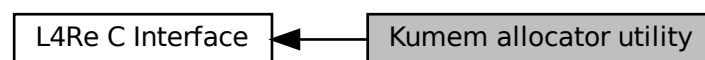
Include File

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

13.56 Kumem allocator utility

Kumem allocator utility C interface.

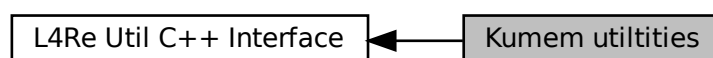
Collaboration diagram for Kumem allocator utility:



Kumem allocator utility C interface.

13.57 Kumem utilities

Collaboration diagram for Kumem utilities:



13.58 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_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.

13.58.1 Detailed Description

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

13.58.2 Enumeration Type Documentation

13.58.2.1 L4_icu_opcode

```
enum L4\_icu\_opcode
```

Opcodes to the ICU interface.

Enumerator

L4_ICU_OP_BIND	Bind opcode. See also l4_icu_bind()
L4_ICU_OP_UNBIND	Unbind opcode. See also l4_icu_unbind()
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 93 of file [icu.h](#).

13.58.2.2 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 94 of file [ipc_gate.h](#).

13.58.2.3 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_ENABLE_OP	enable an offline CPU
L4_PLATFORM_CTL_CPU_DISABLE_OP	disable an online CPU

Definition at line 135 of file [platform_control.h](#).

13.58.2.4 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 254 of file [task.h](#).

13.58.2.5 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 621 of file [thread.h](#).

13.58.2.6 L4_vcon_ops

enum [L4_vcon_ops](#)

Operations on vcon objects.

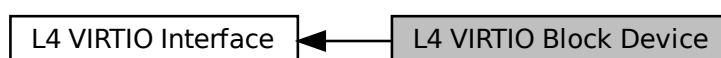
Enumerator

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 264 of file [vcon.h](#).

13.59 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.

13.59.1 Detailed Description

13.59.2 Enumeration Type Documentation

13.59.2.1 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](#).

13.59.2.2 L4virtio_block_status

enum [L4virtio_block_status](#)

Status of a finished block request.

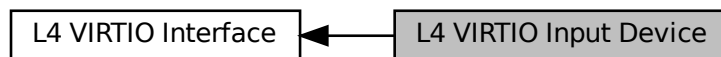
Enumerator

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](#).

13.60 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.

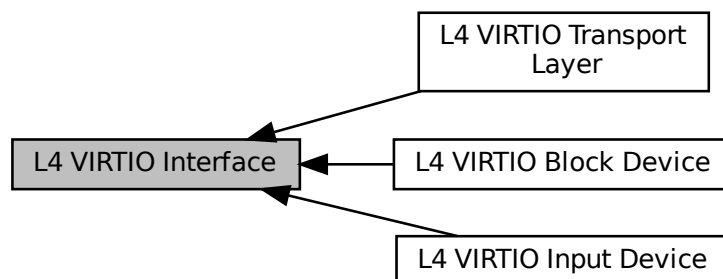
Enumerations

- enum [L4virtio_input_config_select](#)
Device information selectors.

13.60.1 Detailed Description

13.61 L4 VIRTIO Interface

Collaboration diagram for L4 VIRTIO Interface:



Modules

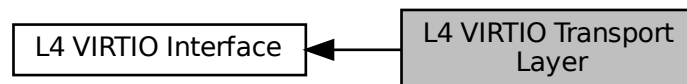
- [L4 VIRTIO Block Device](#)
- [L4 VIRTIO Input Device](#)
- [L4 VIRTIO Transport Layer](#)
L4 specific VIRTIO Transport layer.

13.61.1 Detailed Description

13.62 L4 VIRTIO Transport Layer

L4 specific VIRTIO Transport layer.

Collaboration diagram for L4 VIRTIO Transport Layer:



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` , `L4VIRTIO_OP_REGISTER_IFACE` , `L4VIRTIO_OP_REGISTER_DS` }
L4-VIRTIO opcodes.
- 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_CRYPT` = 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`
Write the VIRTIO status register.
- `int l4virtio_config_queue (l4_cap_idx_t cap, unsigned queue) L4_NOTHROW`
Trigger queue configuration of the given queue.
- `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`
Register a shared data space with VIRTIO host.
- `int l4virtio_register_iface (l4_cap_idx_t cap, l4_cap_idx_t guest_irq, l4_cap_idx_t host_irq, l4_cap_idx_t config_ds) L4_NOTHROW`
Register client to the L4-VIRTIO device.
- `int l4virtio_device_config_ds (l4_cap_idx_t cap, l4_cap_idx_t config_ds, l4_addr_t *ds_offset) L4_NOTHROW`
Get the dataspace with the L4virtio configuration page.
- `int l4virtio_device_notification_irq (l4_cap_idx_t cap, unsigned index, l4_cap_idx_t irq) L4_NOTHROW`
Get the notification interrupt corresponding to the given index.

13.62.1 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.

13.62.2 Typedef Documentation

13.62.2.1 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.

13.62.3 Enumeration Type Documentation

13.62.3.1 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 111 of file [virtio.h](#).

13.62.3.2 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

<code>L4VIRTIO_IRQ_STATUS_VRING</code>	VRING IRQ pending flag.
<code>L4VIRTIO_IRQ_STATUS_CONFIG</code>	CONFIG IRQ pending flag.

Definition at line 102 of file [virtio.h](#).

13.62.3.3 L4_virtio_opcodes

```
enum L4_virtio_opcodes
```

L4-VIRTIO opcodes.

Enumerator

<code>L4VIRTIO_OP_SET_STATUS</code>	Set status register in device config.
<code>L4VIRTIO_OP_CONFIG_QUEUE</code>	Set queue config in device config.
<code>L4VIRTIO_OP_REGISTER_IFACE</code>	Register a transport driver to the device.
<code>L4VIRTIO_OP_REGISTER_DS</code>	Register a data space as transport memory.

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

13.62.3.4 L4virtio_device_ids

```
enum L4virtio_device_ids
```

Virtio device IDs as reported in the driver's config space.

Enumerator

<code>L4VIRTIO_ID_NET</code>	Virtual ethernet card.
<code>L4VIRTIO_ID_BLOCK</code>	General block device.
<code>L4VIRTIO_ID_CONSOLE</code>	Simple device for data IO via ports.
<code>L4VIRTIO_ID_RNG</code>	Entropy source.
<code>L4VIRTIO_ID_BALLOON</code>	Memory ballooning device.
<code>L4VIRTIO_ID_RPMMSG</code>	Device using rpmsg protocol.
<code>L4VIRTIO_ID_SCSI</code>	SCSI host device.
<code>L4VIRTIO_ID_9P</code>	Device using 9P transport protocol.

Enumerator

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 58 of file [virtio.h](#).

13.62.3.5 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 79 of file [virtio.h](#).

13.62.3.6 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 90 of file [virtio.h](#).

13.62.4 Function Documentation

13.62.4.1 l4virtio_config_queue()

```
int l4virtio_config_queue (
    l4_cap_idx_t cap,
    unsigned queue )
```

Trigger queue configuration of the given queue.

Parameters

<i>cap</i>	Capability to the VIRTIO host.
------------	--------------------------------

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.

13.62.4.2 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 232 of file [virtio.h](#).

References [l4virtio_config_hdr_t::queues_offset](#).

13.62.4.3 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 243 of file [virtio.h](#).

13.62.4.4 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 )
```

Get the dataspace with the [L4virtio](#) configuration page.

Parameters

<i>cap</i>	Capability to the L4-VIRTIO host
<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.

13.62.4.5 l4virtio_device_notification_irq()

```
int l4virtio_device_notification_irq (
    l4_cap_idx_t cap,
    unsigned index,
    l4_cap_idx_t irq )
```

Get the notification interrupt corresponding to the given index.

Parameters

	<i>cap</i>	Capability to the L4-VIRTIO host
	<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.

13.62.4.6 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 )
```

Register a shared data space with VIRTIO host.

Parameters

<i>cap</i>	Capability to the VIRTIO host
<i>ds_cap</i>	Data-space 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

Returns

0 on success, < 0 on error

13.62.4.7 l4virtio_register_iface()

```
int l4virtio_register_iface (
    l4_cap_idx_t cap,
```

```

l4_cap_idx_t guest_irq,
l4_cap_idx_t host_irq,
l4_cap_idx_t config_ds )

```

Register client to the L4-VIRTIO device.

Parameters

<i>cap</i>	Capability to the L4-VIRTIO host
<i>guest_irq</i>	IRQ capability for valid IRQ object for device-to-driver notifications.
<i>host_irq</i>	Capability selector for receiving the driver-to-device notifications IRQ capability.
<i>config_ds</i>	Capability for receiving the dataspace capability for the shared L4-VIRTIO config data space.

Return values

<i>L4_EOK</i>	Success.
<i>-L4_ENOSYS</i>	This interface is no longer supported by the server. Use <code>get_device_config()</code> etc. instead.

Deprecated Use `device_config()`, `device_notification_irq()` and `lcu::bind()` instead.

13.62.4.8 l4virtio_set_status()

```

int l4virtio_set_status (
    l4_cap_idx_t cap,
    unsigned status )

```

Write the VIRTIO status register.

Parameters

<i>cap</i>	Capability to the VIRTIO host
<i>status</i>	Status word to write to the VIRTIO status.

Return values

<i>0</i>	on success.
----------	-------------

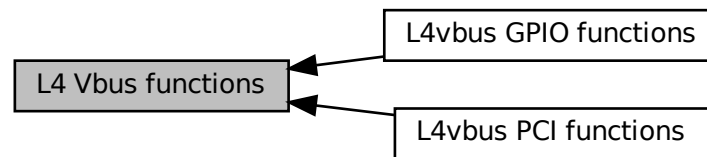
Note

All other registers are accessed via shared memory.

13.63 L4 Vbus functions

C interface of the Vbus API.

Collaboration diagram for L4 Vbus functions:



Modules

- [L4vbus GPIO functions](#)
- [L4vbus PCI 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_request_ioport](#) ([l4_cap_idx_t](#) vbus, [l4vbus_resource_t](#) const *res)
Request an IO port resource.
- int [l4vbus_request_resource](#) ([l4_cap_idx_t](#) vbus, [l4vbus_resource_t](#) const *res, int flags)
Request a resource of a specific type.
- 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 ([L4::Task](#)) 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_release_resource](#) ([l4_cap_idx_t](#) vbus, [l4vbus_resource_t](#) const *res)

Release a previously requested 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.

13.63.1 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.

Include File

```
#include <l4/vbus/vbus.h>
```

Refer to [L4vbus](#) for the C++ API.

13.63.2 Enumeration Type Documentation

13.63.2.1 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 [183](#) of file [vbus.h](#).

13.63.3 Function Documentation

13.63.3.1 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 ([L4::Task](#)) or a [L4Re::Dma_space](#) to a DMA domain.

Parameters

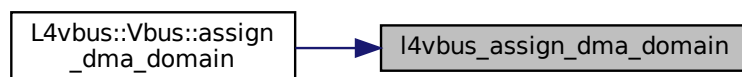
<i>vbus</i>	Capability of the system bus
<i>domain_id</i>	DMA domain ID (resource address of DMA domain found on the vBUS). If the value is ~0U the DMA space of the whole vBUS is used.
<i>flags</i>	A combination of L4vbus_dma_domain_assign_flags .
<i>dma_space</i>	The DMA space capability to bind or 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

<i>0</i>	Operation completed successfully.
<i>-L4_ENOENT</i>	The vbus does not support a global DMA domain or no DMA domain could be found.
<i>-L4_EINVAL</i>	Invalid argument used.
<i>-L4_EBUSY</i>	DMA domain is already active, this means another DMA space is already assigned.

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

Here is the caller graph for this function:



13.63.3.2 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

0	Success.
-L4_ENODEV	No device with the given device handle <i>dev</i> could be found.

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

Here is the caller graph for this function:



13.63.3.3 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

<code>in, out</code>	<code>child</code>	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 (<code>L4VBUS_NULL</code>). If a matching device is found its handle is returned through this parameter.
	<code>hid</code>	HID of the device
	<code>depth</code>	Maximum depth for the recursive lookup
<code>out</code>	<code>devinfo</code>	Device information structure (might be NULL)

Return values

<code>>=0</code>	A device with the given HID was found.
<code>-L4_ENOENT</code>	No device with the given HID could be found on the vbus.
<code>-L4_EINVAL</code>	Invalid or no HID provided.
<code>-L4_ENODEV</code>	Function called on a non-existing device.

13.63.3.4 `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

<code>vbus</code>	Capability of the system bus
<code>dev</code>	Handle of the device
<code>hid</code>	Pointer to a buffer for the HID string
<code>max_len</code>	The size of the buffer (<code>hid</code>)

Returns

the length of the HID string on success, else failure

13.63.3.5 `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)
<i>in, out</i>	<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
<i>out</i>	<i>devinfo</i>	Device information (might be NULL)

Returns

0 on success, else failure

13.63.3.6 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() .
<i>out</i>	<i>res</i>	Descriptor of the resource.

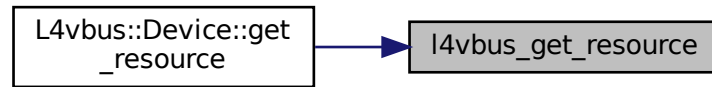
This function returns the resource descriptor of an individual device resource selected by the *res_idx* parameter.

Return values

0	Success.
-L4_ENOENT	Invalid resource index <i>res_idx</i> .

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

Here is the caller graph for this function:



13.63.3.7 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

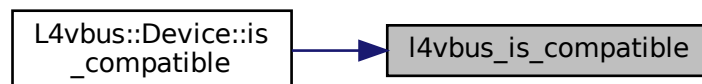
<i>vbus</i>	Capability of the system bus
<i>dev</i>	device handle for which the CID shall be tested
<i>cid</i>	the compatibility ID to test

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:



13.63.3.8 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

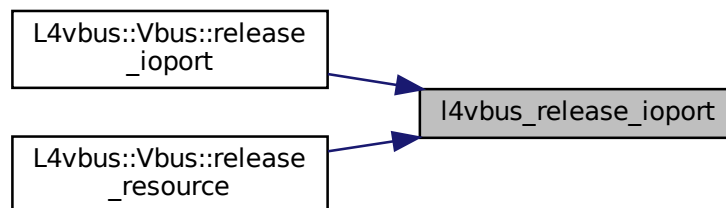
	<i>vbus</i>	Capability of the system bus.
in	<i>res</i>	The IO port resource to be released from the bus.

Returns

>=0 on success, <0 on error.

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

Here is the caller graph for this function:



13.63.3.9 l4vbus_release_resource()

```
int l4vbus_release_resource (
    l4_cap_idx_t vbus,
    l4vbus_resource_t const * res )
```

Release a previously requested resource.

Parameters

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

Returns

≥ 0 on success, < 0 on error.

Deprecated This function is deprecated since Q3 2019. Use [l4vbus_release_ioport\(\)](#) instead.

13.63.3.10 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.

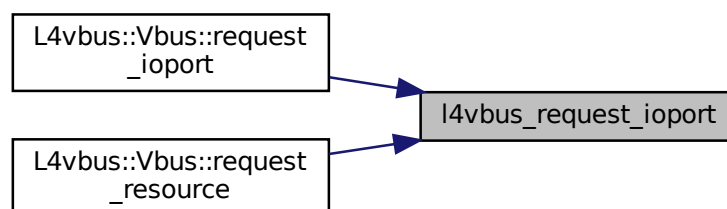
Return values

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\(\)](#), and [L4vbus::Vbus::request_resource\(\)](#).

Here is the caller graph for this function:



13.63.3.11 l4vbus_request_resource()

```
int l4vbus_request_resource (
    l4_cap_idx_t vbus,
    l4vbus_resource_t const * res,
    int flags )
```

Request a resource of a specific type.

Parameters

	<i>vbus</i>	Capability of the system bus.
in	<i>res</i>	Descriptor of the resource.
	<i>flags</i>	Ignored.

Returns

0 on success, else failure.

If any resource is found that contains the requested type and resource address range the resource is obtained.

Deprecated This function is deprecated since Q3 2019. Use [l4vbus_request_ioport\(\)](#) instead.

13.63.3.12 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\(\)](#).

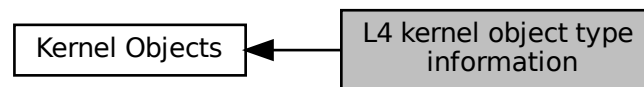
Here is the caller graph for this function:



13.64 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.

13.64.1 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.

13.64.2 Function Documentation

13.64.2.1 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

<i>T</i>	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\(\)](#).

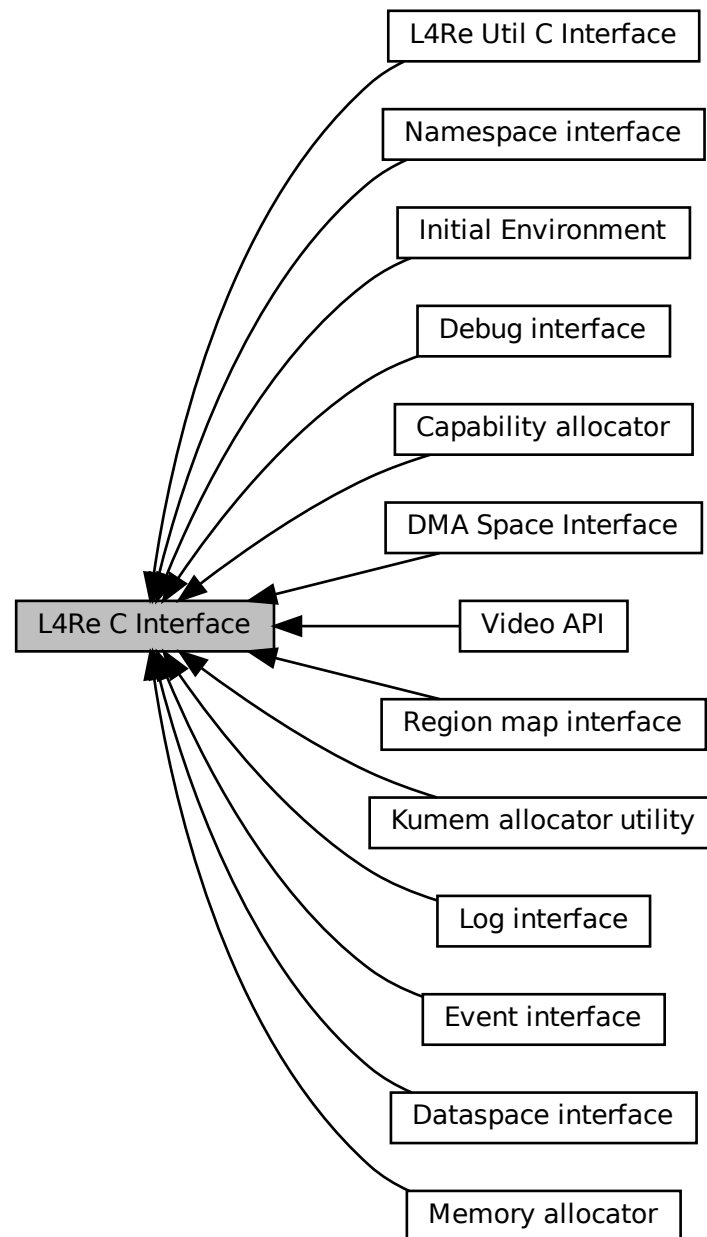
Here is the call graph for this function:



13.65 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](#)

Functions

- [long `l4re_inhibitor_acquire` \(`l4_cap_idx_t` cap, `l4_umword_t` id, char const *reason\)](#)

Inhibitor C interface.

13.65.1 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.

13.65.2 Function Documentation

13.65.2.1 `l4re_inhibitor_acquire()`

```
long l4re_inhibitor_acquire (
    l4_cap_idx_t cap,
    l4_umword_t id,
    char const * reason )
```

Inhibitor C interface.

Acquire an inhibitor lock.

Parameters

<i>cap</i>	Capability for the Inhibitor object (
------------	---------------------------------------

See also

[L4Re::Inhibitor](#))

Parameters

<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

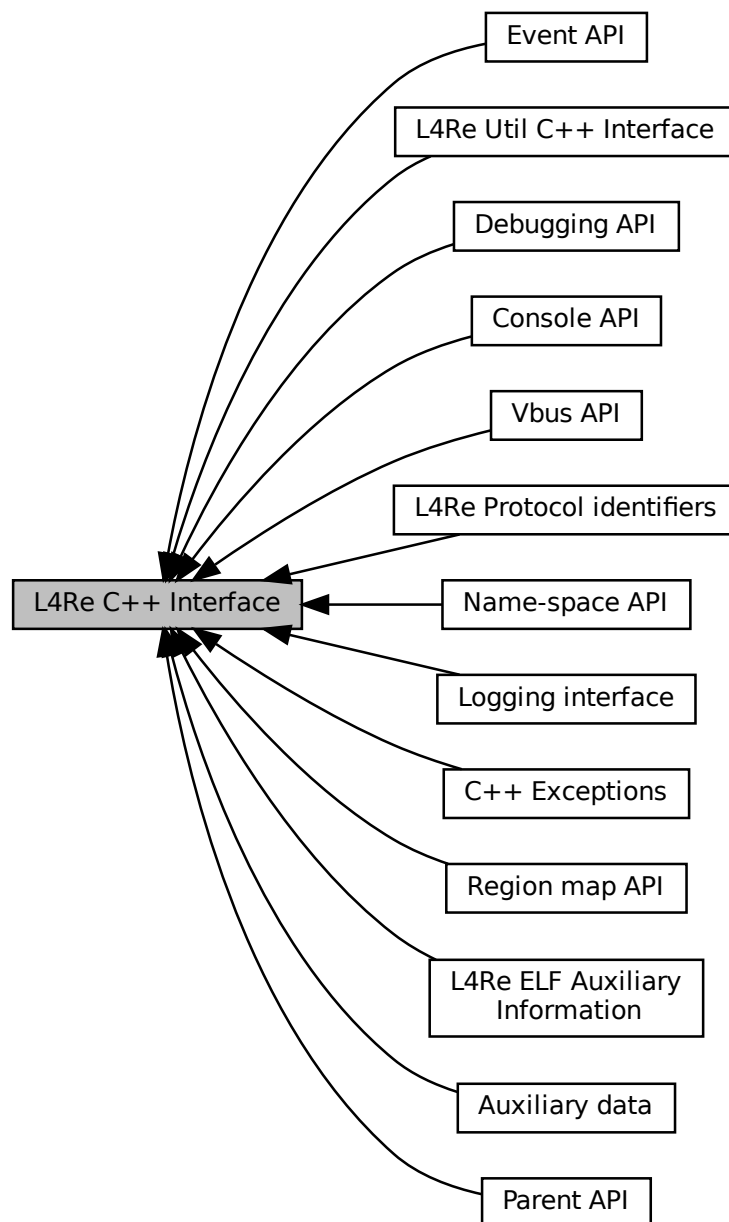
See also

[L4Re::Inhibitor::acquire\(\)](#)

13.66 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.

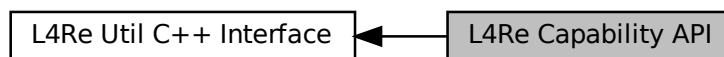
- [Event API](#)
Event API.
- [L4Re ELF Auxiliary Information](#)
API for embedding auxiliary information into binary programs.
- [L4Re Protocol identifiers](#)
Fix [L4Re](#) Protocol Constants.
- [L4Re Util C++ Interface](#)
Documentation of the [L4](#) Runtime Environment utility functionality in C++.
- [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.

13.66.1 Detailed Description

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

13.67 L4Re Capability API

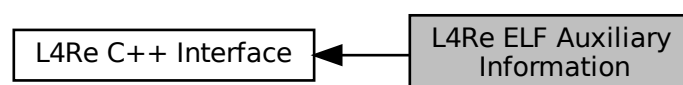
Collaboration diagram for L4Re Capability API:



13.68 L4Re ELF Auxiliary Information

API for embedding auxiliary information into binary programs.

Collaboration diagram for L4Re ELF Auxiliary Information:



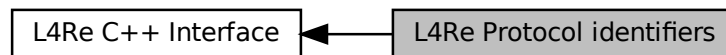
API for embedding auxiliary information into binary programs.

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.

13.69 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.

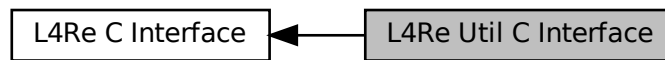
13.69.1 Detailed Description

Fix [L4Re](#) Protocol Constants.

13.70 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.

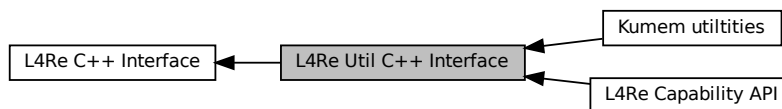
The interface functions closely align with the C++ functions and add no further functionalities.

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

13.71 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

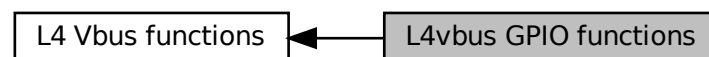
- class [L4Re::Smart_cap_auto< Unmap_flags >](#)
Helper for Unique_cap and Unique_del_cap.
- class [L4Re::Util::Cap_alloc_base](#)
Capability allocator.
- class [L4Re::Util::Br_manager](#)
Buffer-register (BR) manager for L4::Server.
- class [L4Re::Util::Counting_cap_alloc< COUNTERTYPE >](#)
Internal 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.

13.71.1 Detailed Description

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

13.72 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.

13.72.1 Detailed Description

13.72.2 Enumeration Type Documentation

13.72.2.1 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](#).

13.72.2.2 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](#).

13.72.3 Function Documentation

13.72.3.1 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
	<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:



13.72.3.2 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

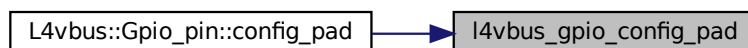
<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>pin</i>	GPIO pin number
<i>func</i>	Hardware specific configuration register, usually offset to the GPIO chip's base address
<i>value</i>	Value which is written into the hardware specific configuration register for the specified pin

Returns

0 if OK, error code otherwise

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

Here is the caller graph for this function:



13.72.3.3 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:



13.72.3.4 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:

**13.72.3.5 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

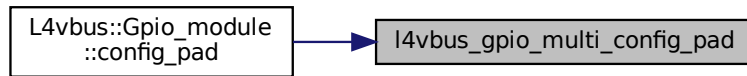
<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>offset</i>	Pin corresponding to the LSB in <i>mask</i> . Note: allowed may be hardware specific.
<i>mask</i>	Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. 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:



13.72.3.6 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:



13.72.3.7 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:



13.72.3.8 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

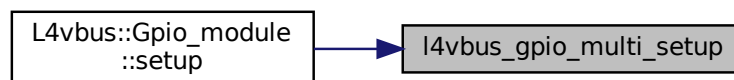
<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>offset</i>	Pin corresponding to the LSB in <i>mask</i> . Note: allowed may be hardware specific.
<i>mask</i>	Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. 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:



13.72.3.9 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:

**13.72.3.10 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

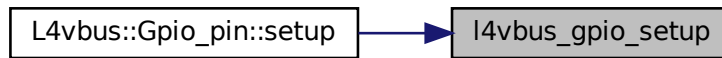
<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>pin</i>	GPIO pin number
<i>mode</i>	GPIO function, see L4vbus_gpio_generic_func for generic functions. Hardware specific functions must be provided in the lower 8 bits.
<i>value</i>	Optional value to set the GPIO pin to if 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:



13.72.3.11 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

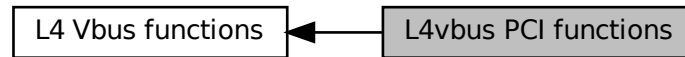
Referenced by [L4vbus::Gpio_pin::to_irq\(\)](#).

Here is the caller graph for this function:



13.73 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.

13.73.1 Detailed Description

13.73.2 Function Documentation

13.73.2.1 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

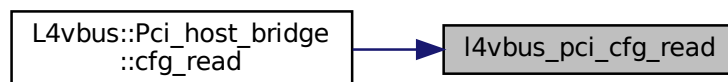
	<i>vbus</i>	Capability of the system bus
	<i>handle</i>	Device handle of the PCI root bridge
	<i>bus</i>	Bus number
	<i>devfn</i>	Device id (upper 16bit) and function (lower 16bit)
	<i>reg</i>	Register in configuration space to read
out	<i>value</i>	Value that has been read
	<i>width</i>	Width to read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



13.73.2.2 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:

**13.73.2.3 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

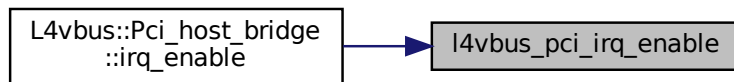
	<i>vbus</i>	Capability of the system bus
	<i>handle</i>	Device handle of the PCI root bridge
	<i>bus</i>	Bus number
	<i>devfn</i>	Device id (upper 16bit) and function (lower 16bit)
	<i>pin</i>	Interrupt pin (normally as reported in configuration register INTR)
out	<i>trigger</i>	False if interrupt is 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:



13.73.2.4 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:



13.73.2.5 l4vbus_pcidev_cfg_write()

```
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.

Parameters

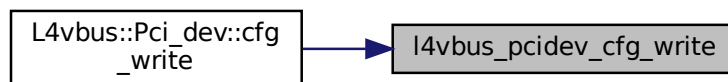
<i>vbus</i>	Capability of the system bus
<i>handle</i>	Device handle of the PCI device
<i>reg</i>	Register in configuration space to write
<i>value</i>	Value to write
<i>width</i>	Width to write in bits (e.g. 8, 16, 32)

Returns

0 on success, else failure

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

Here is the caller graph for this function:



13.73.2.6 l4vbus_pcidev_irq_enable()

```
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.

Parameters

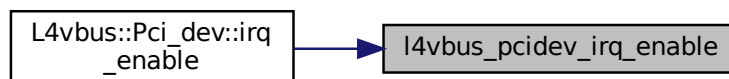
	<i>vbus</i>	Capability of the system bus
	<i>handle</i>	Device handle of the PCI device
out	<i>trigger</i>	False if interrupt is level-triggered
out	<i>polarity</i>	True if interrupt is of low polarity

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:



13.74 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.

13.74.1 Detailed Description

Log C interface.

13.74.2 Function Documentation

13.74.2.1 `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.
---------------	---------------------------------

Returns

0 for success, <0 on error

See also

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

Definition at line 99 of file [log.h](#).

References [l4re_log_print_srv\(\)](#).

Here is the call graph for this function:



13.74.2.2 `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.

Returns

0 for success, <0 on error

See also

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

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

Here is the caller graph for this function:

**13.74.2.3 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.

Returns

0 for success, <0 on error

See also

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

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

References [l4re_log_printn_srv\(\)](#).

Here is the call graph for this function:



13.74.2.4 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.

Returns

0 for success, <0 on error

See also

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

Referenced by [l4re_log_printn\(\)](#).

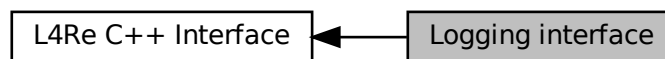
Here is the caller graph for this function:



13.75 Logging interface

Interface for log output.

Collaboration diagram for Logging interface:



Data Structures

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

13.75.1 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.

13.76 Low-Level Thread Functions

Collaboration diagram for Low-Level Thread Functions:



13.77 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_free](#) ([l4re_ds_t](#) const mem) [L4_NOTHROW](#)
Free 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.
- long [l4re_ma_free_srv](#) ([l4_cap_idx_t](#) srv, [l4re_ds_t](#) const mem) [L4_NOTHROW](#)
Free memory.

13.77.1 Detailed Description

Memory allocator C interface.

13.77.2 Enumeration Type Documentation

13.77.2.1 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](#).

13.77.3 Function Documentation

13.77.3.1 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 Mem_alloc_flags::Continuous bit, the allocator tries to allocate as much memory as possible leaving an amount of at least <i>-size</i> bytes within the associated quota.
<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 192 of file [mem_alloc.h](#).

References [l4re_ma_alloc_align_srv\(\)](#).

Here is the call graph for this function:

**13.77.3.2 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.

Return values

<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 200 of file [mem_alloc.h](#).

References [l4re_ma_alloc_align_srv\(\)](#).

Here is the call graph for this function:



13.77.3.3 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

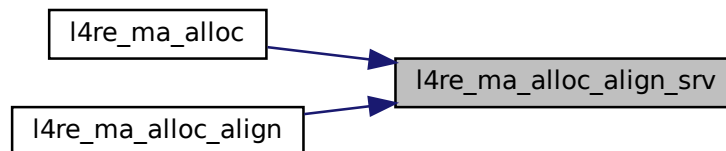
See also

[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:

**13.77.3.4 l4re_ma_free()**

```
long l4re_ma_free (  
    l4re\_ds\_t const mem ) [inline]
```

Free memory.

Parameters

<i>mem</i>	Dataspace to free.
------------	--------------------

Returns

0 on success, <0 on error

See also

[L4Re::Mem_alloc::free](#)

Note

This function is using the [L4Re::Env::env\(\)->mem_alloc\(\)](#) service.

Deprecated This function is deprecated. Use [l4_task_unmap\(\)](#) or similar means to remove a dataspace.

Definition at line 208 of file [mem_alloc.h](#).

References [l4re_ma_free_srv\(\)](#).

Here is the call graph for this function:



13.77.3.5 l4re_ma_free_srv()

```

long l4re_ma_free_srv (
    l4_cap_idx_t srv,
    l4re_ds_t const mem )
  
```

Free memory.

Parameters

<i>srv</i>	Memory allocator service.
<i>mem</i>	Dataspace to free.

Returns

0 on success, <0 on error

See also

[L4Re::Mem_alloc::free](#)

Deprecated This function is deprecated. Use [l4_task_unmap\(\)](#) or similar means to remove a dataspace.

Referenced by [l4re_ma_free\(\)](#).

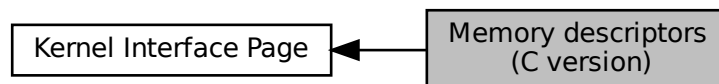
Here is the caller graph for this function:



13.78 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.

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.

13.78.1 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).

13.78.2 Typedef Documentation

13.78.2.1 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.

13.78.3 Enumeration Type Documentation

13.78.3.1 l4_mem_info_sub_type_t

```
enum l4_mem_info_sub_type_t
```

Memory sub types for l4_mem_type_info descriptors.

Enumerator

<code>l4_mem_info_acpi_rsdp</code>	Physical address of the ACPI root pointer.
------------------------------------	--

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

13.78.3.2 `l4_mem_type_t`

```
enum l4_mem_type_t
```

Type of a memory descriptor.

Enumerator

<code>l4_mem_type_undefined</code>	Undefined, unused descriptor.
<code>l4_mem_type_conventional</code>	Conventional memory.
<code>l4_mem_type_reserved</code>	Reserved memory for kernel etc.
<code>l4_mem_type_dedicated</code>	Dedicated memory (some device memory)
<code>l4_mem_type_shared</code>	Shared memory (not implemented)
<code>l4_mem_type_info</code>	Info from the boot loader.
<code>l4_mem_type_bootloader</code>	Memory owned by the boot loader.
<code>l4_mem_type_archspecific</code>	Architecture specific memory (e.g., ACPI memory)

Definition at line 44 of file [memdesc.h](#).

13.78.4 Function Documentation

13.78.4.1 `l4_kernel_info_get_mem_desc_end()`

```
l4_umword_t l4_kernel_info_get_mem_desc_end (
    l4_kernel_info_mem_desc_t * md ) [inline]
```

Get end address of the region described by the memory descriptor.

Returns

End address.

Definition at line 217 of file [memdesc.h](#).

13.78.4.2 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 238 of file [memdesc.h](#).

13.78.4.3 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 210 of file [memdesc.h](#).

13.78.4.4 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 231 of file [memdesc.h](#).

13.78.4.5 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 224 of file [memdesc.h](#).

13.78.4.6 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 188 of file [memdesc.h](#).

13.78.4.7 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 195 of file [memdesc.h](#).

13.79 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.

13.79.1 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>
```

13.79.2 Enumeration Type Documentation

13.79.2.1 L4_mem_op_widths

```
enum L4\_mem\_op\_widths
```

Memory access width definitions.

Enumerator

L4_MEM_WIDTH_1BYTE	Access one byte (8-bit width)
L4_MEM_WIDTH_2BYTE	Access two bytes (16-bit width)
L4_MEM_WIDTH_4BYTE	Access four bytes (32-bit width)

Definition at line 51 of file [mem_op.h](#).

13.79.3 Function Documentation

13.79.3.1 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:



13.79.3.2 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:



13.80 Memory related

Memory related constants, data types and functions.

Collaboration diagram for Memory related:



Macros

- `#define L4_PAGESIZE`
Minimal page size (in bytes).
- `#define L4_PAGEMASK`
Mask for the page number.
- `#define L4_LOG2_PAGESIZE`
Number of bits used for page offset.
- `#define L4_SUPERPAGESIZE`
Size of a large page.
- `#define L4_SUPERPAGEMASK`
Mask for the number of a large page.
- `#define L4_LOG2_SUPERPAGESIZE`
Number of bits used as offset for a large page.
- `#define L4_INVALID_PTR ((void *)L4_INVALID_ADDR)`
Invalid address as pointer type.
- `#define L4_PAGESHIFT 12`
Size of a page, log2-based.
- `#define L4_SUPERPAGESHIFT 21`
Size of a large page, log2-based.
- `#define L4_PAGESHIFT 12`
Size of a page log2-based.
- `#define L4_SUPERPAGESHIFT 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.

13.80.1 Detailed Description

Memory related constants, data types and functions.

13.80.2 Macro Definition Documentation

13.80.2.1 L4_LOG2_PAGESIZE

```
#define L4_LOG2_PAGESIZE
```

Number of bits used for page offset.

Size of page in log2.

Definition at line 325 of file [consts.h](#).

13.80.2.2 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 351 of file [consts.h](#).

13.80.2.3 L4_PAGEMASK

```
#define L4_PAGEMASK
```

Mask for the page number.

Note

The most significant bits are set.

Definition at line 316 of file [consts.h](#).

13.80.2.4 L4_SUPERPAGEMASK

```
#define L4_SUPERPAGEMASK
```

Mask for the number of a large page.

Note

The most significant bits are set.

Definition at line 343 of file [consts.h](#).

13.80.2.5 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 334 of file [consts.h](#).

13.80.3 Enumeration Type Documentation

13.80.3.1 l4_addr_consts_t

```
enum l4_addr_consts_t
```

Address related constants.

Enumerator

L4_INVALID_ADDR	Invalid address.
-----------------	------------------

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

13.80.4 Function Documentation

13.80.4.1 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

bytes	The number of bytes to be translated into machine words.
-------	--

Definition at line 412 of file [consts.h](#).

13.80.4.2 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

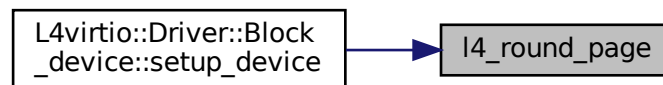
<i>address</i>	The address to round up.
----------------	--------------------------

Definition at line 389 of file [consts.h](#).

References [L4_PAGEMASK](#), and [L4_PAGESIZE](#).

Referenced by [L4virtio::Driver::Block_device::setup_device\(\)](#).

Here is the caller graph for this function:



13.80.4.3 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.

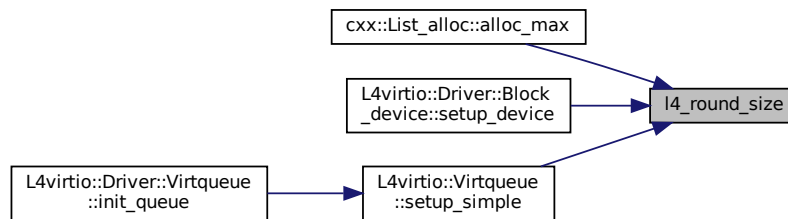
Parameters

<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 400 of file [consts.h](#).

Referenced by [cxx::List_alloc::alloc_max\(\)](#), [L4virtio::Driver::Block_device::setup_device\(\)](#), and [L4virtio::Virtqueue::setup_simple\(\)](#).

Here is the caller graph for this function:



13.80.4.4 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

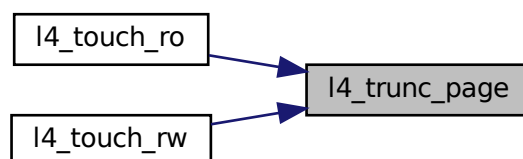
[examples/libs/l4re/c++/mem_alloc/ma+rm.cc](#), and [examples/libs/l4re/c/ma+rm.c](#).

Definition at line 364 of file [consts.h](#).

References [L4_PAGEMASK](#).

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

Here is the caller graph for this function:



13.80.4.5 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*.

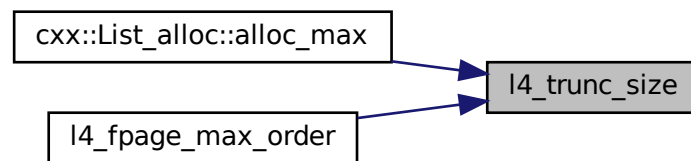
Parameters

<i>address</i>	The address to round.
<i>bits</i>	The size of the flex page (log2).

Definition at line 375 of file [consts.h](#).

Referenced by [cxx::List_alloc::alloc_max\(\)](#), and [l4_fpage_max_order\(\)](#).

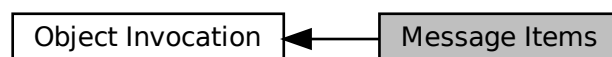
Here is the caller graph for this function:



13.81 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.

13.81.1 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.

13.81.2 Enumeration Type Documentation

13.81.2.1 `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> .
<code>L4_ITEM_CONT</code>	Denote that the following item shall be put into the same receive item as this one.
<code>L4_MAP_ITEM_GRANT</code>	Flag as <i>grant</i> instead of <i>map</i> operation.
<code>L4_MAP_ITEM_MAP</code>	Flag as usual <i>map</i> operation.
<code>L4_RCV_ITEM_SINGLE_CAP</code>	Mark the receive buffer to be a small receive item that describes a buffer for a single capability.
<code>L4_RCV_ITEM_LOCAL_ID</code>	The receiver requests to receive a local ID instead of a mapping whenever possible.

Definition at line 187 of file [consts.h](#).

13.81.3 Function Documentation

13.81.3.1 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.

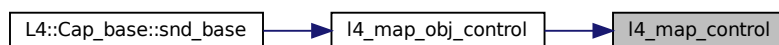
Returns

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

Definition at line 674 of file [__l4_fpage.h](#).

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

Here is the caller graph for this function:



13.81.3.2 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.

Returns

The value to be used as first word in a map item for kernel objects or IO-ports.

Definition at line 681 of file [__l4_fpage.h](#).

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

Referenced by [L4::Cap_base::snd_base\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.82 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](#)

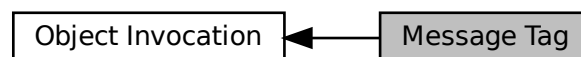
Encapsulation of the message-register block in the UTCB.

13.82.1 Detailed Description

13.83 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`
, `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.

13.83.1 Detailed Description

API related to the message tag data type.

Include File

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

13.83.2 Typedef Documentation

13.83.2.1 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.

13.83.3 Enumeration Type Documentation

13.83.3.1 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_PROPAGATE	Enable IPC propagation. This flag enables IPC propagation, which means an IPC reply-connection from the current caller will be propagated to the new IPC receiver. This makes it possible to propagate an IPC call to a third party.

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

13.83.3.2 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](#).

13.83.3.3 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 [147](#) of file [platform_control.h](#).

13.83.4 Function Documentation

13.83.4.1 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

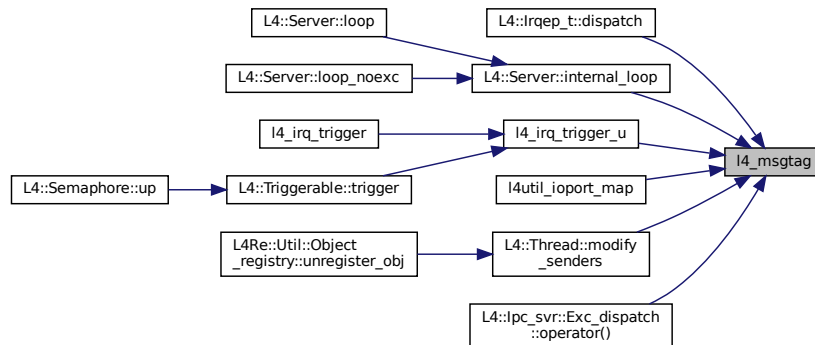
Examples

[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 [408](#) of file [types.h](#).

Referenced by [L4::Irqep_t< Derived, BASE, bool >::dispatch\(\)](#), [L4::Server< LOOP_HOOKS >::internal_loop\(\)](#), [l4_irq_trigger_u\(\)](#), [l4util_ioport_map\(\)](#), [L4::Thread::modify_senders\(\)](#), and [L4::lpc_svr::Exc_dispatch< R, Exc >::operator\(\)](#).

Here is the caller graph for this function:



13.83.4.2 l4_msgtag_flags()

```
unsigned l4_msgtag_flags (
    l4_msgtag_t t ) [inline]
```

Get the flags.

The flag are defined by [l4_msgtag_flags](#).

Parameters

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

Returns

Flags

Definition at line 432 of file [types.h](#).

13.83.4.3 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

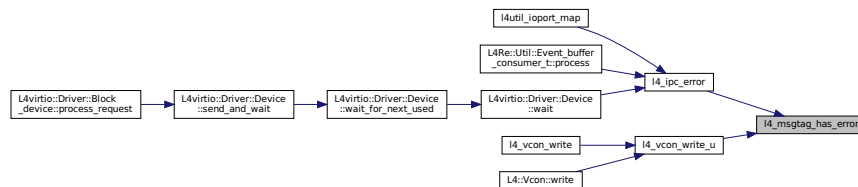
[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

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

References [L4_MSGTAG_ERROR](#).

Referenced by [l4_ipc_error\(\)](#), and [l4_vcon_write_u\(\)](#).

Here is the caller graph for this function:

**13.83.4.4 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

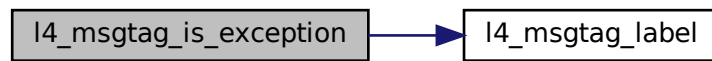
Examples

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

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

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

Here is the call graph for this function:



13.83.4.5 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
----------	---------

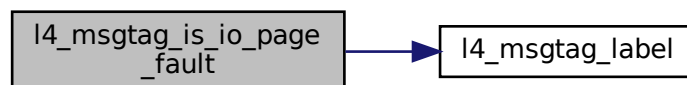
Returns

Boolean value

Definition at line 457 of file [types.h](#).

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

Here is the call graph for this function:



13.83.4.6 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
----------	---------

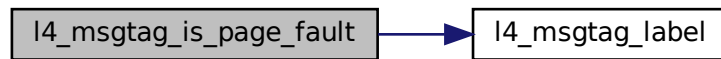
Returns

Boolean value

Definition at line 442 of file [types.h](#).

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

Here is the call graph for this function:

**13.83.4.7 l4_msgtag_is_preemption()**

```
unsigned l4_msgtag_is_preemption (  
    l4_msgtag_t t ) [inline]
```

Test for preemption protocol.

Parameters

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

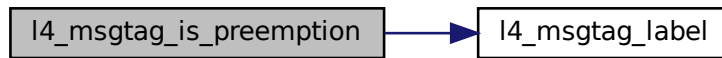
Returns

Boolean value

Definition at line 445 of file [types.h](#).

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

Here is the call graph for this function:



13.83.4.8 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 454 of file [types.h](#).

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

Here is the call graph for this function:



13.83.4.9 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
----------	---------

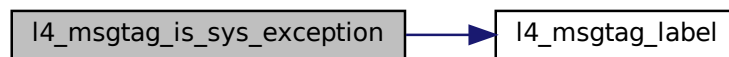
Returns

Boolean value

Definition at line 448 of file [types.h](#).

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

Here is the call graph for this function:

**13.83.4.10 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 428 of file [types.h](#).

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

Here is the caller graph for this function:



13.83.4.11 l4_msgtag_label()

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

Get the protocol of tag.

Parameters

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

Returns

Label

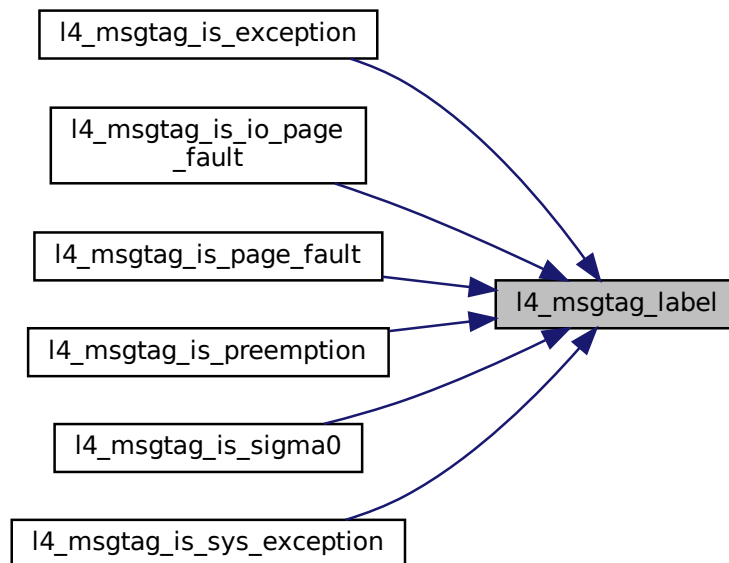
Examples

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

Definition at line 420 of file [types.h](#).

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:



13.83.4.12 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

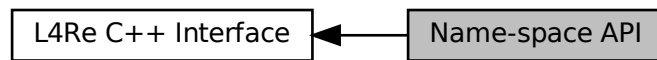
[examples/sys/utcb-ipc/main.c](#).

Definition at line 424 of file [types.h](#).

13.84 Name-space API

API for name spaces that store capabilities.

Collaboration diagram for Name-space API:



Data Structures

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

13.84.1 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.

13.85 Namespace interface

Namespace C interface.

Collaboration diagram for Namespace interface:



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.

13.85.1 Detailed Description

Namespace C interface.

13.85.2 Enumeration Type Documentation

13.85.2.1 [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](#).

13.85.3 Function Documentation

13.85.3.1 [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:



13.85.3.2 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:



13.85.3.3 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 doesn't have necessary permissions.
<i>-L4_ENOMEM</i>	Server has insufficient resources.

Return values

<code>-L4_EINVAL</code>	Invalid parameter.
<code>< 0</code>	IPC errors, see l4_error_code_t .

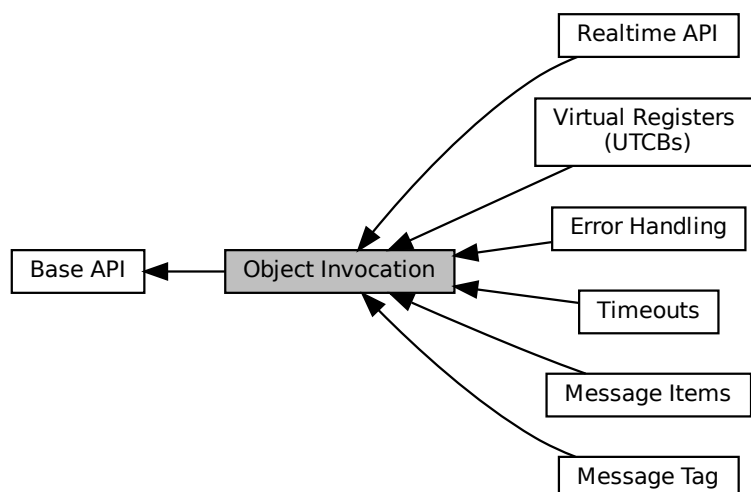
Precondition

requires capability rights: {RW}

13.86 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_ALWAYS_INLINE](#) [l4_msgtag_t](#) [l4_ipc](#) ([l4_cap_idx_t](#) dest, [l4_utcb_t](#) *utcb, [l4_umword_t](#) flags, [l4_umword_t](#) label, [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.
- 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.

13.86.1 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](#)

13.86.2 Enumeration Type Documentation**13.86.2.1 l4_syscall_flags_t**

```
enum l4_syscall_flags_t
```

Capability selector flags.

These flags determine the concrete operation when a kernel object is invoked.

Enumerator

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).
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 .
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 39 of file [consts.h](#).

13.86.3 Function Documentation

13.86.3.1 [l4_ipc\(\)](#)

```
L4_ALWAYS_INLINE 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 of the caller.
	<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).

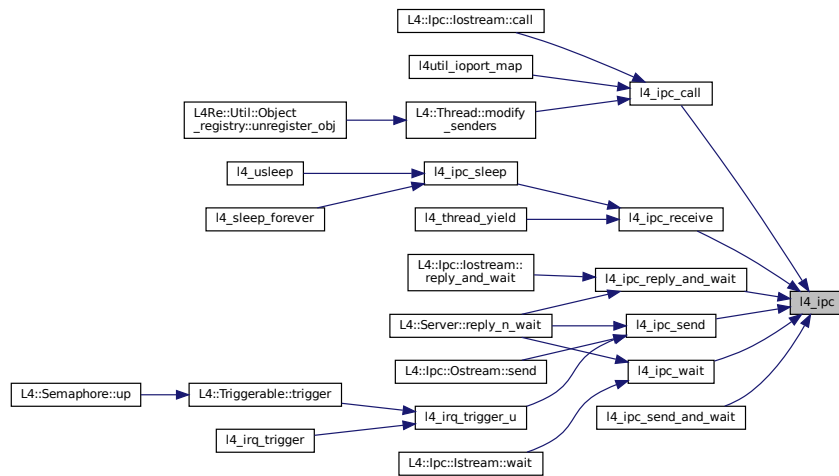
Returns

return tag

Definition at line 34 of file [ipc.h](#).

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:



13.86.3.2 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 of the caller.
<i>tag</i>	Message tag to describe the message to be sent.
<i>timeout</i>	Timeout pair for send and 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.

Examples

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

Definition at line 463 of file [ipc.h](#).

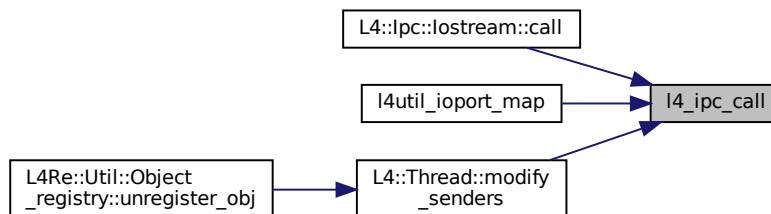
References [l4_ipc\(\)](#), and [L4_SYSF_CALL](#).

Referenced by [L4::ipc::loststream::call\(\)](#), [l4util_ioport_map\(\)](#), and [L4::Thread::modify_senders\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**13.86.3.3 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>timeout</i>	Timeout pair (see l4_timeout_t , only the receive part matters).
<i>utcb</i>	UTCB of the caller.

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.

Examples

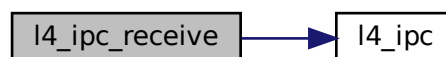
[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 505 of file [ipc.h](#).

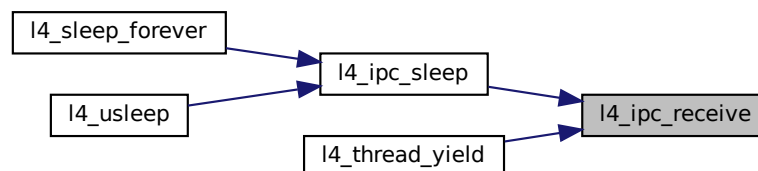
References [l4_ipc\(\)](#), [L4_SYSF_RECV](#), and [l4_msgtag_t::raw](#).

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:



13.86.3.4 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 of the caller.
	<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.

Examples

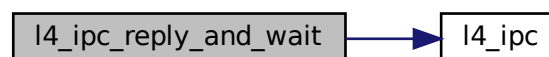
[examples/sys/ipc/ipc_example.c](#).

Definition at line 471 of file [ipc.h](#).

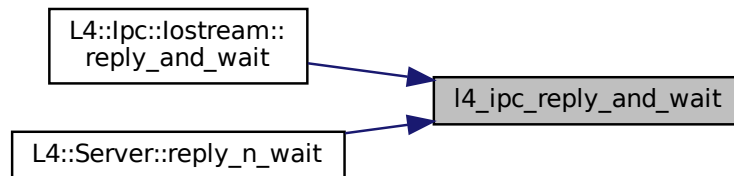
References [L4_INVALID_CAP](#), [l4_ipc\(\)](#), and [L4_SYSF_REPLY_AND_WAIT](#).

Referenced by [L4::lpc::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:



13.86.3.5 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 of the caller.
<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

result tag

A message is sent to the destination object. There is no receive phase included. The invoker continues working after sending the message.

Attention

This is a special-purpose message transfer, objects usually support only invocation via [l4_ipc_call\(\)](#).

Examples

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

Definition at line 488 of file [ipc.h](#).

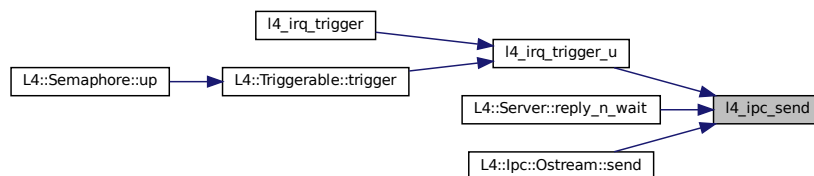
References [l4_ipc\(\)](#), and [L4_SYSF_SEND](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



13.86.3.6 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 of the caller.
	<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.

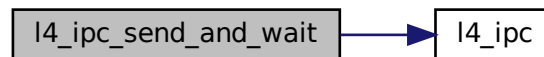
Note

This is a special-purpose operation and shall not be used in general applications.

Definition at line 479 of file [ipc.h](#).

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

Here is the call graph for this function:

**13.86.3.7 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\(\)](#)).

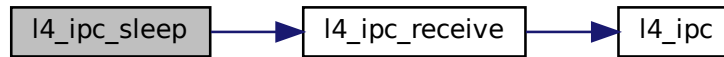
The invoking thread waits until the timeout is expired or the wait was aborted by another thread by [l4_thread_ex_regs\(\)](#).

Definition at line 514 of file [ipc.h](#).

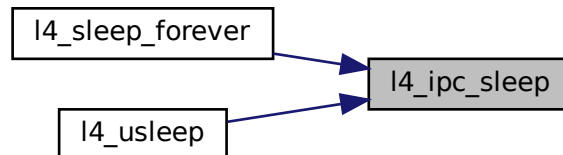
References [L4_INVALID_CAP](#), and [l4_ipc_receive\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



13.86.3.8 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 of the caller.
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\(\)](#).

Examples

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

Definition at line 496 of file [ipc.h](#).

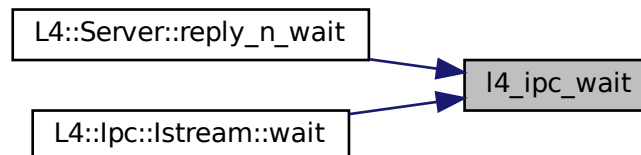
References [L4_INVALID_CAP](#), [l4_ipc\(\)](#), [L4_SYSF_WAIT](#), and [l4_msgtag_t::raw](#).

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:



13.86.3.9 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>tag</i>	Tag to be modified.

Return values

<i>tag</i>	Modified tag, the number of items will be increased, all other values in the tag will be retained.
------------	--

Returns

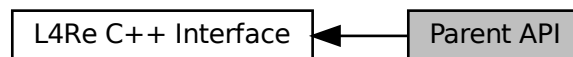
0 on success, negative error code otherwise

Definition at line 574 of file [ipc.h](#).

13.87 Parent API

[Parent](#) interface.

Collaboration diagram for Parent API:

**Data Structures**

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

13.87.1 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.

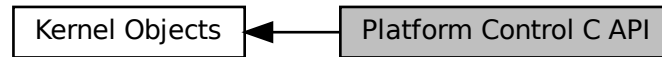
See also

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

13.88 Platform Control C API

C interface for controlling platform-wide properties.

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_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.

13.88.1 Detailed Description

C interface for controlling platform-wide properties.

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](#)

13.88.2 Function Documentation

13.88.2.1 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

Definition at line 232 of file [platform_control.h](#).

13.88.2.2 I4_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

Definition at line 225 of file [platform_control.h](#).

13.88.2.3 I4_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 194 of file [platform_control.h](#).

13.88.2.4 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.

Returns

Syscall return tag

Definition at line 187 of file [platform_control.h](#).

13.89 Producer

Collaboration diagram for Producer:



Functions

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

13.89.1 Detailed Description

13.89.2 Function Documentation

13.89.2.1 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](#).

13.90 Producer

Collaboration diagram for Producer:



Functions

- long [l4shmc_chunk_try_to_take](#) (l4shmc_chunk_t *chunk)
Try to mark chunk busy.
- 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 *chunk)
Check whether chunk is free.

13.90.1 Detailed Description

13.90.2 Function Documentation

13.90.2.1 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.

13.90.2.2 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](#).

13.90.2.3 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](#).

13.90.2.4 l4shmc_is_chunk_clear()

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

Check whether chunk is free.

Parameters

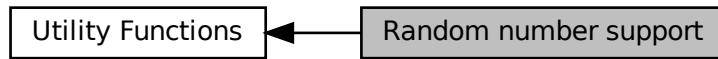
<i>chunk</i>	Chunk to check.
--------------	-----------------

Return values

0	Success.
<0	Error.

13.91 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.

13.91.1 Detailed Description

13.91.2 Function Documentation

13.91.2.1 l4util_rand()

```
l4_uint32_t l4util_rand (  
    void )
```

Deliver next random number.

Returns

A new random number

13.91.2.2 l4util_srand()

```
void l4util_srand (  
    l4_uint32_t seed )
```

Initialize random number generator.

Parameters

<i>seed</i>	Value to initialize
-------------	---------------------

13.92 Realtime API

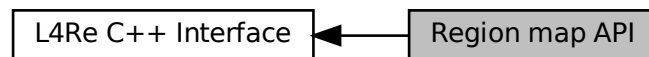
Collaboration diagram for Realtime API:



13.93 Region map API

Virtual address-space management.

Collaboration diagram for Region map API:



Data Structures

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

13.93.1 Detailed Description

Virtual address-space management.

The central purpose of the region-map API is to provide means to manage the virtual memory address space of an [L4](#) task. 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, that allows to attach a data-space object to a region of the virtual address space.

There are two basic concepts provided by a region-map abstraction:

- Regions provide a means to create a view to a data space (or parts of a data space).
- Areas provide a means to reserve areas in a virtual memory address space for special purposes. A reserved area is skipped when searching for an available range of virtual memory, or may be explicitly used to search only within that area.

See also

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

13.94 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](#)
- int [l4re_rm_free_area](#) ([l4_addr_t](#) addr) [L4_NOTHROW](#)
- 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](#)
- int [l4re_rm_detach](#) (void *addr) [L4_NOTHROW](#)

Detach and unmap in current task.

- int [l4re_rm_detach_ds](#) (void *addr, [l4re_ds_t](#) *ds) [L4_NOTHROW](#)

Detach, unmap and return affected dataspace in 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](#)
- 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.

13.94.1 Detailed Description

Region map C interface.

13.94.2 Enumeration Type Documentation

13.94.2.1 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](#).

13.94.3 Function Documentation

13.94.3.1 `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]
```

Parameters

<code>in, out</code>	<code>start</code>	Virtual start address where the region manager shall attach the data space. 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.
	<code>size</code>	Size of the data space to attach (in bytes)
	<code>flags</code>	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.
	<code>mem</code>	Data space
	<code>offs</code>	Offset into the data space to use
	<code>align</code>	Alignment of the virtual region, log2-size, default: a page (L4_PAGESHIFT). This is only meaningful if the L4Re::Rm::F::Search_addr flag is used.

Return values

<code>0</code>	Success
<code>-L4_ENOENT</code>	No area could be found (see L4Re::Rm::F::In_area)
<code>-L4_EPERM</code>	Operation not allowed.
<code>-L4_EINVAL</code>	
<code>-L4_EADDRNOTAVAIL</code>	The given address is not available.
<code><0</code>	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](#)).

Returns

0 on success, <0 on error

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 267 of file [rm.h](#).

References [l4re_rm_attach_srv\(\)](#).

Here is the call graph for this function:

**13.94.3.2 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:



13.94.3.3 l4re_rm_detach()

```
int l4re_rm_detach (
    void * addr ) [inline]
```

Detach and unmap in current task.

Parameters

<i>addr</i>	Address of the region to detach.
-------------	----------------------------------

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 277 of file [rm.h](#).

References [l4re_rm_detach_srv\(\)](#).

Here is the call graph for this function:



13.94.3.4 l4re_rm_detach_ds()

```
int l4re_rm_detach_ds (  
    void * addr,  
    l4re_ds_t * ds ) [inline]
```

Detach, unmap and return affected dataspace in current task.

Parameters

<i>addr</i>	Address of the region to detach.
-------------	----------------------------------

Return values

<i>ds</i>	Returns dataspace that is affected.
-----------	-------------------------------------

Returns

0 on success, <0 on error

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 290 of file [rm.h](#).

References [l4re_rm_detach_srv\(\)](#).

Here is the call graph for this function:



13.94.3.5 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.
-------------	----------------------------------

Return values

<i>ds</i>	Returns dataspace that is affected.
-----------	-------------------------------------

Parameters

<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 297 of file [rm.h](#).

References [l4re_rm_detach_srv\(\)](#).

Here is the call graph for this function:



13.94.3.6 l4re_rm_detach_srv()

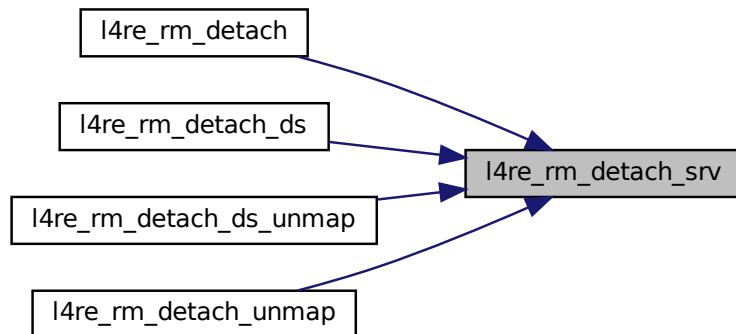
```
int l4re_rm_detach_srv (
    l4_cap_idx_t rm,
    l4_addr_t addr,
    l4re_ds_t * ds,
    l4_cap_idx_t task )
```

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:



13.94.3.7 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 <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 284 of file `rm.h`.

References [l4re_rm_detach_srv\(\)](#).

Here is the call graph for this function:

**13.94.3.8 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]
```

Returns

0 on success, <0 on error

See also

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

Definition at line 304 of file `rm.h`.

References [l4re_rm_find_srv\(\)](#).

Here is the call graph for this function:



13.94.3.9 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:



13.94.3.10 l4re_rm_free_area()

```
int l4re_rm_free_area (
    l4_addr_t addr ) [inline]
```

Returns

0 on success, <0 on error

See also

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

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

Definition at line 261 of file `rm.h`.

References [l4re_rm_free_area_srv\(\)](#).

Here is the call graph for this function:



13.94.3.11 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:



13.94.3.12 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]
```

Returns

0 on success, <0 on error

See also

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

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

Definition at line 253 of file [rm.h](#).

References [l4re_rm_reserve_area_srv\(\)](#).

Here is the call graph for this function:



13.94.3.13 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:



13.94.3.14 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 312 of file [rm.h](#).

References [l4re_rm_show_lists_srv\(\)](#).

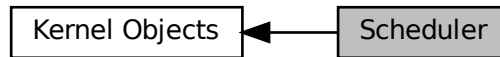
Here is the call graph for this function:



13.95 Scheduler

C interface of the Scheduler kernel object.

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_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_sched_param_t l4_sched_param` (unsigned prio, `l4_cpu_time_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.

13.95.1 Detailed Description

C interface of the Scheduler kernel object.

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.

Include File

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

13.95.2 Enumeration Type Documentation

13.95.2.1 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 201 of file [scheduler.h](#).

13.95.3 Function Documentation

13.95.3.1 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.
<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 211 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:



13.95.3.2 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 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.

Definition at line 323 of file [scheduler.h](#).

13.95.3.3 `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.
in, out	<i>cpus</i>	<i>cpus.offset</i> is first CPU of interest. <i>cpusgranularity</i> (see l4_sched_cpu_set_t). <i>cpus.map</i> Bitmap of online CPUs.

Return values

0	Success.
<code>-L4_EINVAL</code>	The given CPU offset is larger than the maximum number of CPUs.

Definition at line 309 of file [scheduler.h](#).

13.95.3.4 `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 330 of file [scheduler.h](#).

13.95.3.5 `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

<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

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 physical CPUs.
- Two threads with a single identical CPU selected in the CPU set must be scheduled to the same physical 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 316 of file [scheduler.h](#).

13.96 Server-Side IPC framework

Server-Side framework for implementing object-oriented servers.

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::Timed_work< HOOKS >](#)
DEPRECATED.
- 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::Irq_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.

13.96.1 Detailed Description

Server-Side framework for implementing object-oriented servers.

,

13.96.2 Enumeration Type Documentation

13.96.2.1 Reply_mode

```
enum L4::Ipc_svr::Reply_mode
```

Reply mode for server loop.

The reply mode specifies if the server loop shall do a compound reply and wait operation ([Reply_compound](#)), which is the most performant method. Note, `setup_wait()` is called before the reply. The other way is to call reply and wait separately and call `setup_wait` in between.

The actual mode is determined by the return value of the `before_reply()` hook in the `LOOP_HOOKS` of [L4::Server](#).

Enumerator

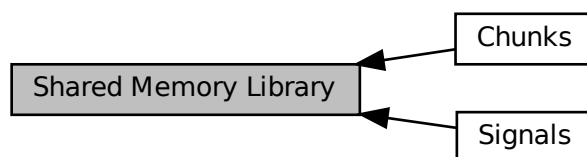
<code>Reply_compound</code>	Server shall use a compound reply and wait (fast).
<code>Reply_separate</code>	Server shall call reply and wait separately.

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

13.97 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](#) (const char *shmc_name, [l4_umword_t](#) shm_size)
Create a shared memory area.
- long [l4shmc_attach](#) (const char *shmc_name, [l4shmc_area_t](#) *shmarea)
Attach to a shared memory area.
- long [l4shmc_attach_to](#) (const char *shmc_name, [l4_umword_t](#) timeout_ms, [l4shmc_area_t](#) *shmarea)
Attach to a shared memory area, with limited waiting.
- 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](#) *shmarea)
Get size of shared memory area.
- long [l4shmc_area_size_free](#) ([l4shmc_area_t](#) *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.

13.97.1 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).

13.97.2 Function Documentation

13.97.2.1 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

13.97.2.2 l4shmc_area_size()

```
long l4shmc_area_size (
    l4shmc_area_t * shmarea ) [inline]
```

Get size of shared memory area.

Parameters

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

Return values

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

13.97.2.3 l4shmc_area_size_free()

```
long l4shmc_area_size_free (
    l4shmc_area_t * 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.
----------------	---------------------

Return values

<i>0</i>	Free capacity in the area.
<i><0</i>	Error.

13.97.2.4 l4shmc_attach()

```
long l4shmc_attach (
    const char * shmc_name,
    l4shmc_area_t * shmarea ) [inline]
```

Attach to a shared memory area.

Parameters

	<i>shmc_name</i>	Name of the shared memory area.
out	<i>shmarea</i>	Pointer to shared memory area descriptor to be filled with information for the shared memory area.

Return values

0	Success.
<0	Error.

Examples

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

13.97.2.5 l4shmc_attach_to()

```
long l4shmc_attach_to (
    const char * shmc_name,
    l4_umword_t timeout_ms,
    l4shmc_area_t * shmarea )
```

Attach to a shared memory area, with limited waiting.

Parameters

	<i>shmc_name</i>	Name of the shared memory area.
	<i>timeout_ms</i>	Timeout to wait for shm area in milliseconds.
out	<i>shmarea</i>	Pointer to shared memory area descriptor to be filled with information for the shared memory area.

Return values

0	Success.
<0	Error.

13.97.2.6 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

13.97.2.7 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.

Returns

0 on success, <0 on error

Examples

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

13.97.2.8 l4shmc_create()

```
long l4shmc_create (
    const char * shm_name,
    l4_umword_t shm_size )
```

Create a shared memory area.

Parameters

<i>shm_name</i>	Name of the shared memory area.
<i>shm_size</i>	Size of the whole shared memory area.

Return values

0	Success.
<0	Error.

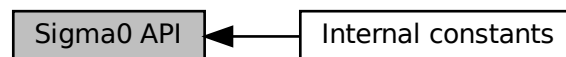
Examples

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

13.98 Sigma0 API

Sigma0 API bindings.

Collaboration diagram for Sigma0 API:



Modules

- [Internal constants](#)
Internal sigma0 definitions.

Files

- file [sigma0.h](#)
Sigma0 interface.

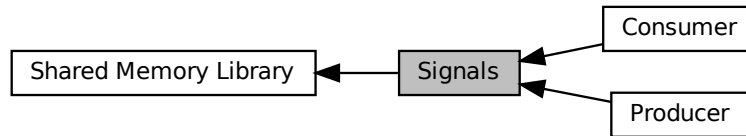
13.98.1 Detailed Description

Sigma0 API bindings.

Convenience bindings for the Sigma0 protocol.

13.99 Signals

Collaboration diagram for Signals:



Modules

- [Consumer](#)
- [Producer](#)

Functions

- long [l4shmc_add_signal](#) (l4shmc_area_t *shmarea, const char *signal_name, l4shmc_signal_t *signal)
Add a signal for the shared memory area.
- long [l4shmc_attach_signal](#) (l4shmc_area_t *shmarea, const char *signal_name, [l4_cap_idx_t](#) thread, l4shmc_signal_t *signal)
Attach to signal.
- long [l4shmc_attach_signal_to](#) (l4shmc_area_t *shmarea, const char *signal_name, [l4_cap_idx_t](#) thread, [l4_umword_t](#) timeout_ms, l4shmc_signal_t *signal)
Attach to signal, with timeout.
- long [l4shmc_get_signal_to](#) (l4shmc_area_t *shmarea, const char *signal_name, [l4_umword_t](#) timeout_ms, l4shmc_signal_t *signal)
Get signal object from the shared memory area.
- [l4_cap_idx_t l4shmc_signal_cap](#) (l4shmc_signal_t *signal)
Get the signal capability of a signal.
- long [l4shmc_check_magic](#) (l4shmc_chunk_t *chunk)
Check magic value of a chunk.

13.99.1 Detailed Description

13.99.2 Function Documentation

13.99.2.1 l4shmc_add_signal()

```

long l4shmc_add_signal (
    l4shmc_area_t * shmarea,
    const char * signal_name,
    l4shmc_signal_t * signal )

```

Add a signal for the shared memory area.

Parameters

	<i>shmarea</i>	The shared memory area to put the chunk in.
	<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](#).

13.99.2.2 l4shmc_attach_signal()

```
long l4shmc_attach_signal (
    l4shmc_area_t * shmarea,
    const char * signal_name,
    l4_cap_idx_t thread,
    l4shmc_signal_t * signal ) [inline]
```

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.

13.99.2.3 l4shmc_attach_signal_to()

```
long l4shmc_attach_signal_to (
    l4shmc_area_t * shmarea,
    const char * signal_name,
    l4_cap_idx_t thread,
```

```
l4_umword_t timeout_ms,
l4shmc_signal_t * signal )
```

Attach to signal, with timeout.

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.
	<i>timeout_ms</i>	Timeout in milliseconds to wait for the chunk to appear in the shared memory area.
out	<i>signal</i>	Signal data structure to fill.

Return values

0	Success.
<0	Error.

Examples

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

13.99.2.4 l4shmc_check_magic()

```
long l4shmc_check_magic (
    l4shmc_chunk_t * 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.

13.99.2.5 l4shmc_get_signal_to()

```
long l4shmc_get_signal_to (
    l4shmc_area_t * shmarea,
    const char * signal_name,
```

```
l4_umword_t timeout_ms,
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.
	<i>timeout_ms</i>	Timeout in milliseconds to wait for signal of a chunk to appear in the shared memory area.
out	<i>signal</i>	Signal data structure to fill.

Return values

0	Success.
<0	Error.

13.99.2.6 l4shmc_signal_cap()

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

Get the signal capability of a signal.

Parameters

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

Returns

Capability of the signal object.

13.100 Small C++ Template Library

Data Structures

- class [L4::Alloc_list](#)
A simple list-based allocator.
- class [cxx::Auto_ptr< T >](#)
Smart pointer with automatic deletion.
- 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 `operator new` version that does not throw exceptions.
- class `cxx::New_allocator< _Type >`
Standard allocator based on `operator new ()`.
- class `L4::String`
A null-terminated string container class.

Functions

- template<typename 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`.
- void * `operator new` (size_t, void *mem, `cxx::Nothrow` const &) throw ()
Simple placement new operator.
- void * `operator new` (size_t, `cxx::Nothrow` const &) throw ()
New operator that does not throw exceptions.

13.100.1 Detailed Description

13.100.2 Function Documentation

13.100.2.1 max()

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

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

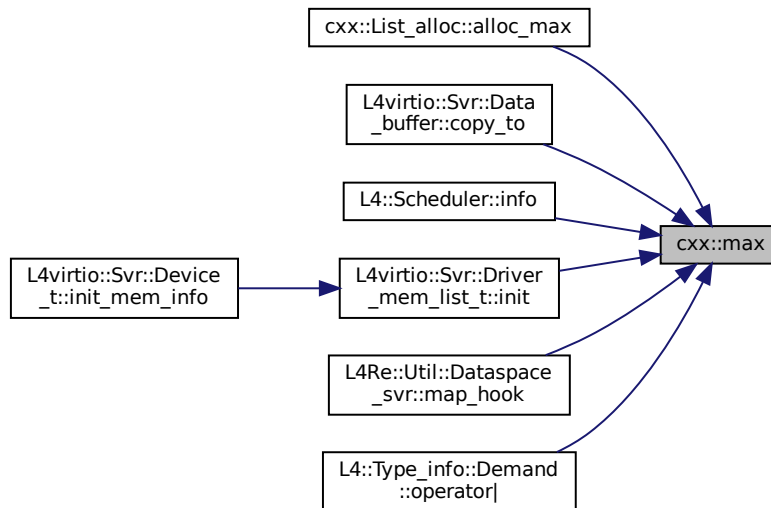
Parameters

<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\(\)](#), [L4virtio::Svr::Data_buffer::copy_to\(\)](#), [L4::Scheduler::info\(\)](#), [L4virtio::Svr::Driver_mem_list_t::init\(\)](#), [L4Re::Util::Dataspace_svr::map_hook\(\)](#), and [L4::Type_info::Demand::operator|\(\)](#).

Here is the caller graph for this function:



13.100.2.2 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.

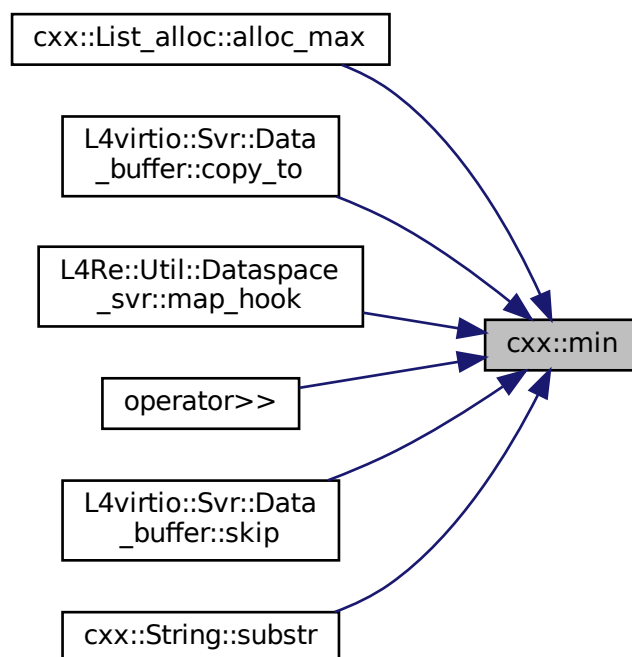
Examples

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

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

Referenced by [cxx::List_alloc::alloc_max\(\)](#), [L4virtio::Svr::Data_buffer::copy_to\(\)](#), [L4Re::Util::Dataspace_svr::map_hook\(\)](#), [operator>>\(\)](#), [L4virtio::Svr::Data_buffer::skip\(\)](#), and [cxx::String::substr\(\)](#).

Here is the caller graph for this function:



13.100.2.3 operator new()

```

void* operator new (
    size_t ,
    void * mem,
    cxx::Nothrow const & ) throw ( )    [inline]

```

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`.

13.101 Task

C interface of the Task kernel object.

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's virtual GICC page to the task.
- `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 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.

13.101.1 Detailed Description

C interface of the Task kernel object.

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>
```

13.101.2 Enumeration Type Documentation

13.101.2.1 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	Flag to tell the unmap operation to unmap all child mappings including the mapping in the invoked task. See also L4::Task::unmap() l4_task_unmap()
L4_FP_DELETE_OBJ	Flag that indicates that the unmap operation on a capability shall try to delete the corresponding objects immediately. See also L4::Task::unmap() l4_task_unmap()
L4_FP_OTHER_SPACES	Counterpart to L4_FP_ALL_SPACES , unmap only child mappings. See also L4::Task::unmap() l4_task_unmap()

Definition at line 157 of file [consts.h](#).

13.101.3 Function Documentation

13.101.3.1 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

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

13.101.3.2 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

Returns

label contains 1 if equal, 0 if not equal

Definition at line 396 of file [task.h](#).

13.101.3.3 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>tag.label()</i>	> 0 Capability is present (refers to an object).
<i>tag.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 390 of file [task.h](#).

13.101.3.4 `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 selector of object to delete

Returns

Syscall return tag

The object will be deleted if the *obj* has sufficient rights. No error will be reported if the rights are insufficient, however, the capability is removed in all cases.

This operation calls [l4_task_unmap\(\)](#) with `L4_FP_DELETE_OBJ`.

Definition at line 369 of file [task.h](#).

13.101.3.5 `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 destination task
<i>src_task</i>	Capability selector of 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.

Returns

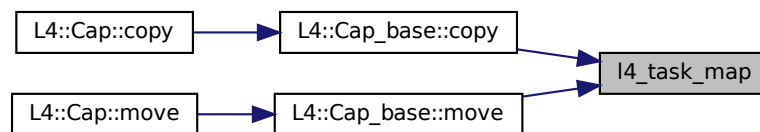
Syscall return tag

This method allows for asynchronous rights delegation from one task to another. It can be used to share memory as well as to delegate access to objects.

Definition at line 339 of file [task.h](#).

Referenced by [L4::Cap_base::copy\(\)](#), and [L4::Cap_base::move\(\)](#).

Here is the caller graph for this function:

**13.101.3.6 l4_task_release_cap()**

```

l4_msgtag_t l4_task_release_cap (
    l4_cap_idx_t task,
    l4_cap_idx_t cap ) [inline]
  
```

Release capability.

Parameters

<i>task</i>	Capability selector of destination task
<i>cap</i>	Capability selector to release

Returns

Syscall return tag

This operation unmaps the capability from the specified task.

Definition at line 384 of file [task.h](#).

13.101.3.7 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 the address space or object 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 and from all the tasks that got the rights delegated from that task (i.e., this operation does a recursive rights revocation).

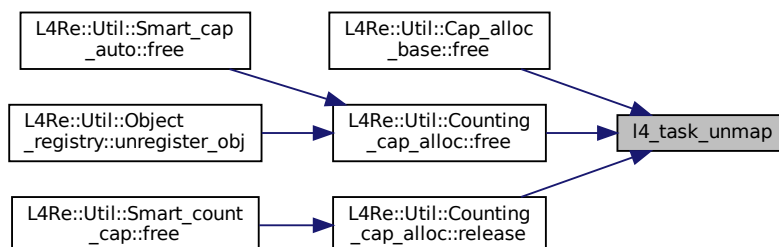
Note

Calling this function on the object space can cause a root capability of an object to be destructed, which destroys the object itself.

Definition at line 346 of file [task.h](#).

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 caller graph for this function:



13.101.3.8 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 that describes an area in the address space or object space of the destination task each
<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

This method allows to revoke rights from the destination task and from all the tasks that got the rights delegated from that task (i.e., this operation does a recursive rights revocation).

Precondition

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

Note

Calling this function on the object space can cause a root capability of an object to be destructed, which destroys the object itself.

Definition at line [353](#) of file [task.h](#).

13.101.3.9 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's virtual GICC page to the task.

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

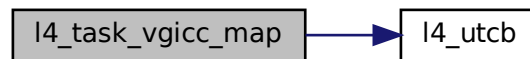
Returns

Syscall return tag

Definition at line 56 of file [__task-arm.h](#).

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

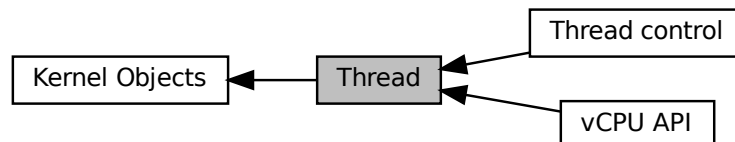
Here is the call graph for this function:



13.102 Thread

Thread object.

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 or disable 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 or disable 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 or disable 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 or disable 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.

13.102.1 Detailed Description

Thread object.

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](#).

13.102.2 Enumeration Type Documentation

13.102.2.1 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 649 of file [thread.h](#).

13.102.2.2 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 672 of file [thread.h](#).

13.102.2.3 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 687 of file [thread.h](#).

13.102.3 Function Documentation

13.102.3.1 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](#).

13.102.3.2 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 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*).

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.

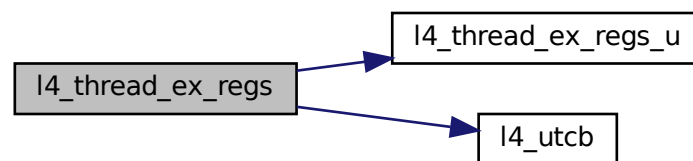
Examples

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

References [l4_thread_ex_regs_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.102.3.3 l4_thread_ex_regs_ret()

```

l4_msgtag_t l4_thread_ex_regs_ret (
    l4_cap_idx_t thread,
    l4_addr_t * ip,
    l4_addr_t * sp,
    l4_umword_t * flags ) [inline]
  
```

Exchange basic thread registers and return previous values.

Parameters

	<i>thread</i>	Capability selector of the thread to manipulate.
in, out	<i>ip</i>	New instruction pointer, use ~0UL to leave the instruction pointer unchanged, return previous instruction pointer.
in, out	<i>sp</i>	New stack pointer, use ~0UL to leave the stack pointer unchanged, returns previous stack pointer.
in, out	<i>flags</i>	Ex-regs flags, see L4_thread_ex_regs_flags , return previous CPU flags of the thread.

Returns

System call return tag

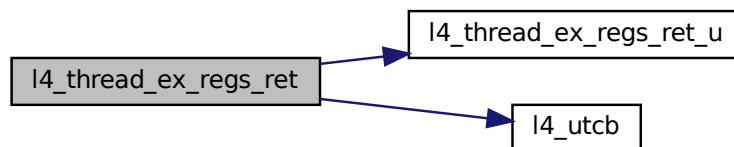
This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see `flags`).

Returned values are valid only if function returns successfully.

Definition at line 845 of file `thread.h`.

References `l4_thread_ex_regs_ret_u()`, and `l4_utcb()`.

Here is the call graph for this function:



13.102.3.4 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 use for this operation.

Returns

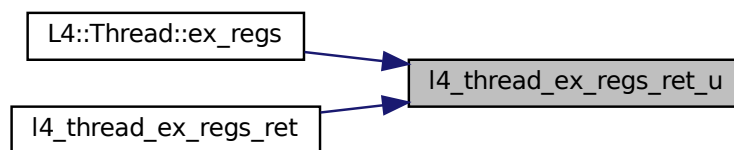
System call return tag. [out] parameters are only valid if the function returns successfully. Use `l4_error()` to check.

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`).

Definition at line 711 of file [thread.h](#).

Referenced by [L4::Thread::ex_regs\(\)](#), and [l4_thread_ex_regs_ret\(\)](#).

Here is the caller graph for this function:



13.102.3.5 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 use for this operation.

Returns

System call return tag

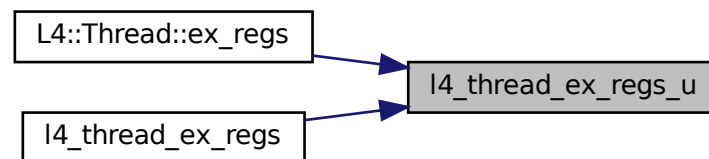
This method allows to manipulate a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see `flags`).

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 700 of file `thread.h`.

Referenced by `L4::Thread::ex_regs()`, and `l4_thread_ex_regs()`.

Here is the caller graph for this function:



13.102.3.6 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 <code>l4_thread_modify_sender_start()</code> or previous <code>l4_thread_modify_sender_add()</code> 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 1018 of file [thread.h](#).

13.102.3.7 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 1029 of file [thread.h](#).

13.102.3.8 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\(\)](#).

See also

[l4_thread_modify_sender_add](#)

[l4_thread_modify_sender_commit](#)

Definition at line 1012 of file [thread.h](#).

13.102.3.9 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.

An example of a deletion event is the removal of an IPC gate that is bound to this thread.

Definition at line 939 of file [thread.h](#).

13.102.3.10 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 907 of file [thread.h](#).

13.102.3.11 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 898 of file [thread.h](#).

13.102.3.12 l4_thread_vcpu_control()

```
l4_msgtag_t l4_thread_vcpu_control (
    l4_cap_idx_t thread,
    l4_addr_t vcpu_state ) [inline]
```

Enable or disable the vCPU feature for the thread.

Parameters

<i>thread</i>	Capability selector of the thread for which the vCPU feature shall be enabled or disabled.
<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

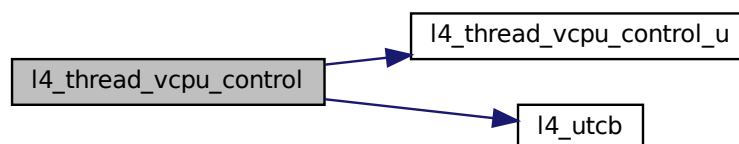
Syscall return tag.

This function enables the vCPU feature of the `thread` if `vcpu_state` is set to a valid kernel-user-memory address, or disables the vCPU feature if `vcpu_state` is 0. (Disable: optional, currently unsupported.)

Definition at line 956 of file [thread.h](#).

References [l4_thread_vcpu_control_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**13.102.3.13 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 or disable 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 or disabled.
<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()).

Returns

Systemcall result message 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` if `ext_vcpu_state` is set to a valid kernel-user-memory address, or disables the vCPU feature if `ext_vcpu_state` is 0.

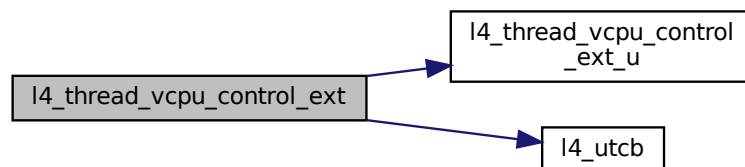
Note

The extended vCPU mode includes the normal vCPU mode.

Definition at line 971 of file [thread.h](#).

References [l4_thread_vcpu_control_ext_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

13.102.3.14 `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 or disable 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 or disabled.
<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 use for this operation.

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` if `ext_vcpu_state` is set to a valid kernel-user-memory address, or disables the vCPU feature if `ext_vcpu_state` is 0.

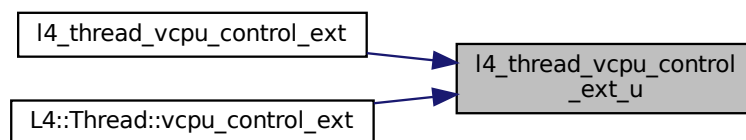
Note

The extended vCPU mode includes the normal vCPU mode.

Definition at line 961 of file [thread.h](#).

Referenced by [l4_thread_vcpu_control_ext\(\)](#), and [L4::Thread::vcpu_control_ext\(\)](#).

Here is the caller graph for this function:



13.102.3.15 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 or disable the vCPU feature for the thread.

Parameters

<i>thread</i>	Capability selector of the thread for which the vCPU feature shall be enabled or disabled.
<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()).
<i>utcb</i>	UTCB to use for this operation.

Returns

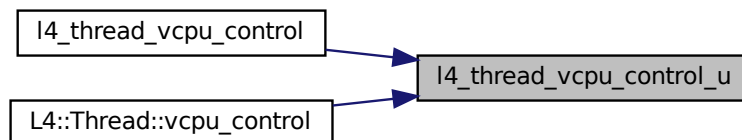
Syscall return tag.

This function enables the vCPU feature of `this` thread if `vcpu_state` is set to a valid kernel-user-memory address, or disables the vCPU feature if `vcpu_state` is 0. (Disable: optional, currently unsupported.)

Definition at line 946 of file [thread.h](#).

Referenced by [l4_thread_vcpu_control\(\)](#), and [L4::Thread::vcpu_control\(\)](#).

Here is the caller graph for this function:



13.102.3.16 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

System call result message tag. In extended vCPU mode and when the virtual interrupts are cleared, the return code 1 flags an incoming IPC message, whereas 0 indicates a VM exit. An error is returned upon:

- Insufficient rights on the given task capability (-L4_EPERM).
- Given task capability is invalid (-L4_ENOENT).
- A supplied mapping failed.

To resume into another address space the capability to the target task must be set in the vCPU-state, with all lower bits in the task capability cleared (see [L4_CAP_MASK](#)). The kernel adds the [L4_SYSF_SEND](#) flag to this field to indicate that the capability has been referenced in the kernel. Consecutive resumes will not reference the task capability again until all bits are cleared again. To release a task use the different task capability or use an invalid capability with the [L4_SYSF_REPLY](#) flag set.

See also

[l4_vcpu_state_t](#)

Definition at line 919 of file [thread.h](#).

13.102.3.17 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 913 of file [thread.h](#).

13.102.3.18 l4_thread_yield()

```
l4_msgtag_t l4_thread_yield (
    void ) [inline]
```

Yield current time slice.

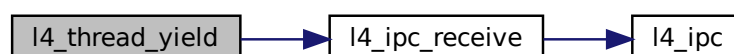
Returns

system call return tag

Definition at line 787 of file [thread.h](#).

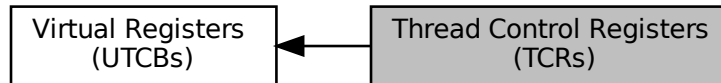
References [L4_INVALID_CAP](#), and [l4_ipc_receive\(\)](#).

Here is the call graph for this function:



13.103 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

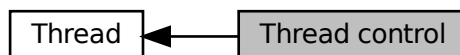
- typedef struct [l4_thread_regs_t](#) [l4_thread_regs_t](#)
Encapsulation of the thread-control-register block of the UTCB.

13.103.1 Detailed Description

13.104 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.

13.104.1 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_utcb_t *u = l4_utcb();
l4_thread_control_start(u);
l4_thread_control_pager(u, pager_cap);
l4_thread_control_bind (u, thread_utcb, task);
l4_thread_control_commit(u, thread_cap);
```

13.104.2 Function Documentation

13.104.2.1 [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.
-----------	--

Alien mode means the thread is not allowed to invoke [L4](#) kernel objects directly and it is also not allowed to allocate FPU state. All those operations result in an exception IPC that gets sent through the pager capability. The responsible pager can then selectively allow an object invocation or allocate FPU state for the thread.

This feature can be used to attach a debugger to a thread and trace all object invocations.

Examples

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

Definition at line [877](#) of file [thread.h](#).

13.104.2.2 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.

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.

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

13.104.2.3 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

system call return tag

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

13.104.2.4 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 864 of file [thread.h](#).

13.104.2.5 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 858 of file [thread.h](#).

13.104.2.6 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, 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 852 of file [thread.h](#).

13.104.2.7 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 883 of file [thread.h](#).

13.105 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.

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.

13.105.1 Detailed Description

All kinds of timeouts and time related functions.

13.105.2 Macro Definition Documentation

13.105.2.1 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](#).

13.105.3 Typedef Documentation

13.105.3.1 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.

13.105.3.2 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.

13.105.4 Function Documentation

13.105.4.1 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 187 of file [__timeout.h](#).

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

13.105.4.2 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
------------	-----------------

Return values

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

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

13.105.4.3 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
------------	--------------

Return values

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

Definition at line 208 of file [__timeout.h](#).

13.105.4.4 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

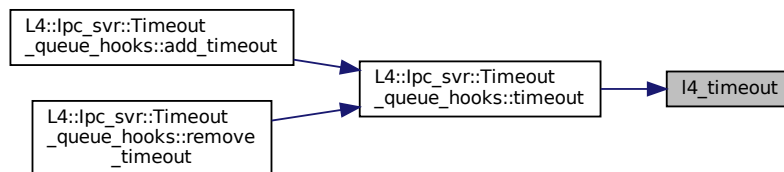
L4 timeout

Definition at line 198 of file `__timeout.h`.

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

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

Here is the caller graph for this function:

**13.105.4.5 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.

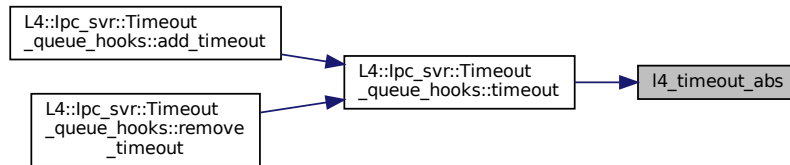
Returns

timeout value

Definition at line 383 of file `utcb.h`.

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

Here is the caller graph for this function:



13.105.4.6 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

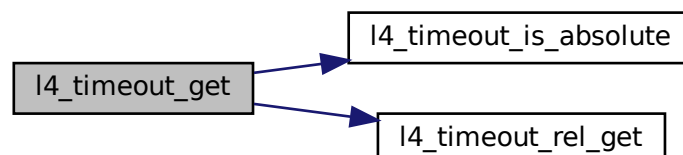
Returns

Clock sum

Definition at line 245 of file `__timeout.h`.

References `l4_timeout_is_absolute()`, and `l4_timeout_rel_get()`.

Here is the call graph for this function:



13.105.4.7 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
-----------	------------

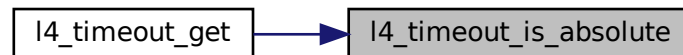
Returns

!= 0 if absolute, 0 if relative

Definition at line 238 of file [__timeout.h](#).

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

Here is the caller graph for this function:



13.105.4.8 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 222 of file [__timeout.h](#).

13.105.4.9 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 229 of file [__timeout.h](#).

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

Here is the caller graph for this function:



13.105.4.10 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
------------	--

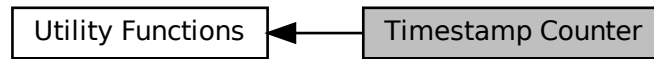
Returns

Index to 64bit message register alias

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

13.106 Timestamp Counter

Collaboration diagram for Timestamp Counter:



Files

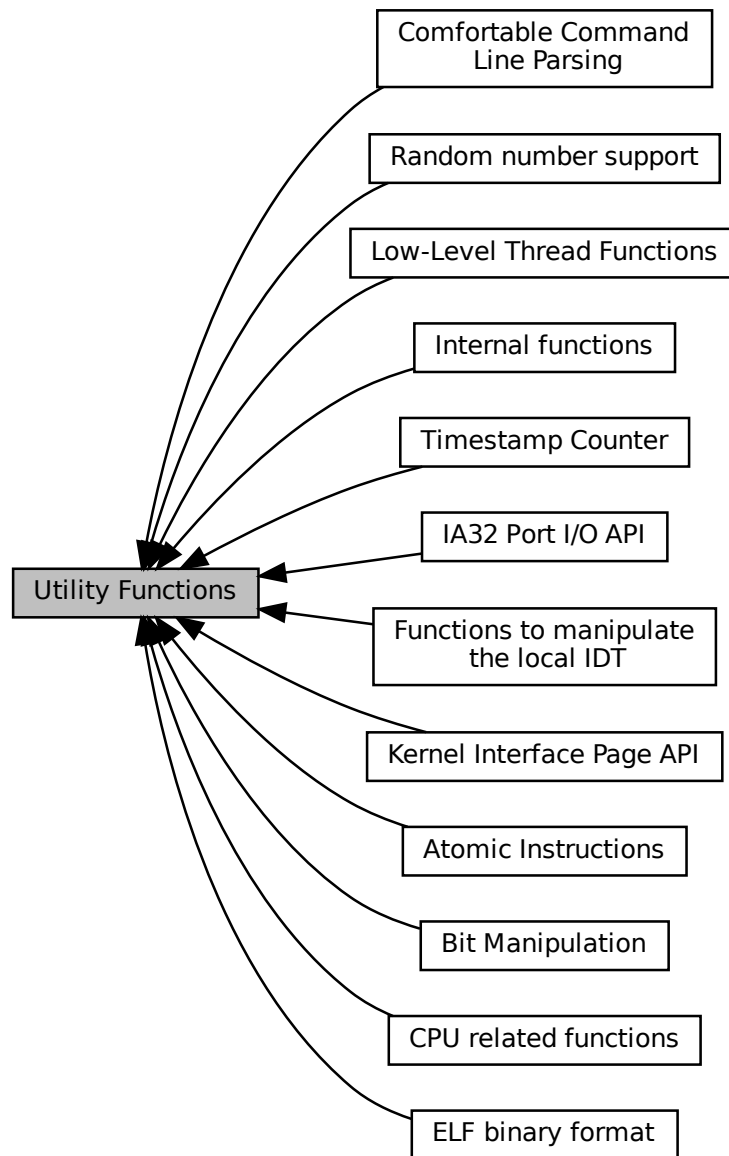
- file [rdtsc.h](#)
time stamp counter related functions
- file [rdtsc.h](#)
time stamp counter related functions

13.106.1 Detailed Description

13.107 Utility Functions

Utilities, generic file.

Collaboration diagram for Utility Functions:



Modules

- [Atomic Instructions](#)
- [Bit Manipulation](#)
- [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)

13.107.1 Detailed Description

Utilities, generic file.

13.107.2 Function Documentation

13.107.2.1 [l4_sleep\(\)](#)

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

Suspend thread for a period of *ms* milliseconds.

Parameters

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

13.107.2.2 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.

Examples

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

Definition at line 94 of file [util.h](#).

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

Here is the call graph for this function:

**13.107.2.3 l4_touch_rw()**

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

Touch data areas to force mapping (read-write)

Parameters

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

Examples

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

Definition at line 107 of file [util.h](#).

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

Here is the call graph for this function:



13.107.2.4 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_sleep\(\)](#).

Here is the call graph for this function:



13.107.2.5 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

13.107.2.6 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.

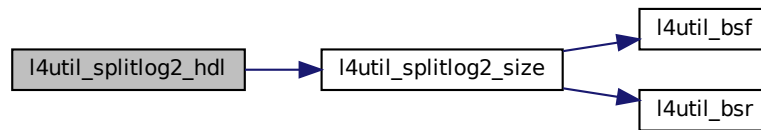
Returns

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:



13.107.2.7 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

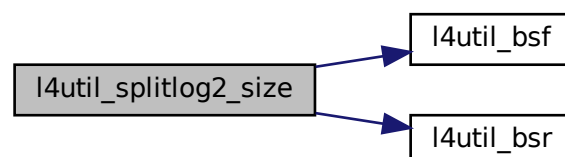
length of elements in log2size (length is $1 \ll \log_2 \text{size}$)

Definition at line 72 of file [splitlog2.h](#).

References [l4util_bsf\(\)](#), and [l4util_bsr\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



13.108 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.

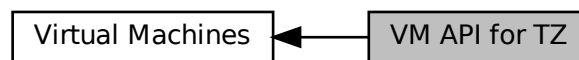
13.108.1 Detailed Description

Virtual machine API for SVM.

13.109 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

13.109.1 Detailed Description

Virtual Machine API for ARM TrustZone.

13.110 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.

13.110.1 Detailed Description

Virtual machine API for VMX.

13.110.2 Enumeration Type Documentation

13.110.2.1 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](#).

13.110.2.2 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_CTLS_REG	True pin-based control caps.
L4_VM_VMX_TRUE_PROCBASED_CTLS_REG	True processor based control caps.
L4_VM_VMX_TRUE_EXIT_CTLS_REG	True exit control caps.
L4_VM_VMX_TRUE_ENTRY_CTLS_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_CTLS2_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](#).

13.110.2.3 L4_vm_vmx_dfl1_regs

enum [L4_vm_vmx_dfl1_regs](#)

Exported VMX capability registers (default to 1 bits).

Enumerator

L4_VM_VMX_PINBASED_CTLS_DFL1_REG	Default 1 bits in pin-based controls.
L4_VM_VMX_PROCBASED_CTLS_DFL1_REG	Default 1 bits in processor-based controls.
L4_VM_VMX_EXIT_CTLS_DFL1_REG	Default 1 bits in exit controls.
L4_VM_VMX_ENTRY_CTLS_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](#).

13.110.3 Function Documentation

13.110.3.1 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:



13.110.3.2 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:



13.110.3.3 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:



13.110.3.4 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](#).

13.110.3.5 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](#).

13.110.3.6 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](#).

13.110.3.7 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 vmptld.

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

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

Here is the call graph for this function:

**13.110.3.8 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.

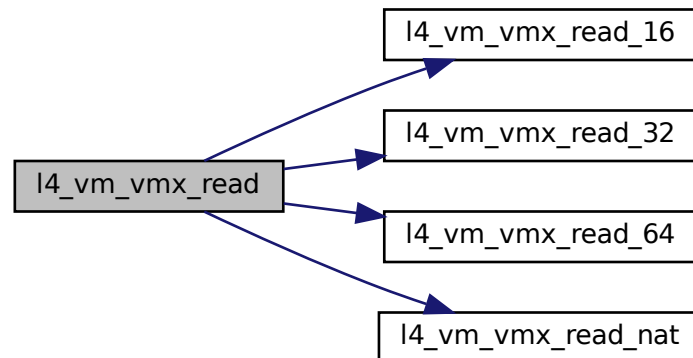
Returns

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:

**13.110.3.9 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:

**13.110.3.10 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:



13.110.3.11 `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:



13.110.3.12 `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:



13.110.3.13 `l4_vm_vmx_write()`

```
void l4_vm_vmx_write (
    void * vmcs,
    unsigned field,
    l4_uint64_t val ) [inline]
```

Write to an arbitrary VMCS field.

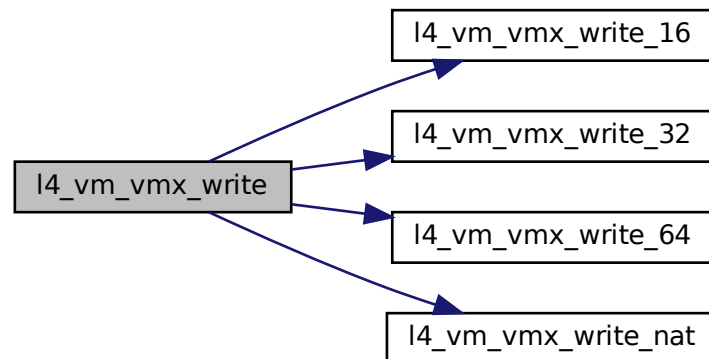
Parameters

<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:



13.110.3.14 `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:



13.110.3.15 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:



13.110.3.16 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:



13.110.3.17 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\(\)](#).

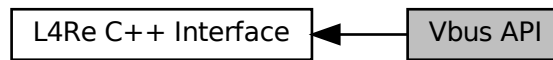
Here is the caller graph for this function:



13.111 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.

13.111.1 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

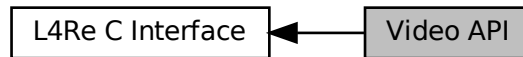
```
#include <l4/vbus/vbus_gpio>
```

Include File

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


13.112 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.

13.112.1 Detailed Description

13.112.2 Typedef Documentation

13.112.2.1 [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.

13.112.3 Enumeration Type Documentation

13.112.3.1 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](#).

13.112.3.2 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](#).

13.112.4 Function Documentation

13.112.4.1 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

13.112.4.2 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.
-------------	--------------------------

Return values

<i>view</i>	the structure will be initialized for the new view.
-------------	---

13.112.4.3 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())

13.112.4.4 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.

13.112.4.5 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.

13.112.4.6 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.

Return values

<i>view</i>	the structure will be 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\(\)](#).

13.112.4.7 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
-------------	-------------

Return values

<i>ginfo</i>	Pointer to goos information structure.
--------------	--

Returns

0 for success, <0 on error

- [-L4_ENODEV](#)
- IPC errors

13.112.4.8 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

13.112.4.9 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.
-------------	------------------------------------

Return values

<i>info</i>	a buffer receiving the information about the view.
-------------	--

13.112.4.10 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

13.112.4.11 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.

13.112.4.12 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.

13.112.4.13 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.

13.113 Virtual Console

Virtual console for simple character based input and output.

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.*

13.113.1 Detailed Description

Virtual console for simple character based input and output.

The interrupt for read events is provided by the virtual key interrupt.

Include File

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

See [L4::Vcon](#) for the C++ interface.

13.113.2 Enumeration Type Documentation

13.113.2.1 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 189 of file [vcon.h](#).

13.113.2.2 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 211 of file [vcon.h](#).

13.113.2.3 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 200 of file [vcon.h](#).

13.113.2.4 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 95 of file [vcon.h](#).

13.113.3 Function Documentation

13.113.3.1 l4_vcon_get_attr()

```
l4_msgtag_t l4_vcon_get_attr (
    l4_cap_idx_t vcon,
    l4_vcon_attr_t * attr ) [inline]
```

Get attributes of a Vcon.

Parameters

	<i>vcon</i>	Vcon object.
out	<i>attr</i>	Attribute structure.

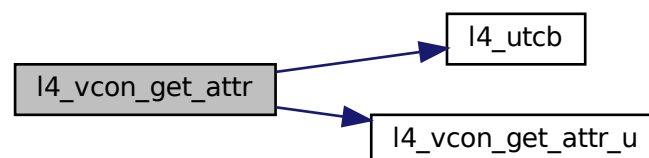
Returns

Syscall return tag

Definition at line 408 of file [vcon.h](#).

References [l4_utcb\(\)](#), and [l4_vcon_get_attr_u\(\)](#).

Here is the call graph for this function:



13.113.3.2 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 pointer of the calling thread.

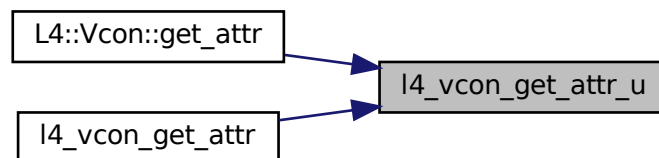
Returns

Syscall return tag.

Definition at line 390 of file [vcon.h](#).

Referenced by [L4::Vcon::get_attr\(\)](#), and [l4_vcon_get_attr\(\)](#).

Here is the caller graph for this function:



13.113.3.3 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><0</code>	Error code.
<code>>size</code>	If more bytes are to read, <code>size</code> bytes are in the buffer <code>buf</code> .
<code><=size</code>	Number of bytes read.

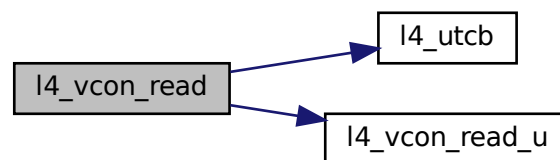
Note

Size must not exceed [L4_VCON_READ_SIZE](#).

Definition at line [364](#) of file [vcon.h](#).

References [l4_utcb\(\)](#), and [l4_vcon_read_u\(\)](#).

Here is the call graph for this function:

13.113.3.4 `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 pointer of the calling thread.

Return values

<code><0</code>	Error code.
--------------------	-------------

Return values

$>size$	More bytes to read, <i>size</i> bytes are in the buffer <i>buf</i> .
$\leq size$	Number of bytes read.

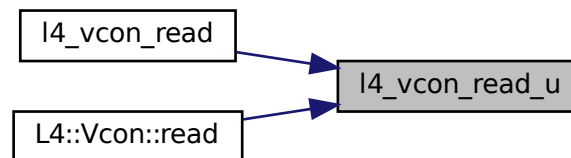
Note

Size must not exceed [L4_VCON_READ_SIZE](#).

Definition at line [354](#) of file [vcon.h](#).

Referenced by [l4_vcon_read\(\)](#), and [L4::Vcon::read\(\)](#).

Here is the caller graph for this function:



13.113.3.5 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><0</code>	Error code.
<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 [348](#) of file [vcon.h](#).

13.113.3.6 l4_vcon_send()

```
l4_msgtag_t l4_vcon_send (
    l4_cap_idx_t vcon,
    char const * buf,
    unsigned size ) [inline]
```

Send data to virtual console.

Parameters

<code>vcon</code>	Vcon object.
<code>buf</code>	Pointer to data buffer.
<code>size</code>	Size of buffer in bytes.

Returns

Syscall return tag

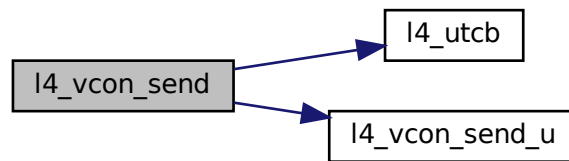
Note

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

Definition at line [288](#) of file [vcon.h](#).

References [l4_utcb\(\)](#), and [l4_vcon_send_u\(\)](#).

Here is the call graph for this function:



13.113.3.7 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>	UTBC pointer of the calling thread.

Returns

Syscall return tag

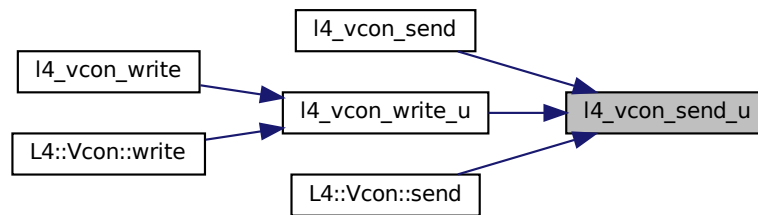
Note

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 275 of file [vcon.h](#).

Referenced by [l4_vcon_send\(\)](#), [l4_vcon_write_u\(\)](#), and [L4::Vcon::send\(\)](#).

Here is the caller graph for this function:



13.113.3.8 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.

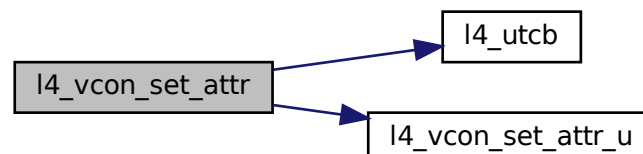
Returns

Syscall return tag

Definition at line 384 of file [vcon.h](#).

References [l4_utcb\(\)](#), and [l4_vcon_set_attr_u\(\)](#).

Here is the call graph for this function:



13.113.3.9 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 pointer of the calling thread.

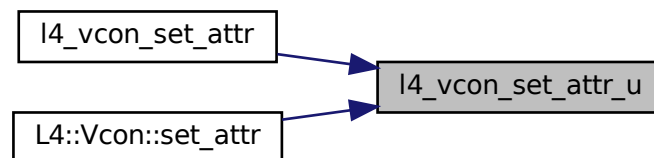
Returns

Syscall return tag.

Definition at line 370 of file [vcon.h](#).

Referenced by [l4_vcon_set_attr\(\)](#), and [L4::Vcon::set_attr\(\)](#).

Here is the caller graph for this function:



13.113.3.10 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.

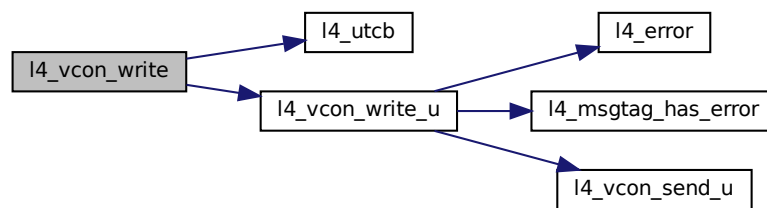
Return values

<code><0</code>	Error.
<code>>=0</code>	Number of bytes written to the virtual console

Definition at line 309 of file [vcon.h](#).

References [l4_utcb\(\)](#), and [l4_vcon_write_u\(\)](#).

Here is the call graph for this function:



13.113.3.11 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 pointer of the calling thread.

Return values

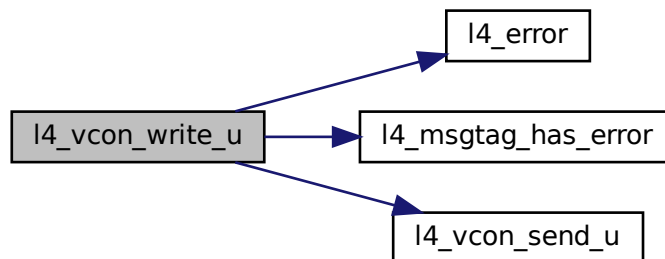
<code><0</code>	Error.
<code>>=0</code>	Number of bytes written to the virtual console.

Definition at line 294 of file [vcon.h](#).

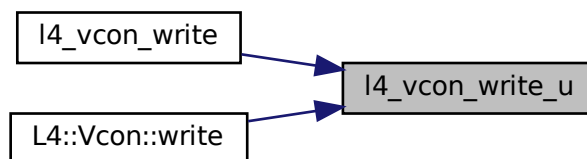
References [l4_error\(\)](#), [l4_msgtag_has_error\(\)](#), [l4_vcon_send_u\(\)](#), and [L4_VCON_WRITE_SIZE](#).

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

Here is the call graph for this function:



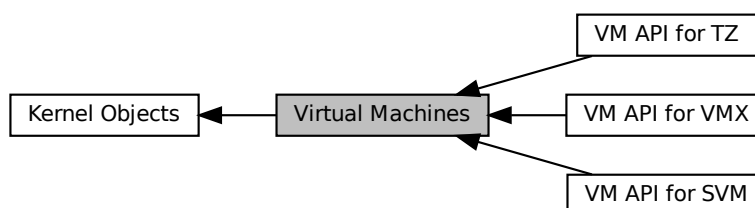
Here is the caller graph for this function:



13.114 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.

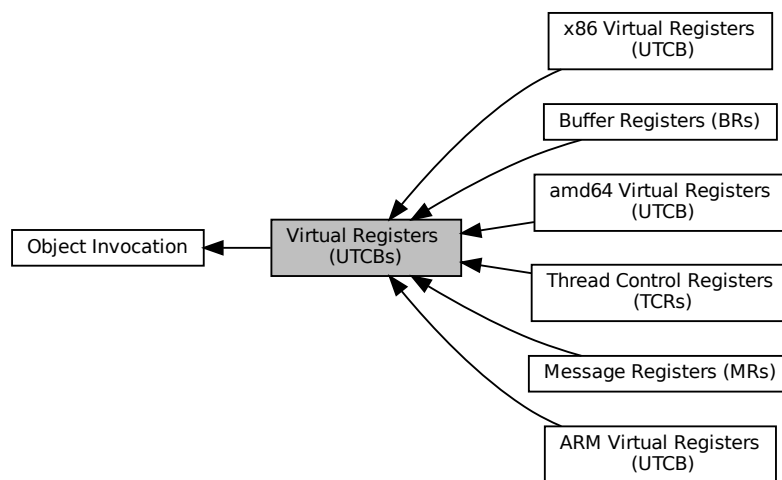
13.114.1 Detailed Description

Virtual Machine API.

13.115 Virtual Registers (UTCBs)

[L4](#) Virtual Registers (UTCb).

Collaboration diagram for Virtual Registers (UTCb):



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.

13.115.1 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.

13.115.2 Typedef Documentation

13.115.2.1 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 1 of file [utcb.h](#).

13.115.3 Function Documentation

13.115.3.1 l4_utcb_br()

```
l4_buf_regs_t * l4_utcb_br (  
    void ) [inline]
```

Get the buffer-register block of a UTCB.

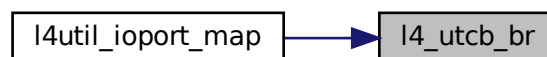
Returns

A pointer to the buffer-register block of `u`.

Definition at line 355 of file [utcb.h](#).

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

Here is the caller graph for this function:



13.115.3.2 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`.

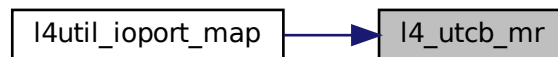
Examples

[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](#).

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

Here is the caller graph for this function:



13.115.3.3 l4_utcb_tcr()

```
l4_thread_regs_t * l4_utcb_tcr (  
    void ) [inline]
```

Get the thread-control-register block of a UTCB.

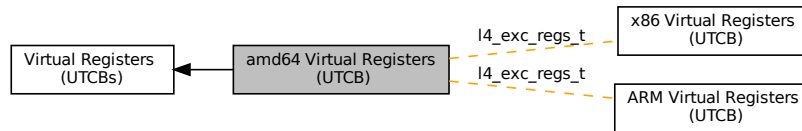
Returns

A pointer to the thread-control-register block of `u`.

Definition at line [358](#) of file [utcb.h](#).

13.116 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

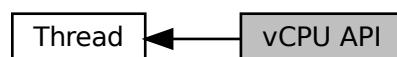
- enum [L4_utcb_consts_amd64](#)
UTCB constants for AMD64.

13.116.1 Detailed Description

13.117 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_DEBUG_EXC](#) = 0x08 ,
[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.

13.117.1 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](#).

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](#)

13.117.2 Enumeration Type Documentation

13.117.2.1 L4_vcpu_state_flags

enum [L4_vcpu_state_flags](#)

State flags of a vCPU.

Enumerator

L4_VCPU_F_IRQ	IRQs (events) enabled.
L4_VCPU_F_PAGE_FAULTS	Page faults enabled.
L4_VCPU_F_EXCEPTIONS	Exception enabled.
L4_VCPU_F_DEBUG_EXC	Debug exception enabled.
L4_VCPU_F_USER_MODE	User task will be used.
L4_VCPU_F_FPU_ENABLED	FPU enabled.

Definition at line 71 of file [vcpu.h](#).

13.117.2.2 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 94 of file [vcpu.h](#).

13.117.2.3 L4_vcpu_sticky_flags

enum [L4_vcpu_sticky_flags](#)

Sticky flags of a vCPU.

Enumerator

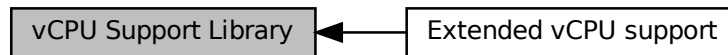
L4_VCPU_SF_IRQ_PENDING	An event (e.g. IRQ) is pending.
------------------------	---------------------------------

Definition at line 85 of file [vcpu.h](#).

13.118 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.

13.118.1 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.

13.118.2 Function Documentation

13.118.2.1 l4vcpu_irq_disable()

```
void l4vcpu_irq_disable (
    l4_vcpu_state_t * vcpu ) [inline]
```

Disable a vCPU for event delivery.

Parameters

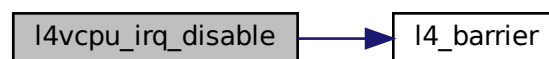
<i>vcpu</i>	Pointer to vCPU area.
-------------	-----------------------

Definition at line 208 of file [vcpu.h](#).

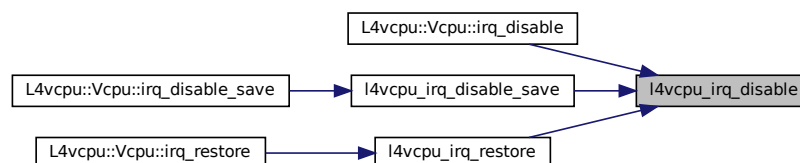
References [l4_barrier\(\)](#), and [L4_VCPU_F_IRQ](#).

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:



13.118.2.2 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

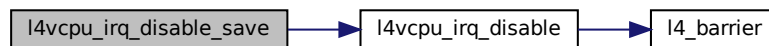
IRQ state before disabling IRQs.

Definition at line 216 of file [vcpu.h](#).

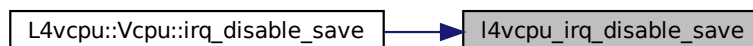
References [l4vcpu_irq_disable\(\)](#).

Referenced by [L4vcpu::Vcpu::irq_disable_save\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.118.2.3 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

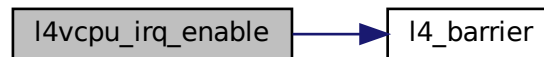
<i>vcpu</i>	Pointer to vCPU area.
<i>utcb</i>	Utc b pointer of the calling vCPU.
<i>do_event_work_cb</i>	Call-back function that is called in case an event (such as an interrupt) is pending.
<i>setup_ipc</i>	Function call-back that is called right before any IPC operation, and before event delivery is enabled.

Definition at line 239 of file [vcpu.h](#).

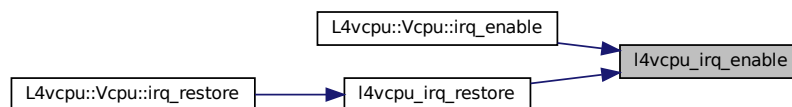
References [l4_barrier\(\)](#), [L4_LIKELY](#), [L4_VCPU_F_IRQ](#), and [L4_VCPU_SF_IRQ_PENDING](#).

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:



13.118.2.4 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

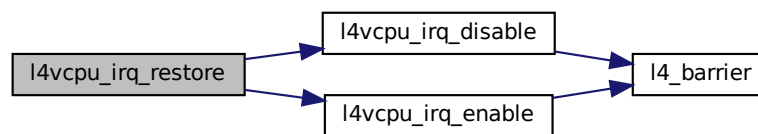
<i>vcpu</i>	Pointer to vCPU area.
<i>s</i>	IRQ state to be restored.
<i>utcb</i>	Utcbl pointer of the calling vCPU.
<i>do_event_work_cb</i>	Call-back function that is called in case an event (such as an interrupt) is pending after enabling.
<i>setup_ipc</i>	Function call-back that is called right before any IPC operation, and before event delivery is enabled.

Definition at line 264 of file [vcpu.h](#).

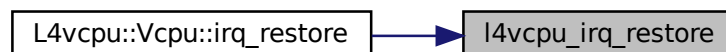
References [L4_VCPU_F_IRQ](#), [l4vcpu_irq_disable\(\)](#), and [l4vcpu_irq_enable\(\)](#).

Referenced by [L4vcpu::Vcpu::irq_restore\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.118.2.5 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.

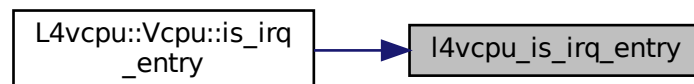
Parameters

<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:



13.118.2.6 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.

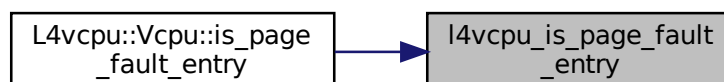
Parameters

<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:



13.118.2.7 l4vcpu_print_state()

```
void l4vcpu_print_state (
    const l4_vcpu_state_t * vcpu,
    const char * prefix )
```

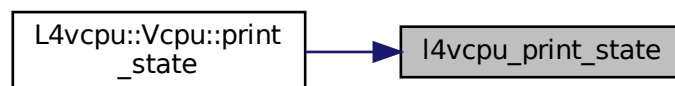
Print the state of a vCPU.

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:



13.118.2.8 l4vcpu_wait_for_event()

```
void l4vcpu_wait_for_event (
    l4_vcpu_state_t * vcpu,
    l4_utcb_t * utcb,
    l4vcpu_event_hndl_t do_event_work_cb,
    l4vcpu_setup_ipc_t setup_ipc ) [inline]
```

Wait for event.

Parameters

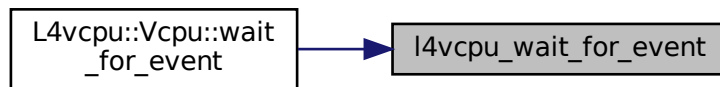
<i>vcpu</i>	Pointer to vCPU area.
<i>utcb</i>	Utc b pointer of the calling vCPU.
<i>do_event_work_cb</i>	Call-back function that is called when the vCPU awakes and needs to handle an event/IRQ.
<i>setup_ipc</i>	Function call-back that is called right before any IPC operation.

Note that event delivery remains disabled after this function returns.

Definition at line 277 of file [vcpu.h](#).

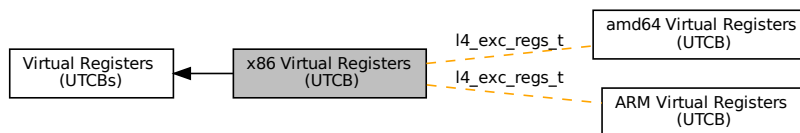
Referenced by [L4vcpu::Vcpu::wait_for_event\(\)](#).

Here is the caller graph for this function:



13.119 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.

13.119.1 Detailed Description

13.119.2 Enumeration Type Documentation

13.119.2.1 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](#).

Chapter 14

Namespace Documentation

14.1 cxx Namespace Reference

Our C++ library.

Namespaces

- [Bits](#)

Internal helpers for the cxx package.

Data Structures

- class [Auto_ptr](#)
Smart pointer with automatic deletion.
- class [Avl_map](#)
AVL tree based associative container.
- class [Avl_set](#)
AVL set for simple compareable items.
- class [Avl_tree_node](#)
Node of an AVL tree.
- class [Avl_tree](#)
A generic AVL tree.
- class [Bitfield](#)
Definition for a member (part) of a bit field.
- class [Bitmap_base](#)
Basic bitmap abstraction.
- class [Bitmap](#)
A static bit map.
- class [H_list_item_t](#)
Basic element type for a double-linked [H_list](#).
- class [H_list](#)
General double-linked list of unspecified [cxx::H_list_item](#) elements.
- struct [H_list_t](#)
Double-linked list of typed [H_list_item_t](#) elements.

- class [List_item](#)
Basic list item.
- class [List](#)
Doubly linked list, with internal allocation.
- class [List_alloc](#)
Standard list-based allocator.
- struct [Pair](#)
Pair of two values.
- class [Pair_first_compare](#)
Comparison functor for [Pair](#).
- class [Ref_ptr](#)
A reference-counting pointer with automatic cleanup.
- struct [Ref_obj_list_item](#)
Item for list linked via [cxx::Ref_ptr](#) with default reference counting.
- class [Base_slab](#)
Basic slab allocator.
- class [Slab](#)
Slab allocator for object of type `Type`.
- class [Base_slab_static](#)
Merged slab allocator (allocators for objects of the same size are merged together).
- class [Slab_static](#)
Merged slab allocator (allocators for objects of the same size are merged together).
- class [S_list](#)
Simple single-linked list.
- class [static_vector](#)
Simple encapsulation for a dynamically allocated array.
- class [Nothrow](#)
Helper type to distinguish the `operator new` version that does not throw exceptions.
- class [New_allocator](#)
Standard allocator based on `operator new ()`.
- struct [Lt_functor](#)
Generic comparator class that defaults to the less-than operator.
- class [String](#)
Allocation free string class with explicit length field.
- class [Weak_ref_base](#)
Generic (base) weak reference to some object.
- class [Weak_ref](#)
Typed weak reference to an object of type `T`.

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.

14.1.1 Detailed Description

Our C++ library.

Small Low-Level C++ Library.

Strings.

Various kinds of C++ utilities.

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 Bst`
Basic binary search tree (BST).
- `struct Direction`
The direction to go in a binary search tree.
- `class Bst_node`
Basic type of a node in a binary search tree (BST).
- `class Basic_list`
Internal: Common functions for all head-based list implementations.
- `class Smart_ptr_list_item`
List item for an arbitrary item in a Smart_ptr_list.
- `class Smart_ptr_list`
List of smart-pointer-managed objects.

14.2.1 Detailed Description

Internal helpers for the cxx package.

14.3 L4 Namespace Reference

[L4](#) low-level kernel interface.

Namespaces

- [lpc](#)
IPC related functionality.
- [lpc_svr](#)
Helper classes for [L4::Server](#) instantiation.
- [Typeid](#)
Definition of interface data-type helpers.
- [Types](#)
[L4](#) basic type helpers for C++.

Data Structures

- struct [Type_info](#)
Dynamic Type Information for [L4Re](#) Interfaces.
- struct [Kobject_typeid](#)
Meta object for handling access to type information of Kobjects.
- struct [Kobject_typeid< void >](#)
Minimalistic ID for `void` interface.
- class [Kobject_t](#)
Helper class to create an [L4Re](#) interface class that is derived from a single base class.
- 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...`
- struct [Proto_t](#)
Data type for defining protocol numbers.
- struct [Kobject_x](#)
Generic [Kobject](#) inheritance template.
- class [Vm](#)
Virtual machine host address space.
- class [Arm_smccc](#)
Wrapper for function calls that follow the ARM SMC/HVC calling convention.
- class [Cap](#)
C++ interface for capabilities.
- class [Cap_base](#)
Base class for all kinds of capabilities.

- struct [Epiface](#)
Base class for interface implementations.
- struct [Epiface_t0](#)
[Epiface](#) mixin for generic Kobject-based interfaces.
- struct [Irqep_t](#)
[Epiface](#) implementation for interrupt handlers.
- class [Registry_iface](#)
Abstract interface for object registries.
- struct [Epiface_t](#)
[Epiface](#) implementation for Kobject-based interface implementations.
- class [Basic_registry](#)
This registry returns the corresponding server object based on the label of an [lpc_gate](#).
- class [Server](#)
Basic server loop for handling client requests.
- class [Debugger](#)
C++ kernel debugger API.
- class [Exception](#)
[Exception](#) interface.
- class [Factory](#)
C++ Factory interface.
- class [lommu](#)
Interface for IO-MMUs used for DMA remapping.
- class [lpc_gate](#)
The C++ IPC gate interface.
- class [Irq_eoi](#)
Interface for sending an acknowledge message to an object.
- struct [Triggerable](#)
Interface that allows an object to be triggered by some source.
- class [Irq](#)
C++ [Irq](#) interface.
- struct [Irq_mux](#)
IRQ multiplexer for shared IRQs.
- class [lcu](#)
C++ [lcu](#) interface.
- class [Kobject](#)
Base class for all kinds of kernel objects and remote objects, referenced by capabilities.
- class [Meta](#)
[Meta](#) interface that shall be implemented by each [L4Re](#) object and gives access to the dynamic type information for [L4Re](#) objects.
- class [lo_pager](#)
[lo_pager](#) interface.
- class [Pager](#)
[Pager](#) interface including the [lo_pager](#) interface.
- class [Platform_control](#)
[L4](#) C++ interface for controlling platform-wide properties.
- class [Rcv_endpoint](#)
Interface for kernel objects that allow to receive IPC from them.
- class [Scheduler](#)
C++ interface of the [Scheduler](#) kernel object.
- struct [Semaphore](#)
Kernel-provided semaphore object.

- class [Smart_cap](#)
Smart capability class.
- class [Task](#)
C++ interface of the [Task](#) kernel object.
- class [Thread](#)
C++ [L4](#) kernel thread interface.
- class [Vcon](#)
C++ [L4](#) [Vcon](#) interface.
- class [Poll_timeout_kipclock](#)
A polling timeout based on the [L4Re](#) clock.
- class [Alloc_list](#)
A simple list-based allocator.
- class [IOModifier](#)
Modifier class for the IO stream.
- class [Exception_tracer](#)
Back-trace support for exceptions.
- class [Base_exception](#)
Base class for all exceptions, thrown by the [L4Re](#) framework.
- class [Runtime_error](#)
Exception for an abstract runtime error.
- class [Out_of_memory](#)
Exception signalling insufficient memory.
- class [Element_already_exists](#)
Exception for duplicate element insertions.
- class [Unknown_error](#)
Exception for an unknown condition.
- class [Element_not_found](#)
Exception for a failed lookup (element not found).
- class [Invalid_capability](#)
Indicates that an invalid object was invoked.
- class [Com_error](#)
Error conditions during IPC.
- class [Bounds_error](#)
Access out of bounds.
- 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.*
- struct [Irq_handler_object](#)
Server object base class for handling IRQ messages.
- class [String](#)
A null-terminated string container class.

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

PROTO_ANY	Default protocol used by Kobject_t and Kobject_x .
PROTO_EMPTY	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 379 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 T.
<i>L4_INVALID_CAP</i>	<i>c</i> does not support the target interface T 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.

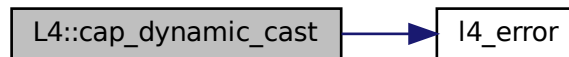
Example code:

```
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 410 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} .
------------	--

Parameters

<i>order</i>	order (2^{order}) to round up to.
--------------	--

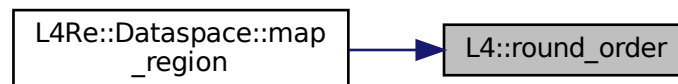
Returns

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, usually the UTCB of the calling thread.

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.

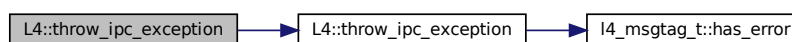
Parameters

<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, usually the UTCB of the calling thread.

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.

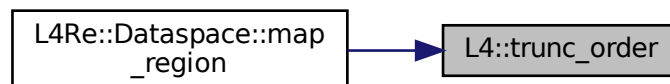
Returns

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

- [Msg](#)

IPC Message related functionality.

Data Structures

- struct [Array_ref](#)
Array reference data type for arrays located in the message.
- struct [Array](#)
Array data type for dynamically sized arrays in RPCs.
- struct [Array_in_buf](#)
Server-side copy in buffer for [Array](#).
- struct [Call](#)
RPC attribute for a standard RPC call.
- struct [Call_zero_send_timeout](#)
RPC attribute for an RPC call, with zero send timeout.
- struct [Call_t](#)
RPC attribute for an RPC call with required rights.
- struct [Send_only](#)
RPC attribute for a send-only RPC.
- struct [Ret_array](#)
Dynamically sized output array of type T.
- struct [Out](#)
Mark an argument as a output value in an RPC signature.
- struct [In_out](#)
Mark an argument as in-out argument.
- struct [As_value](#)
Pass the argument as plain data value.
- struct [Opt](#)
Attribute for defining an optional RPC argument.
- class [Small_buf](#)
A receive item for receiving a single capability.
- class [Snd_item](#)
RPC wrapper for a send item.
- class [Buf_item](#)
RPC wrapper for a receive item.
- class [Gen_fpage](#)
Generic RPC wrapper for [L4](#) flex-pages.
- class [Cap](#)
Capability type for RPC interfaces (see [L4 : : Cap<T>](#)).
- 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.
- class [Str_cp_in](#)
Abstraction for extracting a zero-terminated string from an [lpc::Istream](#).
- class [Msg_ptr](#)
Pointer to an element of type T in an [lpc::Istream](#).
- class [Istream](#)
Input stream for IPC unmarshalling.
- class [Ostream](#)
Output stream for IPC marshalling.
- class [lostream](#)
Input/Output stream for IPC [un]marshalling.

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::lstream](#).

Parameters

	<i>v</i>	Pointer to the array that shall receive the values from the lpc::lstream .
<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::lstream](#). 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 172 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 113 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::lstream .
out	<i>size</i>	The number of elements found in the stream.

This routine provides a possibility to extract an array from an [lpc::lstream](#), without extra copy overhead. In contrast to [buf_cp_in\(\)](#) the data is not copied to a buffer, but a pointer to the array is returned. The user must make sure the UTCB is not used for other purposes while the returned pointer is still in use.

The mechanism is comparable to that of [Msg_ptr](#), however it handles arrays inserted with [buf_cp_out\(\)](#).

See also

[buf_cp_in\(\)](#) and [buf_cp_out\(\)](#).

Definition at line 323 of file [ipc_stream](#).

14.4.2.4 make_cap()

```
template<typename T >
Cap<T> L4::Ipc::make_cap (
    L4::Cap< T > cap,
    unsigned rights ) [noexcept]
```

Make an `L4::lpc::Cap<T>` for the given capability and rights.

Template Parameters

<i>T</i>	(IMPLICIT) type of the referenced interface
----------	---

Parameters

<i>cap</i>	source capability (<code>L4::Cap<T></code>)
<i>rights</i>	rights mask that shall be applied on transfer.

Definition at line 624 of file [ipc_types](#).

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::ipc_gate::bind_thread\(\)](#).

Definition at line 662 of file [ipc_types](#).

References [L4_CAP_FPAGE_RWSD](#), and [L4_FPAGE_C_OBJ_RIGHTS](#).

14.4.2.6 make_cap_rw()

```
template<typename T >
Cap<T> L4::Ipc::make_cap_rw (
    L4::Cap< T > cap ) [noexcept]
```

Make an L4::ipc::Cap<T> for the given capability with [L4_CAP_FPAGE_RW](#) rights.

Template Parameters

<i>T</i>	(IMPLICIT) type of the referenced interface
----------	---

Parameters

<i>cap</i>	source capability (L4::Cap<T>)
------------	--------------------------------

Examples

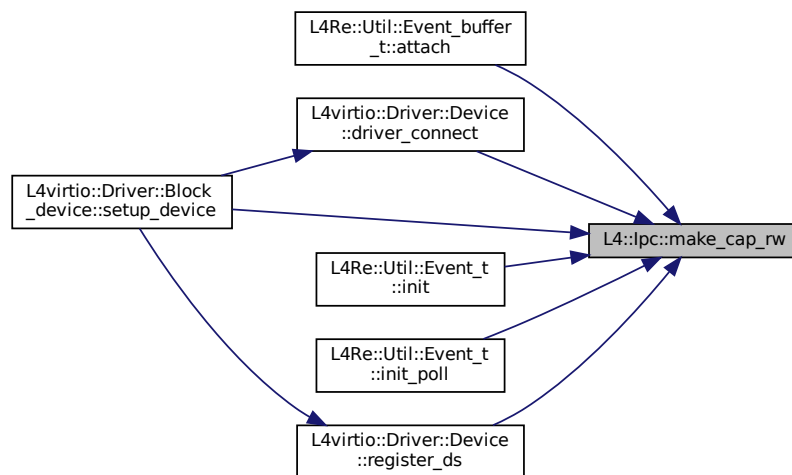
[examples/libs/l4re/c++/mem_alloc/ma+rm.cc](#), [examples/libs/l4re/c++/shared_ds/ds_clnt.cc](#), and [examples/libs/l4re/c++/shared_](#)

Definition at line 634 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\(\)](#), 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 644 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 [ipc::Istream](#). This may be used to avoid copy out of large data structures. (See [Msg_ptr](#).)

Definition at line 265 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 1404 of file [ipc_stream](#).

14.4.2.10 str_cp_in()

```
template<typename T >
Str_cp_in<T> L4::Ipc::str_cp_in (
    T * v,
    unsigned long & size )
```

Create a [Str_cp_in](#) for the given values.

Parameters

	<i>v</i>	Pointer to the array that shall receive the values from the lpc::lstream .
<i>in, out</i>	<i>size</i>	Input: the number of elements the array can take at most Output: the number of elements found in the stream.

This function makes it more convenient to extract arrays from an [lpc::lstream](#) (

See also

[Str_cp_in](#).)

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

14.5 L4::lpc::Msg Namespace Reference

IPC Message related functionality.

Data Structures

- struct [Elem< Array< A, LEN > >](#)
Array as input arguments.
- struct [Elem< Array< A, LEN > & >](#)
Array as output argument.
- struct [Elem< Array_ref< A, LEN > & >](#)
Array_ref as output argument.
- struct [Dir_in](#)
Marker type for input values.
- struct [Dir_out](#)
Marker type for output values.
- struct [Cls_data](#)
Marker type for data values.
- struct [Cls_item](#)
Marker type for item values.
- struct [Cls_buffer](#)
Marker type for receive buffer values.
- struct [Do_in_data](#)
Marker for Input data.
- struct [Do_out_data](#)
Marker for Output data.
- struct [Do_in_items](#)
Marker for Input items.
- struct [Do_out_items](#)
Marker for Output items.
- struct [Do_rcv_buffers](#)
Marker for receive buffers.
- struct [Clnt_val_ops](#)
Defines client-side handling of 'MTYPE' as RPC argument.
- struct [Svr_val_ops](#)
Defines server-side handling for MTYPE server arguments.
- struct [ls_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.

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 [Server_iface](#)
Interface for server-loop related functions.
- struct [Ignore_errors](#)
Mix in for LOOP_HOOKS to ignore IPC errors.
- struct [Default_timeout](#)
Mix in for LOOP_HOOKS to use a 0 send and a infinite receive timeout.
- struct [Compound_reply](#)
Mix in for LOOP_HOOKS to always use compound reply and wait.
- struct [Default_setup_wait](#)
Mix in for LOOP_HOOKS for setup_wait no op.
- class [Timed_work](#)
DEPRECATED.
- 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.
- class [Br_manager_no_buffers](#)
Empty implementation of [Server_iface](#).
- struct [Default_loop_hooks](#)
Default LOOP_HOOKS.
- 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](#)
Standard list of RPCs of an interface.
- struct [Rpc_code](#)
List of RPCs of an interface using a special opcode type.
- struct [Rpc_nocode](#)
List of RPCs of an interface using a single operation without an opcode.
- 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

- class [Flags](#)
Template for defining typical [Flags](#) bitmaps.
- 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 [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 [Bool](#)
Boolean meta type.
- struct [False](#)
[False](#) meta value.
- struct [True](#)
[True](#) meta value.
- struct [Same](#)
Compare two data types for equality.

14.8.1 Detailed Description

[L4](#) basic type helpers for C++.

14.9 L4Re Namespace Reference

[L4Re](#) C++ Interfaces.

Namespaces

- [Util](#)
Documentation of the [L4](#) Runtime Environment utility functionality in C++.
- [Vfs](#)
Virtual file system for interfaces POSIX libc.

Data Structures

- class [Cap_alloc](#)
Capability allocator interface.
- 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 [Console](#)
[Console](#) class.
- class [Dataspace](#)
Interface for memory-like objects.
- class [Debug_obj](#)
Debug interface.
- class [Dma_space](#)
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.

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 (L4::Cap , L4Re::Util::Unique_cap 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](#).

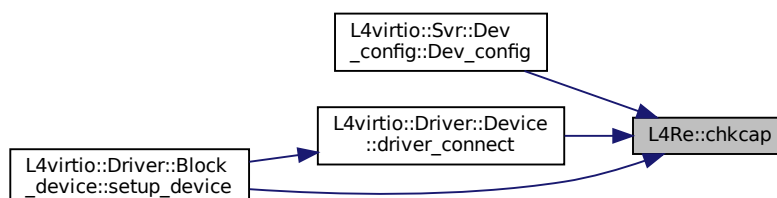
References [L4_UNLIKELY](#), and [throw_error\(\)](#).

Referenced by [L4virtio::Svr::Dev_config::Dev_config\(\)](#), [L4virtio::Driver::Device::driver_connect\(\)](#), and [L4virtio::Driver::Block_device::s](#)

Here is the call graph for this function:



Here is the caller graph for this function:



14.9.3.2 chkipc()

```
l4_msgtag_t L4Re::chkipc (
    l4_msgtag_t tag,
    char const * extra = "",
    l4_utcb_t * utcb = l4_utcb() ) [inline]
```

Test a message tag for IPC errors.

Parameters

<i>tag</i>	Message tag returned by the IPC.
<i>extra</i>	Exception message in case of error.
<i>utcb</i>	The UTCB used in the IPC operation.

Returns

On IPC error an exception is thrown, otherwise `tag` is returned.

Exceptions

<i>L4::Runtime_exception</i>	with the translated IPC error code
------------------------------	------------------------------------

This function does not check the message tag's label value.

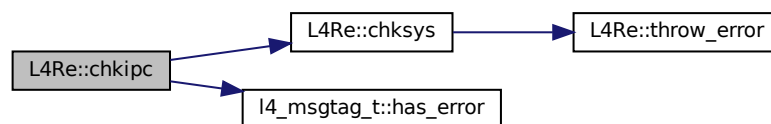
Note

This must be called on a message tag before the UTCB is changed.

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

References [chksys\(\)](#), [l4_msgtag_t::has_error\(\)](#), and [L4_UNLIKELY](#).

Here is the call graph for this function:



14.9.3.3 chksys() [1/3]

```
long L4Re::chksys (
    l4_msgtag_t const & t,
    char const * extra = "",
    l4_utcb_t * utcb = l4_utcb(),
    long ret = 0 ) [inline]
```

Generate C++ exception on error.

Parameters

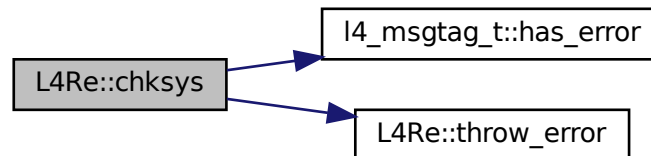
<i>t</i>	Message tag.
<i>extra</i>	Optional text for exception (default "")
<i>utcb</i>	Option UTCB
<i>ret</i>	Optional value for exception, default is error value (err)

This function throws an exception if the message tag contains an error or the label in the message tag is negative. Otherwise the label in the message tag is returned.

Definition at line 89 of file [error_helper](#).

References [l4_msgtag_t::has_error\(\)](#), [L4_UNLIKELY](#), 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

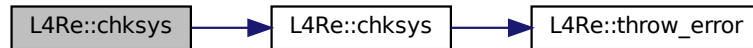
<i>t</i>	Message tag.
<i>utcb</i>	UTCB.
<i>extra</i>	Optional text for exception (default "")

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](#).

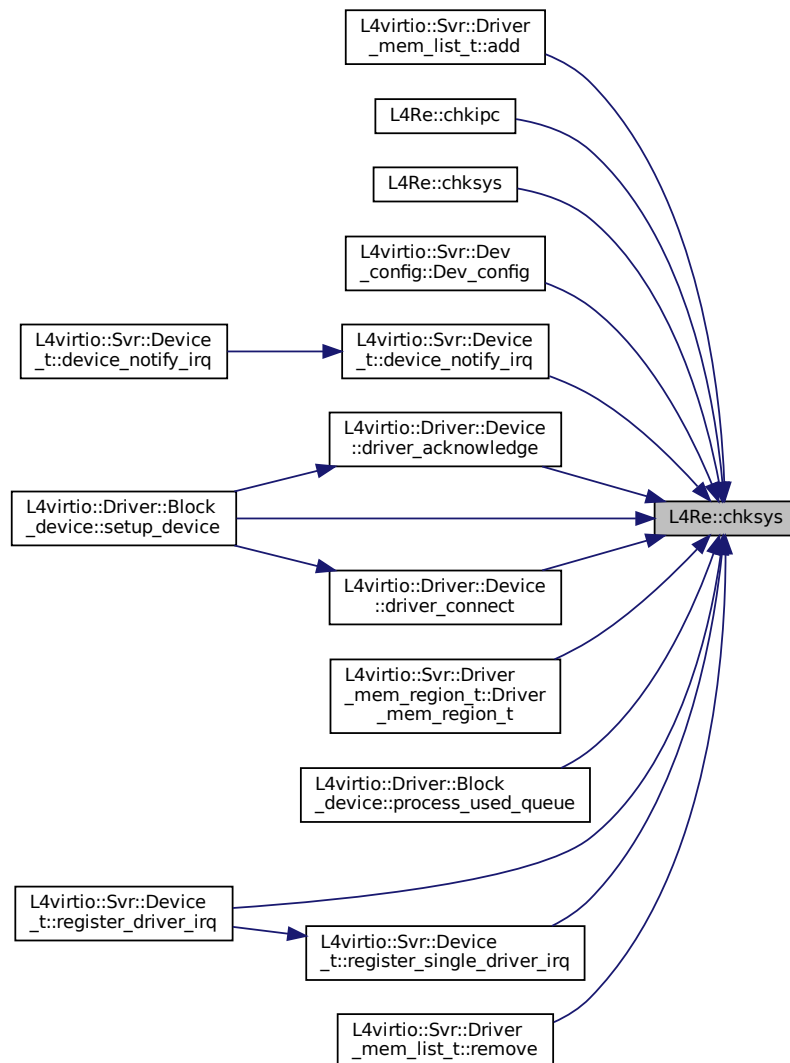
References [L4_UNLIKELY](#), and [throw_error\(\)](#).

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::Driver::Device::driver_acknowledge\(\)](#), [L4virtio::Driver::Device::driver_c](#), [L4virtio::Svr::Driver_mem_region_t< DATA >::Driver_mem_region_t\(\)](#), [L4virtio::Driver::Block_device::process_used_queue\(\)](#), [L4virtio::Svr::Device_t< DATA >::register_driver_irq\(\)](#), [L4virtio::Svr::Device_t< DATA >::register_single_driver_irq\(\)](#), [L4virtio::Svr::Driver_mem_list_t< DATA >::remove\(\)](#), and [L4virtio::Driver::Block_device::setup_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.9.3.6 make_shared_cap()

```

template<typename T >
Shared_cap<T> L4Re::make_shared_cap (
    L4Re::Cap_alloc * ca )

```

Allocate a capability slot and wrap it in a `Shared_cap`.

Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

Parameters

<i>ca</i>	Capability allocator to use.
-----------	------------------------------

Note

This function is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::make_shared_cap<T>\(\)](#).

Definition at line 60 of file [shared_cap](#).

References [L4Re::Cap_alloc::alloc\(\)](#).

Here is the call graph for this function:



14.9.3.7 make_shared_del_cap()

```

template<typename T >
Shared_del_cap<T> L4Re::make_shared_del_cap (
    L4Re::Cap_alloc * ca )
  
```

Allocate a capability slot and wrap it in a Shared_del_cap.

Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

Parameters

<i>ca</i>	Capability allocator to use.
-----------	------------------------------

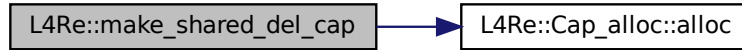
Note

This function is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::make_shared_del_cap<T>\(\)](#).

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

References [L4Re::Cap_alloc::alloc\(\)](#).

Here is the call graph for this function:



14.9.3.8 make_unique_cap()

```

template<typename T >
Unique_cap<T> L4Re::make_unique_cap (
    L4Re::Cap_alloc * ca )
  
```

Allocate a capability slot and wrap it in an Unique_cap.

Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

Parameters

<i>ca</i>	Capability allocator to use.
-----------	------------------------------

Note

This function is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::make_unique_cap<T>\(\)](#).

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

References [L4Re::Cap_alloc::alloc\(\)](#).

Here is the call graph for this function:



14.9.3.9 make_unique_del_cap()

```
template<typename T >
Unique_del_cap<T> L4Re::make_unique_del_cap (
    L4Re::Cap_alloc * ca )
```

Allocate a capability slot and wrap it in an Unique_del_cap.

Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

Parameters

<i>ca</i>	Capability allocator to use.
-----------	------------------------------

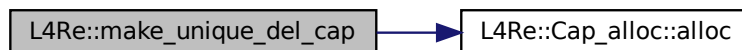
Note

This function is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::make_unique_del_cap<T>\(\)](#).

Definition at line 91 of file [unique_cap](#).

References [L4Re::Cap_alloc::alloc\(\)](#).

Here is the call graph for this function:



14.9.3.10 throw_error()

```
void L4Re::throw_error (
    long err,
    char const * extra = "" ) [inline]
```

Generate C++ exception.

Parameters

<i>err</i>	Error value
<i>extra</i>	Optional text for exception (default "")

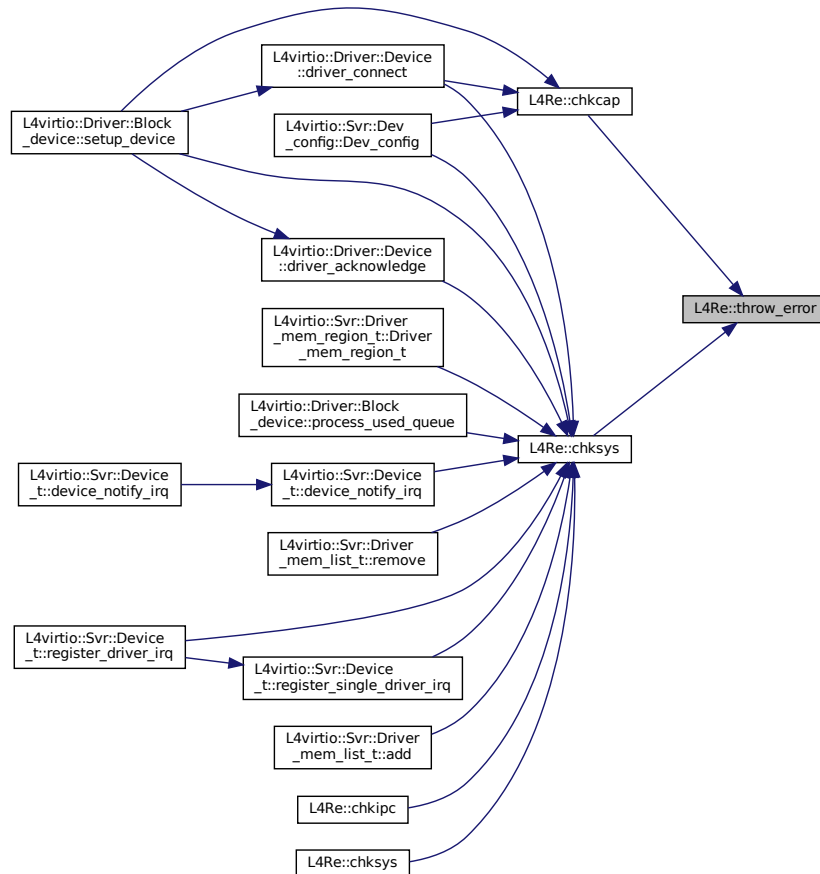
This function throws an [L4](#) exception. The exact exception type depends on the error value (err). This function does never return.

Definition at line 45 of file [error_helper](#).

References [L4_EEXIST](#), [L4_ENOENT](#), [L4_ENOMEM](#), and [L4_ERANGE](#).

Referenced by [chkcap\(\)](#), 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 [Cap_alloc_base](#)
Capability allocator.
- 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 [Smart_cap_auto](#)
Helper for [Unique_cap](#) and [Unique_del_cap](#).
- class [Smart_count_cap](#)
Helper for [Ref_cap](#) and [Ref_del_cap](#).
- 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.
- struct [Counter](#)
[Counter](#) for [Counting_cap_alloc](#) with variable data width.
- class [Counting_cap_alloc](#)
Internal reference-counting cap allocator.
- class [Dataspace_svr](#)
[Dataspace](#) server class.
- class [Event_t](#)
Convenience wrapper for getting access to an event object.
- class [Event_buffer_t](#)
[Event_buffer](#) utility class.
- class [Event_buffer_consumer_t](#)
An event buffer consumer.
- class [Event_svr](#)
Convenience wrapper for implementing an event server.
- 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.
- class [Registry_server](#)
A server loop object which has a [Object_registry](#) included.
- 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

<i>T</i>	Type of the object the capability refers to.
----------	--

The ownership of the capability is managed in the same way as `unique_ptr`.

Usage:

```

{
    L4Re::Util::Unique_cap<L4Re::Dataspace>
        ds_cap = L4Re::Util::make_unique_cap<L4Re::Dataspace>();

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(L4_PAGESIZE, ds_cap.get()));

    ...

    // At the end of the scope ds_cap is unmapped and the capability
    // selector is freed.
}

```

Definition at line 57 of file [unique_cap](#).

14.10.2.7 Unique_del_cap

```

template<typename T >
using L4Re::Util::Unique_del_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>
>

```

Unique capability that implements automatic free and unmap+delete of the capability selector.

Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

The main difference to `Unique_cap` is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Usage:

```
{
    L4Re::Util::Unique_del_cap<L4Re::Dataspace>
        ds_cap = make_unique_del_cap<L4Re::Dataspace>();

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(L4_PAGESIZE, ds_cap.get()));

    ...

    // At the end of the scope ds_cap is unmapped and the capability
    // selector is freed. Because the deletion flag is set the data space
    // shall also be deleted (even if there are other references to this
    // data space).
}
```

Definition at line 97 of file `unique_cap`.

14.10.2.8 unique_del_cap

```
template<typename T >
using L4Re::Util::unique_del_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>
>
```

Unique capability that implements automatic free and unmap+delete of the capability selector.

Template Parameters

<code>T</code>	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 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.

14.10.3.2 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](#).

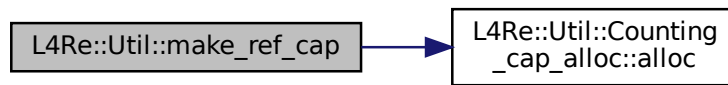
Template Parameters

<i>T</i>	Type of capability the slot is used for.
----------	--

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

References [L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc\(\)](#), and [cap_alloc](#).

Here is the call graph for this function:



14.10.3.3 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](#).

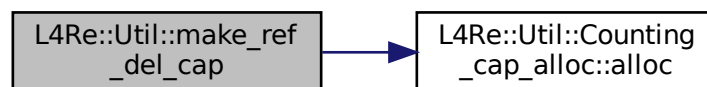
Template Parameters

<i>T</i>	Type of capability the slot is used for.
----------	--

Definition at line 227 of file [cap_alloc](#).

References [L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc\(\)](#), and [cap_alloc](#).

Here is the call graph for this function:



14.10.3.4 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`.

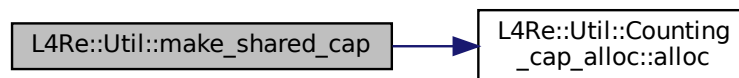
Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

Definition at line 71 of file [shared_cap](#).

References [L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc\(\)](#), and [cap_alloc](#).

Here is the call graph for this function:

14.10.3.5 `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 [L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc\(\)](#), and [cap_alloc](#).

Here is the call graph for this function:



14.10.3.6 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.

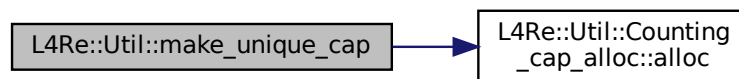
Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

Definition at line 66 of file [unique_cap](#).

References [L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc\(\)](#), and [cap_alloc](#).

Here is the call graph for this function:



14.10.3.7 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.

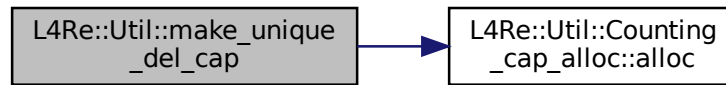
Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

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

References [L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc\(\)](#), and [cap_alloc](#).

Here is the call graph for this function:



14.10.4 Variable Documentation

14.10.4.1 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/ds/ds_srvr.cc](#), and [examples/libs/l4re/streammap/client.cc](#).

Referenced by [L4Re::Util::Smart_count_cap< Unmap_flags >::copy\(\)](#), [L4Re::Util::Smart_cap_auto< Unmap_flags >::free\(\)](#), [L4Re::Util::Smart_count_cap< Unmap_flags >::free\(\)](#), [make_ref_cap\(\)](#), [make_ref_del_cap\(\)](#), [make_shared_cap\(\)](#), [make_shared_del_cap\(\)](#), [make_unique_cap\(\)](#), [make_unique_del_cap\(\)](#), and [L4Re::Util::Object_registry::unregister_obj\(\)](#).

14.11 L4Re::Vfs Namespace Reference

Virtual file system for interfaces 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 [Generic_file](#)
The common interface for an open POSIX file.
- class [Directory](#)
Interface for a POSIX file that is a directory.
- 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.
- class [File](#)
The basic interface for an open POSIX file.
- class [Mman](#)
Interface for the POSIX memory management.
- class [File_system](#)
Basic interface for an [L4Re::Vfs](#) file system.
- class [Fs](#)
POSIX File-system related functionality.
- class [Ops](#)
Interface for the POSIX backends for an application.

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 POSIX libc.

14.12 L4vbus Namespace Reference

C++ interface of the [Vbus](#) API.

Data Structures

- class [Pm](#)
Power-management API mixin.
- class [Device](#)
Device on a [L4vbus::Vbus](#).
- class [lcu](#)
Vbus Interrupt controller API.
- class [Vbus](#)
The virtual bus ([Vbus](#)) interface.
- class [Gpio_pin](#)
A GPIO pin.
- class [Gpio_module](#)
A [Gpio_module](#) groups multiple GPIO pins together.
- class [Pci_host_bridge](#)
A Pci host bridge.
- class [Pci_dev](#)
A PCI device.

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 [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.

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 `cxx::Auto_ptr< T >` Class Template Reference

Smart pointer with automatic deletion.

Collaboration diagram for `cxx::Auto_ptr< T >`:

<code>cxx::Auto_ptr< T ></code>
<div><div>+ <code>Auto_ptr()</code></div><div>+ <code>Auto_ptr()</code></div><div>+ <code>operator=()</code></div><div>+ <code>~Auto_ptr()</code></div><div>+ <code>operator*()</code></div><div>+ <code>operator->()</code></div><div>+ <code>get()</code></div><div>+ <code>release()</code></div><div>+ <code>reset()</code></div><div>+ <code>operator Priv_type *()</code></div></div>

Public Types

- typedef T [Ref_type](#)
The referenced type.

Public Member Functions

- [Auto_ptr](#) (T *p=0) throw ()
Construction by assignment of a normal pointer.
- [Auto_ptr](#) ([Auto_ptr](#) const &o) throw ()
Copy construction, releases the original pointer.
- [Auto_ptr](#) & [operator=](#) ([Auto_ptr](#) const &o) throw ()
Assignment from another smart pointer.
- [~Auto_ptr](#) () throw ()
Destruction, shall delete the object.
- T & [operator*](#) () const throw ()
Dereference the pointer.
- T * [operator->](#) () const throw ()
Member access for the object.
- T * [get](#) () const throw ()
Get the normal pointer.
- T * [release](#) () throw ()
Release the object and get the normal pointer back.
- void [reset](#) (T *p=0) throw ()
Delete the object and reset the smart pointer to NULL.
- [operator Priv_type *](#) () const throw ()
Operator for `if (!ptr)`

15.1.1 Detailed Description

```
template<typename T>
class cxx::Auto_ptr< T >
```

Smart pointer with automatic deletion.

Template Parameters

<i>T</i>	The type of the referenced object.
----------	------------------------------------

This smart pointer calls the delete operator when the destructor is called. This has the effect that the object the pointer points to will be deleted when the pointer goes out of scope, or a new value gets assigned. The smart pointer provides a [release\(\)](#) method to get a normal pointer to the object and set the smart pointer to NULL.

Definition at line 36 of file [auto_ptr](#).

15.1.2 Constructor & Destructor Documentation

15.1.2.1 Auto_ptr() [1/2]

```
template<typename T >
cxx::Auto_ptr< T >::Auto_ptr (
    T * p = 0 ) throw ( )    [inline], [explicit]
```

Construction by assignment of a normal pointer.

Parameters

<code>p</code>	The pointer to the object
----------------	---------------------------

Definition at line 51 of file [auto_ptr](#).

15.1.2.2 `Auto_ptr()` [2/2]

```
template<typename T >
cxx::Auto_ptr< T >::Auto_ptr (
    Auto_ptr< T > const & o ) throw ( )    [inline]
```

Copy construction, releases the original pointer.

Parameters

<code>o</code>	The smart pointer, which shall be copied and released.
----------------	--

Definition at line 57 of file [auto_ptr](#).

15.1.3 Member Function Documentation**15.1.3.1 `get()`**

```
template<typename T >
T* cxx::Auto_ptr< T >::get ( ) const throw ( )    [inline]
```

Get the normal pointer.

Attention

This function will not release the object, the object will be deleted by the smart pointer.

Definition at line 90 of file [auto_ptr](#).

15.1.3.2 `operator=()`

```
template<typename T >
Auto_ptr& cxx::Auto_ptr< T >::operator= (
    Auto_ptr< T > const & o ) throw ( )    [inline]
```

Assignment from another smart pointer.

Parameters

<i>o</i>	The source for the assignment (will be released).
----------	---

Definition at line 65 of file [auto_ptr](#).

References [cxx::Auto_ptr< T >::release\(\)](#).

Here is the call graph for this function:



15.1.3.3 release()

```

template<typename T >
T* cxx::Auto\_ptr< T >::release ( ) throw ( )    [inline]
  
```

Release the object and get the normal pointer back.

After calling this function the smart pointer will point to NULL and the object will not be deleted by the pointer anymore.

Definition at line 98 of file [auto_ptr](#).

Referenced by [cxx::Auto_ptr< T >::operator=\(\)](#).

Here is the caller graph for this function:



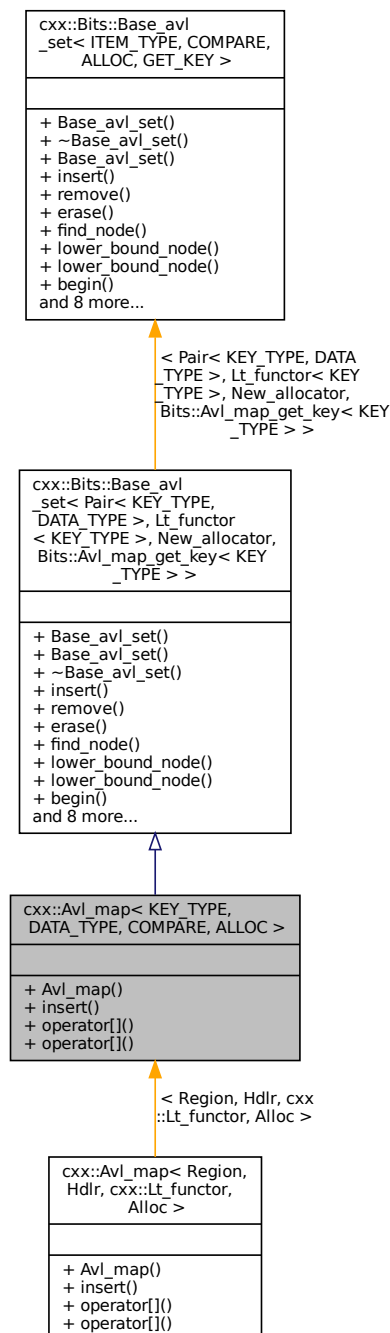
The documentation for this class was generated from the following file:

- `I4/cxx/auto_ptr`

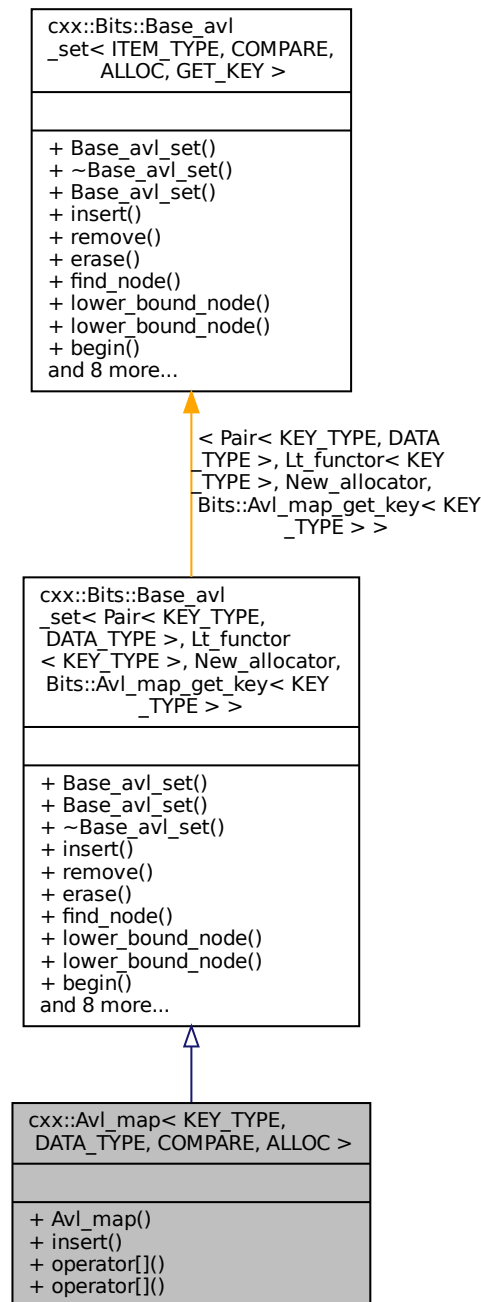
15.2 cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC > Class Template Reference

AVL tree based associative container.

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 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.

15.2.1 Detailed Description

```
template<typename KEY_TYPE, typename DATA_TYPE, template< typename A > class COMPARE = Lt_functor, template< type-
name 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`.

15.2.2 Constructor & Destructor Documentation

15.2.2.1 `Avl_map()`

```
template<typename KEY_TYPE , typename DATA_TYPE , template< typename A > class COMPARE = Lt_
_functor, template< typename B > class ALLOC = New_allocator>
cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::Avl_map (
    Node_allocator const & alloc = Node_allocator() ) [inline]
```

Create an empty AVL-tree based map.

Parameters

<i>alloc</i>	The node allocator.
--------------	---------------------

Definition at line 91 of file [avl_map](#).

15.2.3 Member Function Documentation

15.2.3.1 insert()

```
template<typename KEY_TYPE , typename DATA_TYPE , template< typename A > class COMPARE = Lt↔
_funcutor, template< typename B > class ALLOC = New_allocator>
cxx::Pair<Iterator, int> cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::insert (
    Key_type const & key,
    Data_type const & data ) [inline]
```

Insert a <key, data> pair into the map.

Parameters

<i>key</i>	The key value.
<i>data</i>	The data value to insert.

Returns

A pair of iterator (*first*) and return value (*second*). *second* will be 0 if the element was inserted into the set and `-#E_exist` if the key was already in the set and the set was therefore not updated. In both cases, *first* contains an iterator that points to the element. *second* may also be `-#E_nomem` when memory for the new node could not be allocated. *first* is then invalid.

Definition at line 110 of file [avl_map](#).

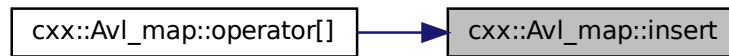
References [cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::insert\(\)](#).

Referenced by [cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::operator\[\]\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.2.3.2 `operator[]()` [1/2]

```

template<typename KEY_TYPE , typename DATA_TYPE , template< typename A > class COMPARE = Lt↵
_functor, template< typename B > class ALLOC = New_allocator>
Data_type& cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::operator[] (
    Key_type const & key ) [inline]
  
```

Get or insert data for the given key.

Parameters

<i>key</i>	The key value to use for lookup.
------------	----------------------------------

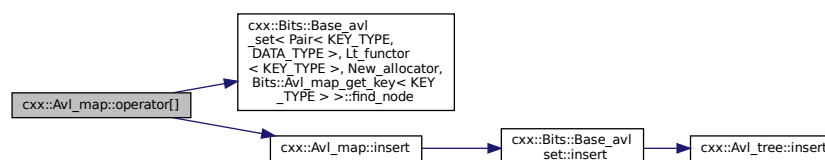
Returns

If the item already exists, a reference to the data item. Otherwise a new data item is default-constructed and inserted under the given key before a reference is returned.

Definition at line 130 of file [avl_map](#).

References [cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, Lt_functor< KEY_TYPE >, New_allocator, Bits::Avl_map_get_key< KEY_TYPE >>::find_node](#), [cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::insert\(\)](#), and [cxx::Pair< First, Second >::second](#).

Here is the call graph for this function:



15.2.3.3 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.
------------	----------------------------------

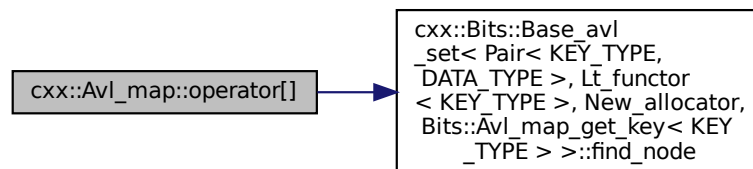
Precondition

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< Pair< KEY_TYPE, DATA_TYPE >, Lt_functor< KEY_TYPE >, New_allocator, Bits::Avl_map_](#)
and [cxx::Pair< First, Second >::second](#).

Here is the call graph for this function:



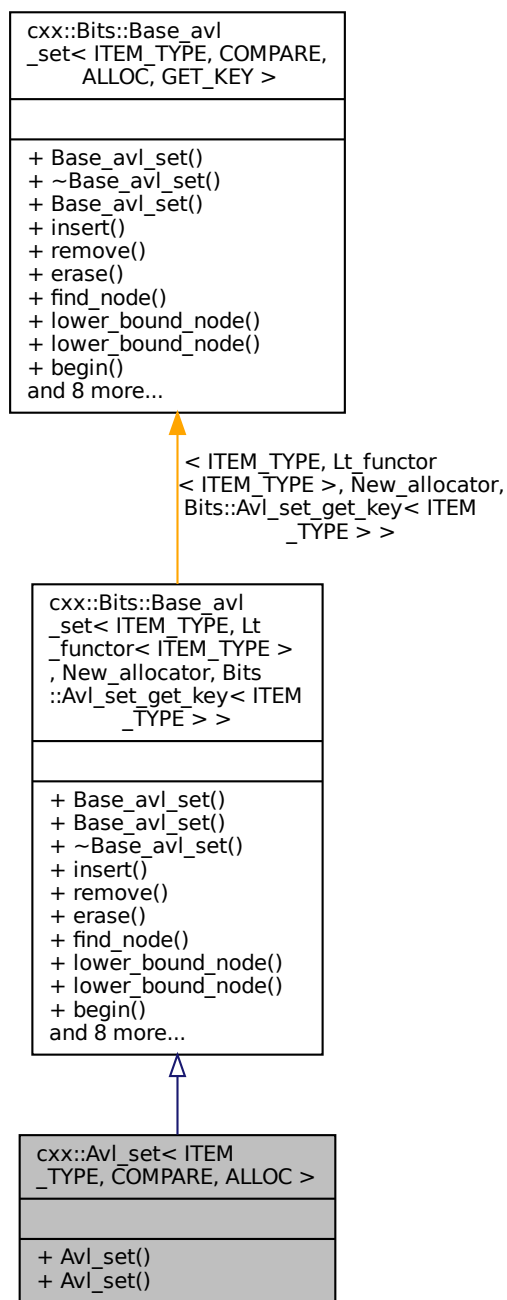
The documentation for this class was generated from the following file:

- [l4/cxx/avl_map](#)

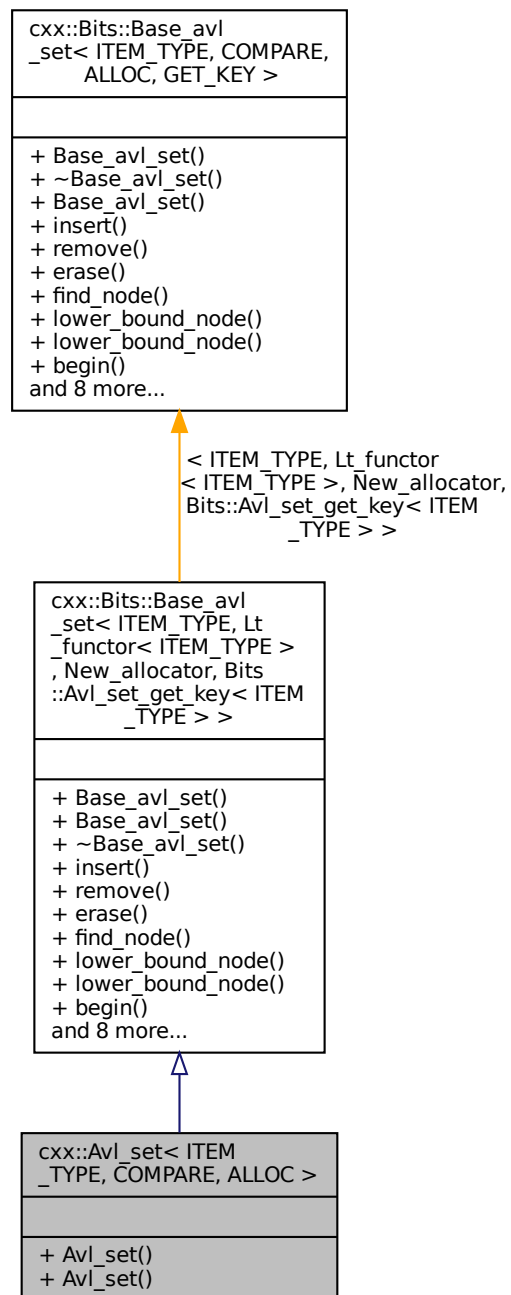
15.3 cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC > Class Template Reference

AVL set for simple comparable items.

Inheritance diagram for cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >:



Collaboration diagram for `cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >`:



Additional Inherited Members

15.3.1 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 440 of file [avl_set](#).

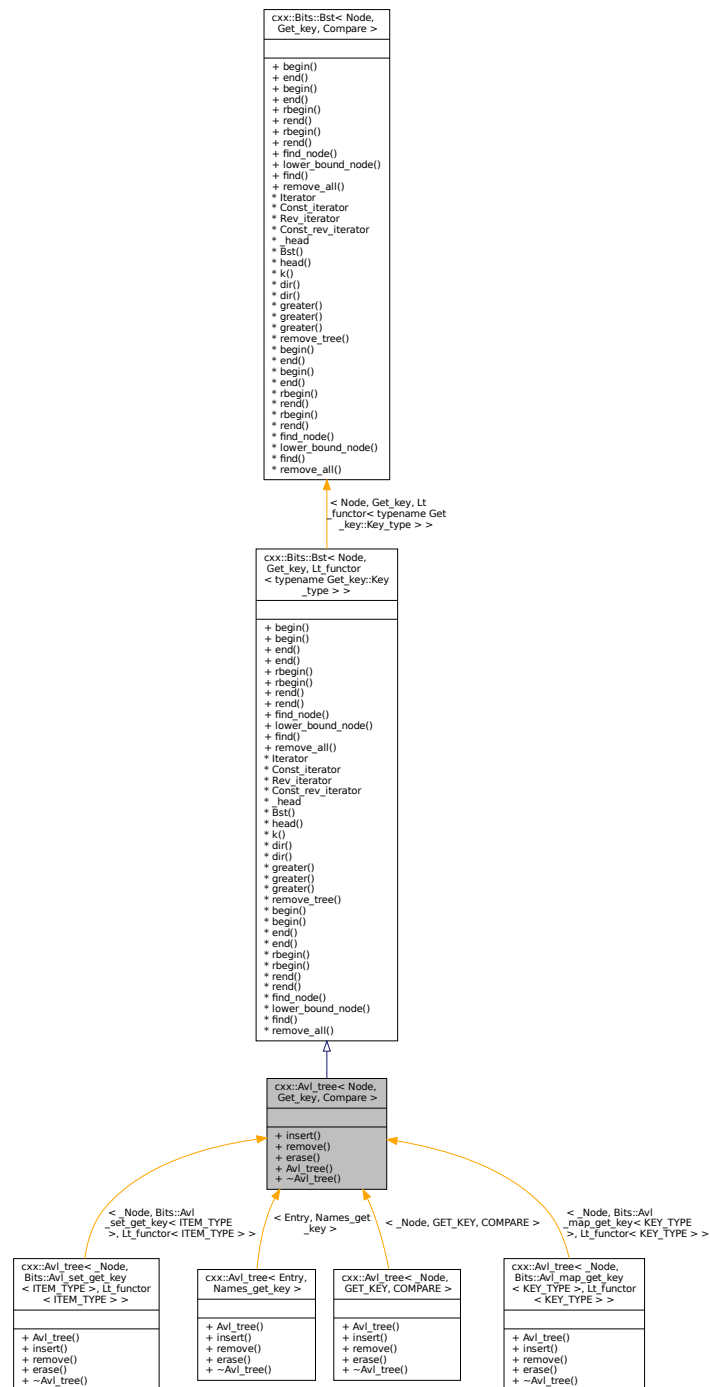
The documentation for this class was generated from the following file:

- l4/cxx/[avl_set](#)

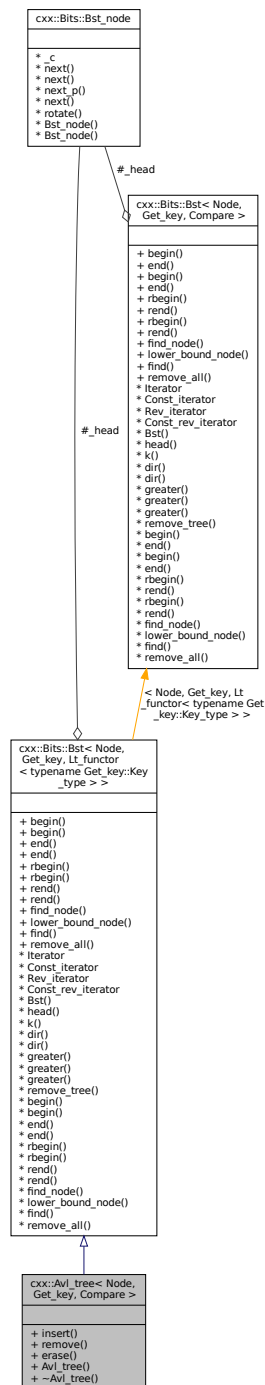
15.4 cxx::Avl_tree< Node, Get_key, Compare > Class Template Reference

A generic AVL tree.

Inheritance diagram for `cxx::Avl_tree< Node, Get_key, Compare >`:



Collaboration diagram for cxx::Avl_tree< Node, Get_key, Compare >:



Public Types

- typedef [Bst::Iterator](#) `Iterator`
Forward iterator for the tree.
- typedef [Bst::Const_iterator](#) `Const_iterator`
Constant forward iterator for the tree.
- typedef [Bst::Rev_iterator](#) `Rev_iterator`

Backward iterator for the tree.

- typedef [Bst::Const_rev_iterator](#) [Const_rev_iterator](#)

Constant backward iterator for the tree.

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.

Additional Inherited Members

15.4.1 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](#).

15.4.2 Member Function Documentation

15.4.2.1 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.
-----------------	----------------------------

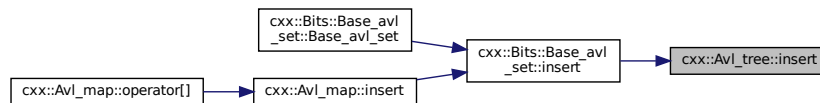
Returns

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:



15.4.2.2 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.
------------	--------------------------------

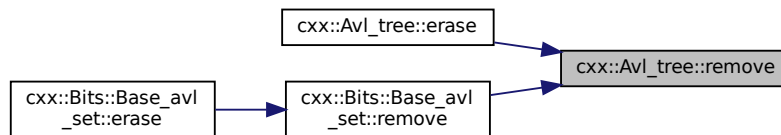
Returns

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:



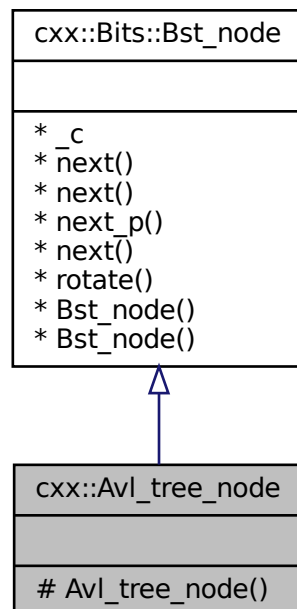
The documentation for this class was generated from the following file:

- [I4/cxx/avl_tree](#)

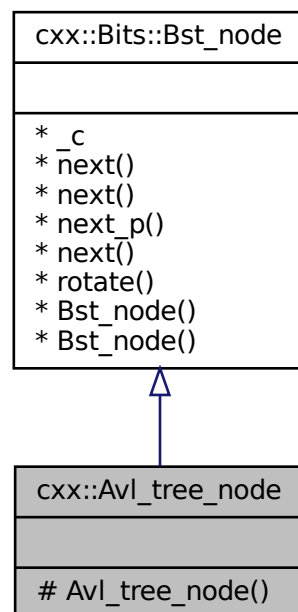
15.5 cxx::Avl_tree_node Class Reference

Node of an 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.

Additional Inherited Members

15.5.1 Detailed Description

Node of an AVL tree.

Examples

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

Definition at line 38 of file [avl_tree](#).

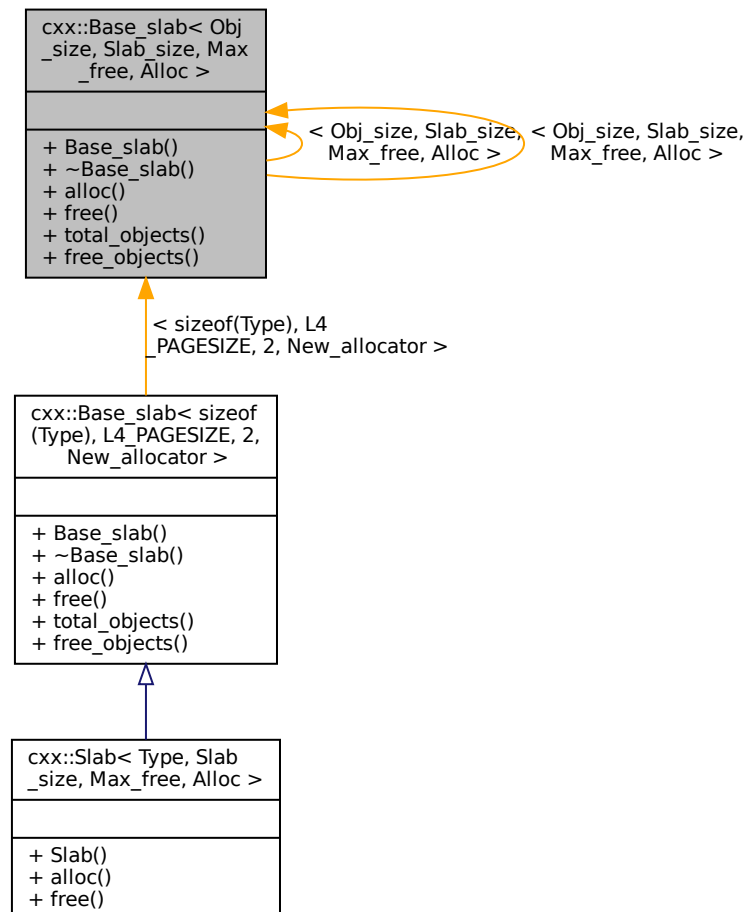
The documentation for this class was generated from the following file:

- [l4/cxx/avl_tree](#)

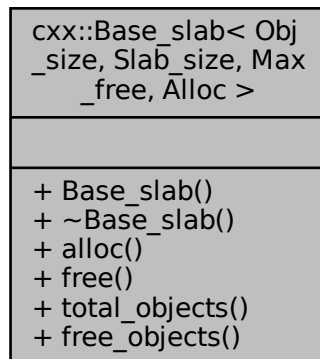
15.6 cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc > Class Template Reference

Basic slab allocator.

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](#) () throw ()
Allocate a new object.
- void [free](#) (void *_o) throw ()
Free the given object (_o).
- unsigned [total_objects](#) () const throw ()
Get the total number of objects managed by the slab allocator.
- unsigned [free_objects](#) () const throw ()
Get the number of objects which can be allocated before a new empty slab needs to be added to the slab allocator.

15.6.1 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](#).

15.6.2 Member Enumeration Documentation

15.6.2.1 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](#).

15.6.3 Member Function Documentation

15.6.3.1 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 ( ) throw ( ) [inline]
```

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

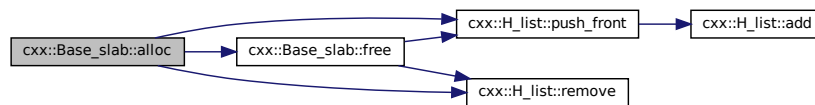
The user is responsible for initializing the object.

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

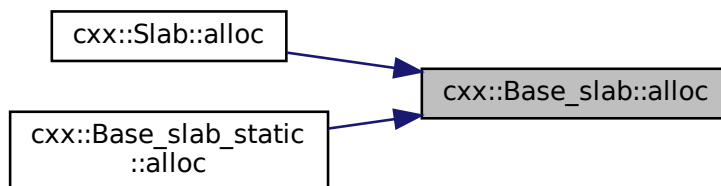
References [cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::free\(\)](#), [cxx::H_list< T, POLICY >::push_front\(\)](#), and [cxx::H_list< T, POLICY >::remove\(\)](#).

Referenced by [cxx::Slab< Type, 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:

**15.6.3.2 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 ) throw ( )    [inline]
  
```

Free the given object (`_o`).

Precondition

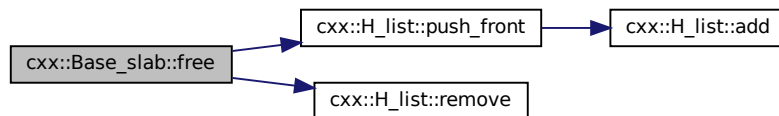
The object must have been allocated with this allocator.

Definition at line 257 of file [slab_alloc](#).

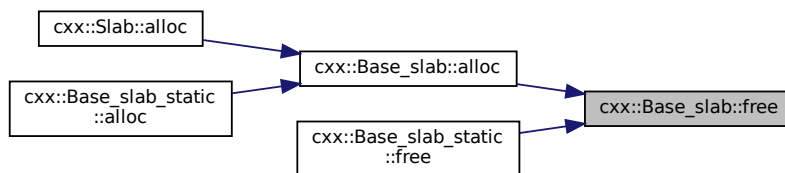
References [cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::max_free_slabs](#), [cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::push_front\(\)](#), [cxx::H_list< T, POLICY >::push_front\(\)](#), [cxx::H_list< T, POLICY >::remove\(\)](#), and [cxx::Base_slab< Obj_size, Slab_size, Max_free, 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:

**15.6.3.3 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 throw
( ) [inline]
  
```

Get the number of objects which can be allocated before a new empty slab needs to be added to the slab allocator.

Returns

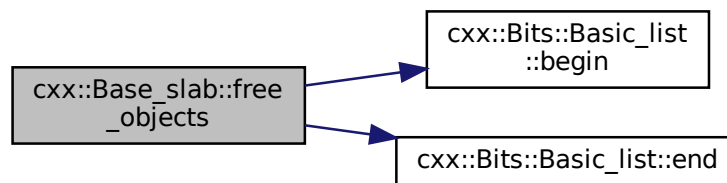
The number of free objects in the slab allocator.

Definition at line 319 of file `slab_alloc`.

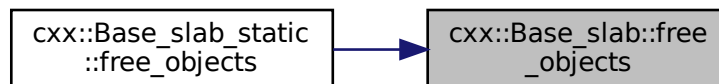
References `cxx::Bits::Basic_list< POLICY >::begin()`, `cxx::Bits::Basic_list< POLICY >::end()`, and `cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::free_objects()`.

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:

**15.6.3.4 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 throw
( ) [inline]
  
```

Get the total number of objects managed by the slab allocator.

Returns

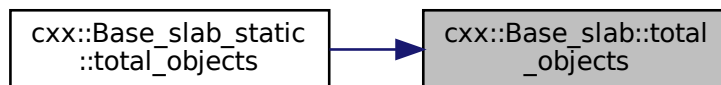
The number of objects managed by the allocator (including the free objects).

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:

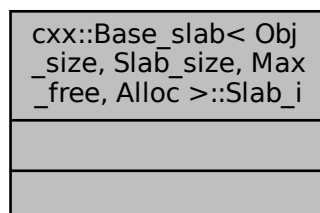
- `I4/cxx/slab_alloc`

15.7 `cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i` Struct Reference

Type of a slab.

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`:



15.7.1 Detailed Description

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class Alloc = New_allocator>
struct cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i
```

Type of a slab.

Definition at line 97 of file [slab_alloc](#).

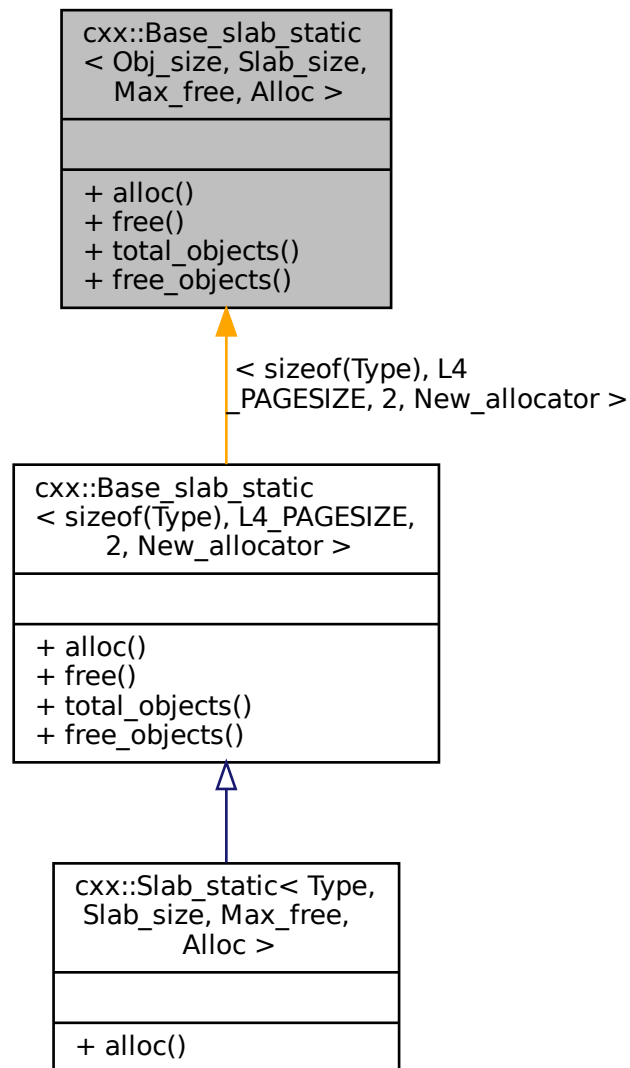
The documentation for this struct was generated from the following file:

- `l4/cxx/slab_alloc`

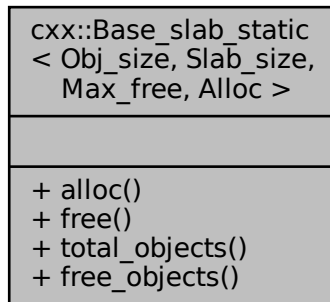
15.8 `cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >` Class Template Reference

Merged slab allocator (allocators for objects of the same size are merged together).

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` () throw ()
Allocate an object.
- void `free` (void *p) throw ()
Free the given object (p).
- unsigned `total_objects` () const throw ()
Get the total number of objects managed by the slab allocator.
- unsigned `free_objects` () const throw ()
Get the number of free objects in the slab allocator.

15.8.1 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](#).

15.8.2 Member Enumeration Documentation

15.8.2.1 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](#).

15.8.3 Member Function Documentation

15.8.3.1 `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 ( ) throw ( ) [inline]
```

Allocate an object.

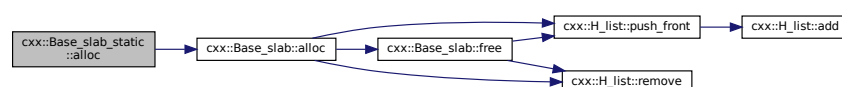
Note

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:



15.8.3.2 `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 ) throw ( )    [inline]
```

Free the given object (`p`).

Parameters

<code>p</code>	The pointer to the object to free.
----------------	------------------------------------

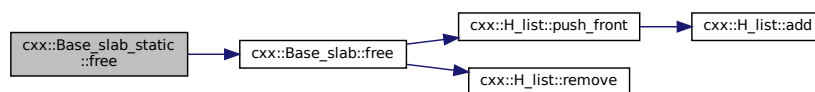
Precondition

`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:



15.8.3.3 `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
throw ( )    [inline]
```

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.

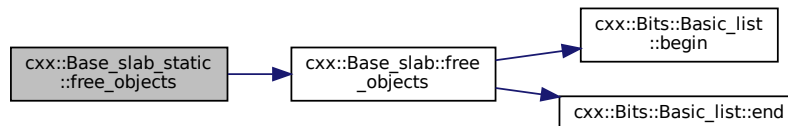
Note

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:

**15.8.3.4 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 throw ( )    [inline]
  
```

Get the total number of objects managed by the slab allocator.

Returns

The number of objects managed by the allocator (including the free objects).

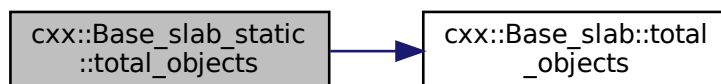
Note

The value is the merged value for all equal parameterized [Base_slab_static](#) instances.

Definition at line 441 of file [slab_alloc](#).

References [cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::total_objects\(\)](#).

Here is the call graph for this function:



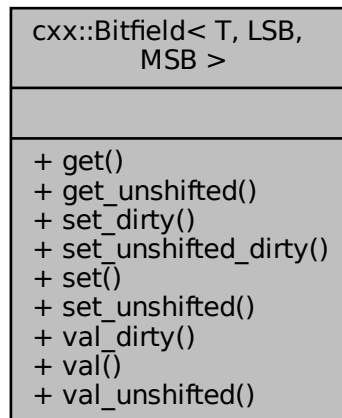
The documentation for this class was generated from the following file:

- `l4/cxx/slab_alloc`

15.9 `cxx::Bitfield< T, LSB, MSB >` Class Template Reference

Definition for a member (part) of a bit field.

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 }
- 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.

15.9.1 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](#).

15.9.2 Member Typedef Documentation

15.9.2.1 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 82 of file [bitfield](#).

15.9.2.2 `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 90 of file `bitfield`.

15.9.3 Member Enumeration Documentation

15.9.3.1 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`.

15.9.3.2 Masks

```
template<typename T , unsigned LSB, unsigned MSB>
enum cxx::Bitfield::Masks : T
```

Enumerator

Low_mask	Mask value to get <code>Bits</code> bits.
Mask	Mask value to the bits out of a <code>T</code> .

Definition at line 68 of file `bitfield`.

15.9.4 Member Function Documentation

15.9.4.1 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.
------------------	---------------------------------------

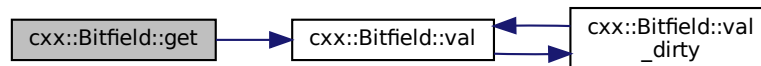
Returns

The bits from `Lsb` to `Msb` shifted to the right.

Definition at line 107 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:



15.9.4.2 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 120 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:



15.9.4.3 `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.

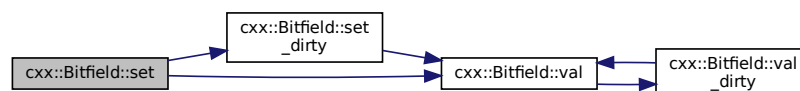
Returns

The new value of the whole bit field.

Definition at line 169 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:



15.9.4.4 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 [135](#) of file [bitfield](#).

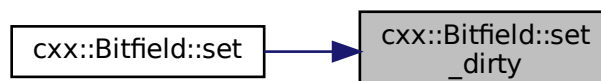
References [cxx::Bitfield< T, LSB, MSB >::Lsb](#), [cxx::Bitfield< T, LSB, MSB >::Mask](#), and [cxx::Bitfield< T, LSB, MSB >::val\(\)](#).

Referenced by [cxx::Bitfield< T, LSB, MSB >::set\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.9.4.5 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.

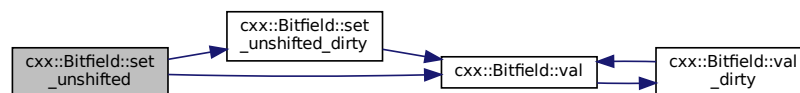
Returns

the new value of the whole bit field.

Definition at line [181](#) 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:



15.9.4.6 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

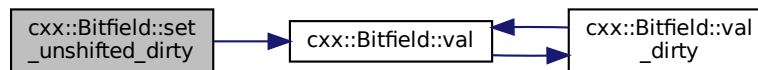
This function does not mask `val` to the right number of bits.

Definition at line 155 of file [bitfield](#).

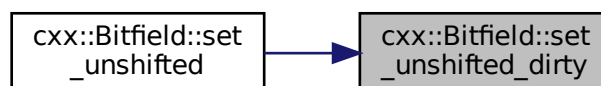
References [cxx::Bitfield< T, LSB, MSB >::Mask](#), and [cxx::Bitfield< T, LSB, MSB >::val\(\)](#).

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:

**15.9.4.7 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

<i>val</i>	The value to set into the bits.
------------	---------------------------------

Returns

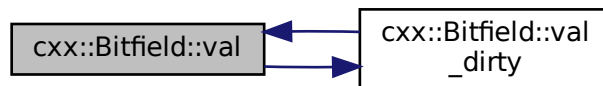
The raw bit field value.

Definition at line 204 of file [bitfield](#).

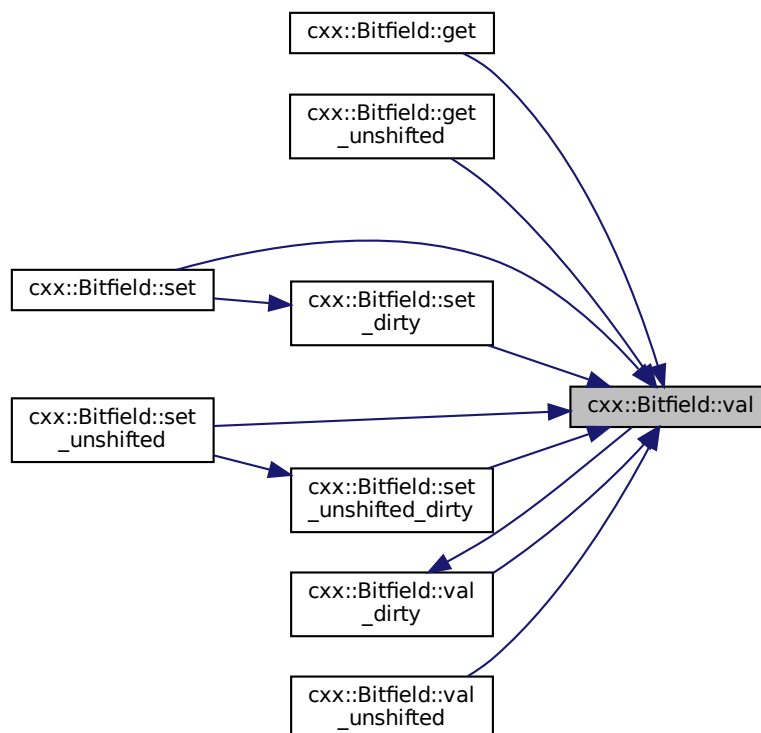
References [cxx::Bitfield< T, LSB, MSB >::Low_mask](#), and [cxx::Bitfield< T, LSB, MSB >::val_dirty\(\)](#).

Referenced by [cxx::Bitfield< T, LSB, MSB >::get\(\)](#), [cxx::Bitfield< T, LSB, MSB >::get_unshifted\(\)](#), [cxx::Bitfield< T, LSB, MSB >::set\(\)](#), [cxx::Bitfield< T, LSB, MSB >::set_dirty\(\)](#), [cxx::Bitfield< T, LSB, MSB >::set_unshifted\(\)](#), [cxx::Bitfield< T, LSB, MSB >::set_unshifted_dirty\(\)](#), [cxx::Bitfield< T, LSB, MSB >::val_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:



15.9.4.8 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

<code>val</code>	The value to set into the bits.
------------------	---------------------------------

Returns

The raw bit field value.

Precondition

`val` must not contain more than [Bits](#) bits.

Note

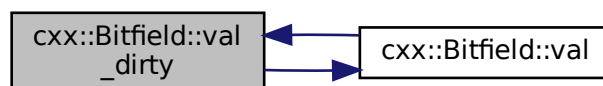
This function does not mask `val` to the right number of bits.

Definition at line [195](#) of file [bitfield](#).

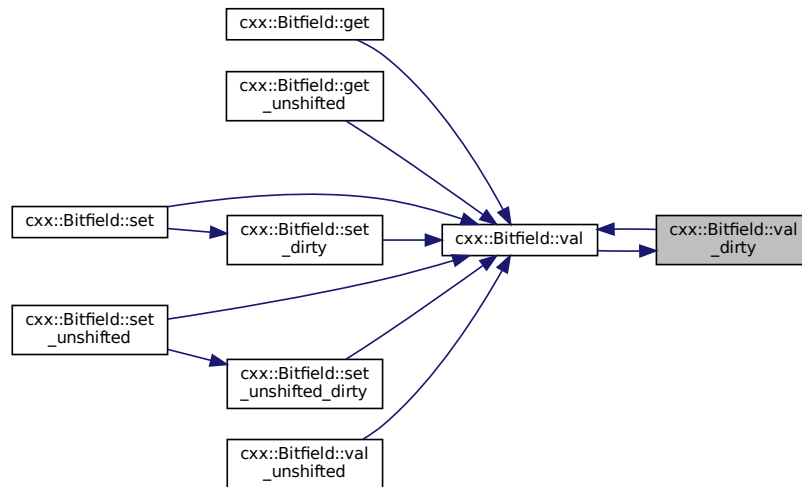
References [cxx::Bitfield< T, LSB, MSB >::Lsb](#), and [cxx::Bitfield< T, LSB, MSB >::val\(\)](#).

Referenced by [cxx::Bitfield< T, LSB, MSB >::val\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.9.4.9 `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 214 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:



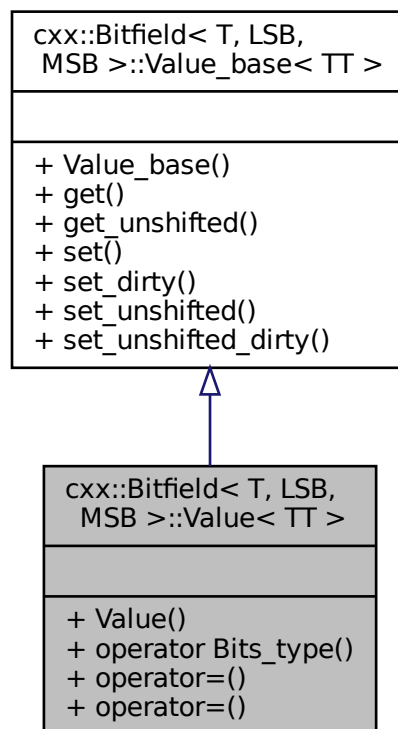
The documentation for this class was generated from the following file:

- l4/cxx/bitfield

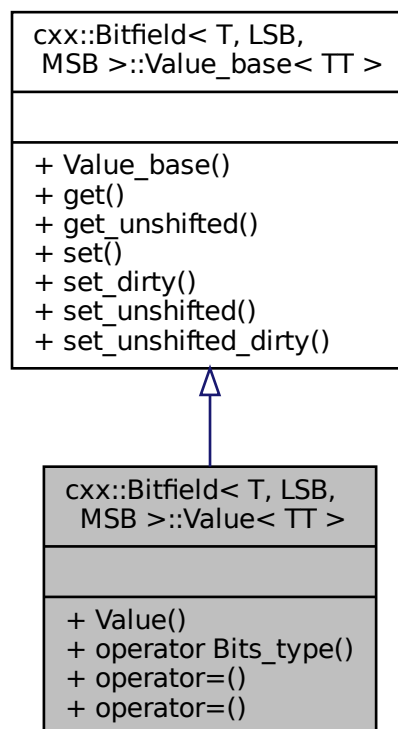
15.10 `cxx::Bitfield< T, LSB, MSB >::Value< TT >` Class Template Reference

Internal helper type.

Inheritance diagram for `cxx::Bitfield< T, LSB, MSB >::Value< TT >`:



Collaboration diagram for `cxx::Bitfield< T, LSB, MSB >::Value< TT >`:



15.10.1 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 236 of file [bitfield](#).

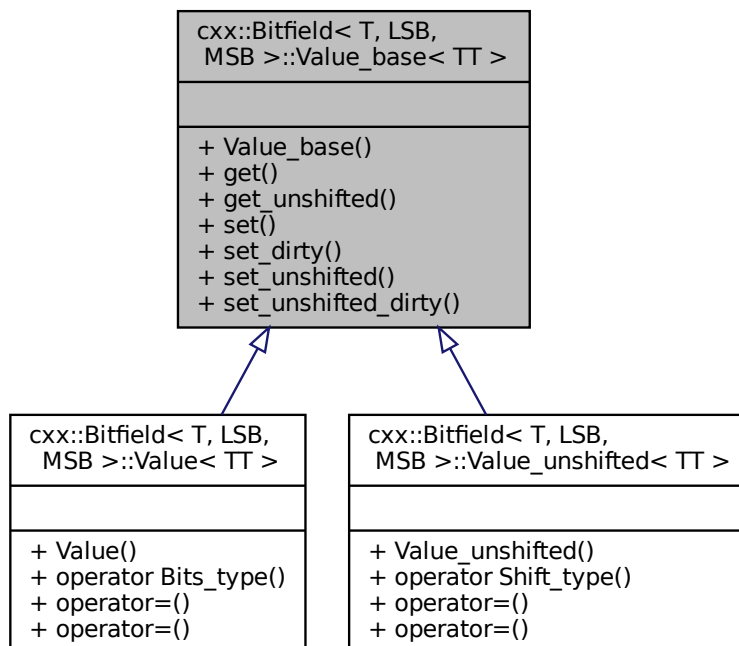
The documentation for this class was generated from the following file:

- `I4/cxx/bitfield`

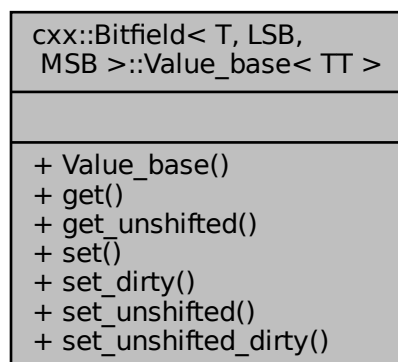
15.11 `cxx::Bitfield< T, LSB, MSB >::Value_base< TT >` Class Template Reference

Internal helper type.

Inheritance diagram for `cxx::Bitfield< T, LSB, MSB >::Value_base< TT >`:



Collaboration diagram for `cxx::Bitfield< T, LSB, MSB >::Value_base< TT >`:



15.11.1 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 218 of file [bitfield](#).

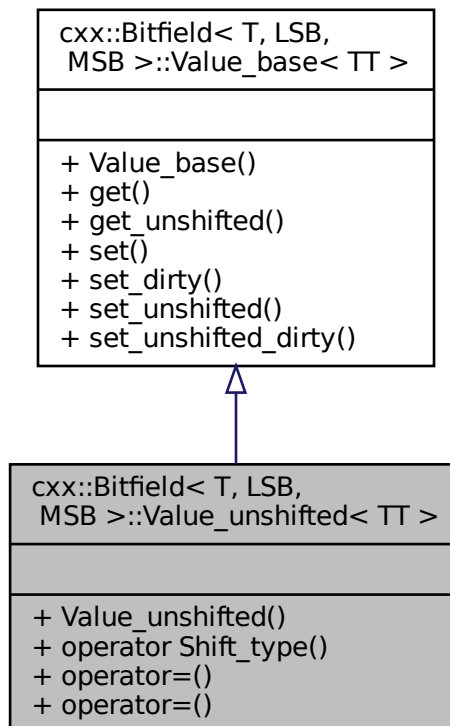
The documentation for this class was generated from the following file:

- `I4/cxx/bitfield`

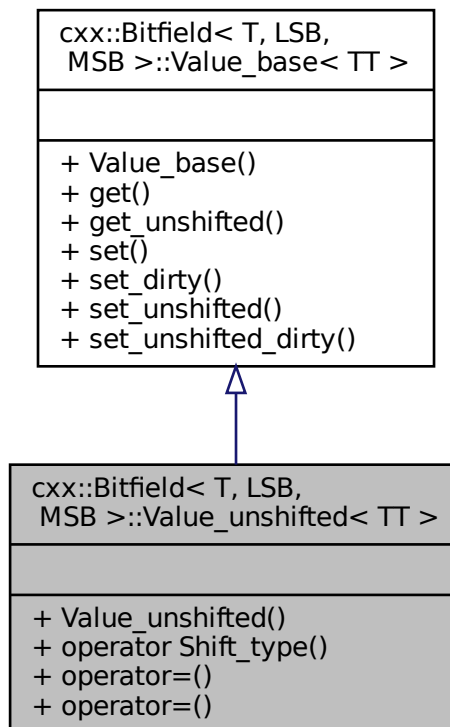
15.12 `cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >` Class Template Reference

Internal helper type.

Inheritance diagram for `cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >`:



Collaboration diagram for `cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >`:



15.12.1 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 247 of file [bitfield](#).

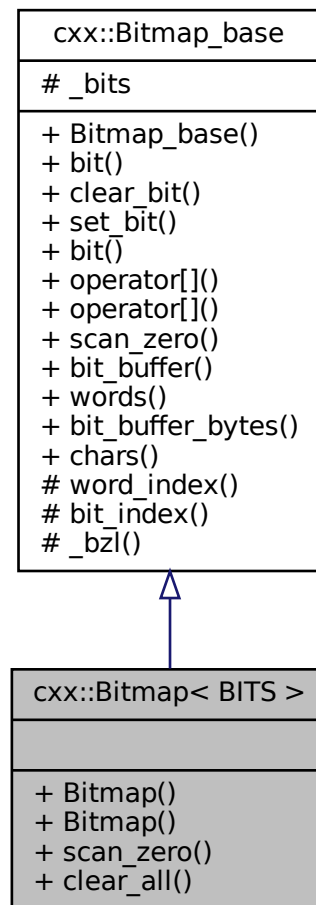
The documentation for this class was generated from the following file:

- `I4/cxx/bitfield`

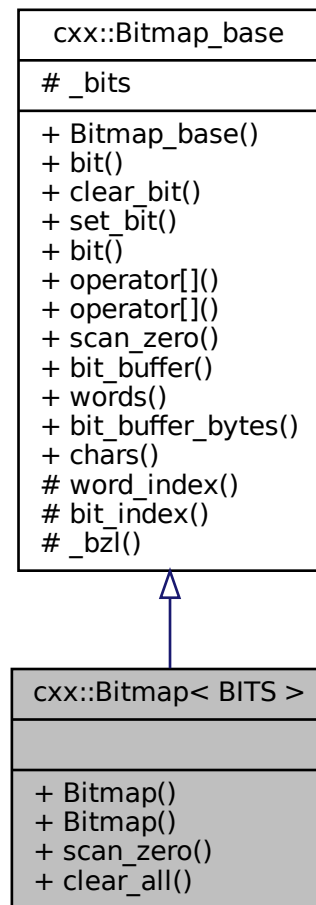
15.13 `cxx::Bitmap< BITS >` Class Template Reference

A static bit map.

Inheritance diagram for cxx::Bitmap< BITS >:



Collaboration diagram for `cxx::Bitmap< BITS >`:



Public Member Functions

- `Bitmap () throw ()`
Create a bitmap with BITS bits.
- `long scan_zero (long start_bit=0) const throw ()`
Scan for the first zero bit.

Additional Inherited Members

15.13.1 Detailed Description

```
template<int BITS>
class cxx::Bitmap< BITS >
```

A static bit map.

Parameters

<i>BITS</i>	the number of bits that shall be in the bitmap.
-------------	---

Definition at line 180 of file [bitmap](#).

15.13.2 Member Function Documentation

15.13.2.1 `scan_zero()`

```
template<int BITS>
long cxx::Bitmap< BITS >::scan_zero (
    long start_bit = 0 ) const throw ( )    [inline]
```

Scan for the first zero bit.

Parameters

<i>start_bit</i>	Hint at the number of the first bit to look at. Zero bits below <i>start_bit</i> may or may not be taken into account by the implementation.
------------------	--

Return values

<code>>=</code>	0 Number of first zero bit found.
<code>-1</code>	All bits at <i>start_bit</i> or higher are set.

Compared to [Bitmap_base::scan_zero\(\)](#), the upper bound is set to BITS.

Definition at line 285 of file [bitmap](#).

References [cxx::Bitmap_base::scan_zero\(\)](#).

Here is the call graph for this function:



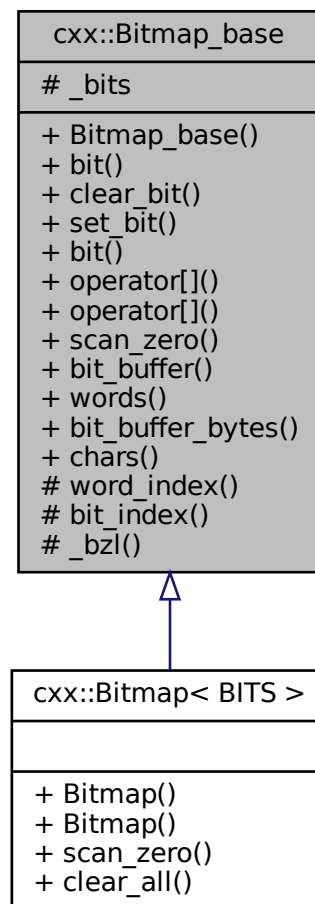
The documentation for this class was generated from the following file:

- `I4/cxx/bitmap`

15.14 cxx::Bitmap_base Class Reference

Basic bitmap abstraction.

Inheritance diagram for cxx::Bitmap_base:



Collaboration diagram for cxx::Bitmap_base:

cxx::Bitmap_base
_bits
+ Bitmap_base() + bit() + clear_bit() + set_bit() + bit() + operator[]() + operator[]() + scan_zero() + bit_buffer() + words() + bit_buffer_bytes() + chars() # word_index() # bit_index() # _bzl()

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) throw ()
Set the value of bit bit to on.
- void [clear_bit](#) (long bit) throw ()
Clear bit bit.
- void [set_bit](#) (long bit) throw ()
Set bit bit.
- [word_type](#) [bit](#) (long bit) const throw ()
Get the truth value of a bit.
- [word_type](#) [operator\[\]](#) (long bit) const throw ()
Get the bit at index bit.
- [Bit](#) [operator\[\]](#) (long bit) throw ()
Get the lvalue for the bit at index bit.
- long [scan_zero](#) (long max_bit, long start_bit=0) const throw ()
Scan for the first zero bit.

Static Public Member Functions

- static long [words](#) (long bits) throw ()
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.

15.14.1 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](#).

15.14.2 Member Enumeration Documentation

15.14.2.1 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](#).

15.14.3 Member Function Documentation

15.14.3.1 bit() [1/2]

```
Bitmap_base::word_type cxx::Bitmap_base::bit (  
    long bit ) const throw ( )    [inline]
```

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](#).

15.14.3.2 bit() [2/2]

```
void cxx::Bitmap_base::bit (  
    long bit,  
    bool on ) throw ( )    [inline]
```

Set the value of bit *bit* to *on*.

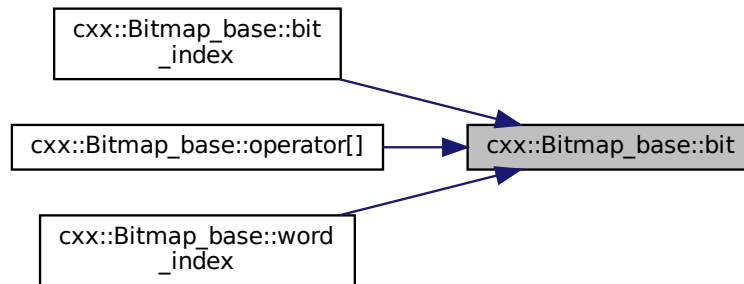
Parameters

<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 [bit_index\(\)](#), [operator\[\]\(\)](#), and [word_index\(\)](#).

Here is the caller graph for this function:



15.14.3.3 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.
------------	------------------------------

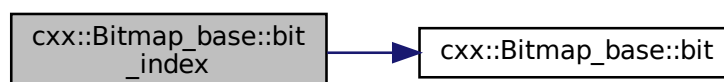
Returns

the bit index within `word_type` (`bit % W_bits`).

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

References [bit\(\)](#), and [W_bits](#).

Here is the call graph for this function:



15.14.3.4 clear_bit()

```
void cxx::Bitmap_base::clear_bit (
    long bit ) throw ( )    [inline]
```

Clear bit *bit*.

Parameters

<i>bit</i>	the number of the bit to clear.
------------	---------------------------------

Definition at line 220 of file [bitmap](#).

15.14.3.5 operator[]() [1/2]

```
Bit cxx::Bitmap_base::operator[] (
    long bit ) throw ( )    [inline]
```

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:

**15.14.3.6 operator[]() [2/2]**

```
word_type cxx::Bitmap_base::operator[] (
    long bit ) const throw ( )    [inline]
```

Get the bit at index *bit*.

Parameters

<i>bit</i>	the number of the bit to read.
------------	--------------------------------

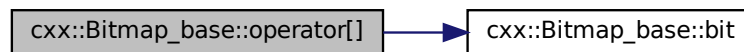
Returns

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:

**15.14.3.7 scan_zero()**

```

long cxx::Bitmap_base::scan_zero (
    long max_bit,
    long start_bit = 0 ) const throw ( )    [inline]
  
```

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.

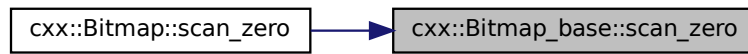
Return values

<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:



15.14.3.8 set_bit()

```
void cxx::Bitmap_base::set_bit (
    long bit ) throw ( )    [inline]
```

Set bit *bit*.

Parameters

<i>bit</i>	the number of the bit to set,
------------	-------------------------------

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

15.14.3.9 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.
------------	-----------------------------------

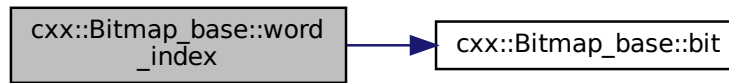
Returns

the index in [Bitmap_base::_bits](#) for the given bit (bit / W_bits).

Definition at line 54 of file [bitmap](#).

References [bit\(\)](#), and [W_bits](#).

Here is the call graph for this function:



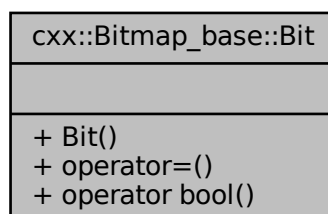
The documentation for this class was generated from the following file:

- `I4/cxx/bitmap`

15.15 cxx::Bitmap_base::Bit Class Reference

A writeable bit in a bitmap.

Collaboration diagram for `cxx::Bitmap_base::Bit`:



15.15.1 Detailed Description

A writeable bit in a bitmap.

Definition at line [66](#) of file [bitmap](#).

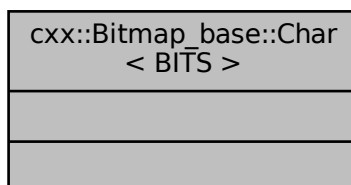
The documentation for this class was generated from the following file:

- `I4/cxx/bitmap`

15.16 cxx::Bitmap_base::Char< BITS > Class Template Reference

Helper abstraction for a byte contained in the bitmap.

Collaboration diagram for cxx::Bitmap_base::Char< BITS >:



15.16.1 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](#).

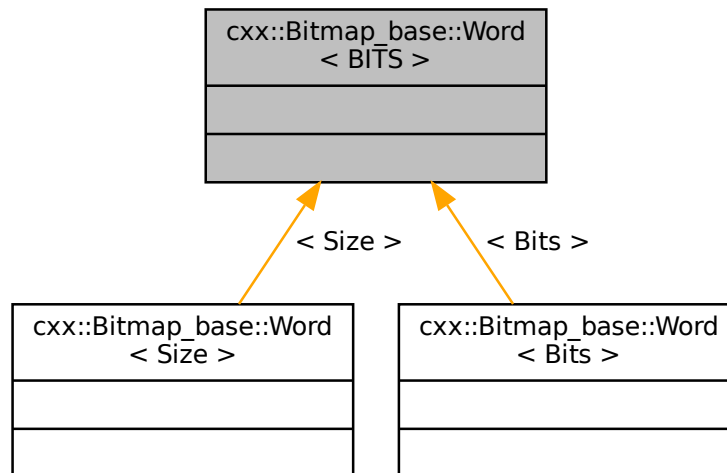
The documentation for this class was generated from the following file:

- `I4/cxx/bitmap`

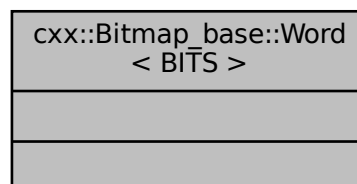
15.17 cxx::Bitmap_base::Word< BITS > Class Template Reference

Helper abstraction for a word contained in the bitmap.

Inheritance diagram for `cxx::Bitmap_base::Word< BITS >`:



Collaboration diagram for `cxx::Bitmap_base::Word< BITS >`:



15.17.1 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](#).

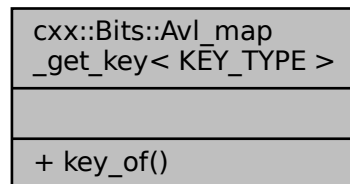
The documentation for this class was generated from the following file:

- `I4/cxx/bitmap`

15.18 cxx::Bits::Avl_map_get_key< KEY_TYPE > Struct Template Reference

Key-getter for [Avl_map](#).

Collaboration diagram for cxx::Bits::Avl_map_get_key< KEY_TYPE >:



15.18.1 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](#).

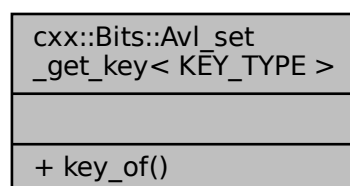
The documentation for this struct was generated from the following file:

- [l4/cxx/avl_map](#)

15.19 cxx::Bits::Avl_set_get_key< KEY_TYPE > Struct Template Reference

Internal, key-getter for [Avl_set](#) nodes.

Collaboration diagram for cxx::Bits::Avl_set_get_key< KEY_TYPE >:



15.19.1 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](#).

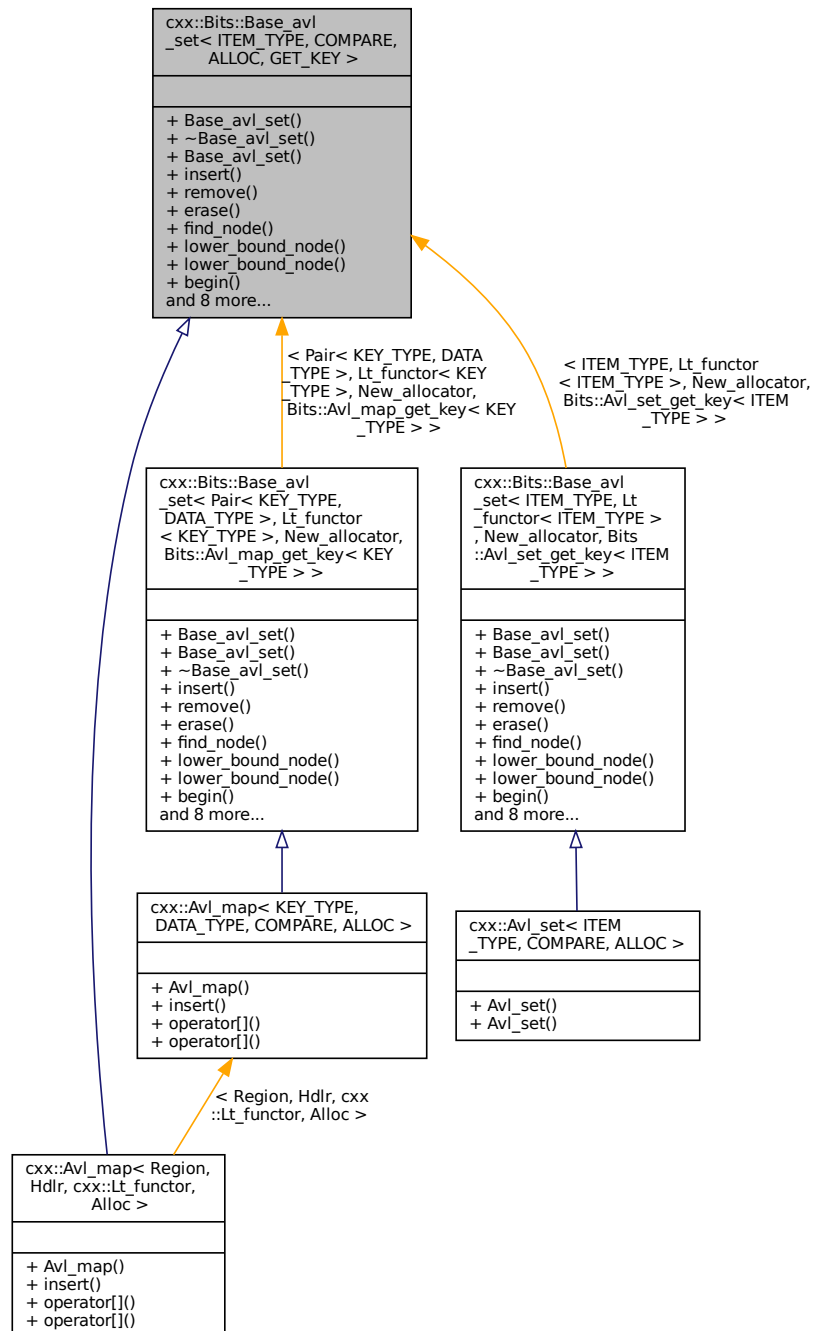
The documentation for this struct was generated from the following file:

- [I4/cxx/avl_set](#)

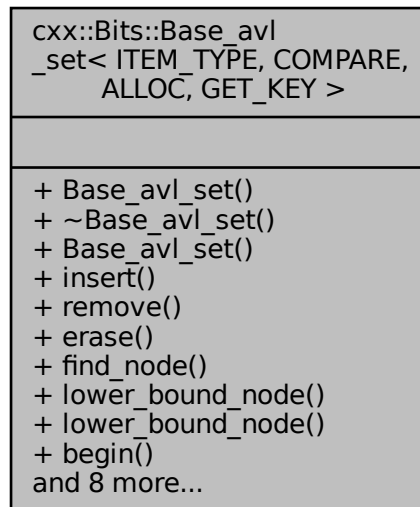
15.20 **cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY > Class Template Reference**

Internal: AVL set with internally managed nodes.

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.

15.20.1 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 <code>Key_type</code> and sort-key for an item (of <code>ITEM_TYPE</code>)).

Definition at line 131 of file [avl_set](#).

15.20.2 Member Enumeration Documentation

15.20.2.1 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](#).

15.20.3 Constructor & Destructor Documentation

15.20.3.1 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](#).

15.20.3.2 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

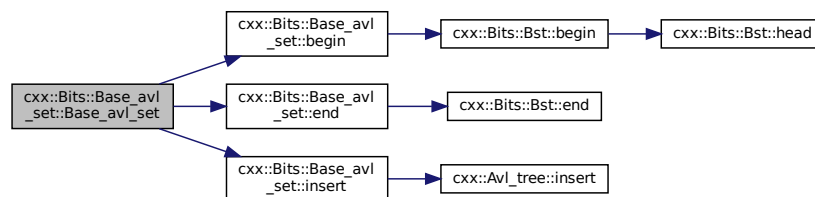
<i>o</i>	The set to copy.
----------	------------------

Creates a deep copy of the set with all its items.

Definition at line 398 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:



15.20.4 Member Function Documentation

15.20.4.1 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 359 of file [avl_set](#).

References [cxx::Bits::Bst< Node, Get_key, Compare >::begin\(\)](#).

Here is the call graph for this function:

**15.20.4.2 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 348 of file [avl_set](#).

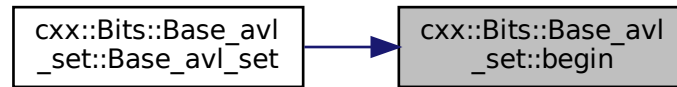
References [cxx::Bits::Bst< Node, Get_key, Compare >::begin\(\)](#).

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:



15.20.4.3 `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 364 of file `avl_set`.

References `cxx::Bits::Bst< Node, Get_key, Compare >::end()`.

Here is the call graph for this function:



15.20.4.4 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

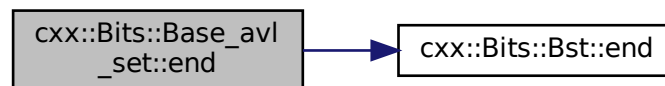
The end marker for the constant forward iterator.

Definition at line 353 of file [avl_set](#).

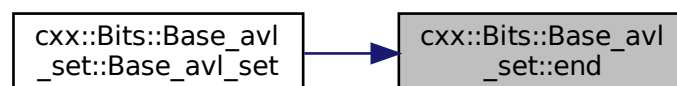
References [cxx::Bits::Bst< Node, Get_key, Compare >::end\(\)](#).

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:



15.20.4.5 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.

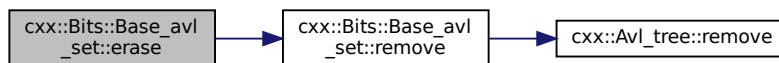
Parameters

<i>item</i>	The key of the item to remove.
-------------	--------------------------------

Definition at line 316 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:



15.20.4.6 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.
-------------	--------------------------

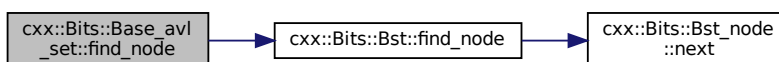
Returns

A smart pointer to the element found. If no element was found the smart pointer will be invalid.

Definition at line 327 of file [avl_set](#).

References [cxx::Bits::Bst< Node, Get_key, Compare >::find_node\(\)](#).

Here is the call graph for this function:



15.20.4.7 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.

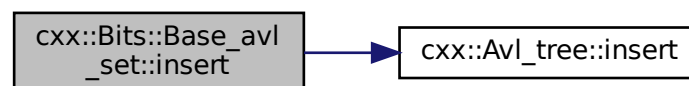
Insert a new item into the set, each item can only be once in the set.

Definition at line 408 of file [avl_set](#).

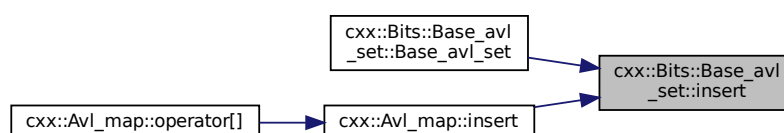
References [cxx::Pair< First, Second >::first](#), [cxx::Avl_tree< Node, Get_key, Compare >::insert\(\)](#), and [cxx::Pair< First, Second >::second](#).

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:



15.20.4.8 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

<code>key</code>	Minimum key to look for.
------------------	--------------------------

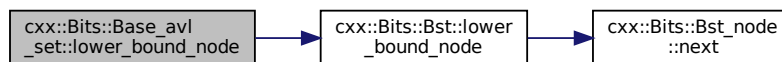
Returns

Smart pointer to the first node greater or equal to `key`. Will be invalid if no such element was found.

Definition at line 338 of file `avl_set`.

References `cxx::Bits::Bst< Node, Get_key, Compare >::lower_bound_node()`.

Here is the call graph for this function:



15.20.4.9 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 381 of file `avl_set`.

References `cxx::Bits::Bst< Node, Get_key, Compare >::rbegin()`.

Here is the call graph for this function:



15.20.4.10 `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.

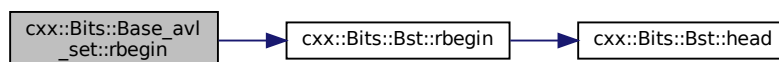
Returns

The constant backward iterator for the last element in the set.

Definition at line 370 of file [avl_set](#).

References [cxx::Bits::Bst< Node, Get_key, Compare >::rbegin\(\)](#).

Here is the call graph for this function:



15.20.4.11 `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

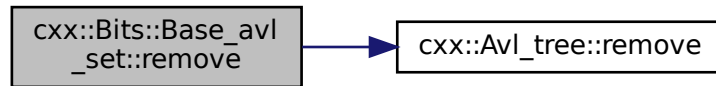
<i>0</i>	Success
<i>-E_noent</i>	Item does not exist
<i>-E_inval</i>	Internal error

Definition at line 296 of file [avl_set](#).

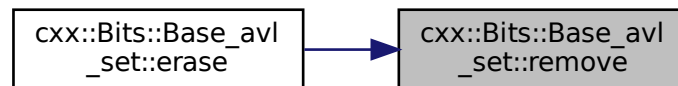
References [cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::E_inval](#), [cxx::Bits::Base_avl_set< ITEM_TYP](#) and [cxx::Avl_tree< Node, Get_key, Compare >::remove\(\)](#).

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:



15.20.4.12 `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.

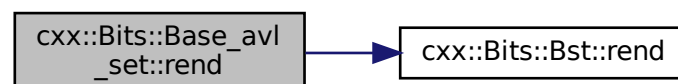
Returns

The end marker for mutable backward iterator.

Definition at line 386 of file `avl_set`.

References `cxx::Bits::Bst< Node, Get_key, Compare >::rend()`.

Here is the call graph for this function:



15.20.4.13 `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.

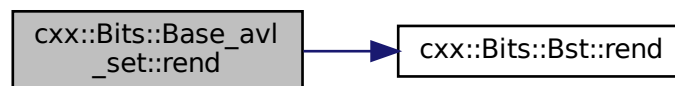
Returns

The end marker for the constant backward iterator.

Definition at line 375 of file [avl_set](#).

References [cxx::Bits::Bst< Node, Get_key, Compare >::rend\(\)](#).

Here is the call graph for this function:



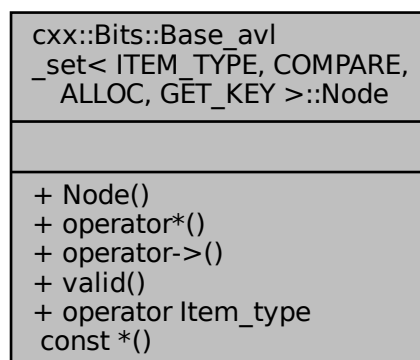
The documentation for this class was generated from the following file:

- [l4/cxx/avl_set](#)

15.21 `cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node` Class Reference

A smart pointer to a tree item.

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.

15.21.1 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](#).

15.21.2 Member Function Documentation

15.21.2.1 `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](#).

15.21.2.2 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](#).

15.21.2.3 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:

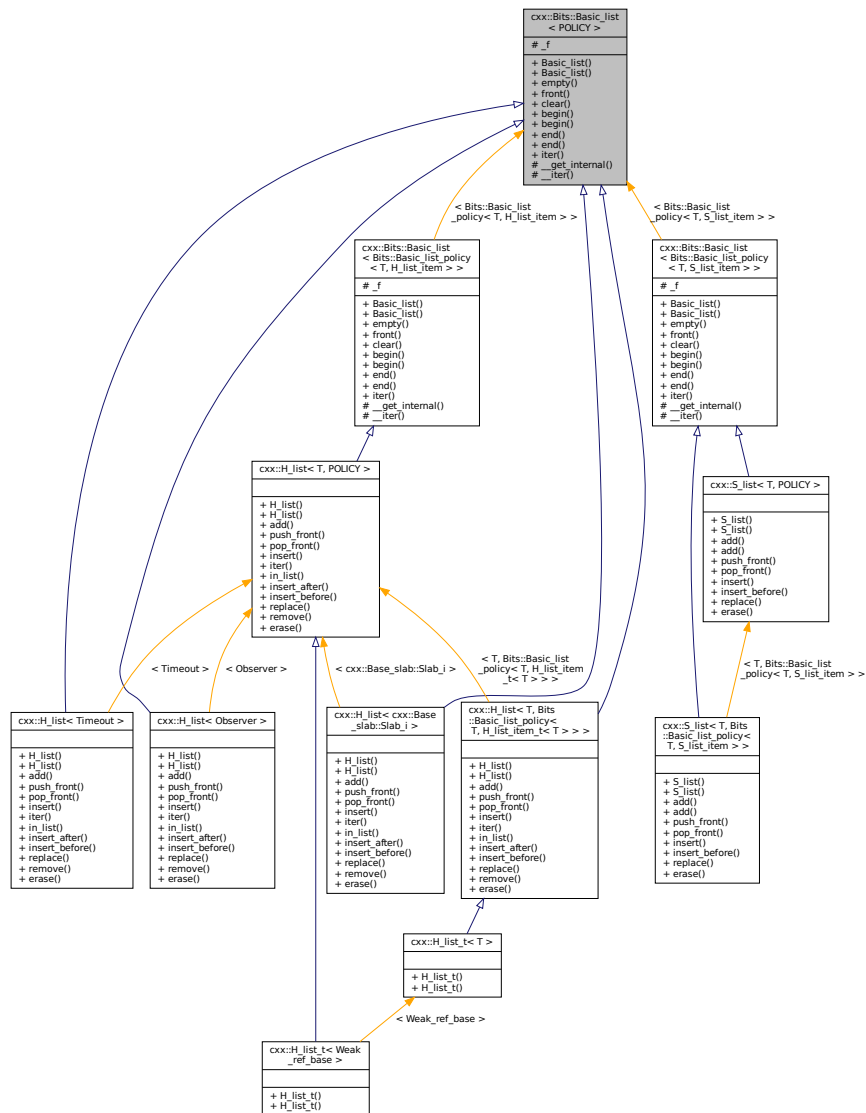
- [l4/cxx/avl_set](#)

15.22 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 >`:

<code>cxx::Bits::Basic_list < POLICY ></code>
<code># _f</code>
<code>+ Basic_list()</code> <code>+ Basic_list()</code> <code>+ empty()</code> <code>+ front()</code> <code>+ clear()</code> <code>+ begin()</code> <code>+ begin()</code> <code>+ end()</code> <code>+ end()</code> <code>+ iter()</code> <code># __get_internal()</code> <code># __iter()</code>

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.

15.22.1 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](#).

15.22.2 Member Function Documentation

15.22.2.1 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 135 of file [list_basics.h](#).

References [cxx::Bits::Basic_list< POLICY >::_f](#).

15.22.2.2 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

<i>c</i>	Element where the iterator should start.
----------	--

Precondition

The element *c* must already be in a list.

Definition at line 148 of file [list_basics.h](#).

The documentation for this class was generated from the following file:

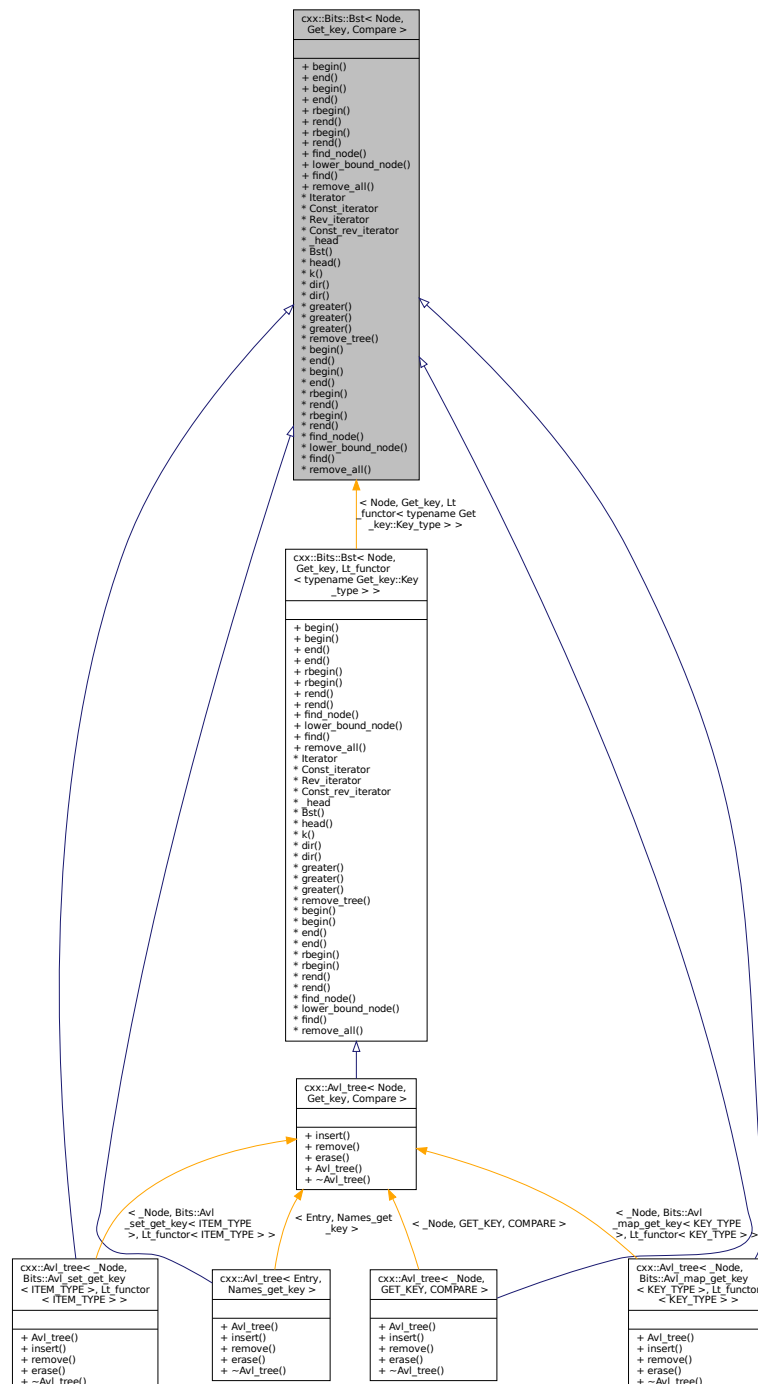
- [I4/cxx/bits/list_basics.h](#)

15.23 cxx::Bits::Bst< Node, Get_key, Compare > Class Template Reference

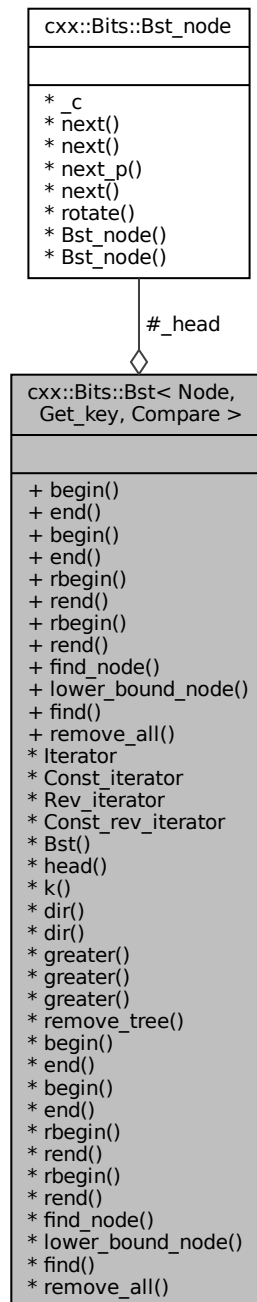
Basic binary search tree (BST).

```
#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.

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.
- template<typename FUNC >
static void [remove_tree](#) ([Bst_node](#) *[head](#), FUNC &&[callback](#))
Remove all elements in the subtree of head.

15.23.1 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](#).

15.23.2 Member Function Documentation

15.23.2.1 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:

**15.23.2.2 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](#).

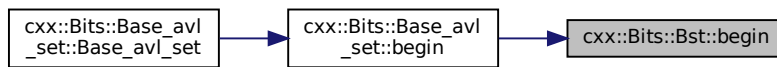
References [cxx::Bits::Bst< Node, Get_key, Compare >::head\(\)](#).

Referenced by [cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::begin\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.23.2.3 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.

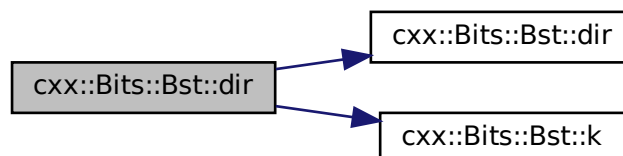
Return values

<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:



15.23.2.4 `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:

**15.23.2.5** `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`.

15.23.2.6 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.

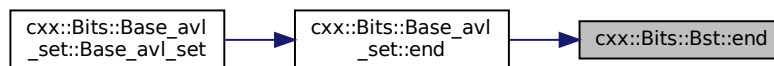
Returns

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\(\)](#).

Here is the caller graph for this function:

**15.23.2.7 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.
------------	---

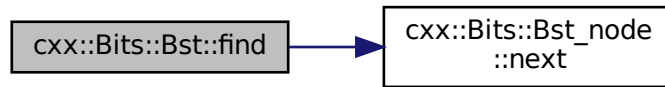
Returns

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:



15.23.2.8 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

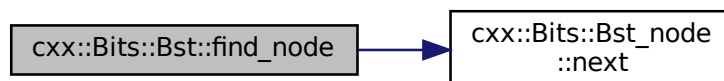
A pointer to the node with the given *key*, or NULL if *key* was not found.

Definition at line 276 of file [bst.h](#).

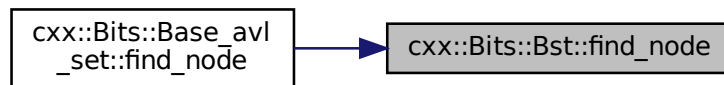
References [cxx::Bits::Bst_node::next\(\)](#).

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:



15.23.2.9 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

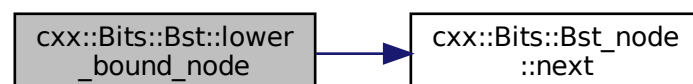
A pointer to the found node, or `NULL` if no node was found.

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

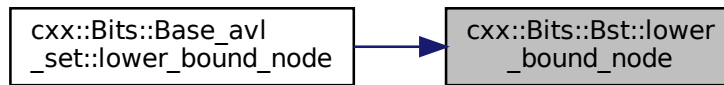
References [cxx::Bits::Bst_node::next\(\)](#).

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:



15.23.2.10 `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:



15.23.2.11 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\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.23.2.12 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

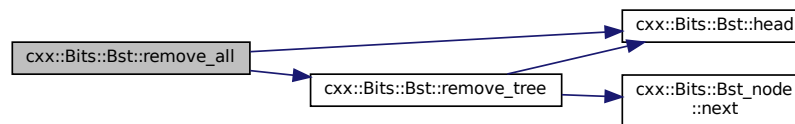
<i>callback</i>	Optional function to be called on each removed element.
-----------------	---

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\(\)](#).

Here is the call graph for this function:



15.23.2.13 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

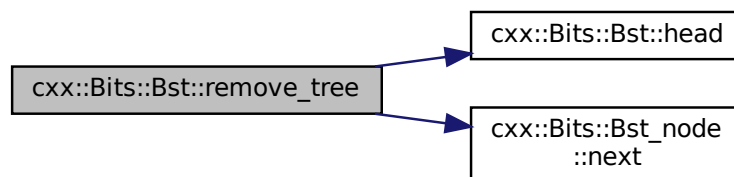
<i>head</i>	Head of the the subtree to remove
<i>callback</i>	Optional function called on each removed element.

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\(\)](#), and [cxx::Bits::Direction::R](#).

Referenced by [cxx::Bits::Bst< Node, Get_key, Compare >::remove_all\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.23.2.14 `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](#).

15.23.2.15 `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.

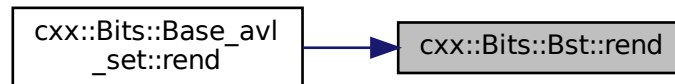
Returns

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\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

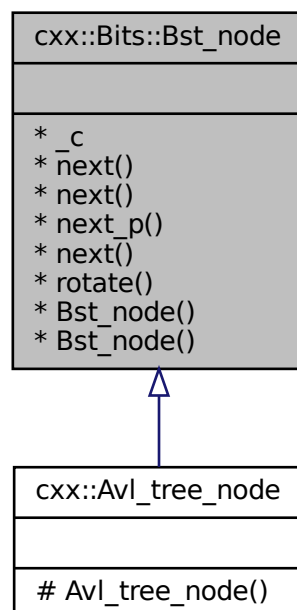
- [l4/cxx/bits/bst.h](#)

15.24 cxx::Bits::Bst_node Class Reference

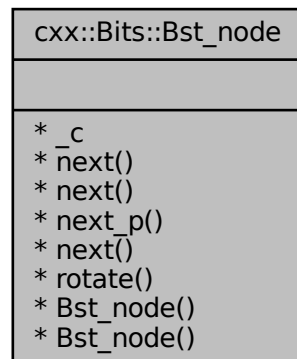
Basic type of a node in a binary search tree (BST).

```
#include <bst_base.h>
```

Inheritance diagram for cxx::Bits::Bst_node:



Collaboration diagram for cxx::Bits::Bst_node:



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.
- [Bst_node](#) ()
Create uninitialized node.
- [Bst_node](#) (bool)
Create initialized node.

15.24.1 Detailed Description

Basic type of a node in a binary search tree (BST).

Definition at line 77 of file [bst_base.h](#).

The documentation for this class was generated from the following file:

- I4/cxx/bits/[bst_base.h](#)

15.25 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
+ d
+ 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
- bool [operator!=](#) ([Direction_e](#) o) const
- bool [operator==](#) ([Direction](#) o) const
- bool [operator!=](#) ([Direction](#) o) const

15.25.1 Detailed Description

The direction to go in a binary search tree.

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

15.25.2 Member Enumeration Documentation

15.25.2.1 `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](#).

15.25.3 Member Function Documentation

15.25.3.1 `operator"!()`

```
Direction cxx::Bits::Direction::operator! ( ) const [inline]
```

Negate the direction.

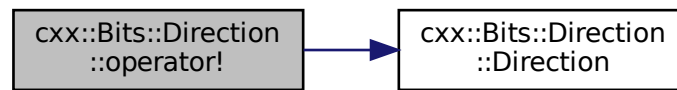
Note

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:

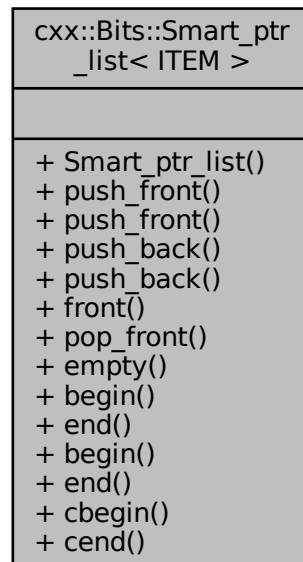
- [l4/cxx/bits/bst_base.h](#)

15.26 cxx::Bits::Smart_ptr_list< ITEM > Class Template Reference

[List](#) of smart-pointer-managed objects.

```
#include <smart_ptr_list.h>
```

Collaboration diagram for cxx::Bits::Smart_ptr_list< ITEM >:



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.

15.26.1 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 46 of file [smart_ptr_list.h](#).

15.26.2 Member Function Documentation

15.26.2.1 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 149 of file [smart_ptr_list.h](#).

The documentation for this class was generated from the following file:

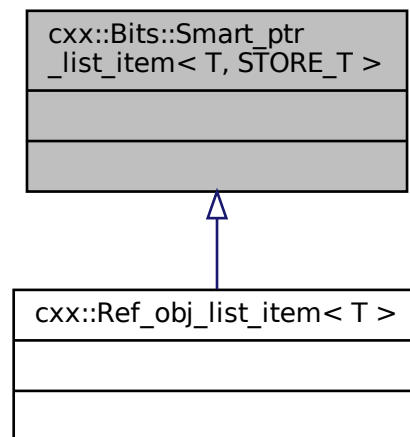
- I4/cxx/bits/smart_ptr_list.h

15.27 `cxx::Bits::Smart_ptr_list_item< T, STORE_T >` Class Template Reference

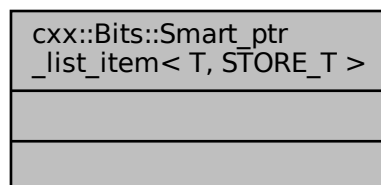
List item for an arbitrary item in a [Smart_ptr_list](#).

```
#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 >`:



15.27.1 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↔ _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 27 of file [smart_ptr_list.h](#).

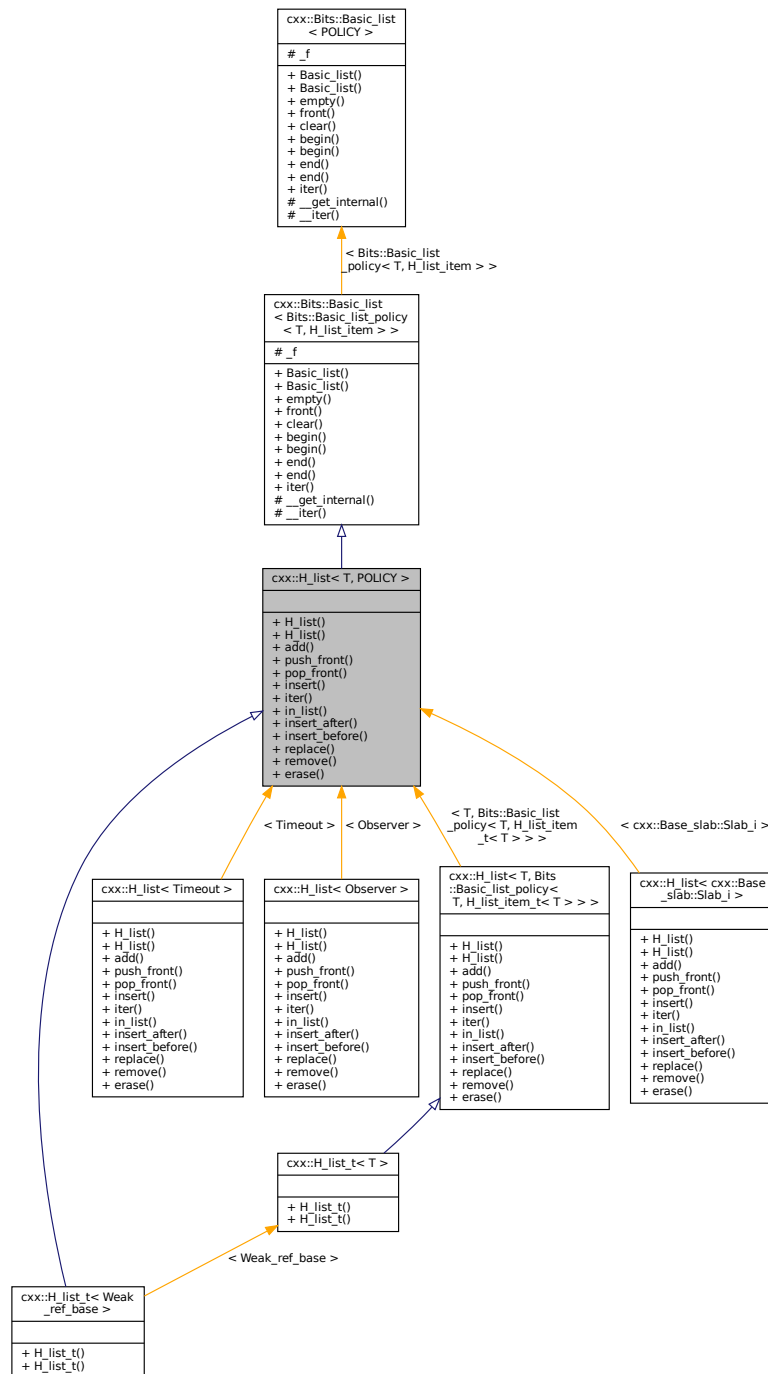
The documentation for this class was generated from the following file:

- `I4/cxx/bits/smart_ptr_list.h`

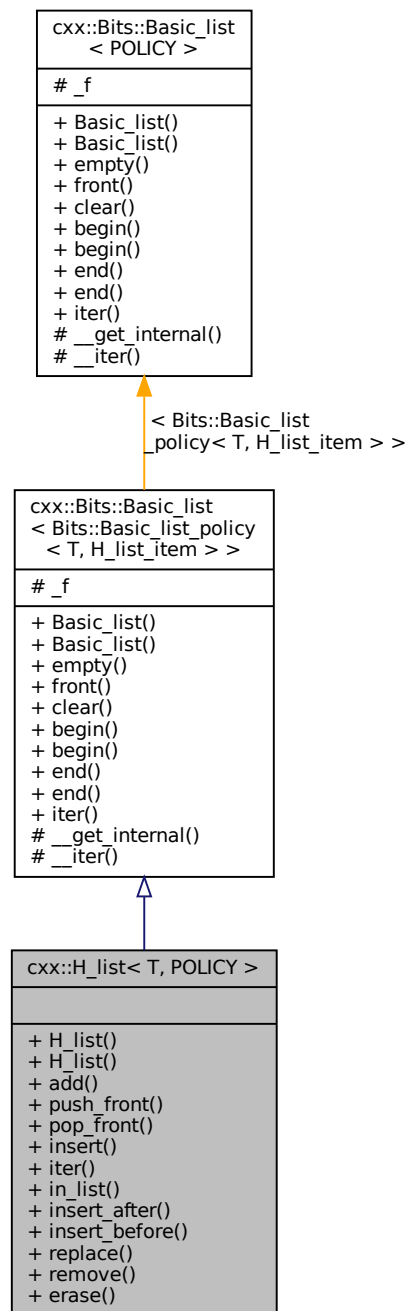
15.28 `cxx::H_list< T, POLICY >` Class Template Reference

General double-linked list of unspecified [cxx::H_list_item](#) elements.

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.

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.

Additional Inherited Members

15.28.1 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](#).

15.28.2 Member Function Documentation

15.28.2.1 [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 <code>end()</code> .
----------	--

Returns

New iterator pointing to the element after the removed one.

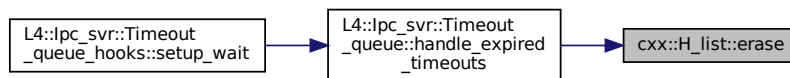
Note

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:

15.28.2.2 `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 <code>end()</code> is given, the element will be inserted at the beginning of the queue.

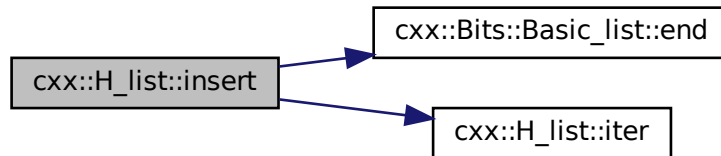
Returns

Iterator pointing to the newly inserted element.

Definition at line 144 of file `hlist`.

References [cxx::Bits::Basic_list< Bits::Basic_list_policy< T, H_list_item > >::_f](#), [cxx::Bits::Basic_list< POLICY >::end\(\)](#), and [cxx::H_list< T, POLICY >::iter\(\)](#).

Here is the call graph for this function:



15.28.2.3 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.

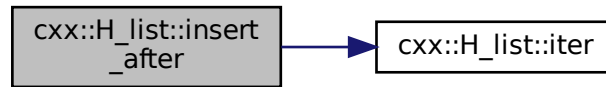
Precondition

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:



15.28.2.4 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.

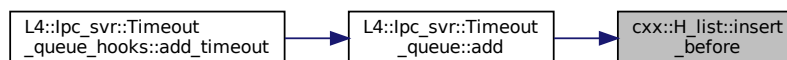
Parameters

<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 end() .

Definition at line 191 of file [hlist](#).

Referenced by [L4::lpc_svr::Timeout_queue::add\(\)](#).

Here is the caller graph for this function:



15.28.2.5 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

<i>c</i>	List element to start the iteration.
----------	--------------------------------------

Returns

A mutable forward iterator.

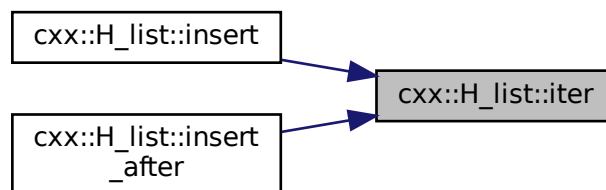
Precondition

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:

**15.28.2.6 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.

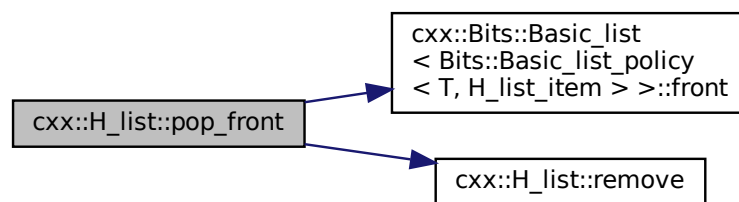
Precondition

The list must not be empty or the behaviour will be undefined.

Definition at line 127 of file [hlist](#).

References [cxx::Bits::Basic_list< Bits::Basic_list_policy< T, H_list_item > >::front\(\)](#), and [cxx::H_list< T, POLICY >::remove\(\)](#).

Here is the call graph for this function:

**15.28.2.7 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.

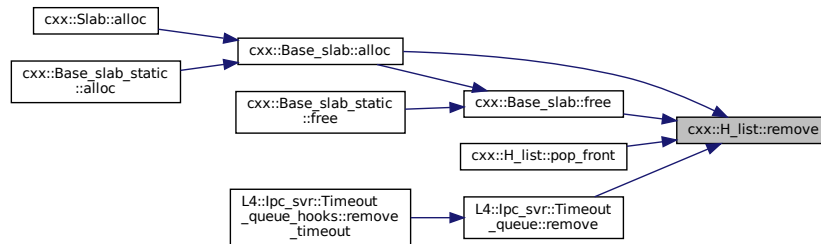
Parameters

<i>e</i>	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 >::pop_front\(\)](#), and [L4::lpc_svr::Timeout_queue::remove\(\)](#).

Here is the caller graph for this function:



15.28.2.8 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](#).

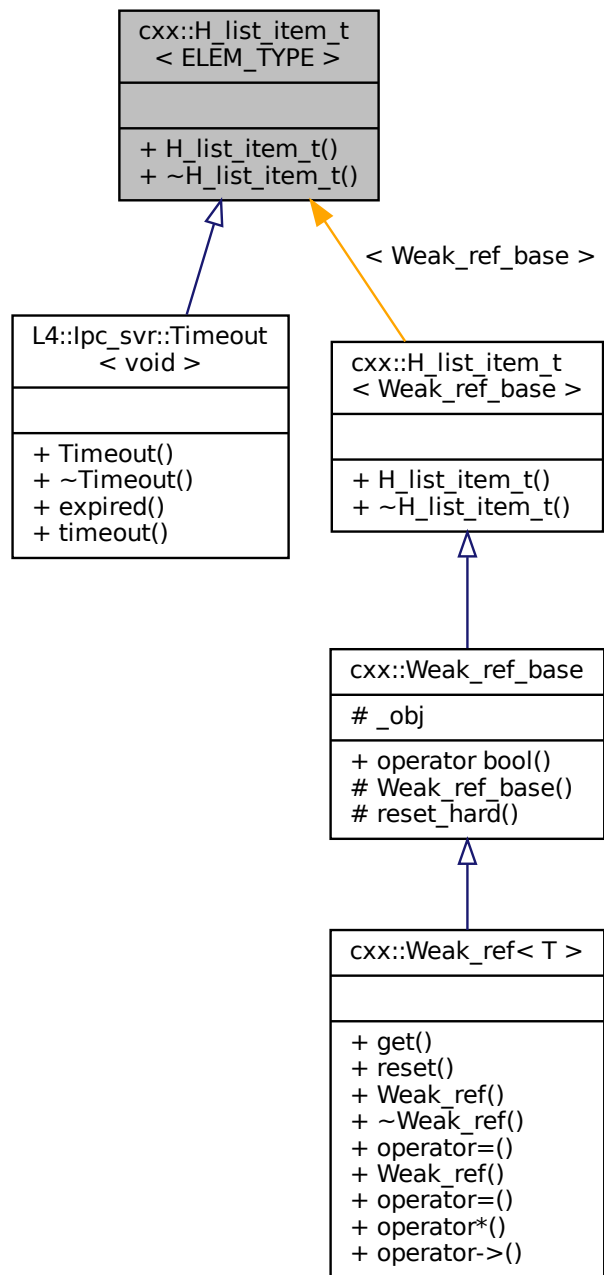
The documentation for this class was generated from the following file:

- [l4/cxx/hlist](#)

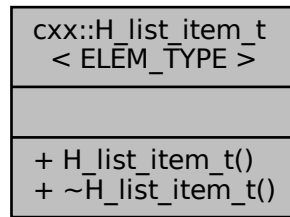
15.29 cxx::H_list_item_t< ELEM_TYPE > Class Template Reference

Basic element type for a double-linked [H_list](#).

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.

15.29.1 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

<code>ELEM_TYPE</code>	Base class of the list element.
------------------------	---------------------------------

Definition at line 33 of file [hlist](#).

15.29.2 Constructor & Destructor Documentation

15.29.2.1 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](#).

15.29.2.2 `~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](#).

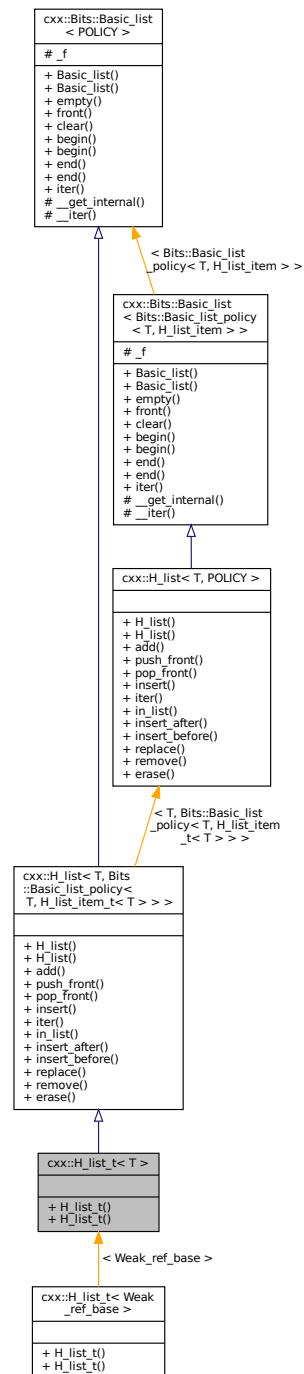
The documentation for this class was generated from the following file:

- `I4/cxx/hlist`

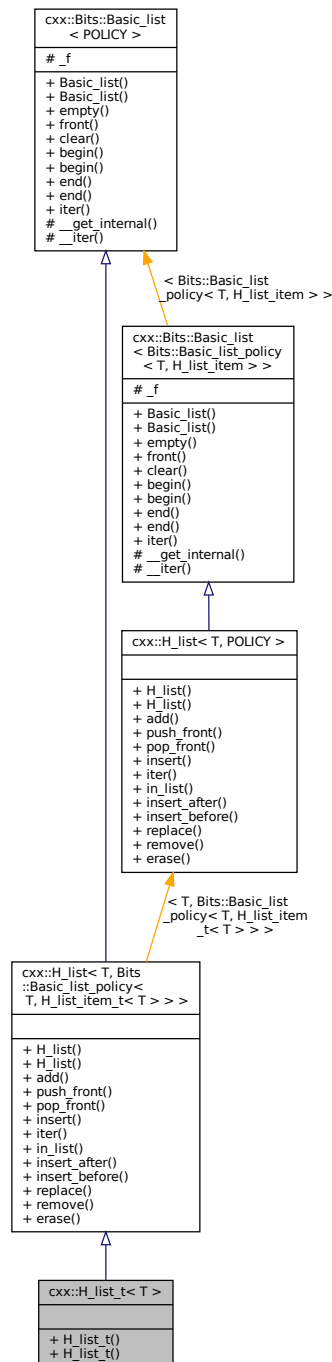
15.30 `cxx::H_list_t< T >` Struct Template Reference

Double-linked list of typed [H_list_item_t](#) elements.

Inheritance diagram for `cxx::H_list_t< T >`:



Collaboration diagram for cxx::H_list_t< T >:



Additional Inherited Members

15.30.1 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](#).

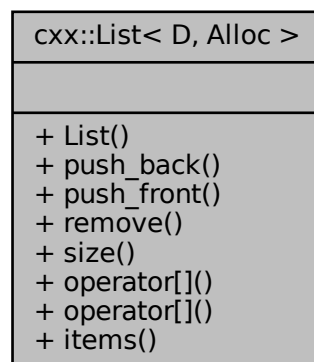
The documentation for this struct was generated from the following file:

- I4/cxx/hlist

15.31 cxx::List< D, Alloc > Class Template Reference

Doubly linked list, with internal allocation.

Collaboration diagram for cxx::List< D, Alloc >:



Data Structures

- class [Iter](#)
Iterator.

Public Member Functions

- void [push_back](#) (D const &d) throw ()
Add element at the end of the list.
- void [push_front](#) (D const &d) throw ()
Add element at the beginning of the list.
- void [remove](#) ([Iter](#) const &i) throw ()
Remove element pointed to by the iterator.
- unsigned long [size](#) () const throw ()
Get the length of the list.
- D const & [operator\[\]](#) (unsigned long idx) const throw ()
Random access.
- D & [operator\[\]](#) (unsigned long idx) throw ()
Random access.
- [Iter](#) [items](#) () throw ()
Get iterator for the list elements.

15.31.1 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](#).

15.31.2 Member Function Documentation

15.31.2.1 `operator[]()` [1/2]

```
template<typename D , template< typename A > class Alloc = New_allocator>
D& cxx::List< D, Alloc >::operator[] (
    unsigned long idx ) throw ( )    [inline]
```

Random access.

Complexity is O(n).

Definition at line 408 of file [list](#).

15.31.2.2 `operator[]()` [2/2]

```
template<typename D , template< typename A > class Alloc = New_allocator>
D const& cxx::List< D, Alloc >::operator[] (
    unsigned long idx ) const throw ( )    [inline]
```

Random access.

Complexity is O(n).

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

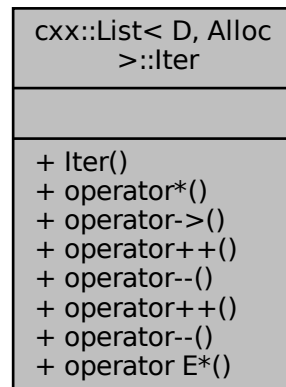
The documentation for this class was generated from the following file:

- `I4/cxx/list`

15.32 cxx::List< D, Alloc >::Iter Class Reference

Iterator.

Collaboration diagram for cxx::List< D, Alloc >::Iter:



Public Member Functions

- [operator E*](#) () const throw ()
operator for testing validity (syntactically equal to pointers)

15.32.1 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](#)

15.33 cxx::List_alloc Class Reference

Standard list-based allocator.

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.

15.33.1 Detailed Description

Standard list-based allocator.

Definition at line 31 of file [list_alloc](#).

15.33.2 Constructor & Destructor Documentation

15.33.2.1 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](#).

15.33.3 Member Function Documentation

15.33.3.1 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](#).

15.33.3.2 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

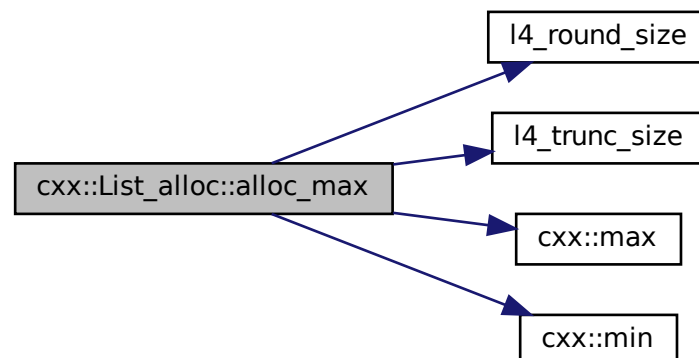
$0 < \text{min} \leq \sim 0\text{UL} - 32$.

$0 < \text{max}$.

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:

15.33.3.3 `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`.

15.33.3.4 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:

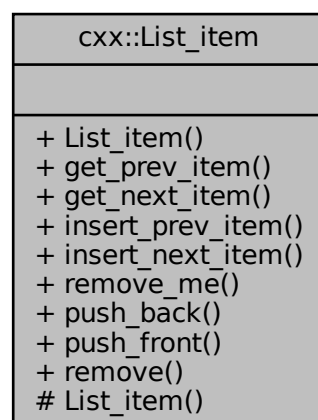
- [l4/cxx/list_alloc](#)

15.34 cxx::List_item Class Reference

Basic list item.

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 throw ()
Get previous item.
- [List_item](#) * [get_next_item](#) () const throw ()
Get next item.
- void [insert_prev_item](#) ([List_item](#) *p) throw ()
Insert item p before this item.
- void [insert_next_item](#) ([List_item](#) *p) throw ()
Insert item p after this item.
- void [remove_me](#) () throw ()
Remove this item from the list.

Static Public Member Functions

- template<typename C , typename N >
static C * [push_back](#) (C *head, N *p) throw ()
Append item to a list.
- template<typename C , typename N >
static C * [push_front](#) (C *head, N *p) throw ()
Prepend item to a list.
- template<typename C , typename N >
static C * [remove](#) (C *head, N *p) throw ()
Remove item from a list.

15.34.1 Detailed Description

Basic list item.

Basic item that can be member of a doubly linked, cyclic list.

Definition at line 37 of file [list](#).

15.34.2 Member Function Documentation

15.34.2.1 [push_back\(\)](#)

```
template<typename C , typename N >
C * cxx::List_item::push_back (
    C * head,
    N * p ) throw ( )    [inline], [static]
```

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](#).

15.34.2.2 push_front()

```
template<typename C , typename N >
C * cxx::List_item::push_front (
    C * head,
    N * p ) throw ( )    [inline], [static]
```

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](#).

15.34.2.3 remove()

```
template<typename C , typename N >
C * cxx::List_item::remove (
    C * head,
    N * p ) throw ( )    [inline], [static]
```

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](#).

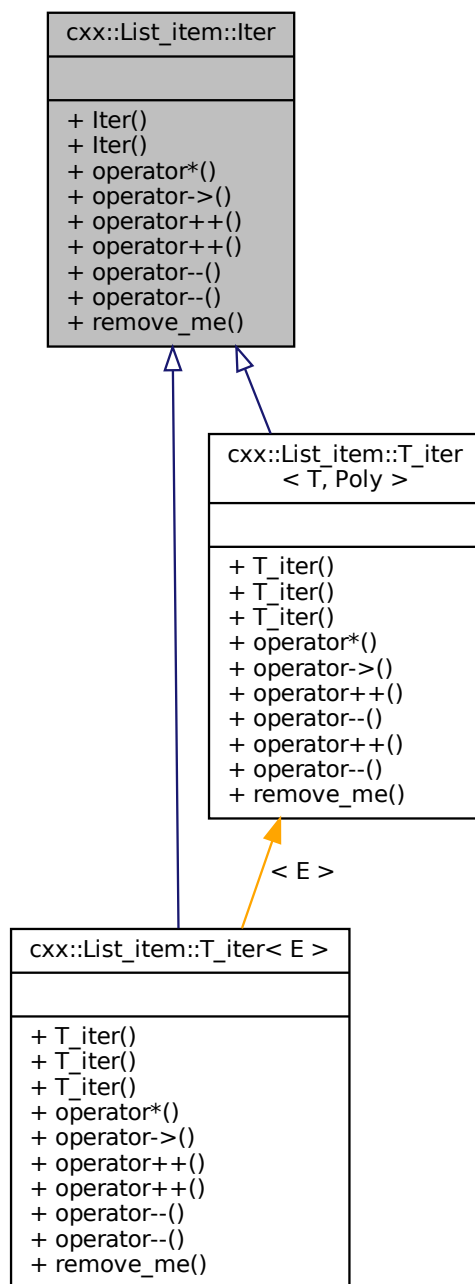
The documentation for this class was generated from the following file:

- `l4/cxx/list`

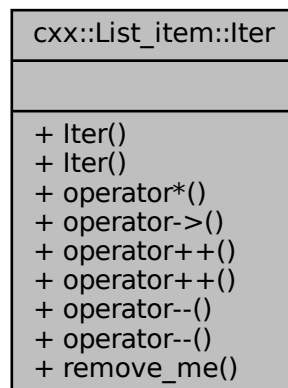
15.35 cxx::List_item::Iter Class Reference

Iterator for a list of ListItem-s.

Inheritance diagram for `cxx::List_item::Iter`:



Collaboration diagram for cxx::List_item::Iter:



Public Member Functions

- [List_item](#) * [remove_me](#) () throw ()
Remove item pointed to by iterator, and return pointer to element.

15.35.1 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](#).

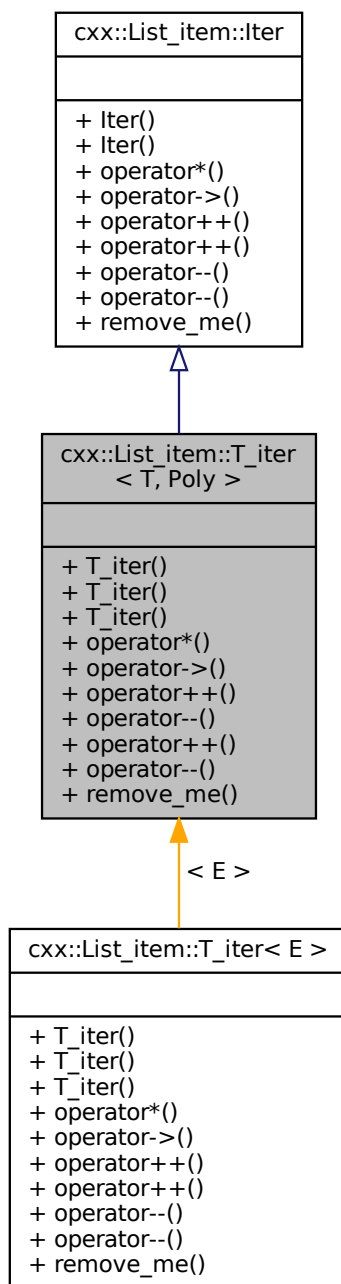
The documentation for this class was generated from the following file:

- `I4/cxx/list`

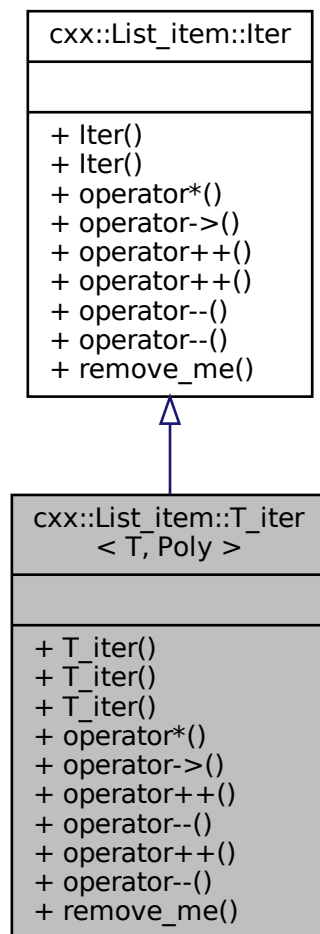
15.36 cxx::List_item::T_iter< T, Poly > Class Template Reference

Iterator for derived classes from ListItem.

Inheritance diagram for `cxx::List_item::T_iter< T, Poly >`:



Collaboration diagram for cxx::List_item::T_iter< T, Poly >:



Additional Inherited Members

15.36.1 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](#).

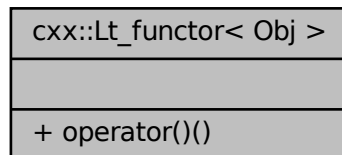
The documentation for this class was generated from the following file:

- `I4/cxx/list`

15.37 `cxx::Lt_functor< Obj >` Struct Template Reference

Generic comparator class that defaults to the less-than operator.

Collaboration diagram for `cxx::Lt_functor< Obj >`:



15.37.1 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](#).

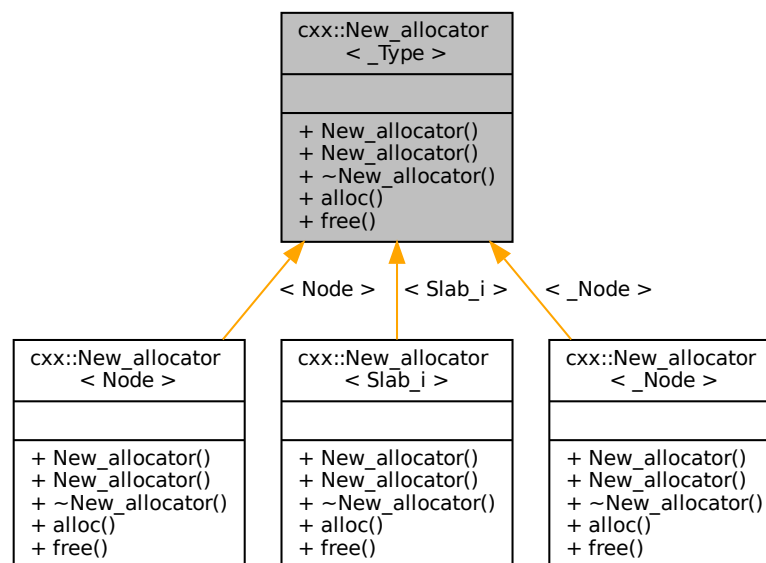
The documentation for this struct was generated from the following file:

- `I4/cxx/std_ops`

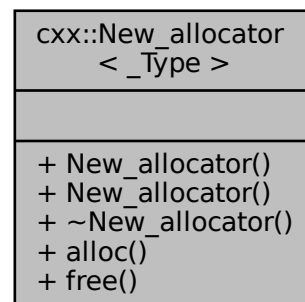
15.38 `cxx::New_allocator< _Type >` Class Template Reference

Standard allocator based on `operator new ()`.

Inheritance diagram for `cxx::New_allocator<_Type>`:



Collaboration diagram for `cxx::New_allocator<_Type>`:



15.38.1 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 60 of file [std_alloc](#).

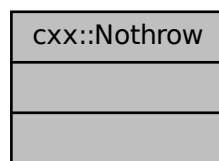
The documentation for this class was generated from the following file:

- `I4/cxx/std_alloc`

15.39 `cxx::Nothrow` Class Reference

Helper type to distinguish the `oeprator new` version that does not throw exceptions.

Collaboration diagram for `cxx::Nothrow`:



15.39.1 Detailed Description

Helper type to distinguish the `oeprator new` version that does not throw exceptions.

Definition at line 30 of file [std_alloc](#).

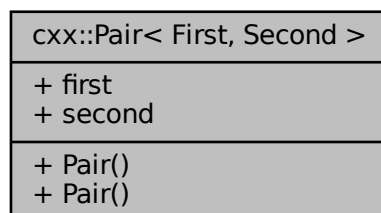
The documentation for this class was generated from the following file:

- `I4/cxx/std_alloc`

15.40 `cxx::Pair< First, Second >` Struct Template Reference

[Pair](#) of two values.

Collaboration diagram for `cxx::Pair< First, Second >`:



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.

15.40.1 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](#).

15.40.2 Constructor & Destructor Documentation

15.40.2.1 `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](#).

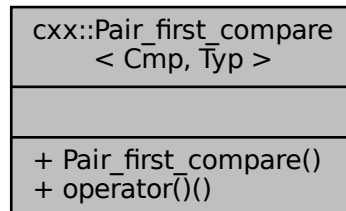
The documentation for this struct was generated from the following file:

- [l4/cxx/pair](#)

15.41 cxx::Pair_first_compare< Cmp, Typ > Class Template Reference

Comparison functor for [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.

15.41.1 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](#).

15.41.2 Constructor & Destructor Documentation

15.41.2.1 `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](#).

15.41.3 Member Function Documentation

15.41.3.1 `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](#).

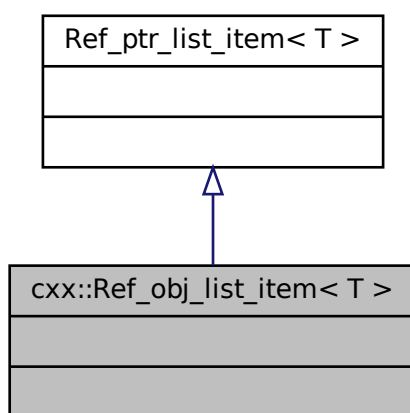
The documentation for this class was generated from the following file:

- [l4/cxx/pair](#)

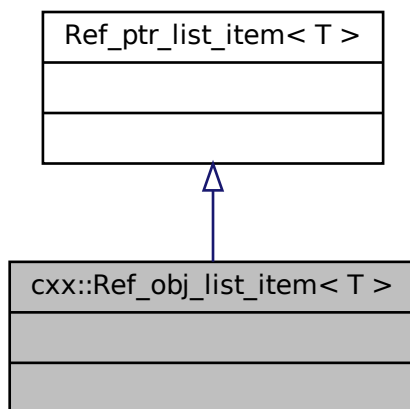
15.42 cxx::Ref_obj_list_item< T > Struct Template Reference

Item for list linked via [cxx::Ref_ptr](#) with default reference counting.

Inheritance diagram for cxx::Ref_obj_list_item< T >:



Collaboration diagram for cxx::Ref_obj_list_item< T >:



15.42.1 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 26 of file [ref_ptr_list](#).

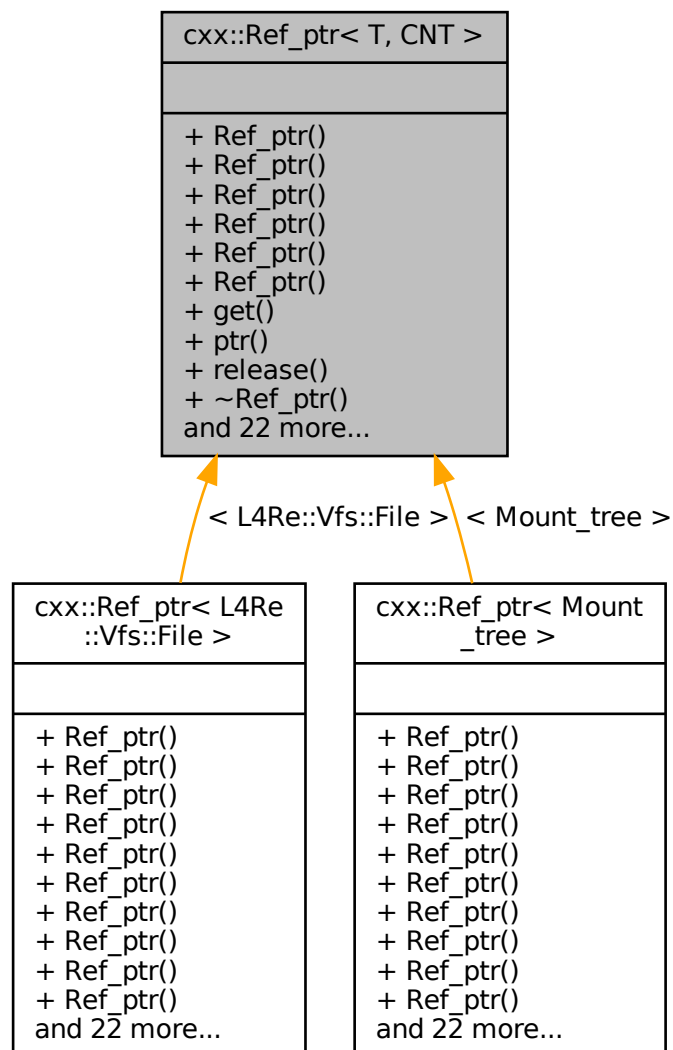
The documentation for this struct was generated from the following file:

- l4/cxx/ref_ptr_list

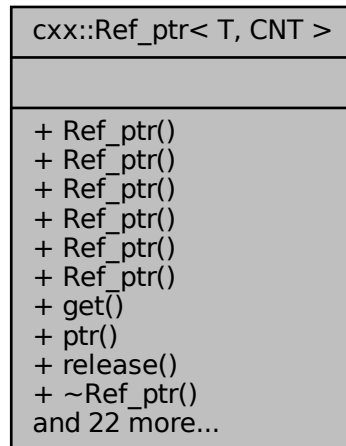
15.43 cxx::Ref_ptr< T, CNT > Class Template Reference

A reference-counting pointer with automatic cleanup.

Inheritance diagram for cxx::Ref_ptr< T, CNT >:



Collaboration diagram for `cxx::Ref_ptr< T, CNT >`:



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.

15.43.1 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](#).

15.43.2 Constructor & Destructor Documentation

15.43.2.1 `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](#).

15.43.2.2 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](#).

15.43.2.3 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](#).

15.43.3 Member Function Documentation

15.43.3.1 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](#).

15.43.3.2 `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`.

15.43.3.3 `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`.

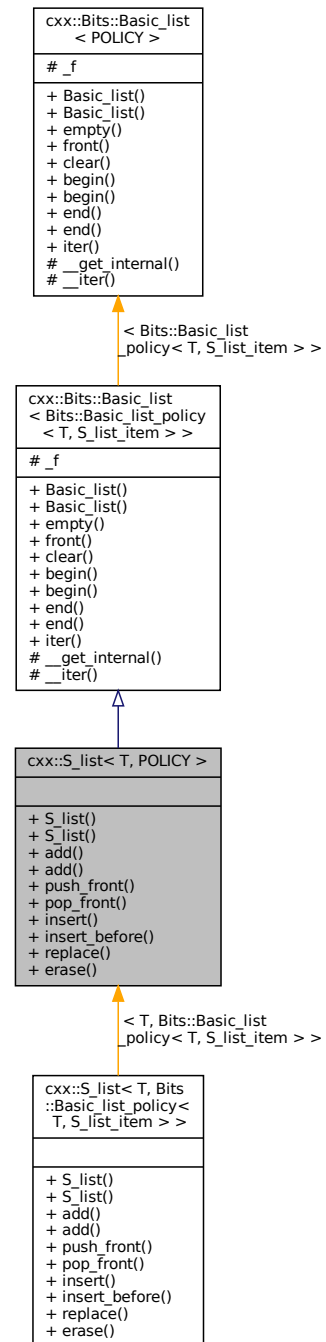
The documentation for this class was generated from the following file:

- `l4/cxx/ref_ptr`

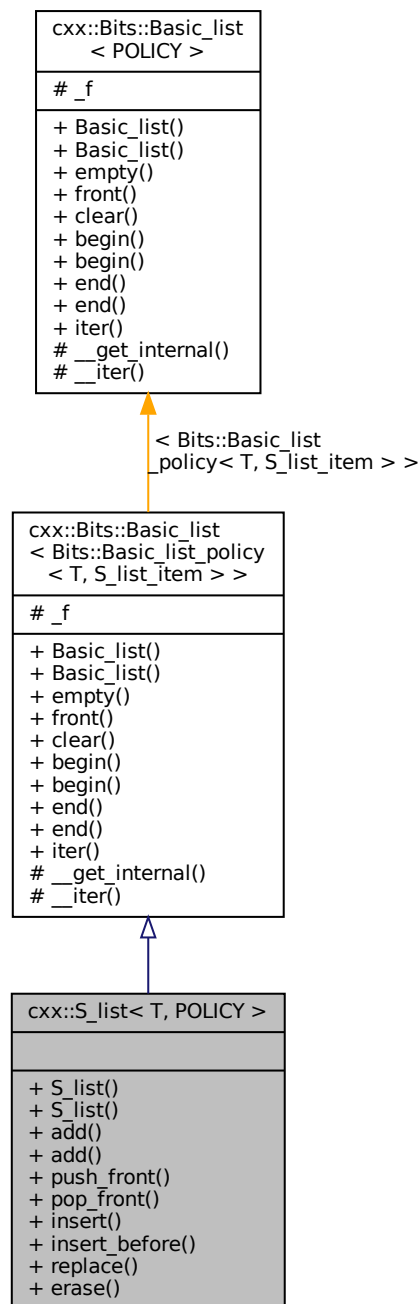
15.44 `cxx::S_list< T, POLICY >` Class Template Reference

Simple single-linked list.

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.

Additional Inherited Members

15.44.1 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

T	Type of elements saved in the list. Must inherit from cxx::S_list_item
----------	--

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

15.44.2 Member Function Documentation

15.44.2.1 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.

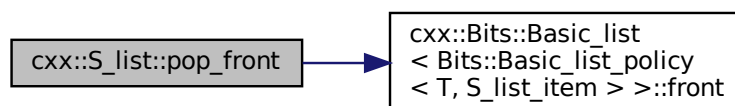
Precondition

The list must not be empty or the behaviour will be undefined.

Definition at line 91 of file [slist](#).

References [cxx::Bits::Basic_list< Bits::Basic_list_policy< T, S_list_item > >::f](#), and [cxx::Bits::Basic_list< Bits::Basic_list_policy< T](#)

Here is the call graph for this function:



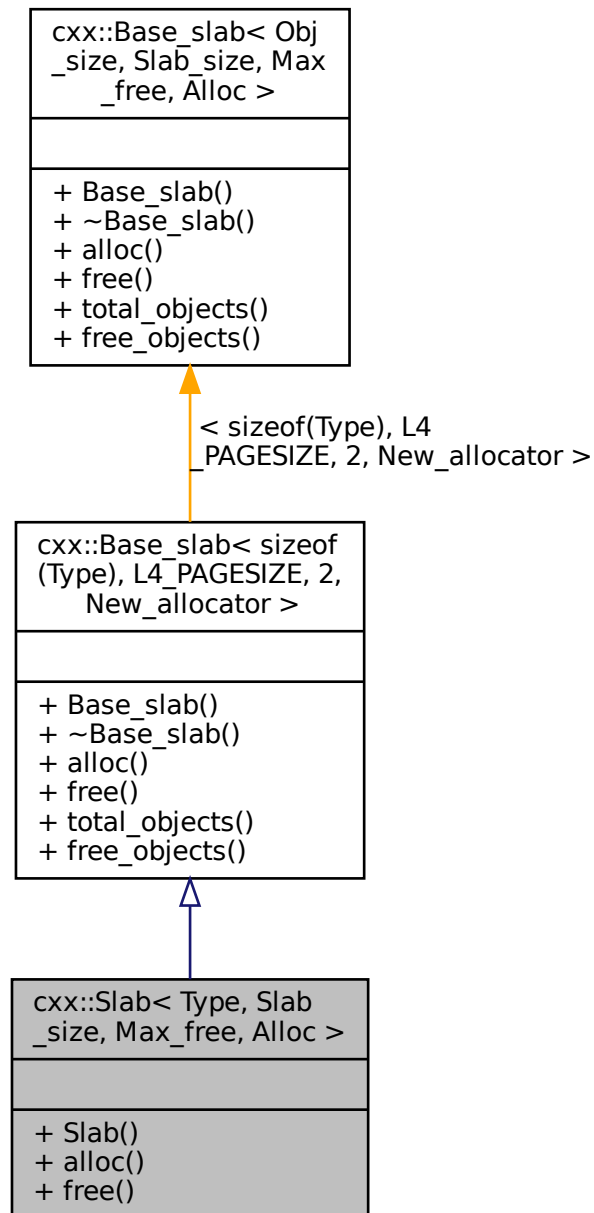
The documentation for this class was generated from the following file:

- `I4/cxx/slist`

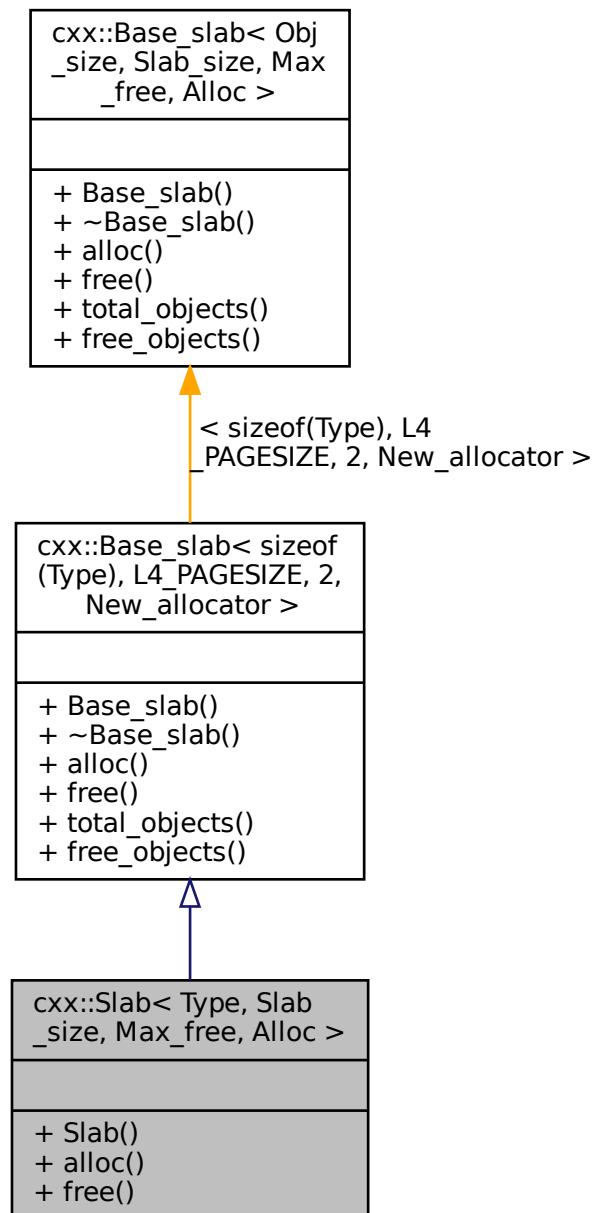
15.45 cxx::Slab< Type, Slab_size, Max_free, Alloc > Class Template Reference

[Slab](#) allocator for object of type `Type`.

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 () throw ()`
Allocate an object of type `Type`.
- `void free (Type *o) throw ()`
Free the object addressed by `o`.

Additional Inherited Members

15.45.1 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](#).

15.45.2 Member Function Documentation

15.45.2.1 `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 ( ) throw ( )    [inline]
```

Allocate an object of type `Type`.

Returns

A pointer to the object just allocated, or 0 on failure.

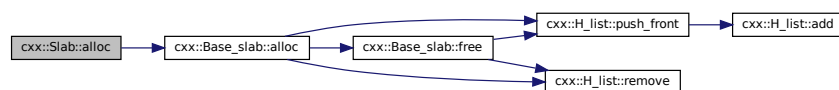
Note

The user is responsible for initializing the object.

Definition at line [366](#) of file [slab_alloc](#).

References [cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::alloc\(\)](#).

Here is the call graph for this function:



15.45.2.2 free()

```
template<typename Type , int Slab_size = L4_PAGE_SIZE, int Max_free = 2, template< typename A
> class Alloc = New_allocator>
void cxx::Slab< Type, Slab_size, Max_free, Alloc >::free (
    Type * o ) throw ( )    [inline]
```

Free the object addressed by o.

Parameters

<i>o</i>	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](#).

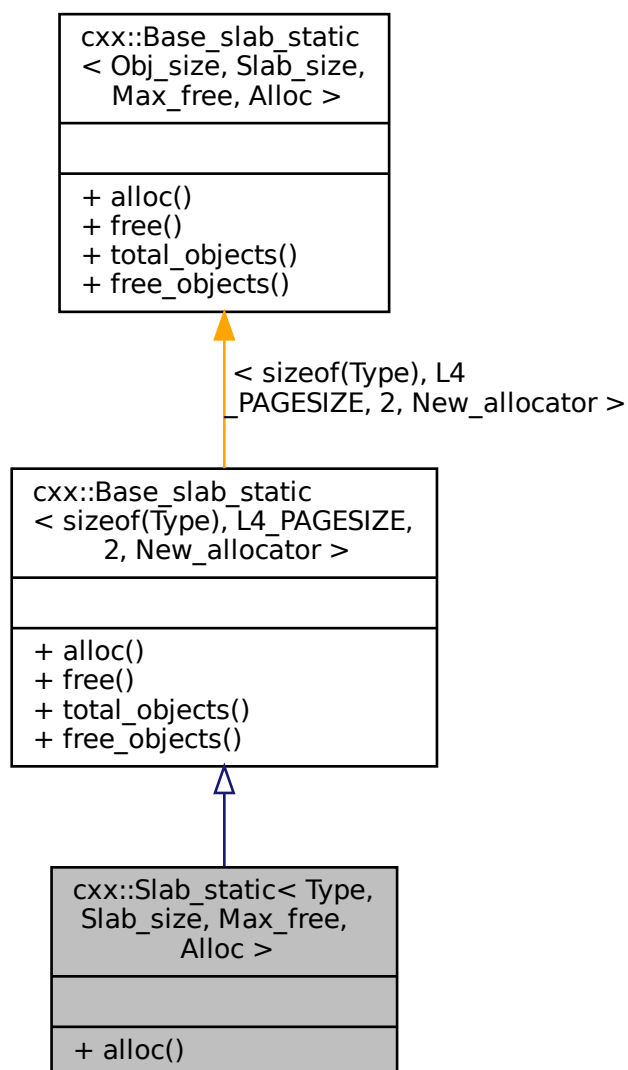
The documentation for this class was generated from the following file:

- l4/cxx/slab_alloc

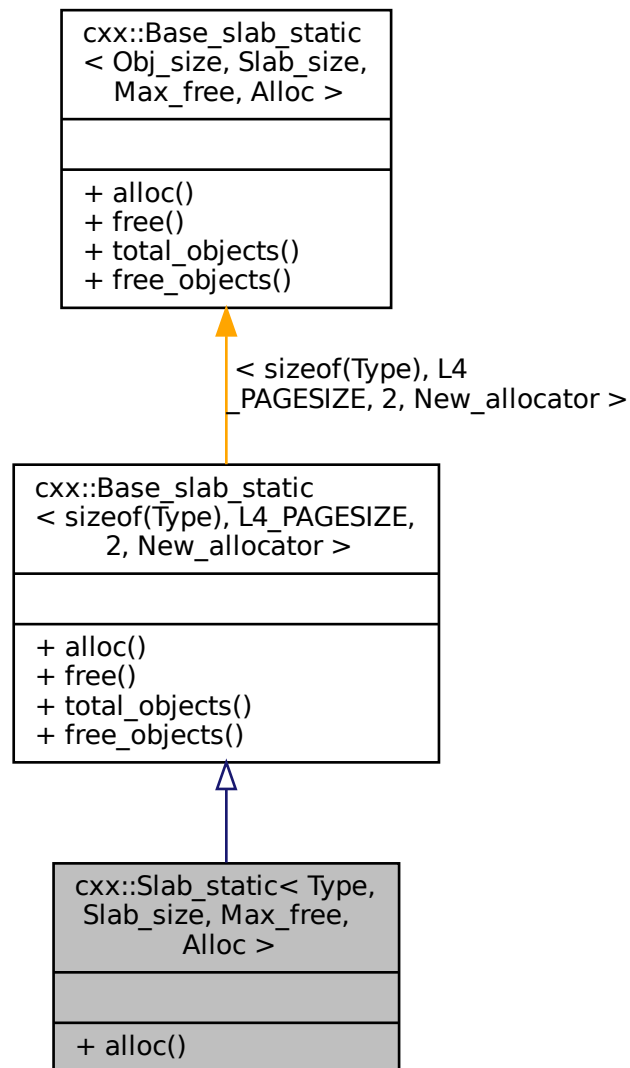
15.46 cxx::Slab_static< Type, Slab_size, Max_free, Alloc > Class Template Reference

Merged slab allocator (allocators for objects of the same size are merged together).

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 () throw ()`
Allocate an object of type `Type`.

Additional Inherited Members

15.46.1 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](#).

15.46.2 Member Function Documentation

15.46.2.1 `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 ( ) throw ( )    [inline]
```

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](#).

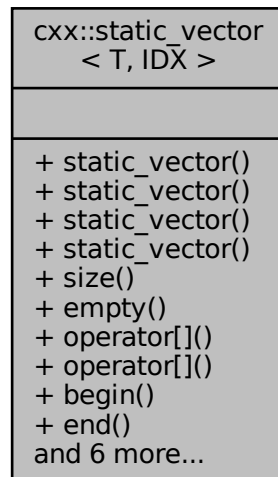
The documentation for this class was generated from the following file:

- `I4/cxx/slab_alloc`

15.47 `cxx::static_vector< T, IDX >` Class Template Reference

Simple encapsulation for a dynamically allocated array.

Collaboration diagram for `cxx::static_vector< T, IDX >`:



Public Member Functions

- `template<typename X, typename = typename enable_if<is_convertible<X, T>::value>::type>`
`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.

15.47.1 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`.

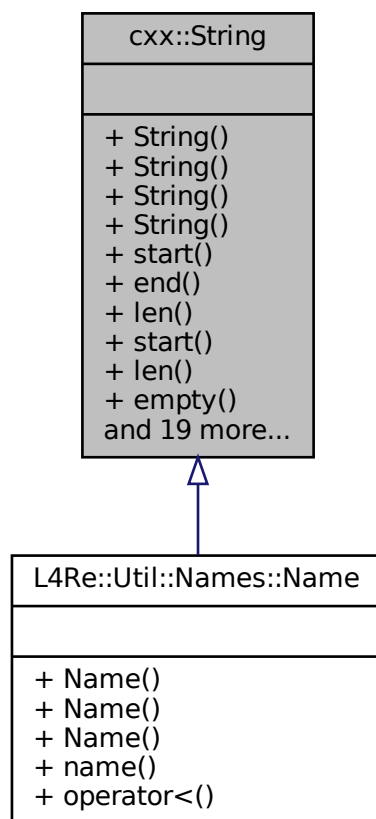
The documentation for this class was generated from the following file:

- `I4/cxx/static_vector`

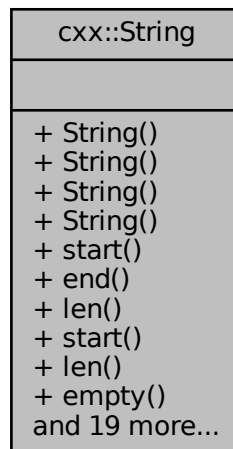
15.48 cxx::String Class Reference

Allocation free string class with explicit length field.

Inheritance diagram for cxx::String:



Collaboration diagram for cxx::String:



Public Types

- typedef char const * [Index](#)
Character index type.

Public Member Functions

- [String](#) (char const *s) throw ()
Initialize from a zero-terminated string.
- [String](#) (char const *s, unsigned long len) throw ()
Initialize from a pointer to first character and a length.
- [String](#) (char const *s, char const *e) throw ()
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.*

15.48.1 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.

Strings 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](#).

15.48.2 Constructor & Destructor Documentation

15.48.2.1 String()

```
cxx::String::String (
    char const * s,
    char const * e ) throw ( )    [inline]
```

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](#).

15.48.3 Member Function Documentation

15.48.3.1 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

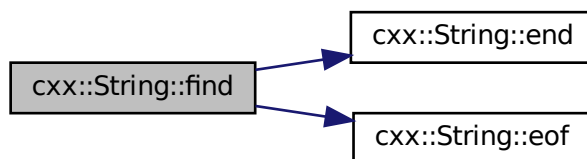
Return values

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:



15.48.3.2 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	<i>v</i>	conversion result
-----	----------	-------------------

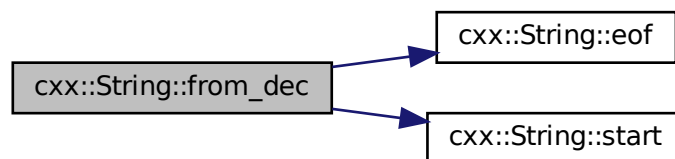
Returns

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:



15.48.3.3 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	<i>v</i>	conversion result
-----	----------	-------------------

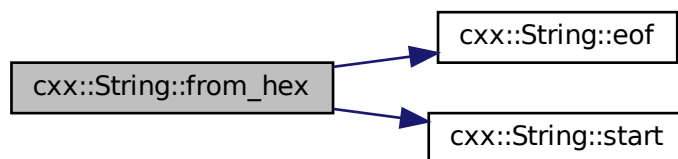
Return values

<i>-1</i>	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:



15.48.3.4 starts_with()

```
Index cxx::String::starts_with (  
    cxx::String const & c ) const [inline]
```

Check if `c` is a prefix of string.

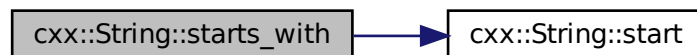
Returns

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:



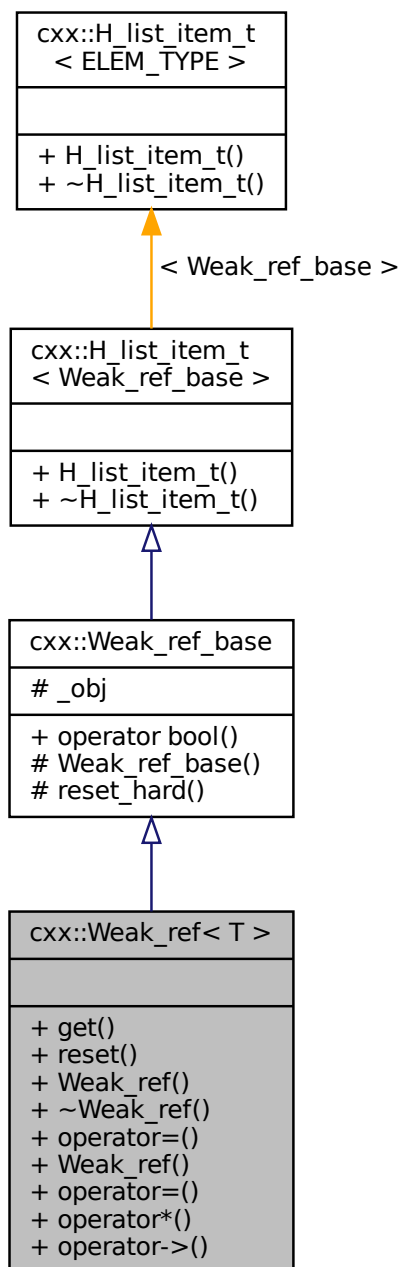
The documentation for this class was generated from the following file:

- `I4/cxx/string`

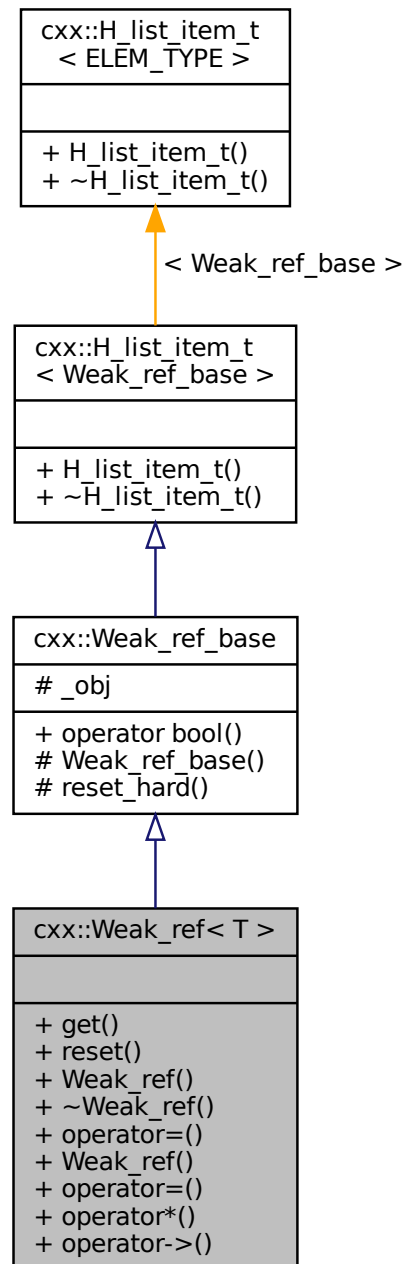
15.49 cxx::Weak_ref< T > Class Template Reference

Typed weak reference to an object of type T.

Inheritance diagram for cxx::Weak_ref< T >:



Collaboration diagram for `cxx::Weak_ref< T >`:



Additional Inherited Members

15.49.1 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](#).

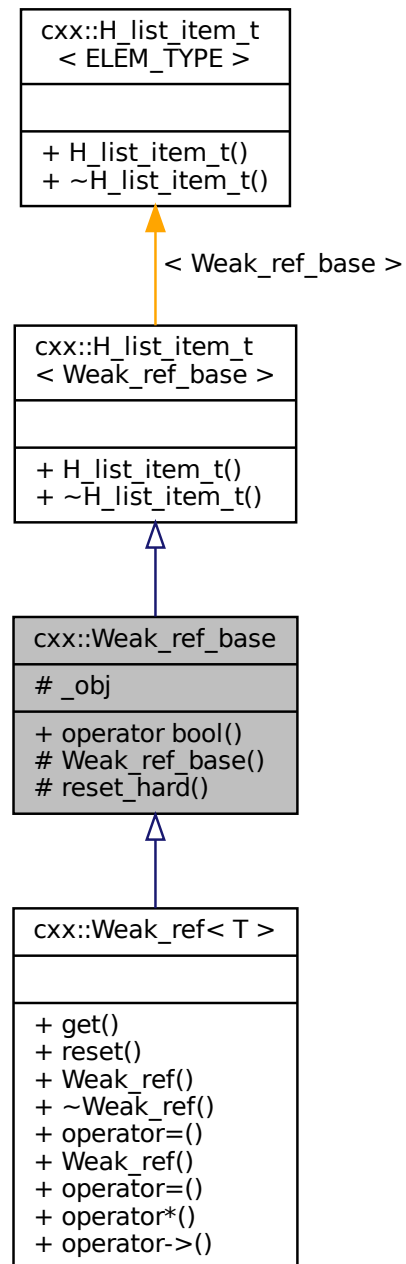
The documentation for this class was generated from the following file:

- `I4/cxx/weak_ref`

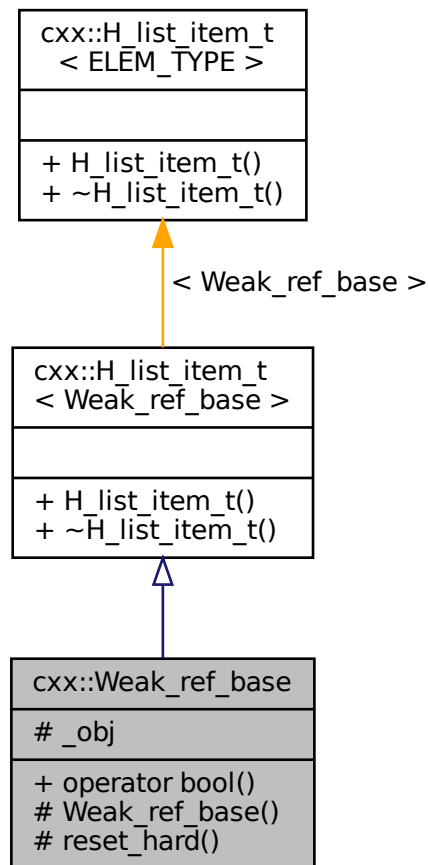
15.50 cxx::Weak_ref_base Class Reference

Generic (base) weak reference to some object.

Inheritance diagram for `cxx::Weak_ref_base`:



Collaboration diagram for cxx::Weak_ref_base:



Additional Inherited Members

15.50.1 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:

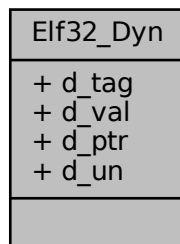
- `I4/cxx/weak_ref`

15.51 Elf32_Dyn Struct Reference

ELF32 dynamic entry.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Dyn:



Data Fields

- [Elf32_Sword d_tag](#)
see DT_ values
- [Elf32_Word d_val](#)
integer values with various interpret.
- [Elf32_Addr d_ptr](#)
program virtual addresses

15.51.1 Detailed Description

ELF32 dynamic entry.

Definition at line [499](#) of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

15.52 Elf32_Ehdr Struct Reference

ELF32 header.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Ehdr:

Elf32_Ehdr
<div>+ e_ident</div> <div>+ e_type</div> <div>+ e_machine</div> <div>+ e_version</div> <div>+ e_entry</div> <div>+ e_phoff</div> <div>+ e_shoff</div> <div>+ e_flags</div> <div>+ e_ehsize</div> <div>+ e_phentsize</div> <div>+ e_phnum</div> <div>+ e_shentsize</div> <div>+ e_shnum</div> <div>+ e_shstrndx</div>

Data Fields

- [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 eip
- [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](#)
- [Elf32_Half e_shentsize](#)
size of section header entry
- [Elf32_Half e_shnum](#)
- [Elf32_Half e_shstrndx](#)
sect.head.tab.idx of strtab

15.52.1 Detailed Description

ELF32 header.

Definition at line [122](#) of file [elf.h](#).

15.52.2 Field Documentation

15.52.2.1 e_phnum

[Elf32_Half](#) [Elf32_Ehdr::e_phnum](#)

of entries in prog. head. tab.

Definition at line [133](#) of file [elf.h](#).

15.52.2.2 e_shnum

[Elf32_Half](#) [Elf32_Ehdr::e_shnum](#)

of entries in sect. head. tab.

Definition at line [135](#) of file [elf.h](#).

The documentation for this struct was generated from the following file:

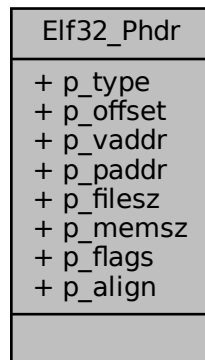
- [l4/util/elf.h](#)

15.53 Elf32_Phdr Struct Reference

ELF32 program header.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Phdr:



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

15.53.1 Detailed Description

ELF32 program header.

Definition at line 418 of file [elf.h](#).

The documentation for this struct was generated from the following file:

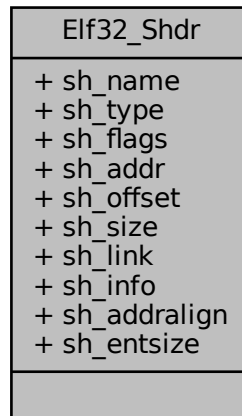
- [l4/util/elf.h](#)

15.54 Elf32_Shdr Struct Reference

ELF32 section header - figure 1-9, page 1-9.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Shdr:



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

15.54.1 Detailed Description

ELF32 section header - figure 1-9, page 1-9.

Definition at line 342 of file [elf.h](#).

The documentation for this struct was generated from the following file:

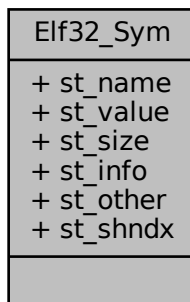
- [l4/util/elf.h](#)

15.55 Elf32_Sym Struct Reference

ELF32 symbol table entry.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Sym:



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

15.55.1 Detailed Description

ELF32 symbol table entry.

Definition at line 781 of file [elf.h](#).

The documentation for this struct was generated from the following file:

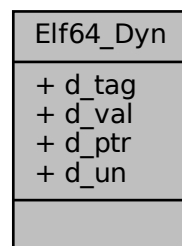
- [l4/util/elf.h](#)

15.56 Elf64_Dyn Struct Reference

ELF64 dynamic entry.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Dyn:



Data Fields

- [Elf64_Sxword d_tag](#)
see DT_ values
- [Elf64_Xword d_val](#)
integer values with various interpret.
- [Elf64_Addr d_ptr](#)
program virtual addresses

15.56.1 Detailed Description

ELF64 dynamic entry.

Definition at line 508 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

15.57 Elf64_Ehdr Struct Reference

ELF64 header.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Ehdr:

Elf64_Ehdr
<div>+ e_ident</div> <div>+ e_type</div> <div>+ e_machine</div> <div>+ e_version</div> <div>+ e_entry</div> <div>+ e_phoff</div> <div>+ e_shoff</div> <div>+ e_flags</div> <div>+ e_ehsize</div> <div>+ e_phentsize</div> <div>+ e_phnum</div> <div>+ e_shentsize</div> <div>+ e_shnum</div> <div>+ e_shstrndx</div>

Data Fields

- [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 eip
- [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](#)
of entries in prog.
- [Elf64_Half e_shentsize](#)
size of section header entry
- [Elf64_Half e_shnum](#)
of entries in sect.
- [Elf64_Half e_shstrndx](#)
sect.head.tab.idx of strtab

15.57.1 Detailed Description

ELF64 header.

Definition at line 142 of file [elf.h](#).

15.57.2 Field Documentation

15.57.2.1 e_phnum

[Elf64_Half](#) Elf64_Ehdr::e_phnum

of entries in prog.

head. tab.

Definition at line 153 of file [elf.h](#).

15.57.2.2 e_shnum

[Elf64_Half](#) Elf64_Ehdr::e_shnum

of entries in sect.

head. tab.

Definition at line 155 of file [elf.h](#).

The documentation for this struct was generated from the following file:

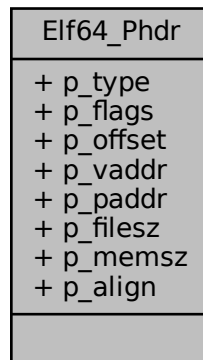
- [l4/util/elf.h](#)

15.58 Elf64_Phdr Struct Reference

ELF64 program header.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Phdr:



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

15.58.1 Detailed Description

ELF64 program header.

Definition at line [430](#) of file [elf.h](#).

The documentation for this struct was generated from the following file:

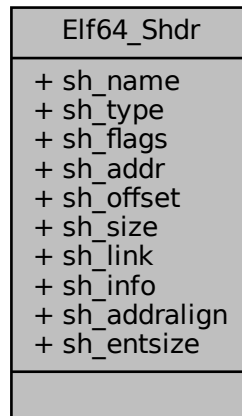
- [l4/util/elf.h](#)

15.59 Elf64_Shdr Struct Reference

ELF64 section header.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Shdr:



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

15.59.1 Detailed Description

Elf64 section header.

Definition at line 356 of file [elf.h](#).

The documentation for this struct was generated from the following file:

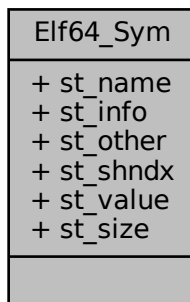
- [l4/util/elf.h](#)

15.60 Elf64_Sym Struct Reference

Elf64 symbol table entry.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Sym:



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

15.60.1 Detailed Description

ELF64 symbol table entry.

Definition at line 791 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

15.61 L4::Alloc_list Class Reference

A simple list-based allocator.

```
#include <alloc.h>
```

Collaboration diagram for L4::Alloc_list:



15.61.1 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:

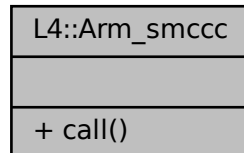
- [l4/cxx/alloc.h](#)

15.62 L4::Arm_smccc Class Reference

Wrapper for function calls that follow the ARM SMC/HVC calling convention.

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.

15.62.1 Detailed Description

Wrapper for function calls that follow the ARM SMC/HVC calling convention.

Definition at line 20 of file [arm_smccc](#).

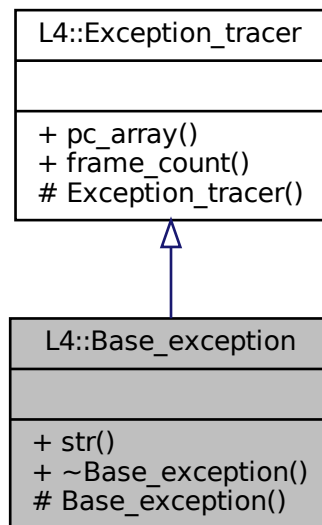
15.62.2 Member Function Documentation

15.62.2.1 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)
  
```


Collaboration diagram for L4::Base_exception:



Public Member Functions

- virtual char const * [str](#) () const =0 throw ()
Return a human readable string for the exception.
- virtual [~Base_exception](#) () throw ()
Destruction.

Protected Member Functions

- [Base_exception](#) () throw ()
Create a base exception.

15.63.1 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](#).

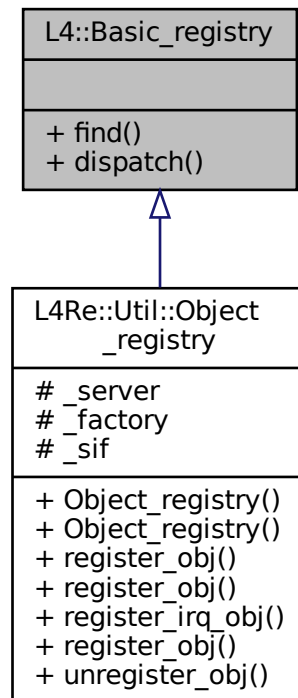
The documentation for this class was generated from the following file:

- [l4/cxx/exceptions](#)

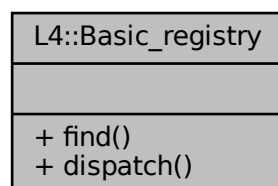
15.64 L4::Basic_registry Class Reference

This registry returns the corresponding server object based on the label of an [lpc_gate](#).

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.

15.64.1 Detailed Description

This registry returns the corresponding server object based on the label of an [lpc_gate](#).

Definition at line 541 of file [ipc_epiface](#).

15.64.2 Member Function Documentation

15.64.2.1 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 566 of file [ipc_epiface](#).

15.64.2.2 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 [550](#) of file [ipc_epiface](#).

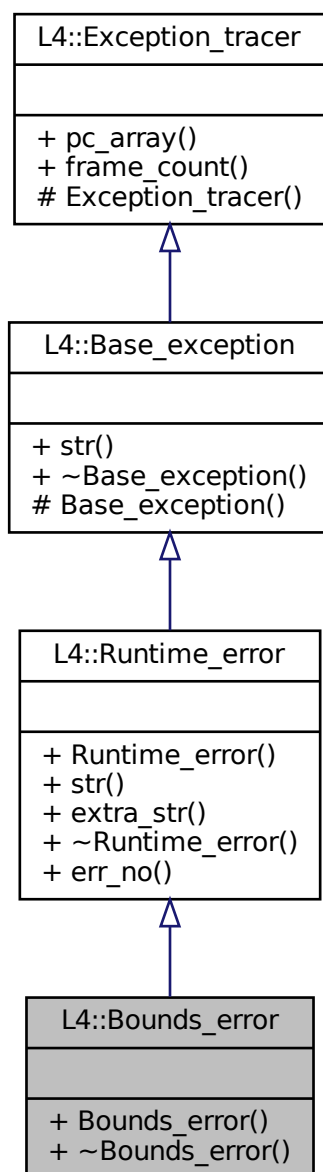
The documentation for this class was generated from the following file:

- `l4/sys/cxx/ipc_epiface`

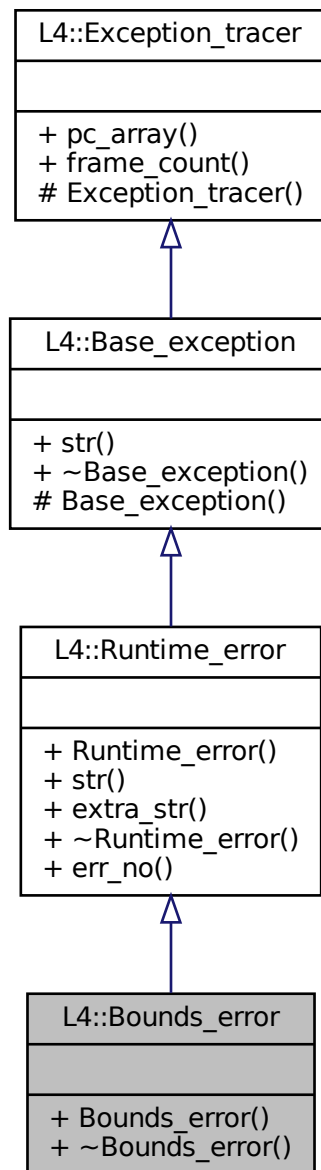
15.65 L4::Bounds_error Class Reference

Access out of bounds.

Inheritance diagram for L4::Bounds_error:



Collaboration diagram for L4::Bounds_error:



Additional Inherited Members

15.65.1 Detailed Description

Access out of bounds.

Definition at line 289 of file [exceptions](#).

The documentation for this class was generated from the following file:

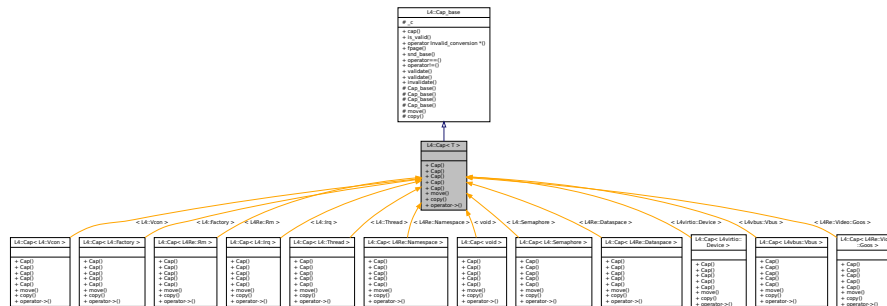
- [l4/cxx/exceptions](#)

15.66 L4::Cap< T > Class Template Reference

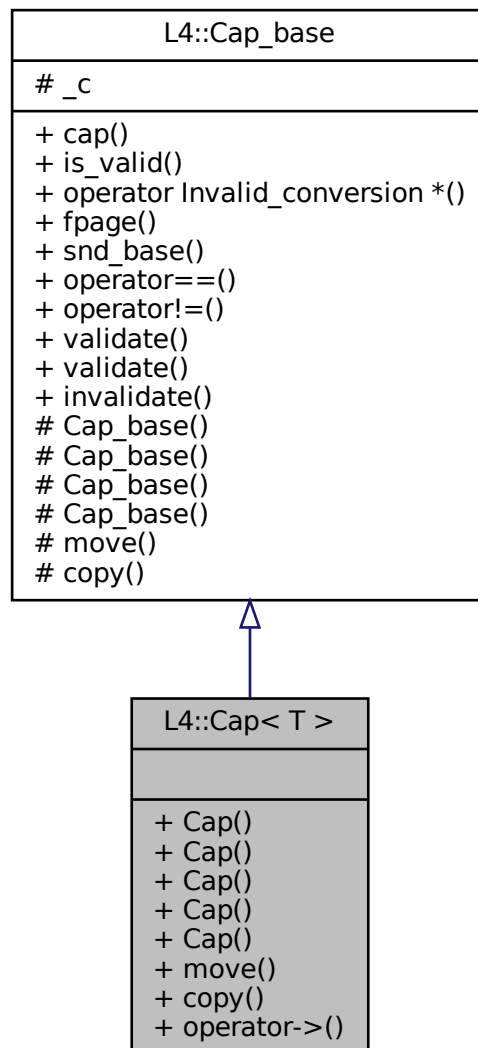
C++ interface for capabilities.

```
#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.

Friends

- class [L4::Kobject](#)

Additional Inherited Members

15.66.1 Detailed Description

```
template<typename T>
class L4::Cap< T >
```

C++ interface for capabilities.

Template Parameters

T	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++/shared_ds/ds_clnt.cc](#), and [examples/libs/l4re/streammap/client.cc](#).

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

15.66.2 Constructor & Destructor Documentation

15.66.2.1 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

<i>o</i>	The source selector that shall be copied (and casted).
----------	--

Definition at line 244 of file [capability.h](#).

15.66.2.2 Cap() [2/4]

```
template<typename T >
L4::Cap< T >::Cap (
    Cap_type cap ) [inline], [noexcept]
```

Constructor to create an invalid capability selector.

Parameters

<i>cap</i>	Capability selector.
------------	----------------------

Definition at line 251 of file [capability.h](#).

15.66.2.3 Cap() [3/4]

```
template<typename T >
L4::Cap< T >::Cap (
    l4_default_caps_t cap ) [inline], [noexcept]
```

Initialize capability with one of the default capability selectors.

Parameters

<i>cap</i>	Capability selector.
------------	----------------------

Definition at line 257 of file [capability.h](#).

15.66.2.4 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 263 of file [capability.h](#).

15.66.3 Member Function Documentation

15.66.3.1 copy()

```
template<typename T >
Cap L4::Cap< T >::copy (
    Cap< T > const & src ) const [inline]
```

Copy a capability to this cap slot.

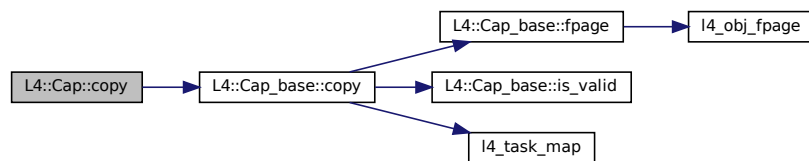
Parameters

<i>src</i>	the source capability slot.
------------	-----------------------------

Definition at line 286 of file [capability.h](#).

References [L4::Cap_base::copy\(\)](#).

Here is the call graph for this function:



15.66.3.2 move()

```
template<typename T >
Cap L4::Cap< T >::move (
    Cap< T > const & src ) const [inline]
```

Move a capability to this cap slot.

Collaboration diagram for L4::Cap_base:

L4::Cap_base
_c
+ cap() + is_valid() + operator Invalid_conversion *() + fpage() + snd_base() + operator==() + operator!=() + validate() + validate() + invalidate() # Cap_base() # Cap_base() # Cap_base() # Cap_base() # move() # copy()

Public Types

- enum [No_init_type](#) { [No_init](#) }
Special value for uninitialized capability objects.
- enum [Cap_type](#) { [Invalid](#) = L4_INVALID_CAP }
Invalid capability type.

Public Member Functions

- [l4_cap_idx_t](#) [cap](#) () const noexcept
Return capability selector.
- bool [is_valid](#) () const noexcept
Test whether the capability is a valid capability index (i.e., not L4_INVALID_CAP).
- [l4_fpage_t](#) [fpage](#) (unsigned rights=[L4_CAP_FPAGE_RWS](#)) const noexcept
Return flex-page for the capability.
- [l4_umword_t](#) [snd_base](#) (unsigned grant=0, [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.

15.67.1 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](#).

15.67.2 Member Enumeration Documentation

15.67.2.1 Cap_type

```
enum L4::Cap_base::Cap_type
```

Invalid capability type.

Enumerator

Invalid	Invalid capability selector.
---------	------------------------------

Definition at line 43 of file [capability.h](#).

15.67.2.2 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](#).

15.67.3 Constructor & Destructor Documentation**15.67.3.1 Cap_base() [1/2]**

```
L4::Cap_base::Cap_base (
    l4_cap_idx_t c ) [inline], [explicit], [protected], [noexcept]
```

Generate a capability from its C representation.

Parameters

c	The C capability
---	------------------

Definition at line 144 of file [capability.h](#).

15.67.3.2 Cap_base() [2/2]

```
L4::Cap_base::Cap_base (
    l4_default_caps_t cap ) [inline], [explicit], [protected], [noexcept]
```

Initialize capability with one of the default capabilities.

Parameters

<i>cap</i>	Capability.
------------	-------------

Definition at line 155 of file [capability.h](#).

15.67.4 Member Function Documentation

15.67.4.1 `cap()`

```
l4_cap_idx_t L4::Cap_base::cap ( ) const [inline], [noexcept]
```

Return capability selector.

Returns

Capability selector.

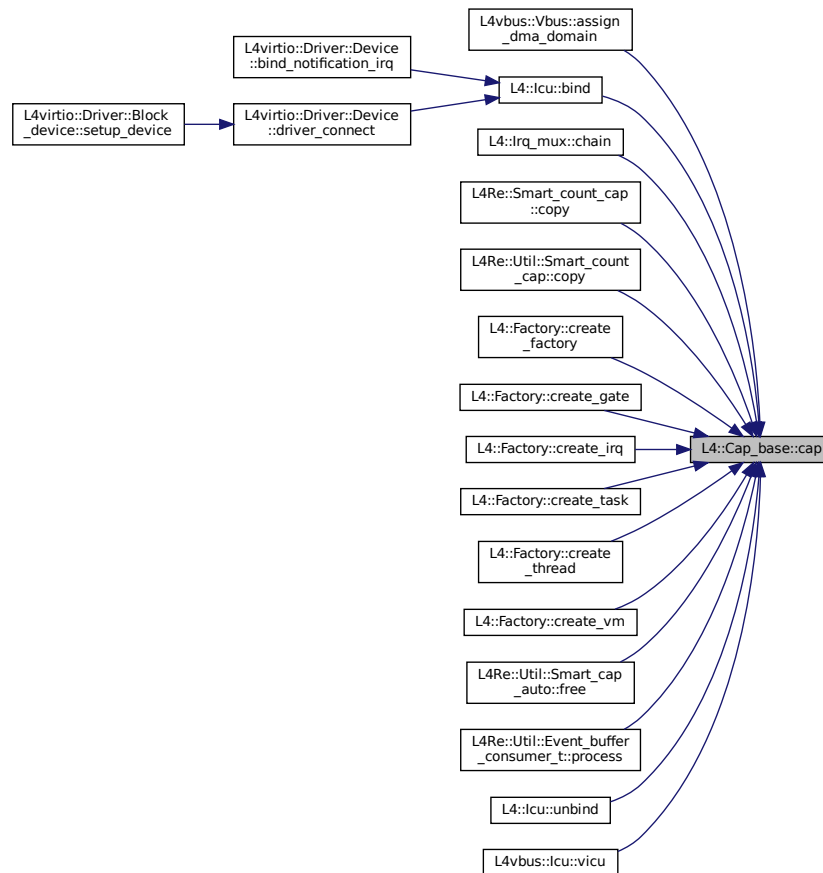
Examples

[examples/libs/l4re/streammap/client.cc](#).

Definition at line 52 of file [capability.h](#).

Referenced by [L4vbus::Vbus::assign_dma_domain\(\)](#), [L4::lcu::bind\(\)](#), [L4::lq_mux::chain\(\)](#), [L4Re::Smart_count_cap< Unmap_flags >::L4Re::Util::Smart_count_cap< Unmap_flags >::copy\(\)](#), [L4::Factory::create_factory\(\)](#), [L4::Factory::create_gate\(\)](#), [L4::Factory::create_irq\(\)](#), [L4::Factory::create_task\(\)](#), [L4::Factory::create_thread\(\)](#), [L4::Factory::create_vm\(\)](#), [L4Re::Util::Smart_cap_auto< Unmap_flags >::free\(\)](#), [L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::process\(\)](#), [L4::lcu::unbind\(\)](#), and [L4vbus::lcu::vicu\(\)](#).

Here is the caller graph for this function:



15.67.4.2 copy()

```
void L4::Cap_base::copy (
    Cap_base const & src ) const [inline], [protected]
```

Copy a capability.

Parameters

<i>src</i>	the source capability.
------------	------------------------

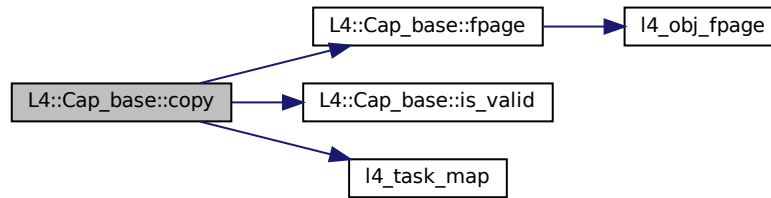
After this operation this capability refers to the same object as *src*.

Definition at line 187 of file [capability.h](#).

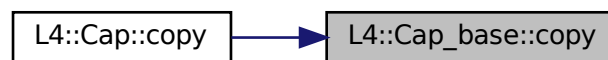
References [fpage\(\)](#), [is_valid\(\)](#), [L4_BASE_TASK_CAP](#), [L4_CAP_FPAGE_RWSD](#), [L4_FPAGE_C_OBJ_RIGHTS](#), and [l4_task_map\(\)](#).

Referenced by [L4::Cap< T >::copy\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.67.4.3 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

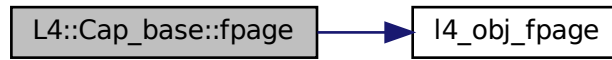
flex-page

Definition at line 72 of file [capability.h](#).

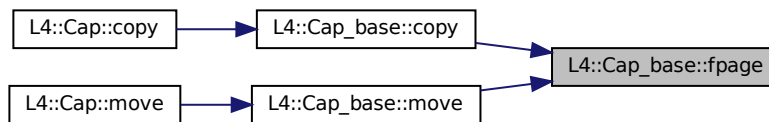
References [l4_obj_fpage\(\)](#).

Referenced by [copy\(\)](#), and [move\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.67.4.4 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

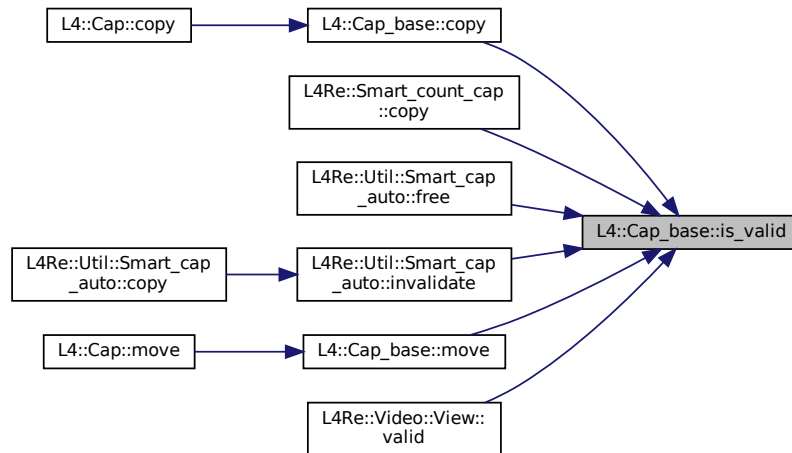
Examples

[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\(\)](#), [move\(\)](#), and [L4Re::Video::View::valid\(\)](#).

Here is the caller graph for this function:



15.67.4.5 move()

```
void L4::Cap_base::move (
    Cap_base const & src ) const [inline], [protected]
```

Replace this capability with the contents of `src`.

Parameters

<code>src</code>	the source capability.
------------------	------------------------

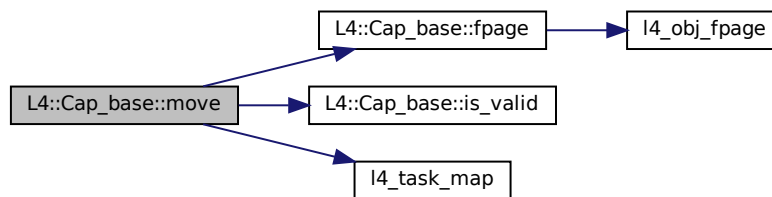
After the operation this capability refers to the object formerly referred to by the source capability `src`, and the source capability no longer refers to an object.

Definition at line 171 of file [capability.h](#).

References [fpage\(\)](#), [is_valid\(\)](#), [L4_BASE_TASK_CAP](#), [L4_CAP_FPAGE_RWSD](#), [L4_FPAGE_C_OBJ_RIGHTS](#), [L4_MAP_ITEM_GRANT](#), and [l4_task_map\(\)](#).

Referenced by [L4::Cap< T >::move\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.67.4.6 snd_base()

```

l4_umword_t L4::Cap_base::snd_base (
    unsigned grant = 0,
    l4_cap_idx_t base = L4_INVALID_CAP ) const [inline], [noexcept]
  
```

Return send base.

Parameters

<i>grant</i>	True object should be granted.
<i>base</i>	Base capability (first in a bundle of aligned capabilities)

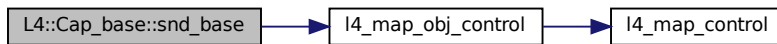
Returns

Map object.

Definition at line 83 of file [capability.h](#).

References [L4_INVALID_CAP](#), and [l4_map_obj_control\(\)](#).

Here is the call graph for this function:



15.67.4.7 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, usually the UTCB of the calling thread.

Return values

<i>tag.label()</i>	> 0 Capability is present (refers to an object).
<i>tag.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](#).

15.67.4.8 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, usually the UTCB of the calling thread.
----------	---

Return values

<i>tag.label()</i>	> 0 Capability is present (refers to an object).
<i>tag.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](#).

The documentation for this class was generated from the following files:

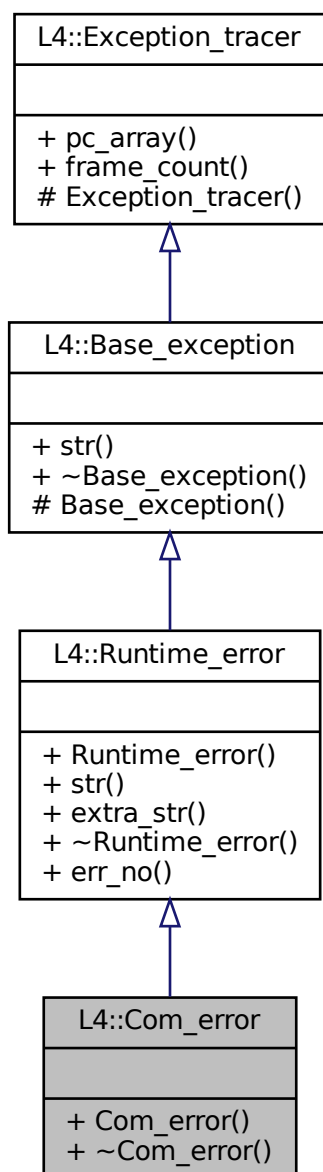
- l4/sys/cxx/capability.h
- l4/sys/[capability](#)

15.68 L4::Com_error Class Reference

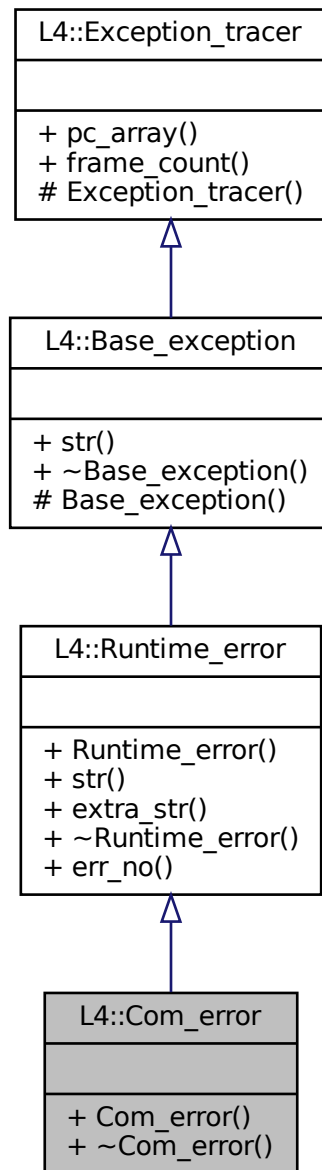
Error conditions during IPC.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Com_error:



Collaboration diagram for L4::Com_error:



Public Member Functions

- [Com_error](#) (long err) throw ()
Create a [Com_error](#) for the given [L4](#) IPC error code.

Additional Inherited Members

15.68.1 Detailed Description

Error conditions during IPC.

This exception encapsulates all IPC error conditions of [L4](#) IPC.

Definition at line [274](#) of file [exceptions](#).

15.68.2 Constructor & Destructor Documentation

15.68.2.1 Com_error()

```
L4::Com_error::Com_error (
    long err ) throw ( )    [inline], [explicit]
```

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](#).

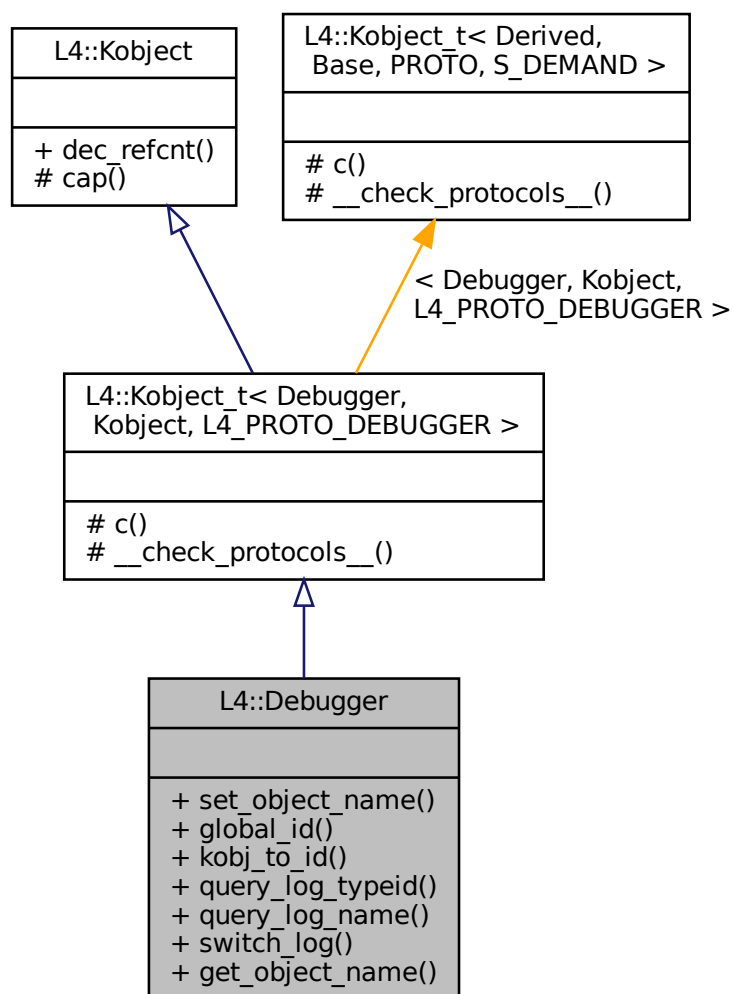
The documentation for this class was generated from the following file:

- [l4/cxx/exceptions](#)

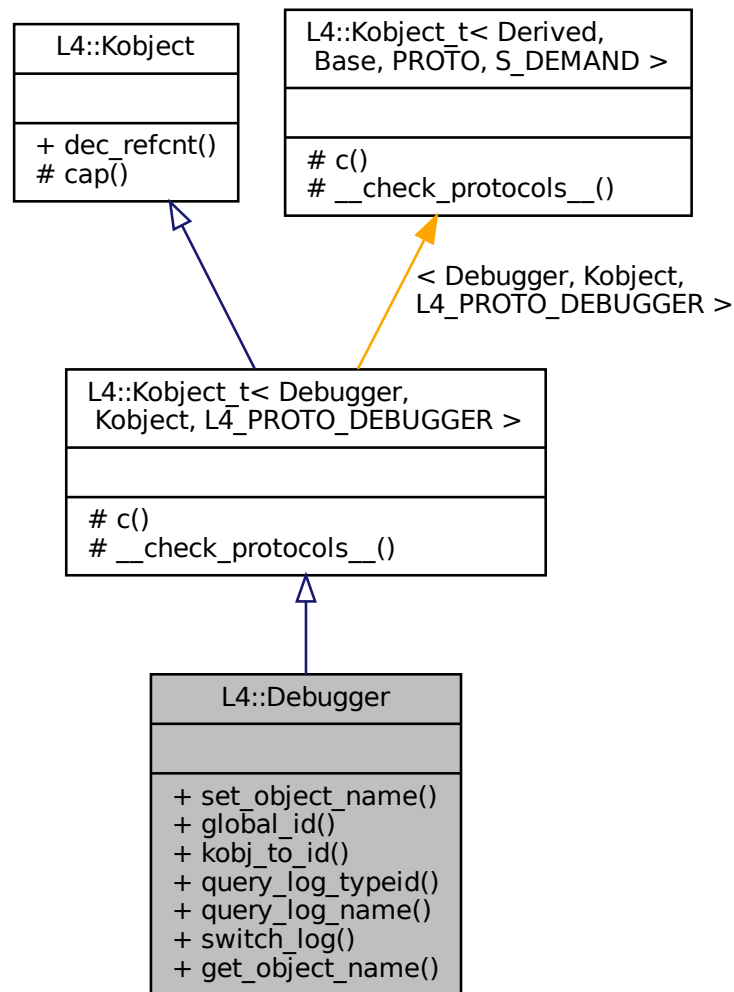
15.69 L4::Debugger Class Reference

C++ kernel debugger API.

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.

Additional Inherited Members

15.69.1 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`.

15.69.2 Member Function Documentation

15.69.2.1 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 <i>name</i> buffer.
	<i>utcb</i>	The UTCTB to use for the operation.

Returns

Syscall return tag

Definition at line 159 of file [debugger](#).

15.69.2.2 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

<i>utcb</i>	The UTCB to use for the operation.
-------------	------------------------------------

Return values

$\sim 0UL$	The capability is invalid.
≥ 0	The global debugger id.

Definition at line 82 of file [debugger](#).

15.69.2.3 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>	The UTCB to use for the operation.

Return values

$\sim 0UL$	The capability or the Kobject pointer are invalid.
≥ 0	The globally unique id.

Definition at line 94 of file [debugger](#).

15.69.2.4 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>	The UTCB to use for the operation.

Return values

0	Success
<0	Error

Definition at line 127 of file [debugger](#).

15.69.2.5 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>	The UTCB to use for the operation.

Return values

>=0	Id
<0	Error

Definition at line 108 of file [debugger](#).

15.69.2.6 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>	The UTCB to use for the operation.

Returns

System call return tag.

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

15.69.2.7 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>	The UTCB to use for the operation.

Returns

Syscall return tag

Definition at line 144 of file [debugger](#).

The documentation for this class was generated from the following file:

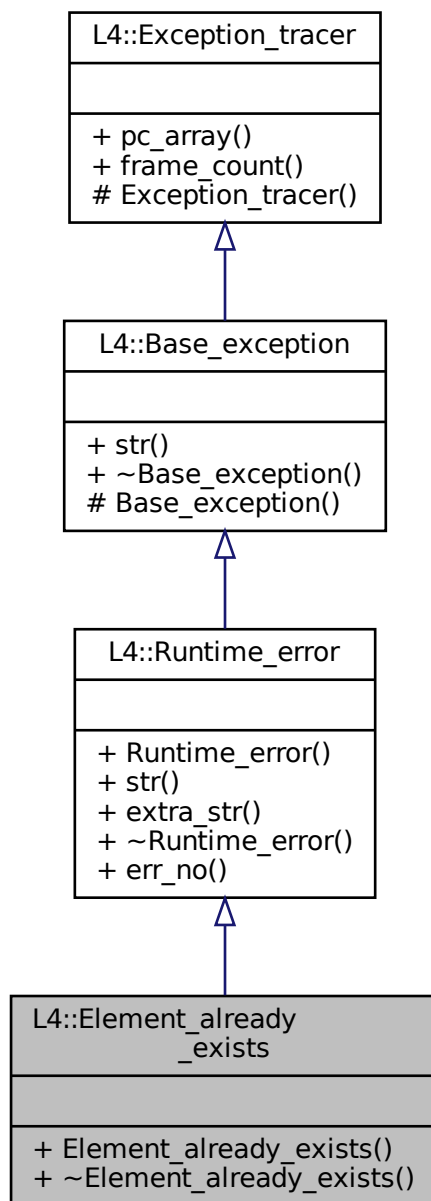
- [l4/sys/debugger](#)

15.70 L4::Element_already_exists Class Reference

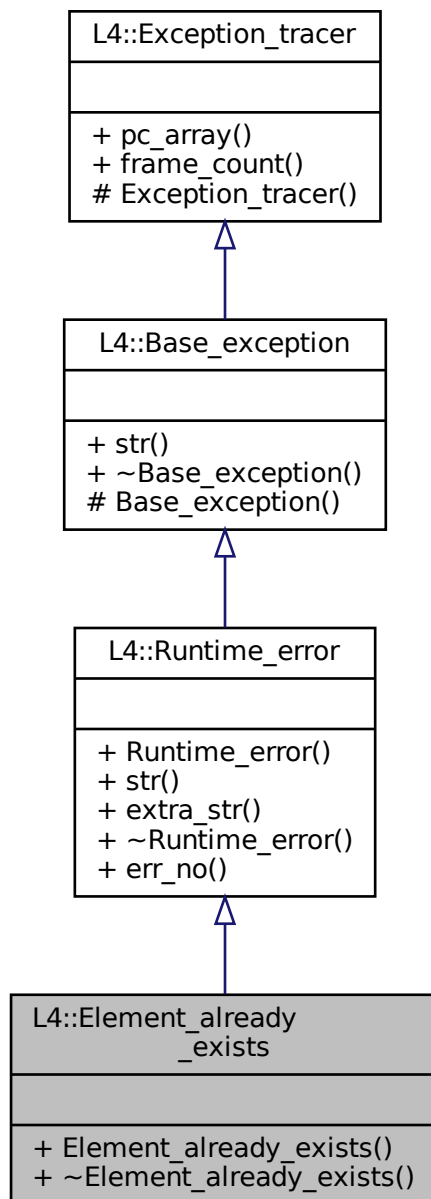
[Exception](#) for duplicate element insertions.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Element_already_exists:



Collaboration diagram for L4::Element_already_exists:



Additional Inherited Members

15.70.1 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:

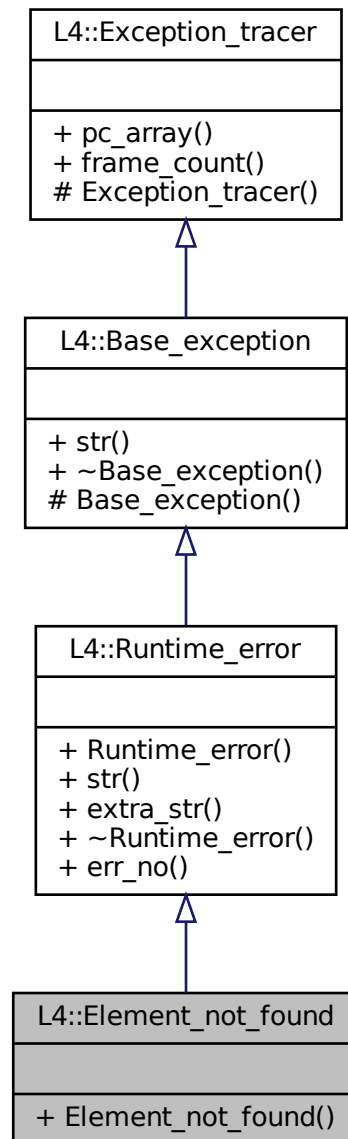
- [l4/cxx/exceptions](#)

15.71 L4::Element_not_found Class Reference

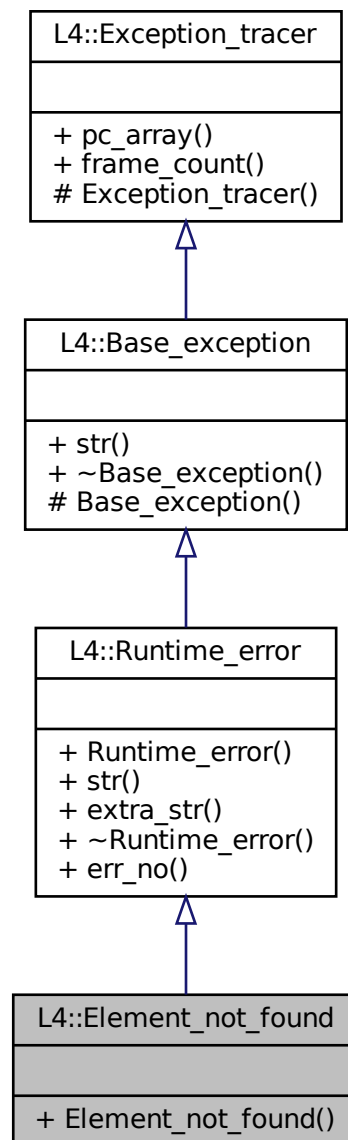
[Exception](#) for a failed lookup (element not found).

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Element_not_found:



Collaboration diagram for L4::Element_not_found:



Additional Inherited Members

15.71.1 Detailed Description

[Exception](#) for a failed lookup (element not found).

Definition at line 231 of file [exceptions](#).

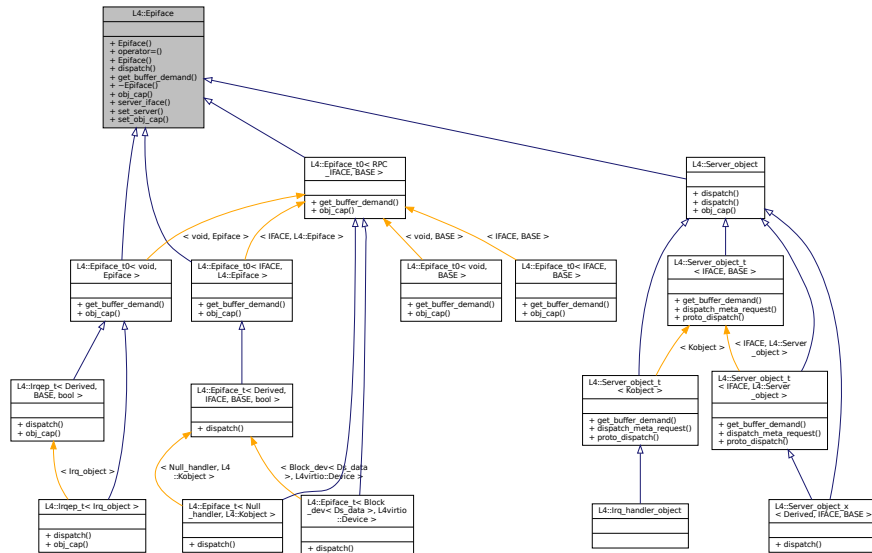
The documentation for this class was generated from the following file:

- [l4/cxx/exceptions](#)

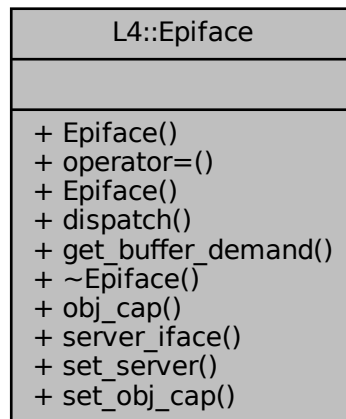
15.72 L4::Epiface Struct Reference

Base class for interface implementations.

Inheritance diagram for L4::Epiface:



Collaboration diagram for L4::Epiface:



Public Types

- typedef [lpc_svr::Server_iface](#) **Server_iface**
Type for abstract server interface.
- typedef [lpc_svr::Server_iface::Demand](#) **Demand**
Type for server-side receive buffer demand.

Public Member Functions

- [Epiface](#) ()
Make a server object.
- virtual [l4_msgtag_t dispatch](#) ([l4_msgtag_t](#) tag, unsigned rights, [l4_utcb_t](#) *utcb)=0
The abstract handler for client requests to the object.
- virtual [Demand get_buffer_demand](#) () const =0
Get the server-side receive buffer demand for this object.
- virtual [~Epiface](#) ()=0
Destroy the object.
- Stored_cap [obj_cap](#) () const
Get the capability to the kernel object belonging to this object.
- [Server_iface](#) * [server_iface](#) () const
Get pointer to server interface at which the object is currently registered.
- int [set_server](#) ([Server_iface](#) *srv, [Cap](#)< void > cap, bool managed=false)
Set server registration info for the object.
- void [set_obj_cap](#) ([Cap](#)< void > const &cap)
Deprecated server registration function.

15.72.1 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 153 of file [ipc_epiface](#).

15.72.2 Member Function Documentation

15.72.2.1 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::lrqep_t< Derived, BASE, bool >](#), [L4::lrqep_t< Irq_object >](#), [L4::Epiface_t< Derived, IFACE, BASE, bool >](#), [L4::Epiface_t< Block_dev< Ds_data >, L4virtio::Device >](#), [L4::Epiface_t< Null_handler, L4::Kobject >](#), and [L4::Server_object](#).

15.72.2.2 `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< RPC_IFACE, BASE >](#), [L4::Epiface_t0< void, Epiface >](#), [L4::Epiface_t0< IFACE, L4::Epiface >](#), [L4::Server_object_t< IFACE, BASE >](#), [L4::Server_object_t< Kobject >](#), and [L4::Server_object_t< IFACE, L4::Server_object >](#).

15.72.2.3 `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 [lrq](#).

Definition at line 214 of file [ipc_epiface](#).

Referenced by [L4Re::Util::Object_registry::unregister_obj\(\)](#).

Here is the caller graph for this function:



15.72.2.4 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 221 of file [ipc_epiface](#).

15.72.2.5 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

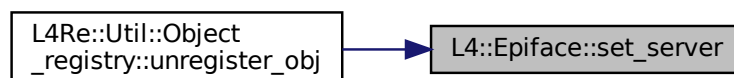
0 on success, -L4_EINVAL if the srv and cap are not consistent.

Definition at line 232 of file [ipc_epiface](#).

References [L4_EINVAL](#).

Referenced by [L4Re::Util::Object_registry::unregister_obj\(\)](#).

Here is the caller graph for this function:



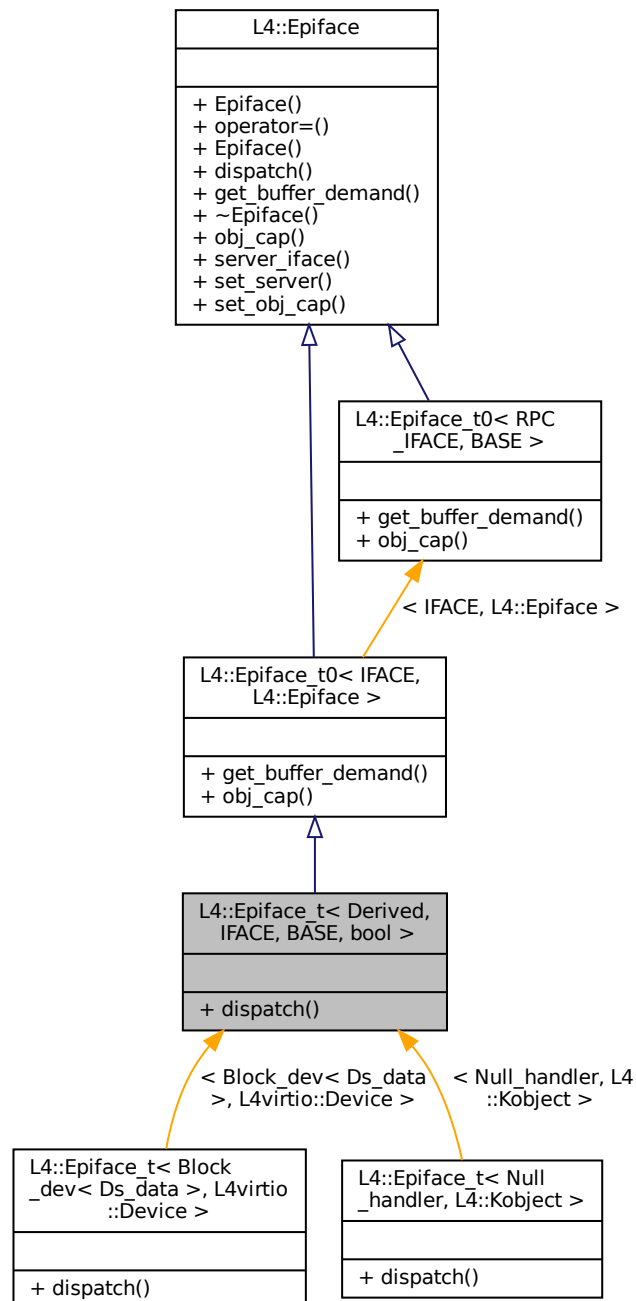
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_epiface](#)

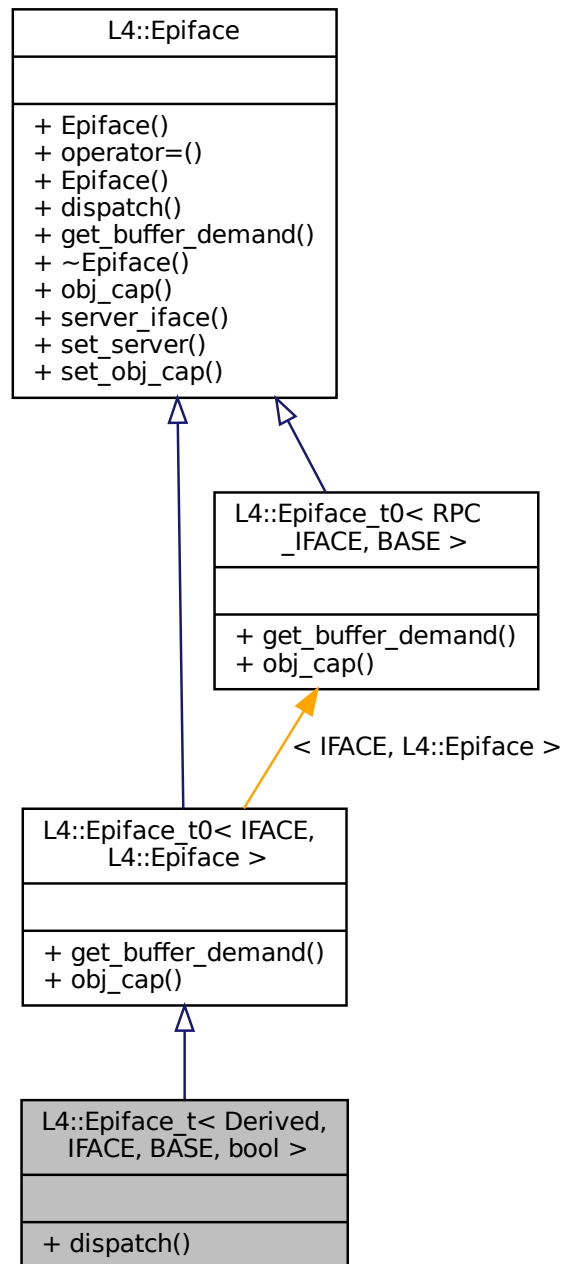
15.73 L4::Epiface_t< Derived, IFACE, BASE, bool > Struct Template Reference

[Epiface](#) implementation for Kobject-based interface implementations.

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.

Additional Inherited Members

15.73.1 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 515 of file [ipc_epiface](#).

15.73.2 Member Function Documentation

15.73.2.1 dispatch()

```
template<typename Derived , typename IFACE , typename BASE = L4::Epiface, bool = cxx::is_↵
polymorphic<BASE>::value>
l4_msgtag_t L4::Epiface_t< Derived, IFACE, 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 518 of file [ipc_epiface](#).

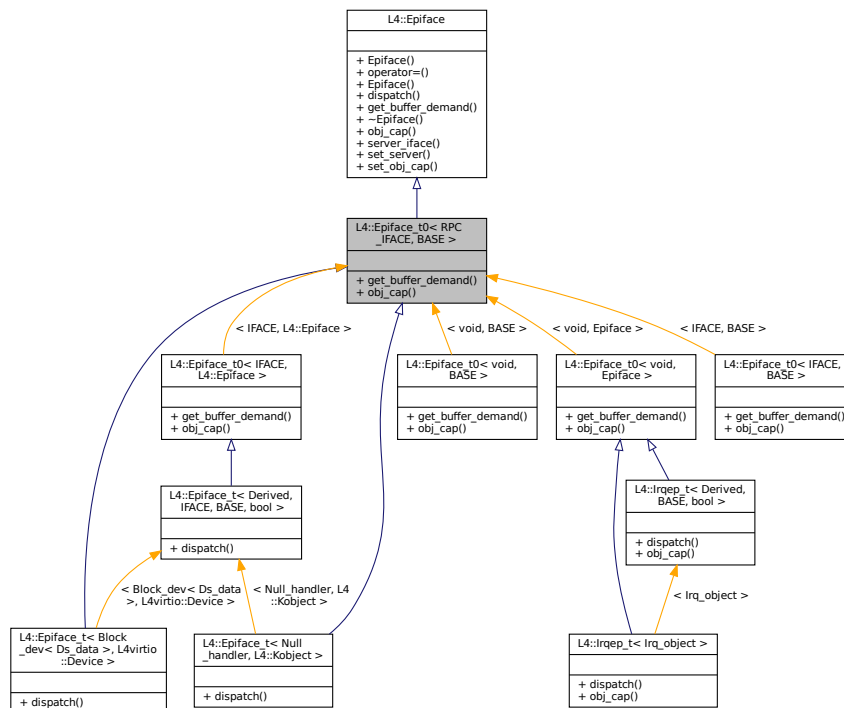
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_epiface`

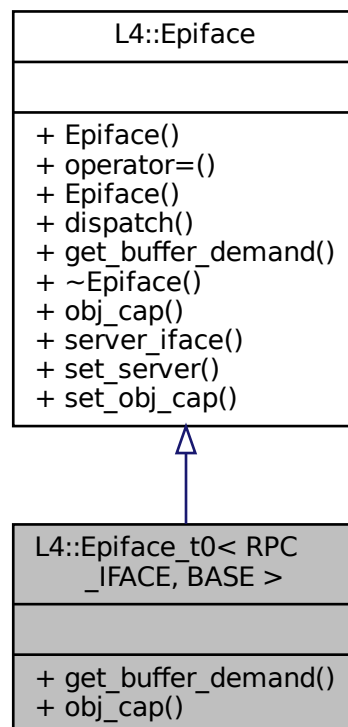
15.74 L4::Epiface_t0< RPC_IFACE, BASE > Struct Template Reference

[Epiface](#) mixin for generic Kobject-based interfaces.

Inheritance diagram for L4::Epiface_t0< RPC_IFACE, BASE >:



Collaboration diagram for L4::Epiface_t0< RPC_IFACE, BASE >:



Public Types

- `typedef RPC_IFACE Interface`
Data type of the IPC interface definition.

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.

15.74.1 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 [264](#) of file [ipc_epiface](#).

15.74.2 Member Function Documentation

15.74.2.1 `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 [277](#) of file [ipc_epiface](#).

The documentation for this struct was generated from the following file:

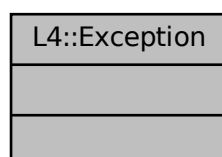
- `l4/sys/cxx/ipc_epiface`

15.75 L4::Exception Class Reference

[Exception](#) interface.

Inherits `L4::Kobject_0t< Derived, PROTO, S_DEMAND >`.

Collaboration diagram for L4::Exception:



Public Types

- typedef [L4::Typeid::Rpc_nocode](#)< exception_t > [Rpc](#)s
Exception call.

15.75.1 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](#).

15.75.2 Member Typedef Documentation

15.75.2.1 Rpc

typedef [L4::Typeid::Rpc_nocode](#)<exception_t> [L4::Exception::Rpc](#)s

[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.

Definition at line 62 of file [exception](#).

The documentation for this class was generated from the following file:

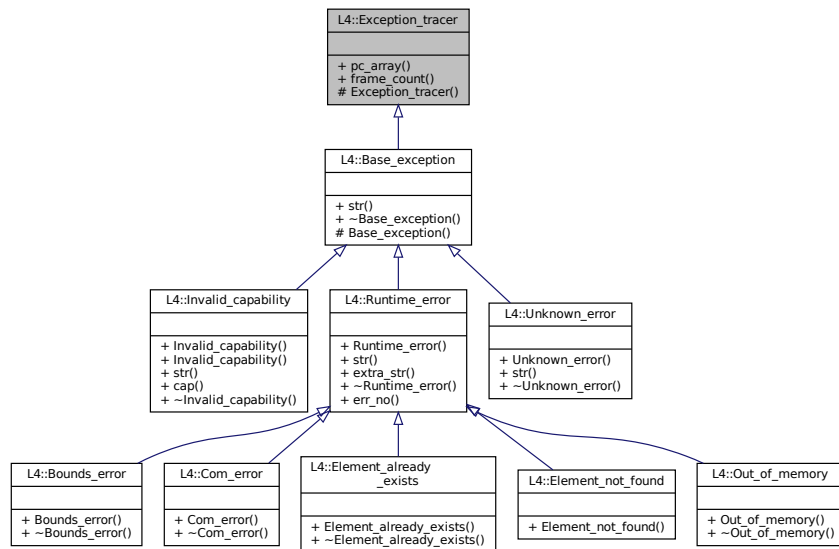
- [l4/sys/exception](#)

15.76 L4::Exception_tracer Class Reference

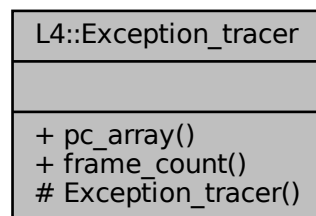
Back-trace support for exceptions.

```
#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 throw ()
Get the array containing the call trace.
- int [frame_count](#) () const throw ()
Get the number of entries that are valid in the call trace.

Protected Member Functions

- [Exception_tracer](#) () throw ()
Create a back trace.

15.76.1 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](#).

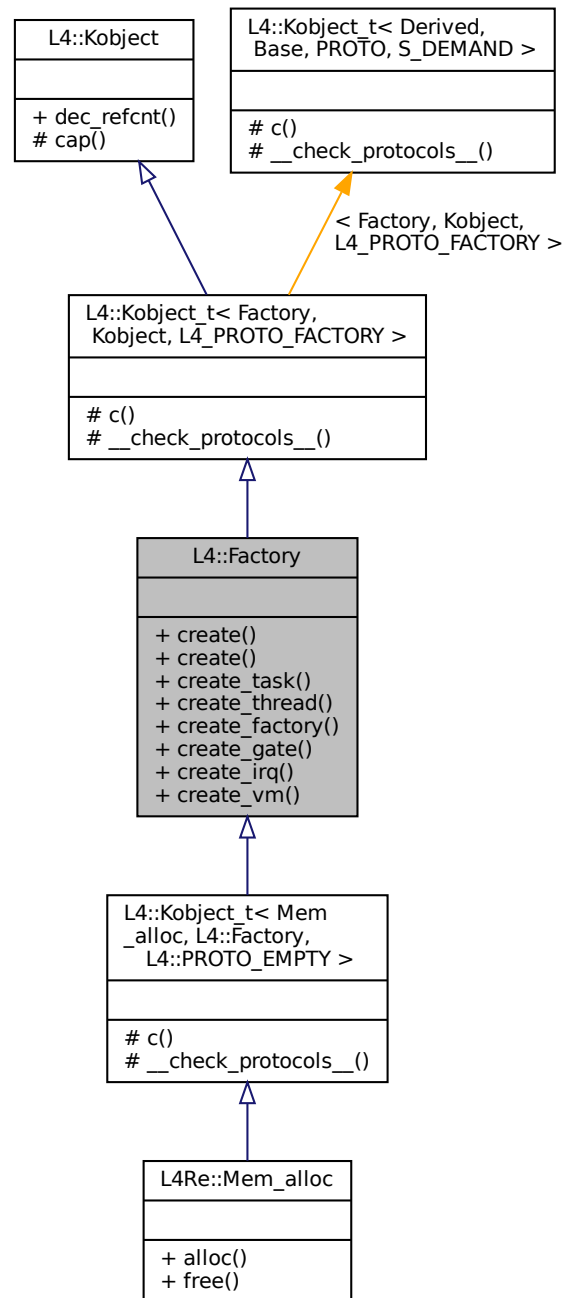
The documentation for this class was generated from the following file:

- [l4/cxx/exceptions](#)

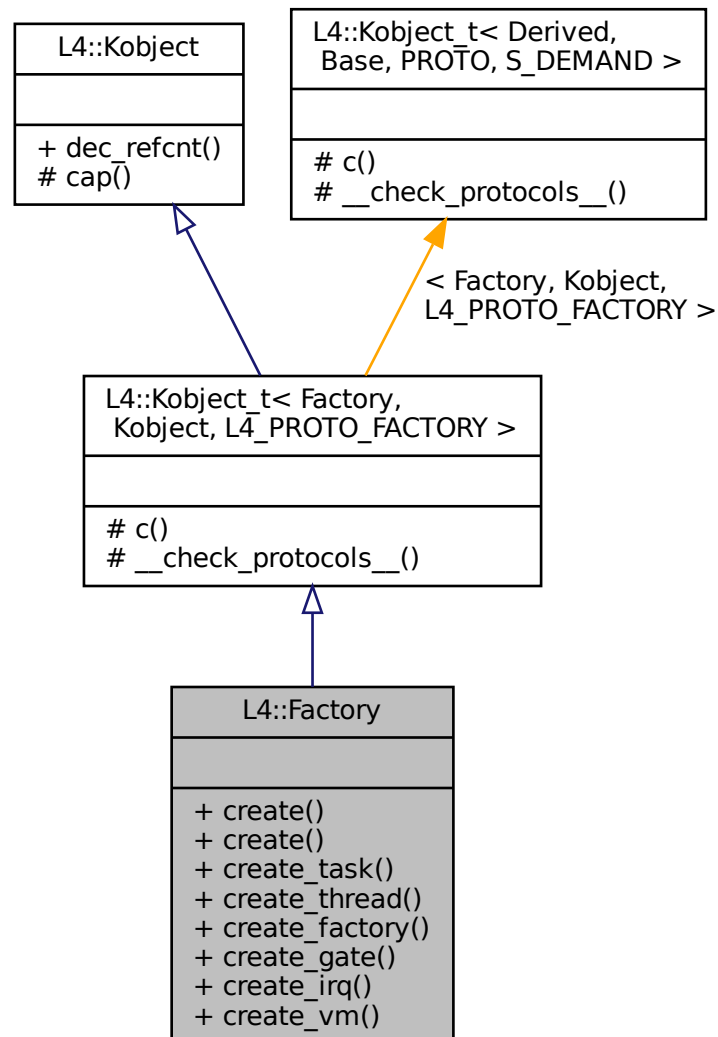
15.77 L4::Factory Class Reference

C++ Factory interface.

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_thread (Cap< Thread > const &target_cap, l4_utcb_t *utcb=l4_utcb()) noexcept`
Create a new thread.
- `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.
- `l4_msgtag_t create_irq (Cap< Irq > const &target_cap, l4_utcb_t *utcb=l4_utcb()) throw ()`
Create a new IRQ.
- `l4_msgtag_t create_vm (Cap< Vm > const &target_cap, l4_utcb_t *utcb=l4_utcb()) noexcept`
Create a new virtual machine.

Additional Inherited Members

15.77.1 Detailed Description

C++ Factory 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](#).

15.77.2 Member Function Documentation

15.77.2.1 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

<i>out</i>	<i>target</i>	Capability of type OBJ.
	<i>utcb</i>	UTCB to use.

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

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:



15.77.2.2 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>	The UTCB to use for the operation.

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

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 277 of file [factory](#).

References [L4::Kobject::cap\(\)](#).

Referenced by [L4Re::Mem_alloc::alloc\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.77.2.3 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>	The UTCB to use for the operation.

Returns

Syscall return tag

Note

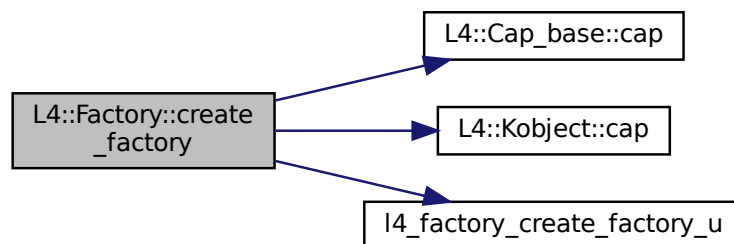
The `limit` (quota) of the new factory is subtracted from the limit of the factory invoked on its creation.

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 383 of file [factory](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4_factory_create_factory_u\(\)](#).

Here is the call graph for this function:



15.77.2.4 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 the thread to bind the gate to. Use L4_INVALID_CAP to create an unbound IPC gate.
	<i>label</i>	Optional label of the gate (is used if <i>thread_cap</i> is valid).
	<i>utcb</i>	The UTCB to use for the operation.

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 lpc_gate object.
<i>-L4_ENOENT</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>thread_cap</i> .

An unbound IPC gate can be bound to a thread using [L4::lpc_gate::bind_thread\(\)](#).

Note

This method is only guaranteed to work with the [Kernel Factory](#).

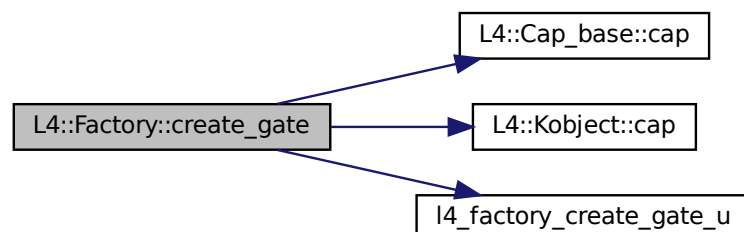
See also

[L4::lpc_gate](#)

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

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4_factory_create_gate_u\(\)](#).

Here is the call graph for this function:



15.77.2.5 create_irq()

```
l4_msgtag_t L4::Factory::create_irq (
    Cap< Irq >const & target_cap,
    l4_utcb_t * utcb = l4_utcb() ) throw ( )    [inline]
```

Create a new IRQ.

Parameters

out	<i>target_cap</i>	The kernel stores the new IRQ's capability into this slot.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

Deprecated Use `create()` with `Cap<Irq>` as argument instead.

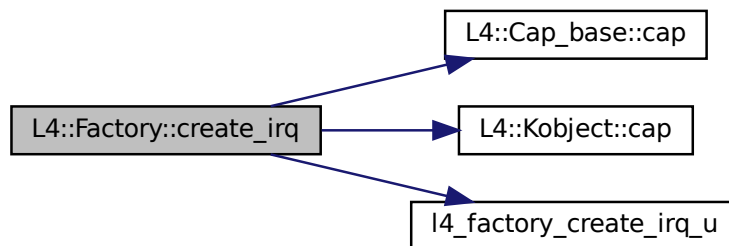
See also

[L4::Irq](#)

Definition at line 434 of file [factory](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4_factory_create_irq_u\(\)](#).

Here is the call graph for this function:



15.77.2.6 create_task()

```
l4_msgtag_t L4::Factory::create_task (
    Cap< Task > const & target_cap,
    l4_fpage_t const & utcb_area,
    l4_utcb_t * utcb = l4_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>	The UTCB to use for the operation.

Returns

Syscall return tag

Note

The size of the UTCB area specifies indirectly the number of UTCBs available for this task. Refer to [L4::Task::add_ku_mem](#) / [l4_task_add_ku_mem\(\)](#) for adding more of this type of memory.

This method is only guaranteed to work with the [Kernel Factory](#).

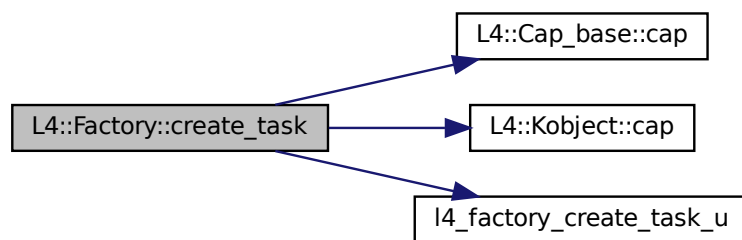
See also

[L4::Task](#)

Definition at line 341 of file [factory](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4_factory_create_task_u\(\)](#).

Here is the call graph for this function:



15.77.2.7 create_thread()

```

l4_msgtag_t L4::Factory::create_thread (
    Cap< Thread > const & target_cap,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]

```

Create a new thread.

Parameters

out	<i>target_cap</i>	The kernel stores the new thread's capability into this slot.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

Deprecated Use `create()` with `Cap<Thread>` as argument instead.

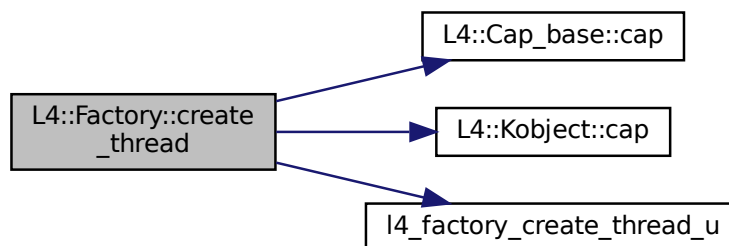
See also

[L4::Thread](#)

Definition at line 359 of file [factory](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4_factory_create_thread_u\(\)](#).

Here is the call graph for this function:



15.77.2.8 create_vm()

```

l4_msgtag_t L4::Factory::create_vm (
    Cap< Vm >const & target_cap,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
  
```

Create a new virtual machine.

Parameters

out	<i>target_cap</i>	The kernel stores the new VM's capability into this slot.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

Deprecated Use `create()` with `Cap<Vm>` as argument instead.

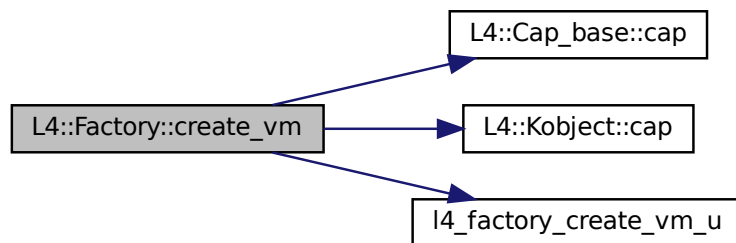
See also

[L4::Vm](#)

Definition at line 452 of file [factory](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4_factory_create_vm_u\(\)](#).

Here is the call graph for this function:



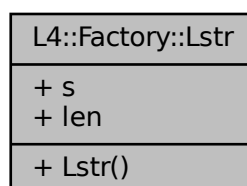
The documentation for this class was generated from the following file:

- [l4/sys/factory](#)

15.78 L4::Factory::Lstr Struct Reference

Special type to add a pascal string into the factory create stream.

Collaboration diagram for `L4::Factory::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.

15.78.1 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](#).

15.78.2 Constructor & Destructor Documentation

15.78.2.1 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](#).

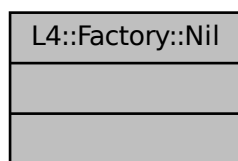
The documentation for this struct was generated from the following file:

- [I4/sys/factory](#)

15.79 L4::Factory::Nil Struct Reference

Special type to add a void argument into the factory create stream.

Collaboration diagram for L4::Factory::Nil:



15.79.1 Detailed Description

Special type to add a void argument into the factory create stream.

Definition at line 57 of file [factory](#).

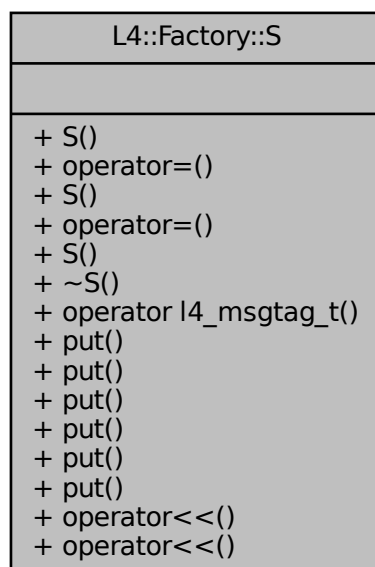
The documentation for this struct was generated from the following file:

- [l4/sys/factory](#)

15.80 L4::Factory::S Class Reference

Stream class for the [create\(\)](#) argument stream.

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.

15.80.1 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](#).

15.80.2 Constructor & Destructor Documentation

15.80.2.1 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](#).

15.80.2.2 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>	The UTCB to use for the operation.

Definition at line 132 of file [factory](#).

15.80.3 Member Function Documentation

15.80.3.1 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.

Definition at line 152 of file [factory](#).

15.80.3.2 put() [1/6]

```
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.
----------	-------------------------------------

Returns

Reference to this stream.

The string will be added with the zero-terminator.

Definition at line 192 of file [factory](#).

15.80.3.3 put() [2/6]

```
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).
----------	--

Returns

Reference to this stream.

Definition at line 230 of file [factory](#).

15.80.3.4 put() [3/6]

```
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.
----------	------------------------------------

Returns

Reference to this stream.

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

15.80.3.5 put() [4/6]

```
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.
----------	------------------------------------

Returns

Reference to this stream.

Definition at line 178 of file [factory](#).

15.80.3.6 put() [5/6]

```
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.
----------	-------------------------------------

Returns

Reference to this stream.

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.

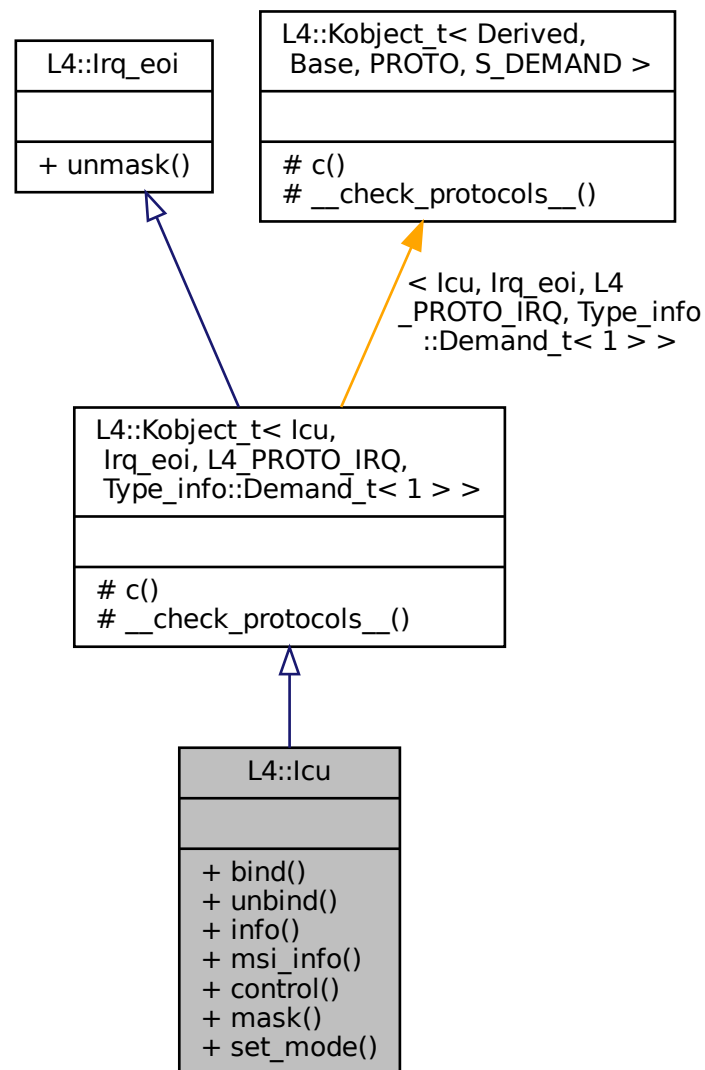
Definition at line 208 of file [factory](#).

15.80.3.7 put() [6/6]

```
void L4::Factory::S::put (
    Nil ) & [inline], [noexcept]
```

Add an empty argument.

Collaboration diagram for L4::lcu:



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 capabilities of the ICU.
- `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.

Additional Inherited Members

15.81.1 Detailed Description

C++ `l4cu` 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, 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 interrupt capability to a thread
3. `bind()`
4. `unmask()` to receive the first interrupt

Include File

```
#include <l4/sys/icu>
```

Definition at line 236 of file `irq`.

15.81.2 Member Function Documentation

15.81.2.1 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, usually the UTCB of the calling thread.

Returns

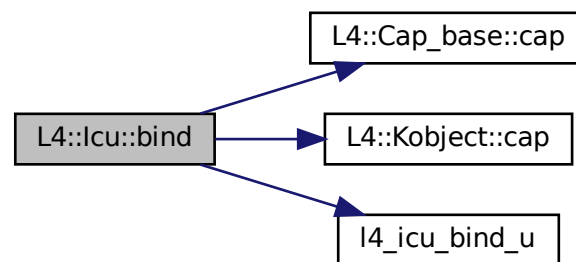
Syscall return tag. The caller should check the return value using [l4_error\(\)](#) to check for errors and to identify the correct method for unmasking the interrupt. Return values < 0 indicate an error. A return value of 0 means a direct unmask via the IRQ object using [L4::Icq::unmask](#). A return value of 1 means that the interrupt has to be unmasked via the ICU using [L4::Icu::unmask](#).

Definition at line 285 of file [irq](#).

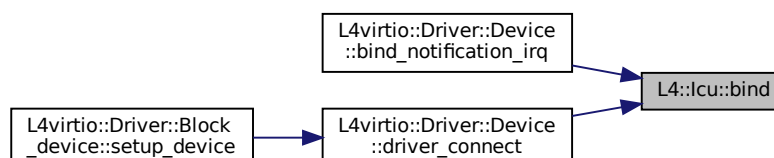
References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4_icu_bind_u\(\)](#).

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:



15.81.2.2 info()

```
l4_msgtag_t L4::Icu::info (
    l4_icu_info_t * info,
    l4_utcb_t * utcb = l4_utcb() )  [inline], [noexcept]
```

Get information about the capabilities of the ICU.

Parameters

<i>out</i>	<i>info</i>	Info structure to be filled with information.
	<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

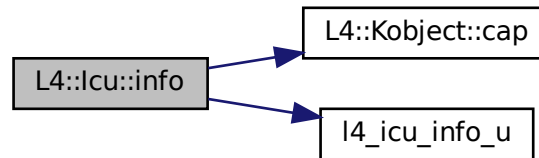
Returns

Syscall return tag

Definition at line 320 of file [irq](#).

References [L4::Kobject::cap\(\)](#), and [l4_icu_info_u\(\)](#).

Here is the call graph for this function:



15.81.2.3 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.
<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, usually the UTCB of the calling thread.

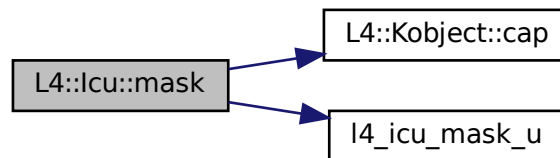
Returns

Syscall return tag

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

References [L4::Kobject::cap\(\)](#), and [l4_icu_mask_u\(\)](#).

Here is the call graph for this function:

**15.81.2.4 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

15.81.2.5 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, usually the UTCB of the calling thread.

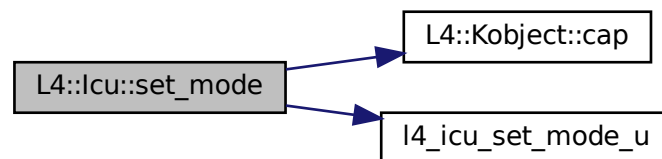
Returns

Syscall return tag

Definition at line 391 of file [irq](#).

References [L4::Kobject::cap\(\)](#), and [l4_icu_set_mode_u\(\)](#).

Here is the call graph for this function:



15.81.2.6 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, usually the UTCB of the calling thread.

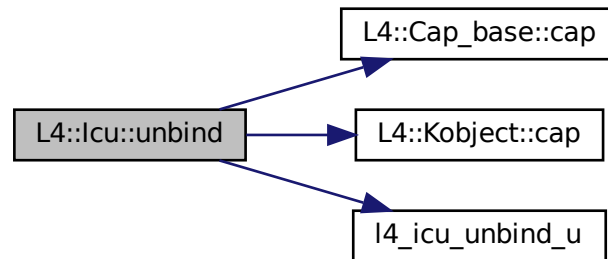
Returns

Syscall return tag

Definition at line 303 of file [irq](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4_icu_unbind_u\(\)](#).

Here is the call graph for this function:



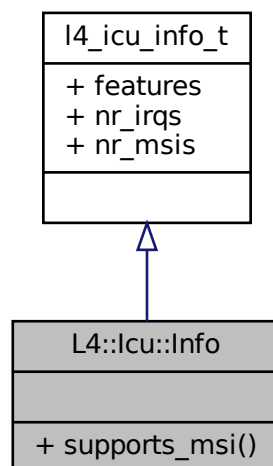
The documentation for this class was generated from the following file:

- [l4/sys/irq](#)

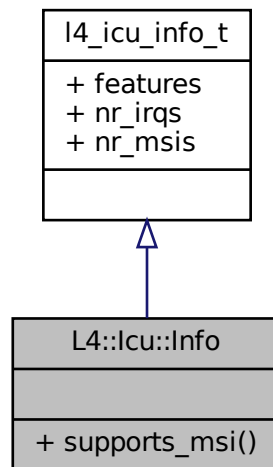
15.82 L4::Icu::Info Class Reference

This class encapsulates information about an ICU.

Inheritance diagram for `L4::Icu::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

15.82.1 Detailed Description

This class encapsulates information about an ICU.

Definition at line 263 of file [irq](#).

The documentation for this class was generated from the following file:

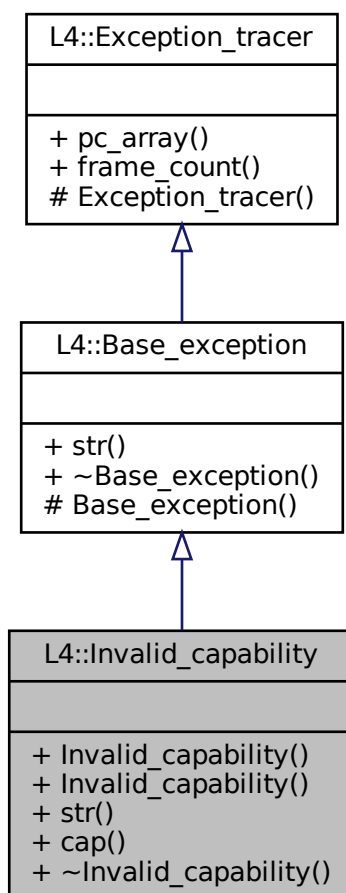
- [l4/sys/irq](#)

15.83 L4::Invalid_capability Class Reference

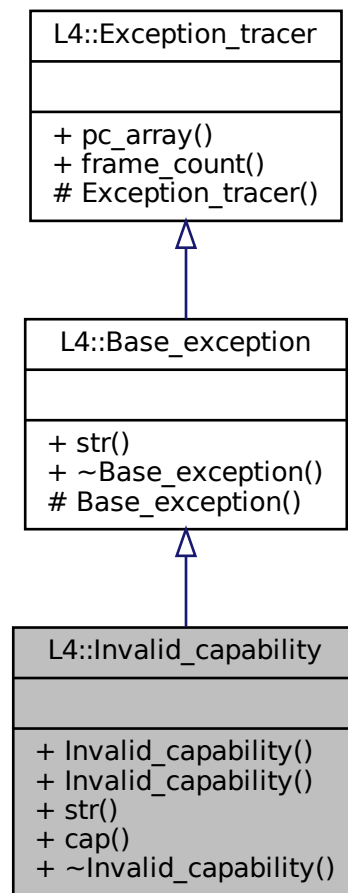
Indicates that an invalid object was invoked.

```
#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) throw ()
Create an *Invalid_object* exception for the Object o.
- char const * [str](#) () const throw ()
Return a human readable string for the exception.
- [Cap](#)< void > const & [cap](#) () const throw ()
Get the object that caused the error.

Additional Inherited Members

15.83.1 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](#).

15.83.2 Constructor & Destructor Documentation

15.83.2.1 Invalid_capability()

```
L4::Invalid_capability::Invalid_capability (
    Cap< void > const & o ) throw ( )    [inline], [explicit]
```

Create an Invalid_object exception for the Object o.

Parameters

<i>o</i>	The object that caused the server side error.
----------	---

Definition at line 255 of file [exceptions](#).

15.83.3 Member Function Documentation

15.83.3.1 cap()

```
Cap<void> const& L4::Invalid_capability::cap ( ) const throw ( )    [inline]
```

Get the object that caused the error.

Returns

The object that caused the error on invocation.

Definition at line 264 of file [exceptions](#).

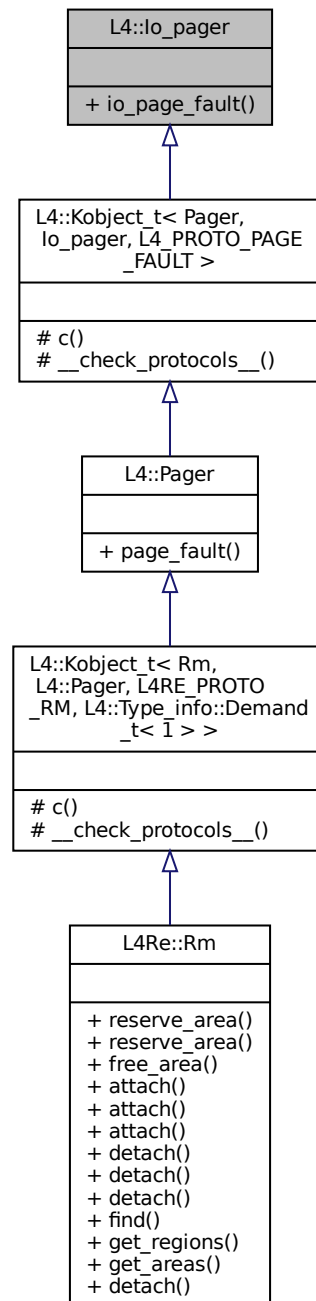
The documentation for this class was generated from the following file:

- [l4/cxx/exceptions](#)

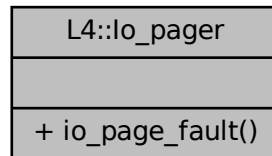
15.84 L4::lo_pager Class Reference

[lo_pager](#) interface.

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.

15.84.1 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](#).

15.84.2 Member Function Documentation

15.84.2.1 `io_page_fault()`

```
l4_msgtag_t L4::Io_pager::io_page_fault (
    l4_fpage_t io_pfa,
    l4_umword_t pc,
    L4::Ipc::Rcv_fpage rwin,
    L4::Ipc::Opt< L4::Ipc::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`.

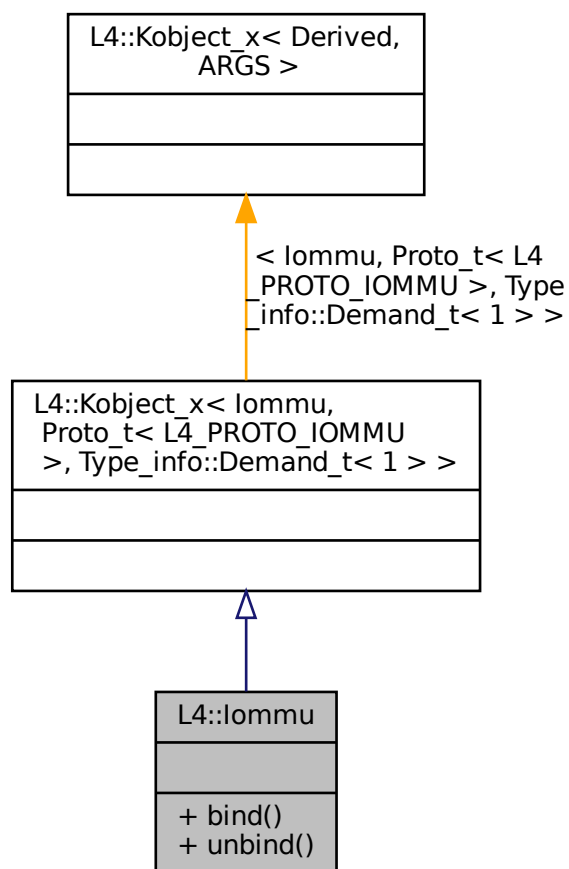
The documentation for this class was generated from the following file:

- `l4/sys/pager`

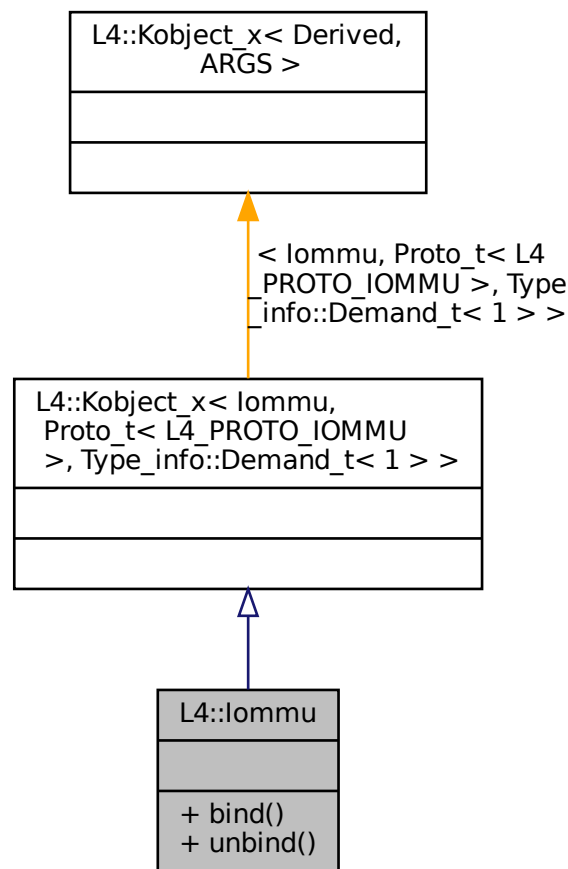
15.85 L4::lomu Class Reference

Interface for IO-MMUs used for DMA remapping.

Inheritance diagram for L4::lomu:



Collaboration diagram for L4::lomu:



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*.

15.85.1 Detailed Description

Interface for IO-MMUs used for DMA remapping.

This interface allows to associate a DMA address space with a platform dependent set of devices.

Definition at line 16 of file `lomu`.

15.85.2 Member Function Documentation

15.85.2.1 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`.

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).

15.85.2.2 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`.

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() .

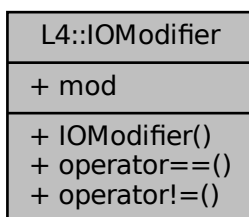
The documentation for this class was generated from the following file:

- `l4/sys/iommu`

15.86 L4::IOModifier Class Reference

Modifier class for the IO stream.

Collaboration diagram for L4::IOModifier:



15.86.1 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](#).

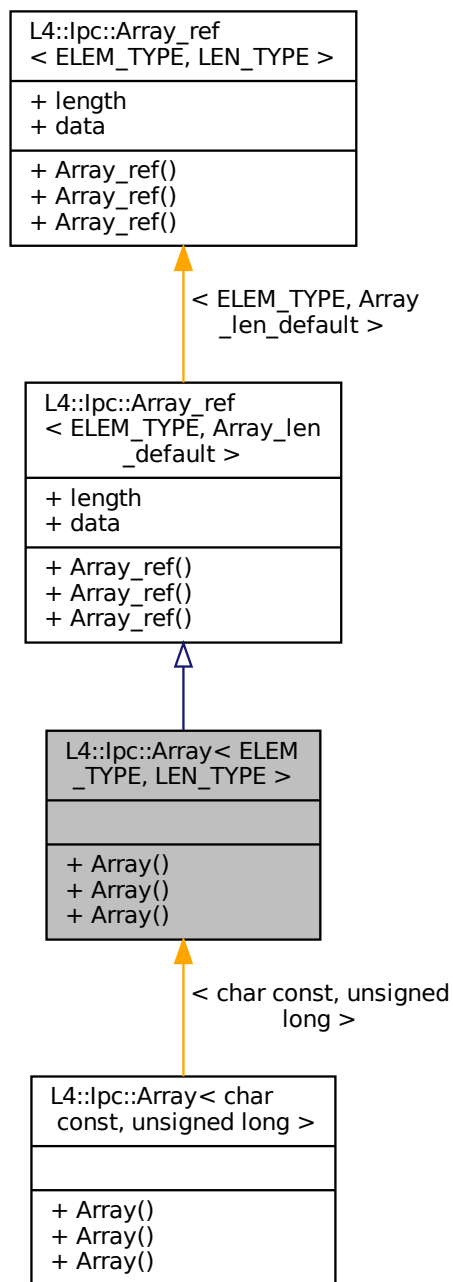
The documentation for this class was generated from the following file:

- I4/cxx/[basic_ostream](#)

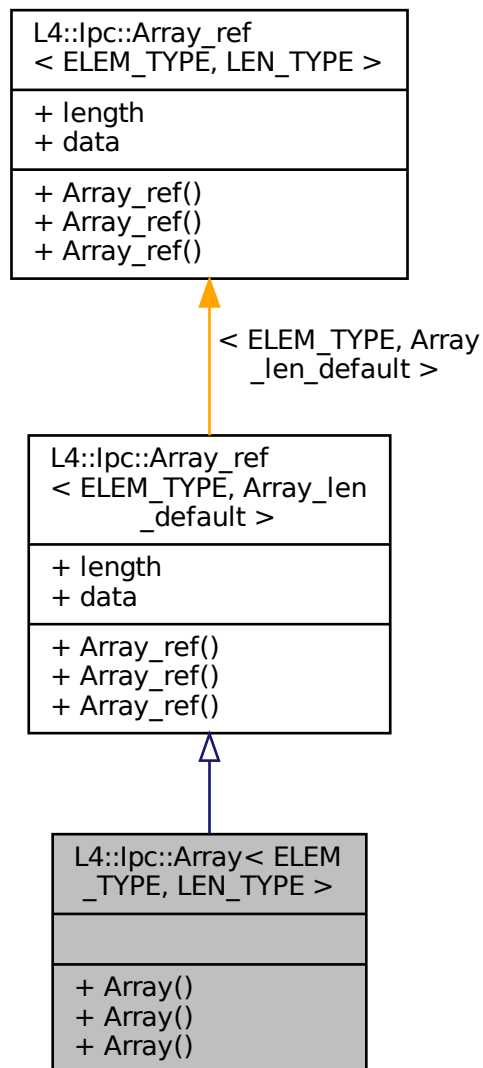
15.87 L4::lpc::Array< ELEM_TYPE, LEN_TYPE > Struct Template Reference

[Array](#) data type for dynamically sized arrays in RPCs.

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.

15.87.1 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 85 of file [ipc_array](#).

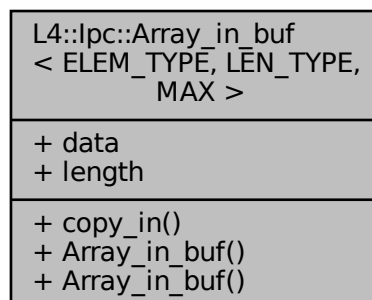
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_array`

15.88 L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX > Struct Template Reference

Server-side copy in buffer for [Array](#).

Collaboration diagram for `L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >`:



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.

15.88.1 Detailed Description

```
template<typename ELEM_TYPE, typename LEN_TYPE = Array_len_default, LEN_TYPE MAX = (L4_UTCB_GENERIC_DATA_SIZE
* sizeof(l4_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 123 of file [ipc_array](#).

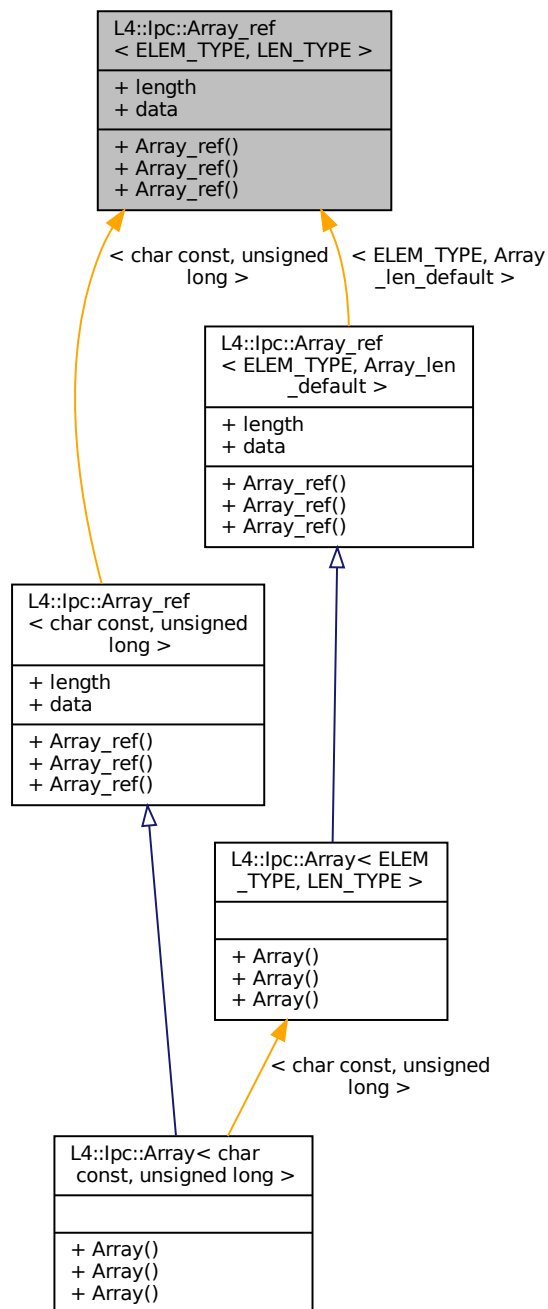
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_array`

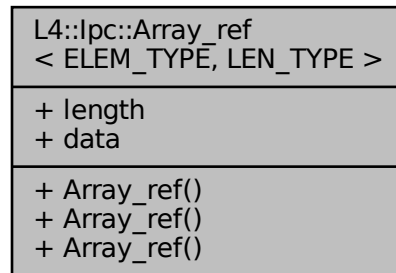
15.89 L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE > Struct Template Reference

[Array](#) reference data type for arrays located in the message.

Inheritance diagram for L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE >:



Collaboration diagram for L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE >:



15.89.1 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](#).

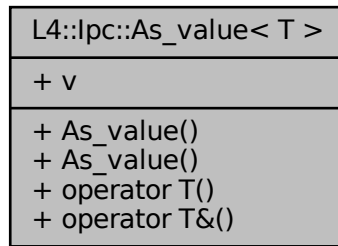
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_array

15.90 L4::lpc::As_value< T > Struct Template Reference

Pass the argument as plain data value.

Collaboration diagram for L4::lpc::As_value< T >:



15.90.1 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](#).

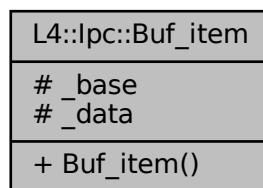
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_types](#)

15.91 L4::lpc::Buf_item Class Reference

RPC warpper for a receive item.

Collaboration diagram for L4::lpc::Buf_item:



15.91.1 Detailed Description

RPC warpper for a receive item.

Definition at line 305 of file [ipc_types](#).

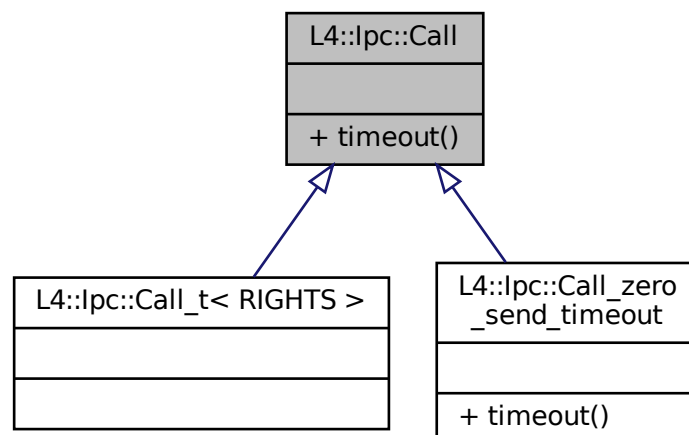
The documentation for this class was generated from the following file:

- [l4/sys/cxx/ipc_types](#)

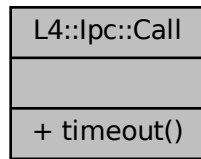
15.92 L4::lpc::Call Struct Reference

RPC attribute for a standard RPC call.

Inheritance diagram for L4::lpc::Call:



Collaboration diagram for L4::lpc::Call:



15.92.1 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::lpc::Call);
```

which is equivalent to:

```
L4_RPC(long, send, (unsigned value));
```

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

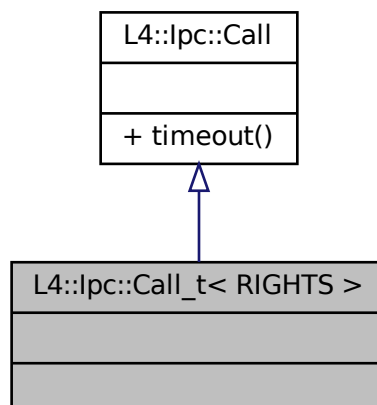
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_iface](#)

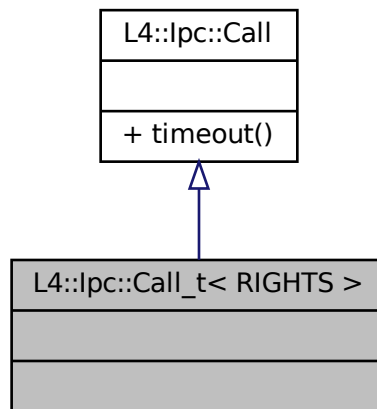
15.93 L4::lpc::Call_t< RIGHTS > Struct Template Reference

RPC attribute for an RPC call with required rights.

Inheritance diagram for L4::lpc::Call_t< RIGHTS >:



Collaboration diagram for L4::lpc::Call_t< RIGHTS >:



15.93.1 Detailed Description

```
template<unsigned RIGHTS>
struct L4::lpc::Call_t< RIGHTS >
```

RPC attribute for an RPC call with required rights.

Template Parameters

<i>RIGHTS</i>	The capability rights required for this call. 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::lpc::Call_t<L4_CAP_FPAGE_RW>);
```

Definition at line 257 of file [ipc_iface](#).

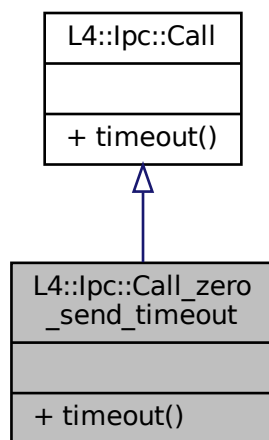
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_iface](#)

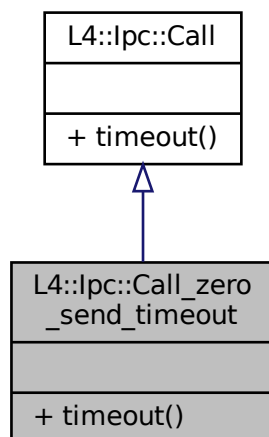
15.94 L4::lpc::Call_zero_send_timeout Struct Reference

RPC attribute for an RPC call, with zero send timeout.

Inheritance diagram for L4::lpc::Call_zero_send_timeout:



Collaboration diagram for L4::lpc::Call_zero_send_timeout:



15.94.1 Detailed Description

RPC attribute for an RPC call, with zero send timeout.

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

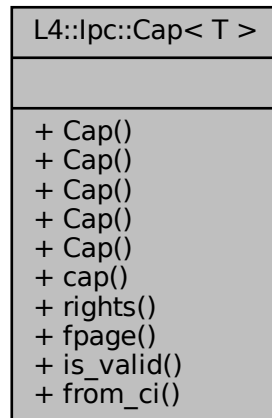
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_iface](#)

15.95 L4::lpc::Cap< T > Class Template Reference

Capability type for RPC interfaces (see [L4 : :Cap<T>](#)).

Collaboration diagram for L4::lpc::Cap< T >:



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.

15.95.1 Detailed Description

```
template<typename T>
class L4::lpc::Cap< T >
```

Capability type for RPC interfaces (see [L4::Cap<T>](#)).

Template Parameters

T	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 [541](#) of file [ipc_types](#).

15.95.2 Member Enumeration Documentation

15.95.2.1 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 [547](#) of file [ipc_types](#).

15.95.3 Constructor & Destructor Documentation

15.95.3.1 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 589 of file [ipc_types](#).

15.95.4 Member Function Documentation

15.95.4.1 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 597 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:

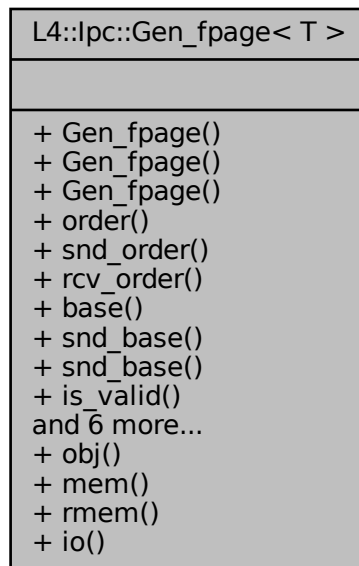
- [l4/sys/cxx/ipc_types](#)

15.96 L4::lpc::Gen_fpage< T > Class Template Reference

Generic RPC wrapper for [L4](#) flex-pages.

Inherits T.

Collaboration diagram for L4::lpc::Gen_fpage< T >:



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 the 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.

15.96.1 Detailed Description

```
template<typename T>
class L4::Ipc::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](#).

15.96.2 Member Function Documentation

15.96.2.1 cap_received()

```
template<typename T >
bool L4::Ipc::Gen_fpage< T >::cap_received ( ) const [inline], [noexcept]
```

Check if the capability has been mapped.

The capability itself can then be retrieved from the cap slot that has been provided in the receive operation.

Definition at line [438](#) of file [ipc_types](#).

15.96.2.2 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 [450](#) of file [ipc_types](#).

15.96.2.3 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 468 of file [ipc_types](#).

15.96.2.4 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 460 of file [ipc_types](#).

The documentation for this class was generated from the following file:

- [l4/sys/cxx/ipc_types](#)

15.97 L4::lpc::In_out< T > Struct Template Reference

Mark an argument as in-out argument.

Collaboration diagram for L4::lpc::In_out< T >:

L4::lpc::In_out< T >
+ v
+ In_out() + In_out() + operator T() + operator T&()

15.97.1 Detailed Description

```
template<typename T>
struct L4::lpc::In_out< T >
```

Mark an argument as in-out argument.

Template Parameters

<i>T</i>	The original argument type, usually a pointer or a reference.
----------	---

In_out<> is used when an otherwise output-only value shall also be used as input value.

Definition at line 52 of file [ipc_types](#).

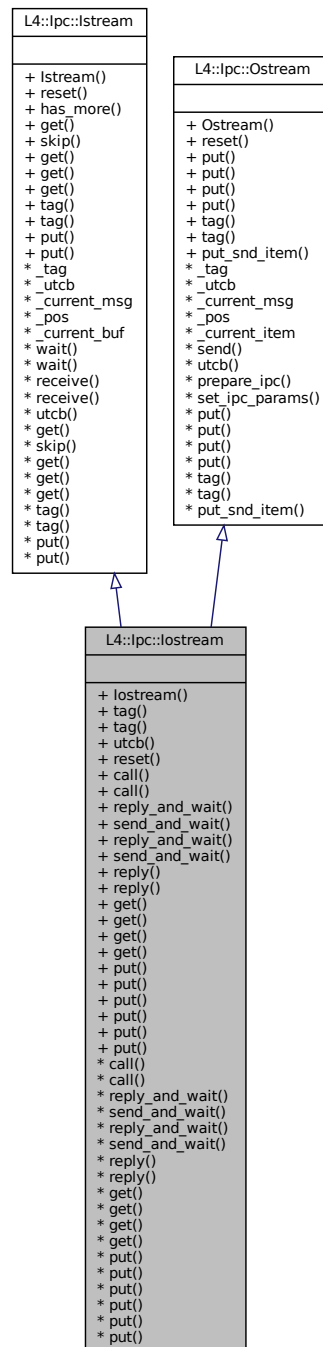
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_types](#)

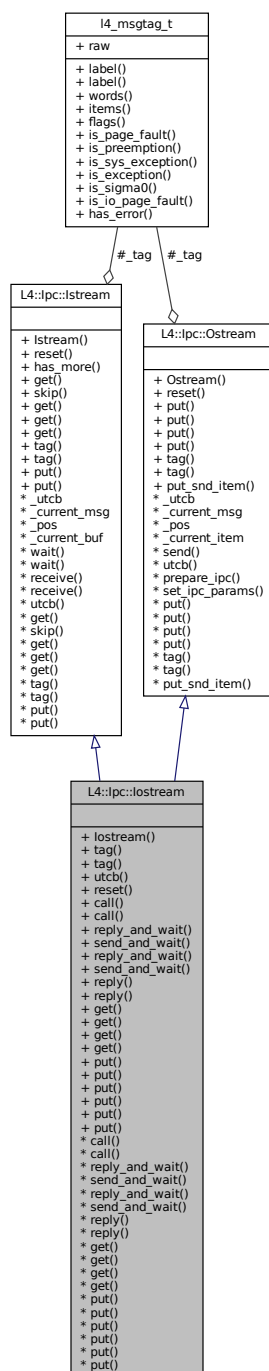
15.98 L4::lpc::lostream Class Reference

Input/Output stream for IPC [un]marshalling.

Inheritance diagram for L4::lpc::lostream:



Collaboration diagram for L4::lpc::loststream:



Public Member Functions

- `loststream (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 call` (`l4_cap_idx_t` dst, long proto=0)
- `l4_msgtag_t reply_and_wait` (`l4_umword_t` *src_dst, long proto=0)
Do an IPC reply and wait.
- `l4_msgtag_t send_and_wait` (`l4_cap_idx_t` dest, `l4_umword_t` *src, long proto=0)
- `l4_msgtag_t reply_and_wait` (`l4_umword_t` *src_dst, `l4_timeout_t` timeout, long proto=0)
Do an IPC reply and wait.
- `l4_msgtag_t send_and_wait` (`l4_cap_idx_t` dest, `l4_umword_t` *src, `l4_timeout_t` timeout, long proto=0)
- `l4_msgtag_t reply` (`l4_timeout_t` timeout, long proto=0)
- `l4_msgtag_t reply` (long proto=0)

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.
- `bool get` (`lpc::Varg` *va)
- `bool put` (`Buf_item` const &)
- `bool put` (`Small_buf` const &)
- `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.
- `int put` (`Varg` const &va)
- `template<typename T >`
`int put` (`Varg_t< T >` const &va)

Additional Inherited Members**15.98.1 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 802 of file `ipc_stream`.

15.98.2 Constructor & Destructor Documentation

15.98.2.1 Iostream()

```
L4::Ipc::Iostream::Iostream (
    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 814 of file [ipc_stream](#).

15.98.3 Member Function Documentation

15.98.3.1 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.

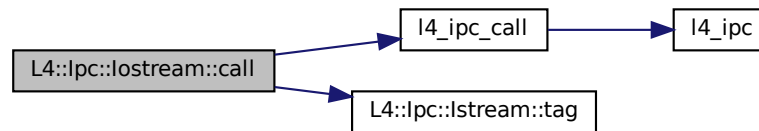
Examples

[examples/libs/l4re/streammap/client.cc](#).

Definition at line 973 of file [ipc_stream](#).

References [l4_ipc_call\(\)](#), and [L4::ipc::Istream::tag\(\)](#).

Here is the call graph for this function:



15.98.3.2 `get()` [1/3]

```
template<typename T >
unsigned long L4::Ipc::Istream::get (
    typename T ) [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 452 of file [ipc_stream](#).

15.98.3.3 `get()` [2/3]

```
template<typename T >
bool L4::Ipc::Istream::get (
    typename T ) [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 [lstream::operator>>\(\)](#)

Definition at line 477 of file [ipc_stream](#).

15.98.3.4 get() [3/3]

```
template<typename T >
unsigned long L4::Ipc::Istream::get (
    typename T ) [inline]
```

Copy out an array of type T with *size* elements.

Parameters

<i>buf</i>	Pointer to a buffer for <i>size</i> elements of type T.
<i>elems</i>	Number of elements of type T to copy out.

Returns

The number of elements copied out.

See [lstream::operator>>\(\)](#)

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

15.98.3.5 put() [1/2]

```
template<typename T >
bool L4::Ipc::Ostream::put (
    typename T ) [inline]
```

Put an array with *size* elements of type T into the stream.

Parameters

<i>buf</i>	A pointer to the array to insert into the buffer.
<i>size</i>	The number of elements in the array.

Definition at line 671 of file [ipc_stream](#).

15.98.3.6 put() [2/2]

```
template<typename T >
bool L4::IpC::Ostream::put (
    typename T ) [inline]
```

Insert an element of type `T` into the stream.

Parameters

<code>v</code>	The element to insert.
----------------	------------------------

Definition at line 689 of file [ipc_stream](#).

15.98.3.7 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.

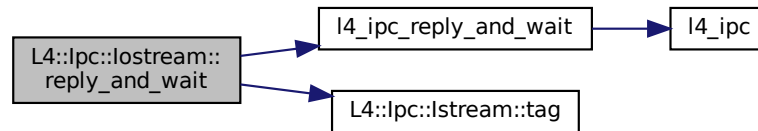
This is a combined IPC operation consisting of a send operation and an open wait for any message.

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 988 of file [ipc_stream](#).

References [l4_ipc_reply_and_wait\(\)](#), and [L4::IpC::Istream::tag\(\)](#).

Here is the call graph for this function:



15.98.3.8 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

<i>in, out</i>	<i>src_dst</i>	Input: the destination for the send operation. Output: the source of the received message.
	<i>proto</i>	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 887 of file [ipc_stream](#).

15.98.3.9 reset()

```

void L4::Ipc::Iostream::reset ( ) [inline]
  
```

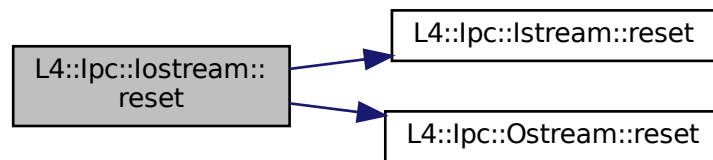
Reset the stream to its initial state.

Input as well as the output stream are reset.

Definition at line 828 of file [ipc_stream](#).

References [L4::lpc::Istream::reset\(\)](#), and [L4::lpc::Ostream::reset\(\)](#).

Here is the call graph for this function:



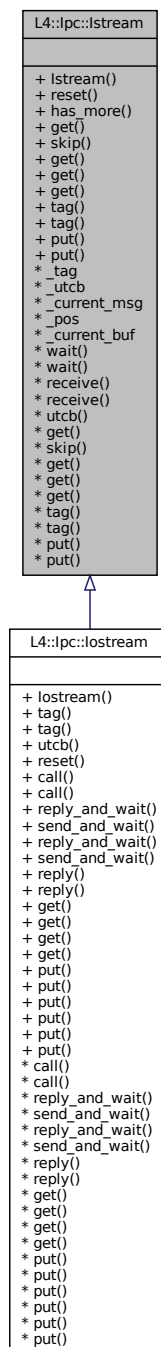
The documentation for this class was generated from the following file:

- [l4/cxx/ipc_stream](#)

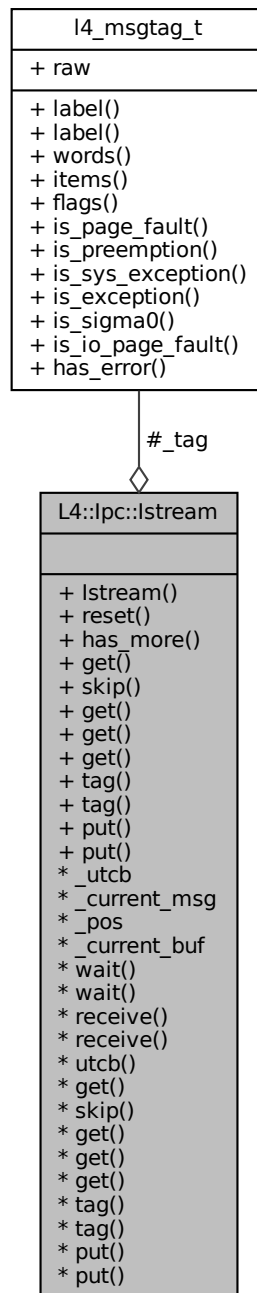
15.99 L4::lpc::Istream Class Reference

Input stream for IPC unmarshalling.

Inheritance diagram for L4::lpc::Istream:



Collaboration diagram for L4::lpc::lstream:



Public Member Functions

- `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.

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.
- `bool get` (lpc::Varg *va)
- `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.
- `bool put` (Buf_item const &)
- `bool put` (Small_buf const &)

IPC operations.

- `l4_msgtag_t _tag`
- `l4_utcb_t * _utcb`
- `char * _current_msg`
- `unsigned _pos`
- `unsigned char _current_buf`
- `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.
- `l4_msgtag_t receive` (l4_cap_idx_t src, l4_timeout_t timeout)
- `l4_utcb_t * utcb` () const
Return utcb pointer.

15.99.1 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 347 of file `ipc_stream`.

15.99.2 Constructor & Destructor Documentation

15.99.2.1 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 [361](#) of file [ipc_stream](#).

15.99.3 Member Function Documentation

15.99.3.1 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 [452](#) of file [ipc_stream](#).

References [L4_UNLIKELY](#).

15.99.3.2 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.
<i>false</i>	An element could not be extracted.

See [Istream::operator>>\(\)](#)

Definition at line 477 of file [ipc_stream](#).

References [L4_UNLIKELY](#).

15.99.3.3 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 <i>size</i> elements of type T.
<i>elems</i>	Number of elements of type T to copy out.

Returns

The number of elements copied out.

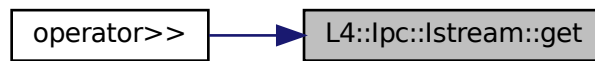
See [Istream::operator>>\(\)](#)

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

References [L4_UNLIKELY](#).

Referenced by [operator>>\(\)](#).

Here is the caller graph for this function:



15.99.3.4 receive()

```
l4_msgtag_t L4::Ipc::Istream::receive (
    l4_cap_idx_t src ) [inline]
```

Wait for a message from the specified sender.

Parameters

<i>src</i>	The sender id to receive from.
------------	--------------------------------

Returns

The IPC result tag ([l4_msgtag_t](#)).

This is commonly known as 'closed wait'.

Definition at line [585](#) of file [ipc_stream](#).

15.99.3.5 reset()

```
void L4::Ipc::Istream::reset ( ) [inline]
```

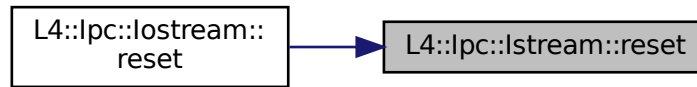
Reset the stream to empty, and ready for [receive\(\)](#)/[wait\(\)](#).

The stream is reset to the same state as on its creation.

Definition at line [371](#) of file [ipc_stream](#).

Referenced by [L4::Ipc::Istream::reset\(\)](#).

Here is the caller graph for this function:



15.99.3.6 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 427 of file [ipc_stream](#).

References [L4_UNLIKELY](#).

15.99.3.7 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 530 of file [ipc_stream](#).

15.99.3.8 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.

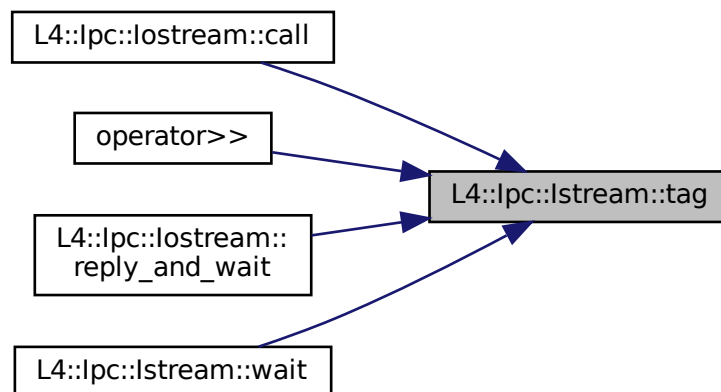
This is in particular useful for handling page faults or exceptions.

See [Istream::operator>>\(\)](#)

Definition at line [518](#) 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:

**15.99.3.9 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 561 of file [ipc_stream](#).

15.99.3.10 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

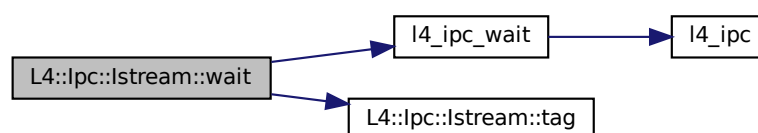
The IPC result tag ([l4_msgtag_t](#)).

This wait is actually known as 'open wait'.

Definition at line 1020 of file [ipc_stream](#).

References [l4_ipc_wait\(\)](#), and [tag\(\)](#).

Here is the call graph for this function:



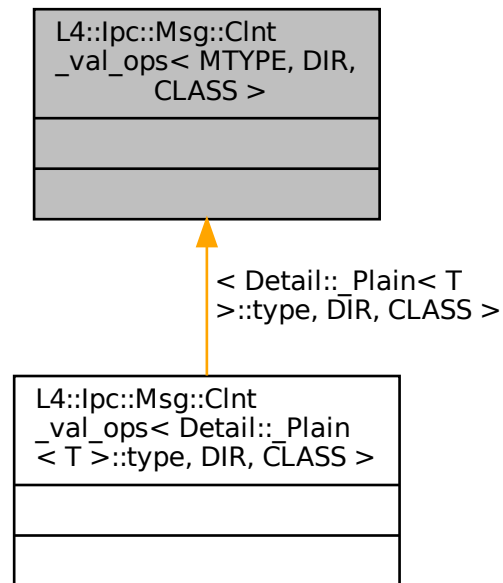
The documentation for this class was generated from the following file:

- [l4/cxx/ipc_stream](#)

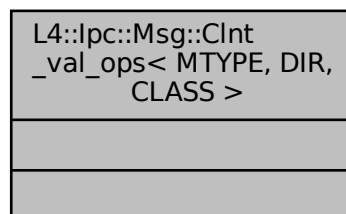
15.100 L4::lpc::Msg::Cnt_val_ops< MTYPE, DIR, CLASS > Struct Template Reference

Defines client-side handling of 'MTYPE' as RPC argument.

Inheritance diagram for L4::lpc::Msg::Cnt_val_ops< MTYPE, DIR, CLASS >:



Collaboration diagram for L4::lpc::Msg::Cnt_val_ops< MTYPE, DIR, CLASS >:



15.100.1 Detailed Description

```
template<typename MTYPE, typename DIR, typename CLASS>  
struct L4::lpc::Msg::Cint_val_ops< MTYPE, DIR, CLASS >
```

Defines client-side handling of 'MTYPE' as RPC argument.

Template Parameters

<i>MTYPE</i>	Elem<T>::arg_type (where T is the type used in the RPC definition)
<i>DIR</i>	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](#).

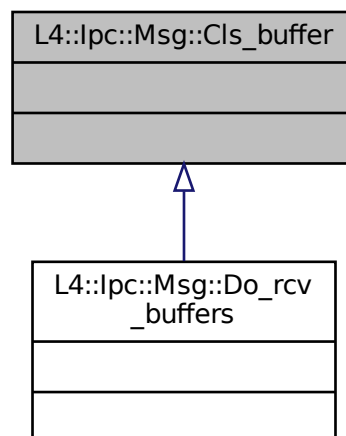
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_basics

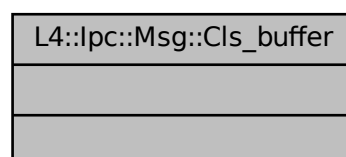
15.101 L4::lpc::Msg::Cls_buffer Struct Reference

Marker type for receive buffer values.

Inheritance diagram for L4::lpc::Msg::Cls_buffer:



Collaboration diagram for L4::lpc::Msg::Cls_buffer:



15.101.1 Detailed Description

Marker type for receive buffer values.

Definition at line 165 of file [ipc_basics](#).

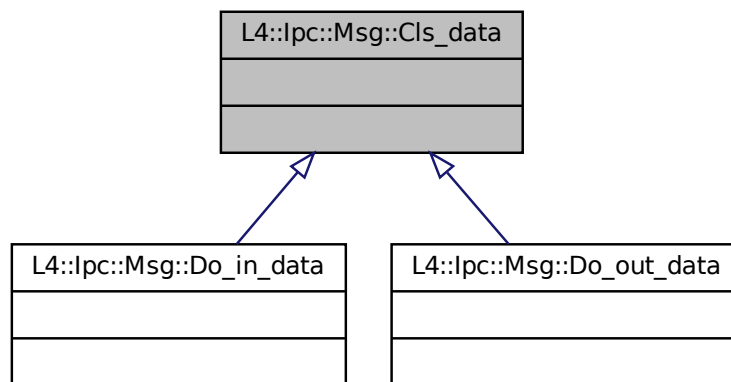
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_basics

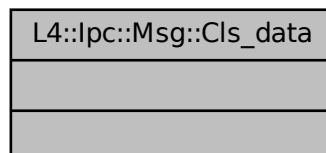
15.102 L4::lpc::Msg::Cls_data Struct Reference

Marker type for data values.

Inheritance diagram for L4::lpc::Msg::Cls_data:



Collaboration diagram for L4::lpc::Msg::Cls_data:



15.102.1 Detailed Description

Marker type for data values.

Definition at line 161 of file [ipc_basics](#).

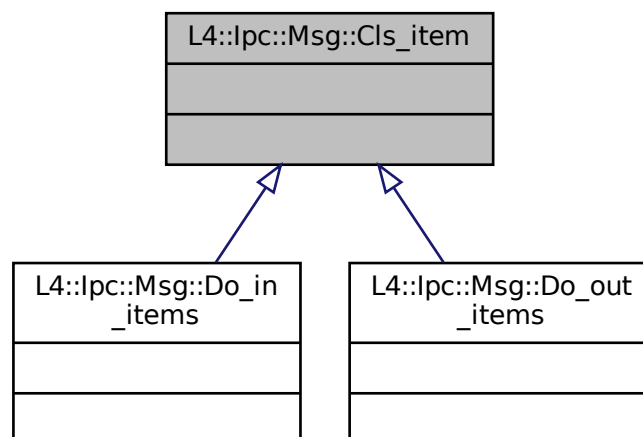
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_basics

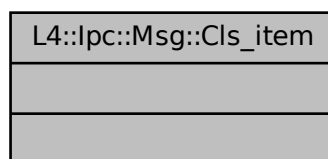
15.103 L4::lpc::Msg::Cls_item Struct Reference

Marker type for item values.

Inheritance diagram for L4::lpc::Msg::Cls_item:



Collaboration diagram for L4::lpc::Msg::Cls_item:



15.103.1 Detailed Description

Marker type for item values.

Definition at line 163 of file [ipc_basics](#).

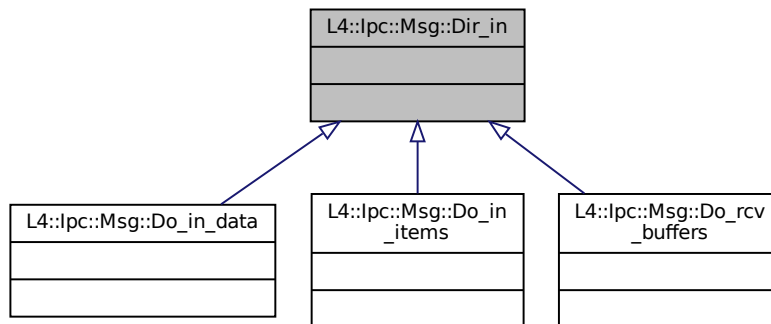
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_basics

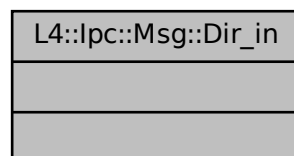
15.104 L4::lpc::Msg::Dir_in Struct Reference

Marker type for input values.

Inheritance diagram for L4::lpc::Msg::Dir_in:



Collaboration diagram for L4::lpc::Msg::Dir_in:



15.104.1 Detailed Description

Marker type for input values.

Definition at line 156 of file [ipc_basics](#).

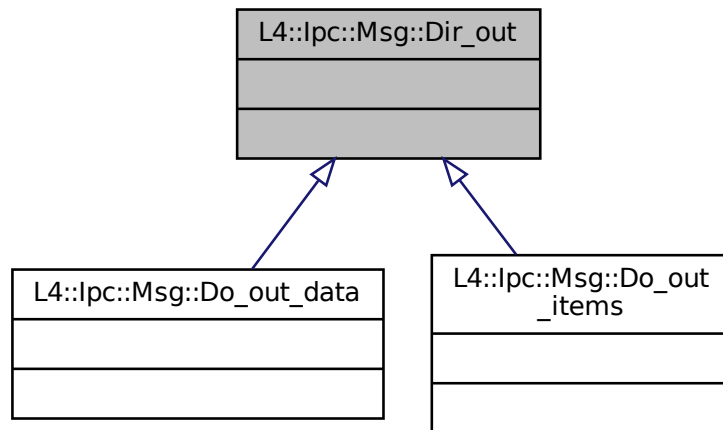
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_basics

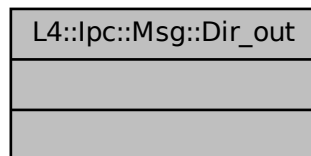
15.105 L4::lpc::Msg::Dir_out Struct Reference

Marker type for output values.

Inheritance diagram for L4::lpc::Msg::Dir_out:



Collaboration diagram for L4::lpc::Msg::Dir_out:



15.105.1 Detailed Description

Marker type for output values.

Definition at line 158 of file [ipc_basics](#).

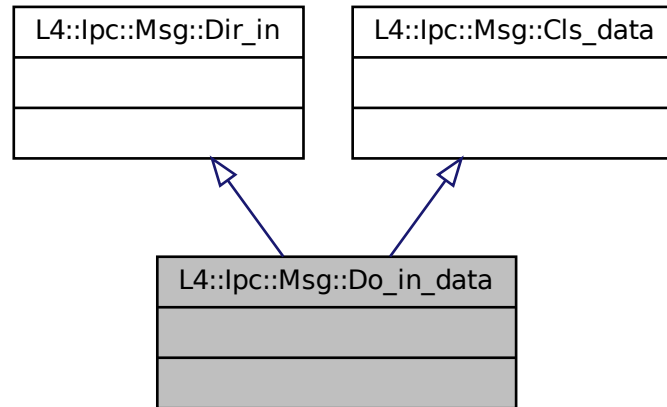
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_basics`

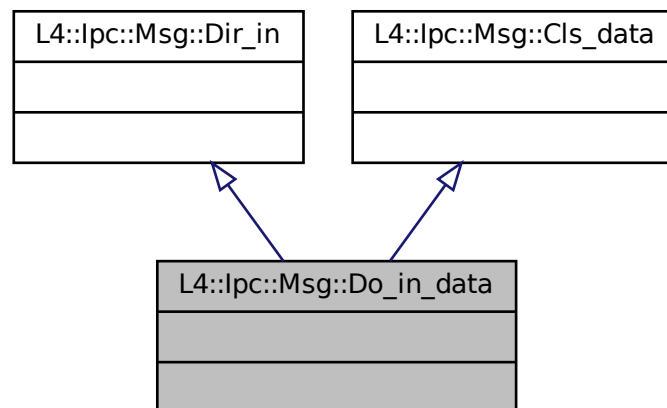
15.106 L4::lpc::Msg::Do_in_data Struct Reference

Marker for Input data.

Inheritance diagram for L4::lpc::Msg::Do_in_data:



Collaboration diagram for L4::lpc::Msg::Do_in_data:



15.106.1 Detailed Description

Marker for Input data.

Definition at line 169 of file [ipc_basics](#).

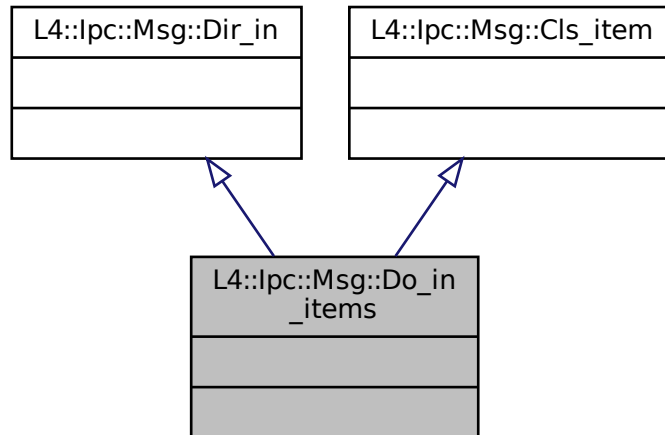
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_basics`

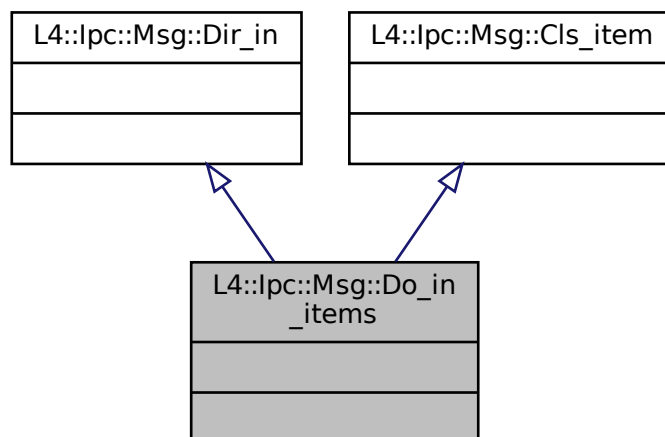
15.107 L4::lpc::Msg::Do_in_items Struct Reference

Marker for Input items.

Inheritance diagram for L4::lpc::Msg::Do_in_items:



Collaboration diagram for L4::lpc::Msg::Do_in_items:



15.107.1 Detailed Description

Marker for Input items.

Definition at line 173 of file [ipc_basics](#).

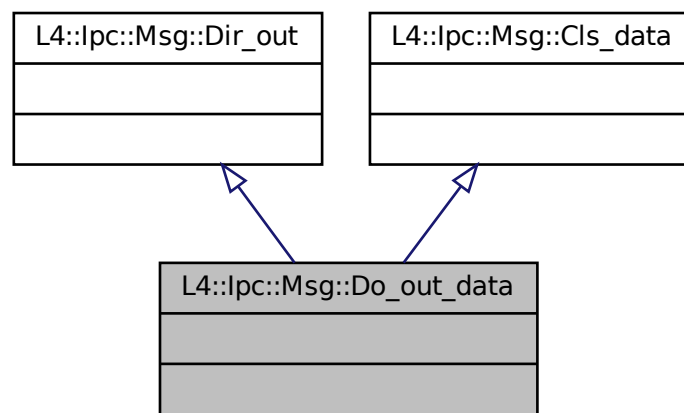
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_basics

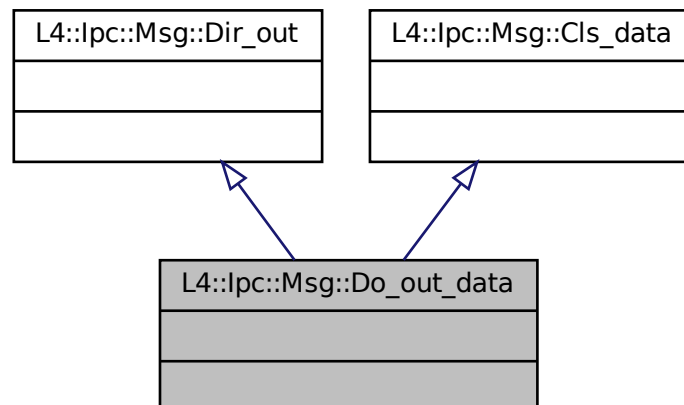
15.108 L4::lpc::Msg::Do_out_data Struct Reference

Marker for Output data.

Inheritance diagram for L4::lpc::Msg::Do_out_data:



Collaboration diagram for L4::lpc::Msg::Do_out_data:



15.108.1 Detailed Description

Marker for Output data.

Definition at line 171 of file [ipc_basics](#).

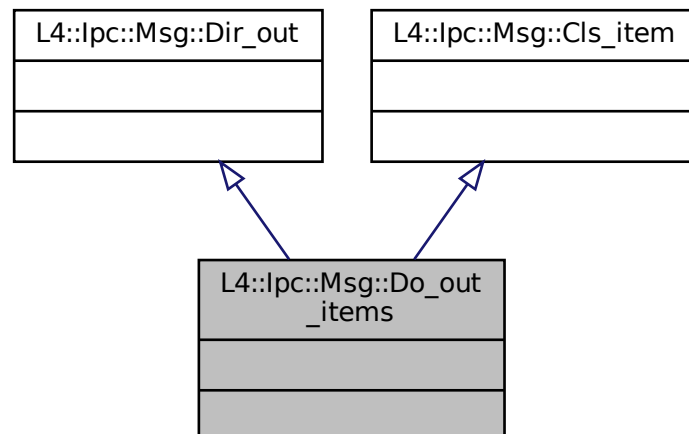
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_basics`

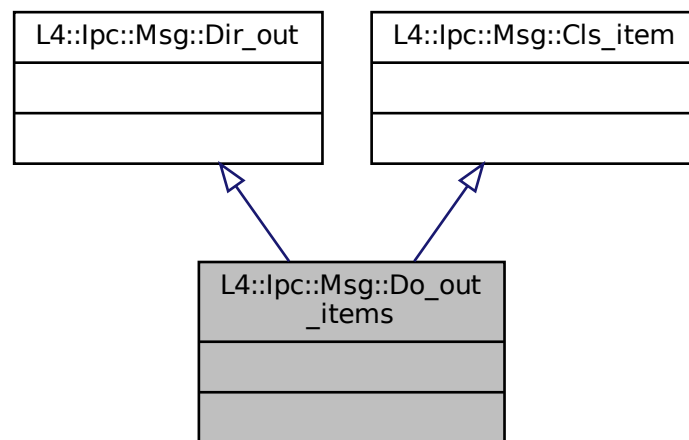
15.109 L4::lpc::Msg::Do_out_items Struct Reference

Marker for Output items.

Inheritance diagram for L4::lpc::Msg::Do_out_items:



Collaboration diagram for L4::lpc::Msg::Do_out_items:



15.109.1 Detailed Description

Marker for Output items.

Definition at line 175 of file [ipc_basics](#).

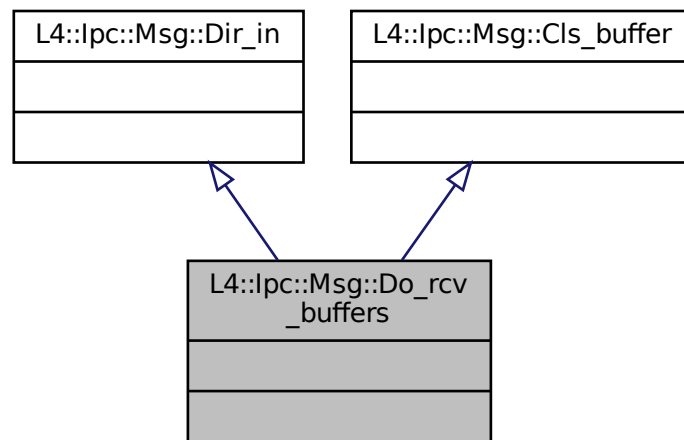
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_basics

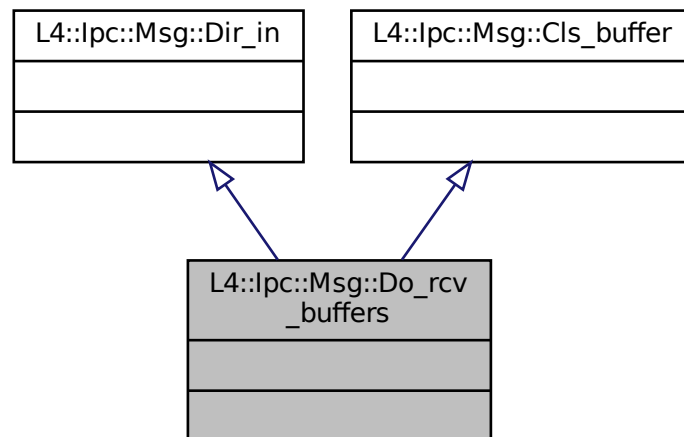
15.110 L4::lpc::Msg::Do_rcv_buffers Struct Reference

Marker for receive buffers.

Inheritance diagram for L4::lpc::Msg::Do_rcv_buffers:



Collaboration diagram for L4::lpc::Msg::Do_rcv_buffers:



15.110.1 Detailed Description

Marker for receive buffers.

Definition at line 177 of file [ipc_basics](#).

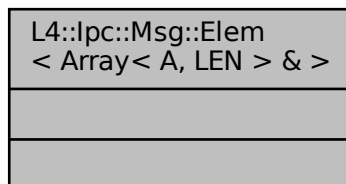
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_basics`

15.111 L4::lpc::Msg::Elem< Array< A, LEN > & > Struct Template Reference

[Array](#) as output argument.

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.

15.111.1 Detailed Description

```
template<typename A, typename LEN>
struct L4::lpc::Msg::Elem< Array< A, LEN > & >
```

[Array](#) as output argument.

Definition at line 167 of file [ipc_array](#).

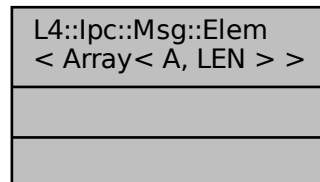
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_array

15.112 L4::ipc::Msg::Elem< Array< A, LEN > > Struct Template Reference

[Array](#) as input arguments.

Collaboration diagram for L4::ipc::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.

15.112.1 Detailed Description

```
template<typename A, typename LEN>
struct L4::ipc::Msg::Elem< Array< A, LEN > >
```

[Array](#) as input arguments.

Definition at line [155](#) of file [ipc_array](#).

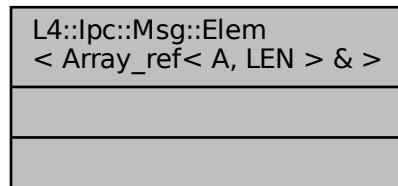
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_array`

15.113 L4::lpc::Msg::Elem< Array_ref< A, LEN > & > Struct Template Reference

[Array_ref](#) as output argument.

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.

15.113.1 Detailed Description

```
template<typename A, typename LEN>
struct L4::lpc::Msg::Elem< Array_ref< A, LEN > & >
```

[Array_ref](#) as output argument.

Definition at line 180 of file [ipc_array](#).

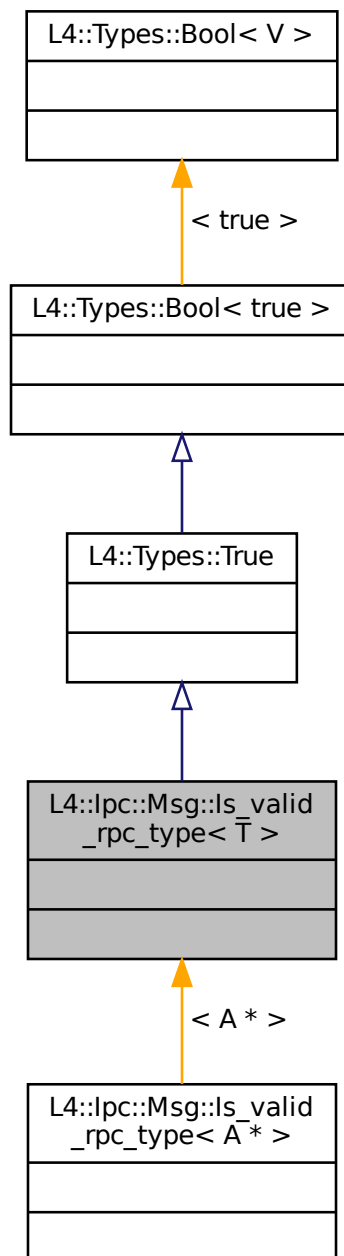
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_array

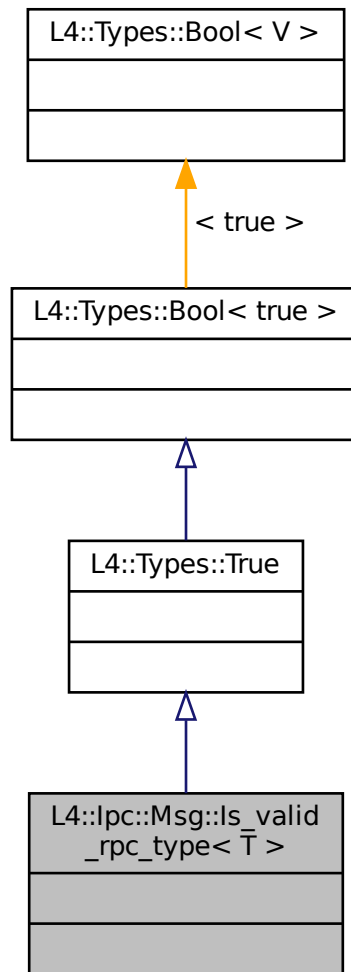
15.114 L4::lpc::Msg::ls_valid_rpc_type< T > Struct Template Reference

Type trait defining a valid RPC parameter type.

Inheritance diagram for L4::lpc::Msg::ls_valid_rpc_type< T >:



Collaboration diagram for L4::ipc::Msg::ls_valid_rpc_type< T >:



Additional Inherited Members

15.114.1 Detailed Description

```
template<typename T>
struct L4::ipc::Msg::ls_valid_rpc_type< T >
```

Type trait defining a valid RPC parameter type.

Definition at line 350 of file [ipc_basics](#).

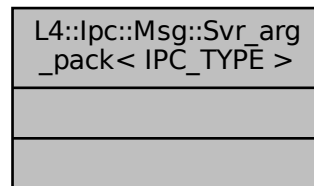
The documentation for this struct was generated from the following file:

- `I4/sys/cxx/ipc_basics`

15.115 L4::lpc::Msg::Svr_arg_pack< IPC_TYPE > Struct Template Reference

Server-side RPC arguments data structure used to provide arguments to the server-side implementation of an RPC function.

Collaboration diagram for L4::lpc::Msg::Svr_arg_pack< IPC_TYPE >:



15.115.1 Detailed Description

```
template<typename IPC_TYPE>
struct 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.

Definition at line 155 of file [ipc_server](#).

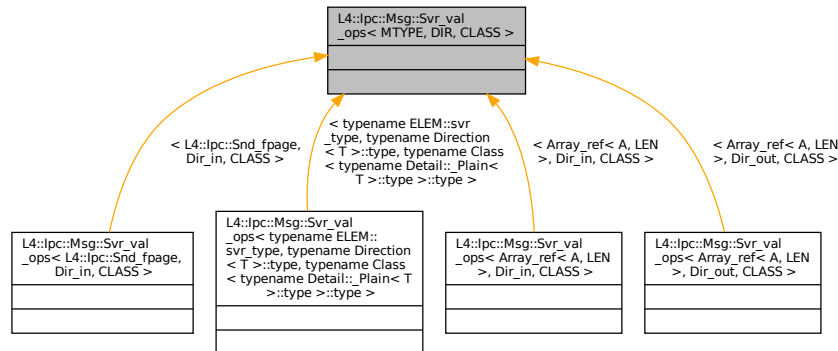
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_server](#)

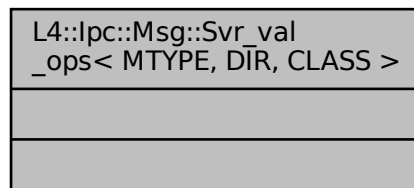
15.116 L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS > Struct Template Reference

Defines server-side handling for `MTYPE` server arguments.

Inheritance diagram for L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >:



Collaboration diagram for L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >:



15.116.1 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](#).

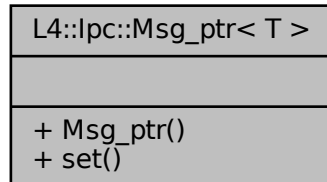
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_basics`

15.117 L4::lpc::Msg_ptr< T > Class Template Reference

Pointer to an element of type T in an [lpc::lstream](#).

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.

15.117.1 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 242 of file [ipc_stream](#).

15.117.2 Constructor & Destructor Documentation

15.117.2.1 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 253 of file [ipc_stream](#).

The documentation for this class was generated from the following file:

- [l4/cxx/ipc_stream](#)

15.118 L4::lpc::Opt< T > Struct Template Reference

Attribute for defining an optional RPC argument.

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.

15.118.1 Detailed Description

```
template<typename T>  
struct L4::ipc::Opt< T >
```

Attribute for defining an optional RPC argument.

Definition at line [147](#) of file [ipc_types](#).

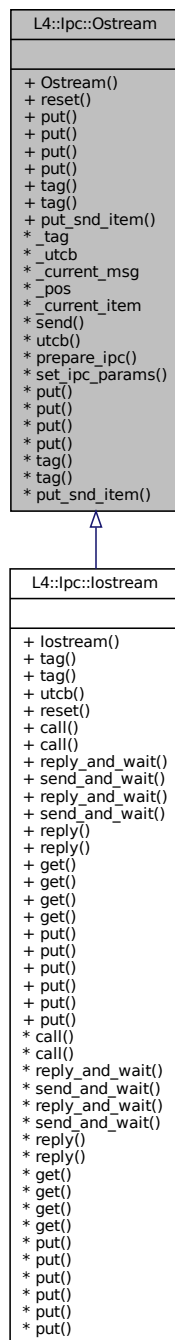
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_types](#)

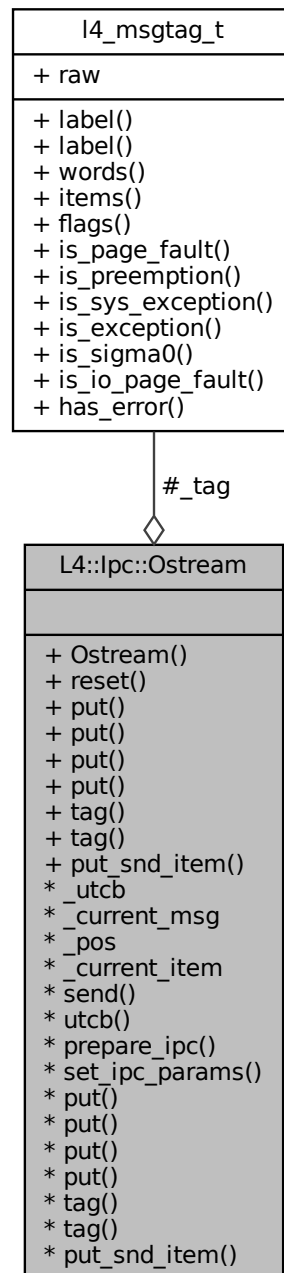
15.119 L4::ipc::Ostream Class Reference

Output stream for IPC marshalling.

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.

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.
- `int put (Varg const &va)`
- `template<typename T >`
`int put (Varg_t< T > const &va)`
- `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.
- `bool put_snd_item (Snd_item const &)`

IPC operations.

- `l4_msgtag_t _tag`
- `l4_utcb_t * _utcb`
- `char * _current_msg`
- `unsigned _pos`
- `unsigned char _current_item`
- `l4_msgtag_t send (l4_cap_idx_t dst, long proto=0, unsigned flags=0)`
Send the message via IPC to the given receiver.
- `l4_utcb_t * utcb () const`
Return utcb pointer.
- `l4_msgtag_t prepare_ipc (long proto=0, unsigned flags=0)`
- `void set_ipc_params (l4_msgtag_t tag)`

15.119.1 Detailed Description

Output stream for IPC marshalling.

`lpc::Ostream` is part of the dynamic IPC marshalling infrastructure, as well as `lpc::Istream` and `lpc::lostream`.

`lpc::Ostream` is an output stream supporting insertion of values into an IPC message buffer. A IPC message can be marshalled using the usual insertion operator <<, see [IPC stream operators](#) .

There exist some special wrapper classes to insert arrays (see `lpc::Buf_cp_out`) and indirect strings (see `Msg_↔out_buffer` and `Msg_io_buffer`).

Definition at line 634 of file `ipc_stream`.

15.119.2 Member Function Documentation**15.119.2.1 put() [1/2]**

```
template<typename T >
bool L4::Ipc::Ostream::put (
    T * buf,
    unsigned long size ) [inline]
```

Put an array with `size` elements of type `T` into the stream.

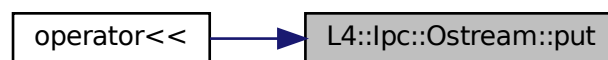
Parameters

<i>buf</i>	A pointer to the array to insert into the buffer.
<i>size</i>	The number of elements in the array.

Definition at line 671 of file [ipc_stream](#).

Referenced by [operator<<\(\)](#).

Here is the caller graph for this function:



15.119.2.2 put() [2/2]

```
template<typename T >
bool L4::Ipc::Ostream::put (
    T const & v ) [inline]
```

Insert an element of type T into the stream.

Parameters

<i>v</i>	The element to insert.
----------	------------------------

Definition at line 689 of file [ipc_stream](#).

15.119.2.3 send()

```
l4\_msgtag\_t L4::Ipc::Ostream::send (
    l4\_cap\_idx\_t dst,
    long proto = 0,
    unsigned flags = 0 ) [inline]
```

Send the message via IPC to the given receiver.

Parameters

<i>dst</i>	The destination for the message.
<i>proto</i>	Protocol to use.
<i>flags</i>	Flags to use.

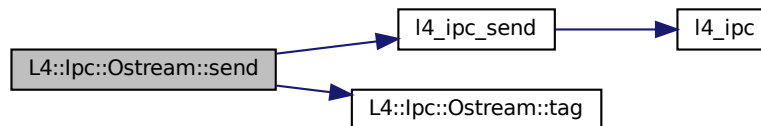
Returns

The syscall return tag.

Definition at line 966 of file [ipc_stream](#).

References [l4_ipc_send\(\)](#), [L4_MSGTAG_FLAGS](#), and [tag\(\)](#).

Here is the call graph for this function:



15.119.2.4 tag() [1/2]

```
l4\_msgtag\_t& L4::Ipc::Ostream::tag ( ) [inline]
```

Extract a reference to the [L4](#) message tag from the stream.

Returns

A reference to the [L4](#) message tag.

Definition at line 724 of file [ipc_stream](#).

15.119.2.5 tag() [2/2]

```
l4_msgtag_t L4::Ipc::Ostream::tag ( ) const [inline]
```

Extract the [L4](#) message tag from the stream.

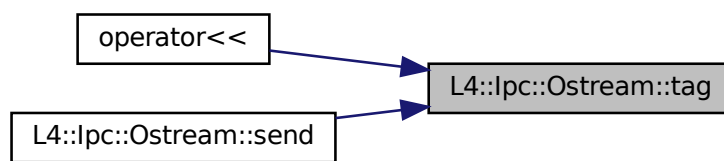
Returns

The extracted [L4](#) message tag.

Definition at line 717 of file [ipc_stream](#).

Referenced by [operator<<\(\)](#), and [send\(\)](#).

Here is the caller graph for this function:



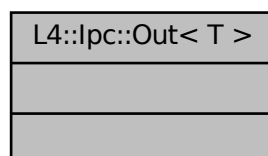
The documentation for this class was generated from the following file:

- [l4/cxx/ipc_stream](#)

15.120 L4::lpc::Out< T > Struct Template Reference

Mark an argument as a output value in an RPC signature.

Collaboration diagram for `L4::lpc::Out< T >`:



15.120.1 Detailed Description

```
template<typename T>
struct L4::lpc::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](#).

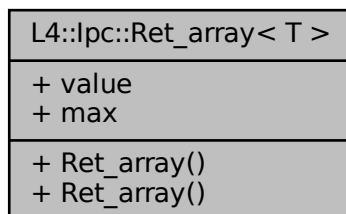
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_types](#)

15.121 L4::lpc::Ret_array< T > Struct Template Reference

Dynamically sized output array of type T.

Collaboration diagram for L4::lpc::Ret_array< T >:



15.121.1 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](#).

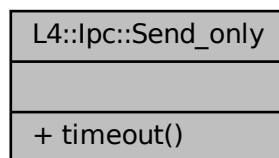
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_ret_array](#)

15.122 L4::lpc::Send_only Struct Reference

RPC attribute for a send-only RPC.

Collaboration diagram for L4::lpc::Send_only:



15.122.1 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](#).

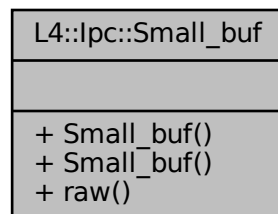
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_iface](#)

15.123 L4::lpc::Small_buf Class Reference

A receive item for receiving a single capability.

Collaboration diagram for L4::lpc::Small_buf:



Public Member Functions

- [Small_buf](#) ([L4::Cap](#)< void > cap, unsigned long flags=0) noexcept
Create a receive item from a C++ cap.
- [Small_buf](#) ([l4_cap_idx_t](#) cap, unsigned long flags=0) noexcept
Create a receive item from a C cap.

15.123.1 Detailed Description

A receive item for receiving a single capability.

This class is the main abstraction for receiving capabilities via [lpc::lstream](#). To receive a 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](#).

15.123.2 Constructor & Destructor Documentation

15.123.2.1 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](#).

15.123.2.2 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](#).

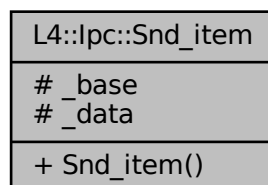
The documentation for this class was generated from the following file:

- [l4/sys/cxx/ipc_types](#)

15.124 L4::lpc::Snd_item Class Reference

RPC wrapper for a send item.

Collaboration diagram for L4::lpc::Snd_item:



15.124.1 Detailed Description

RPC wrapper for a send item.

Definition at line 294 of file [ipc_types](#).

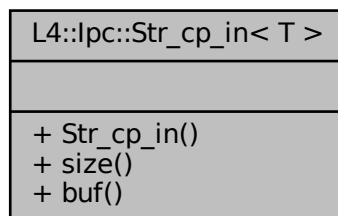
The documentation for this class was generated from the following file:

- [l4/sys/cxx/ipc_types](#)

15.125 L4::lpc::Str_cp_in< T > Class Template Reference

Abstraction for extracting a zero-terminated string from an [lpc::lstream](#).

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](#).

15.125.1 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 from 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 191 of file [ipc_stream](#).

15.125.2 Constructor & Destructor Documentation

15.125.2.1 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::Istream](#).

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 202 of file [ipc_stream](#).

The documentation for this class was generated from the following file:

- [l4/cxx/ipc_stream](#)

15.126 L4::lpc::Varg Class Reference

Variably sized RPC argument.

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... + nil()

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.

15.126.1 Detailed Description

Variably sized RPC argument.

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

15.126.2 Member Function Documentation

15.126.2.1 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\(\)](#).

Here is the caller graph for this function:



15.126.2.2 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](#).

15.126.2.3 is_nil()

```
bool L4::Ipc::Varg::is_nil ( ) const [inline]
```

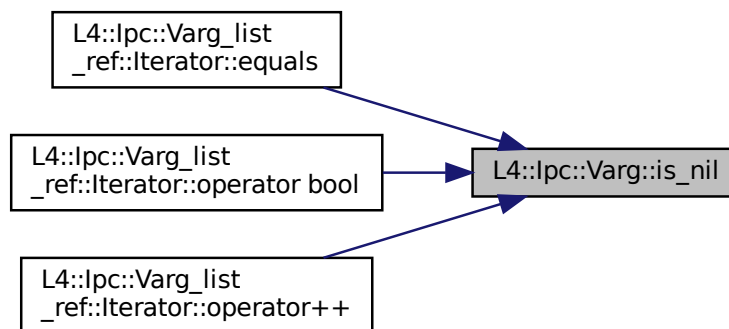
Returns

true if the [Varg](#) is of nil type.

Definition at line 171 of file [ipc_varg](#).

Referenced by [L4::lpc::Varg_list_ref::Iterator::equals\(\)](#), [L4::lpc::Varg_list_ref::Iterator::operator bool\(\)](#), and [L4::lpc::Varg_list_ref::Iterator::operator++\(\)](#).

Here is the caller graph for this function:



15.126.2.4 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:

**15.126.2.5 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:



15.126.2.6 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](#).

15.126.2.7 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](#).

15.126.2.8 type()

```
L4_varg_type L4::Ipc::Varg::type ( ) const [inline]
```

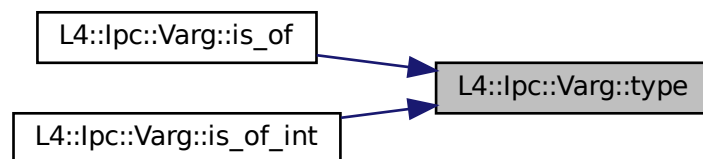
Returns

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:



15.126.2.9 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](#).

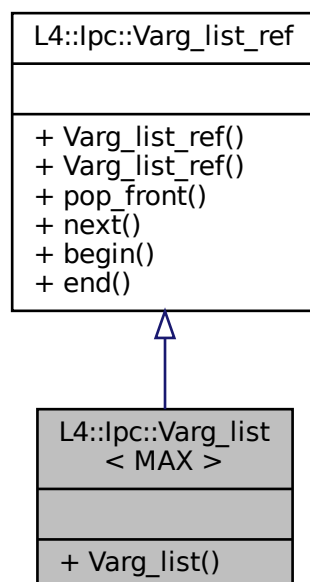
The documentation for this class was generated from the following file:

- [l4/sys/cxx/ipc_varg](#)

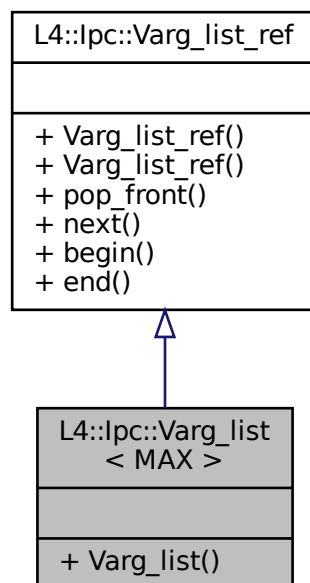
15.127 L4::lpc::Varg_list< MAX > Class Template Reference

Self-contained list of variable-sized RPC parameters.

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.

15.127.1 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](#).

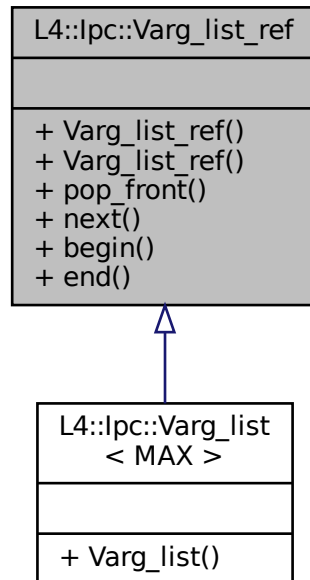
The documentation for this class was generated from the following file:

- `l4/sys/cxx/ipc_varg`

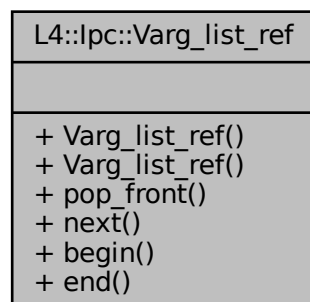
15.128 L4::lpc::Varg_list_ref Class Reference

List of variable-sized RPC parameters as received by the server.

Inheritance diagram for L4::lpc::Varg_list_ref:



Collaboration diagram for L4::lpc::Varg_list_ref:



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.

15.128.1 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](#).

15.128.2 Constructor & Destructor Documentation

15.128.2.1 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](#)

15.129 L4::lpc::Varg_list_ref::Iterator Class Reference

[Iterator](#) for Valists.

Collaboration diagram for L4::lpc::Varg_list_ref::Iterator:

L4::lpc::Varg_list_ref::Iterator
<ul style="list-style-type: none"> + Iterator() + operator bool() + operator++() + operator*() + equals() + operator==(()) + operator!=(())

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

15.129.1 Detailed Description

[Iterator](#) for Valists.

Definition at line 337 of file [ipc_varg](#).

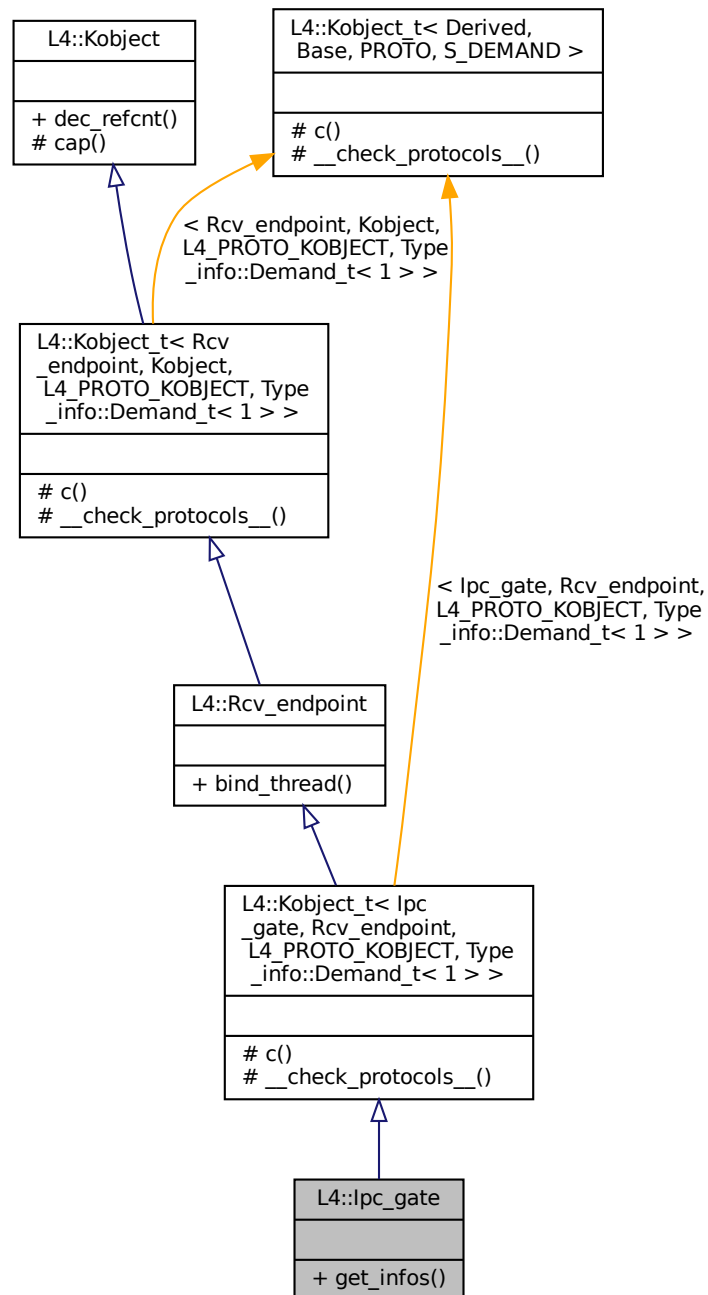
The documentation for this class was generated from the following file:

- l4/sys/cxx/ipc_varg

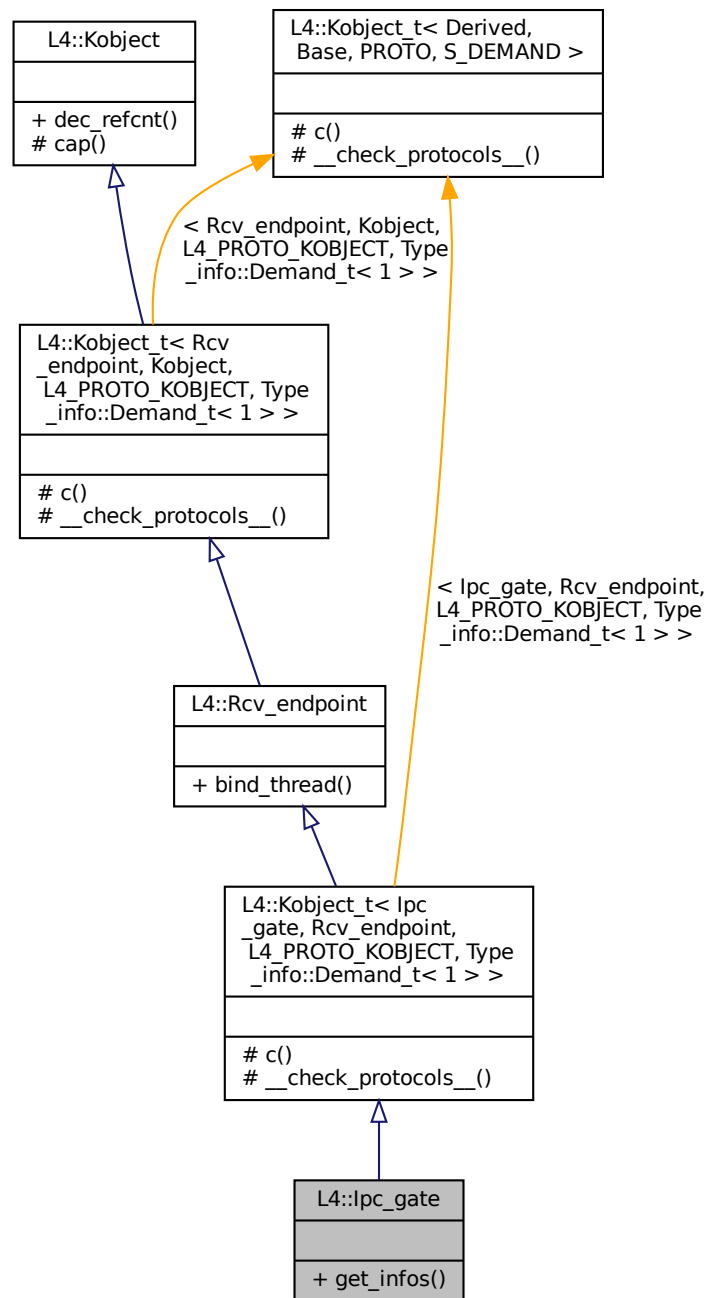
15.130 L4::lpc_gate Class Reference

The C++ IPC gate interface.

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.

Additional Inherited Members

15.130.1 Detailed Description

The C++ IPC gate interface.

IPC gates are used to create secure communication channels between protection domains. An IPC gate can be created using the [L4::Factory](#) interface. [L4::lpc_gate::bind_thread\(\)](#) binds an [L4::Thread](#) as the receiver of all messages to an IPC gate.

The [bind_thread\(\)](#) call allows to assign each IPC gate a kernel protected, machine-word sized payload called a *label*. It securely identifies the gate. The two least significant bits of the *label* are ORed with the [L4_CAP_FPAGE_S](#) and [L4_CAP_FPAGE_W](#) bits stored in the capability when transferred to the receiver. This means the *label* should usually have its two least significant bits set to zero. The *label* is only visible in the [L4::Task](#) which is running the thread the IPC gate was bound to and cannot be altered by the sender.

The IPC gate's methods can only be invoked, if the gate was mapped with the [L4_FPAGE_C_IPCGATE_SVR](#) right. To ensure that the *label* assigned to the gate cannot be altered by the client, the client should receive the gate capability without the [L4_FPAGE_C_IPCGATE_SVR](#) right.

Include File

```
#include <l4/sys/ipc_gate>
```

For the C interface refer to the C [IPC-Gate API](#).

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

15.130.2 Member Function Documentation

15.130.2.1 get_infos()

```
l4_msgtag_t L4::IpC_gate::get_infos (
    l4_umword_t * label )
```

Get information about the IPC-gate.

Parameters

out	<i>label</i>	The label of the IPC gate is returned here.
-----	--------------	---

Returns

System call return tag.

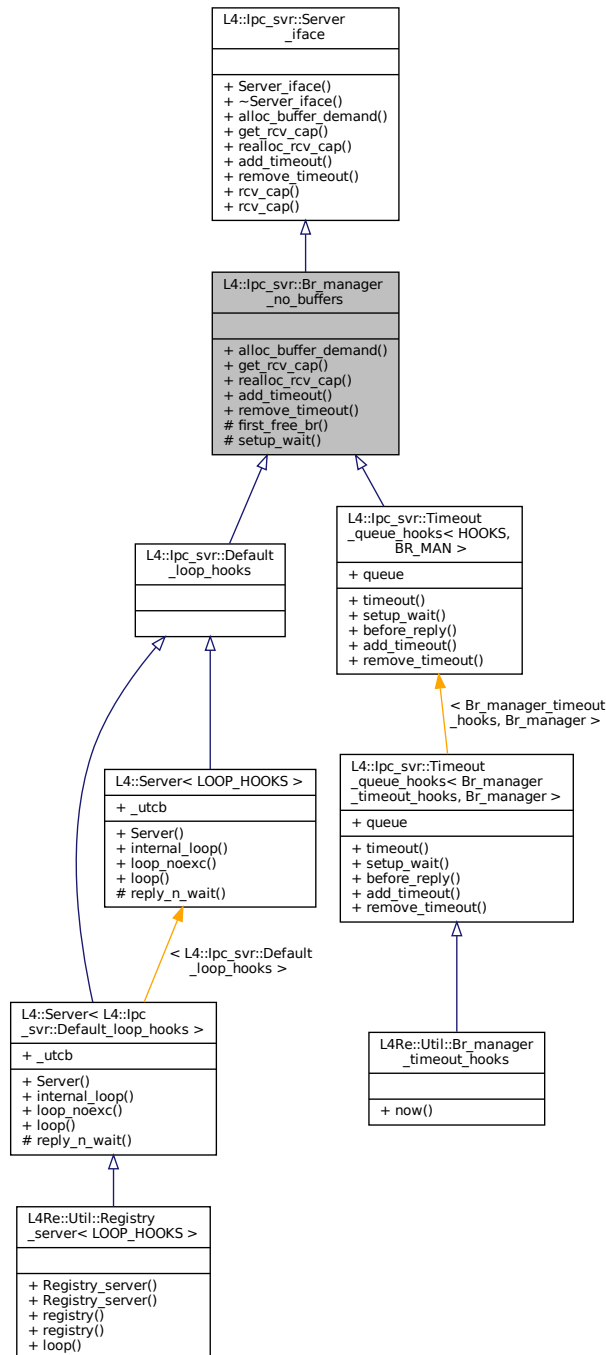
The documentation for this class was generated from the following file:

- [l4/sys/ipc_gate](#)

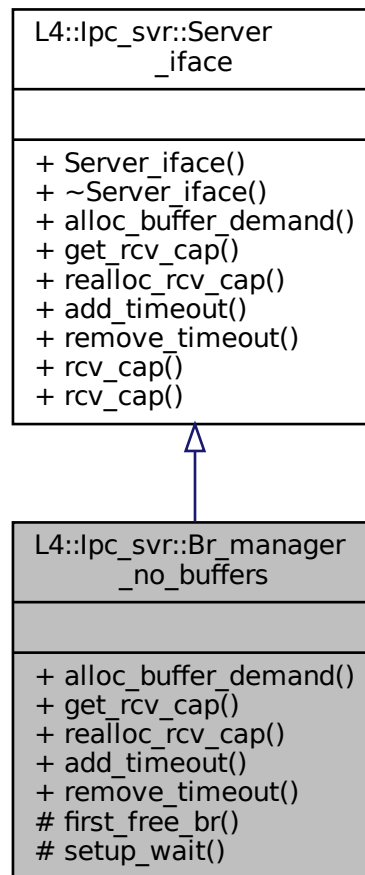
15.131 L4::lpc_svr::Br_manager_no_buffers Class Reference

Empty implementation of [Server_iface](#).

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)
Tells the server to allocate buffers for the given demand.
- [L4::Cap](#)< void > [get_rcv_cap](#) (int) const
Returns [L4::Cap](#)<void>::Invalid, we have no buffer management.
- int [realloc_rcv_cap](#) (int)
Returns -L4_ENOMEM, we have no buffer management.
- int [add_timeout](#) ([Timeout](#) *, [l4_kernel_clock_t](#))
Returns -L4_ENOSYS, we have no timeout queue.
- int [remove_timeout](#) ([Timeout](#) *)
Returns -L4_ENOSYS, we have no timeout queue.

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

15.131.1 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 224 of file [ipc_server_loop](#).

15.131.2 Member Function Documentation

15.131.2.1 `alloc_buffer_demand()`

```
int L4::Ipc_svr::Br_manager_no_buffers::alloc_buffer_demand (
    Demand const & demand ) [inline], [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

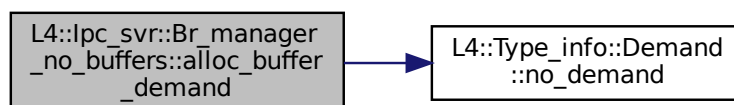
success (0) if demand is empty, -L4_ENOMEM else.

Implements [L4::ipc_svr::Server_iface](#).

Definition at line 231 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:



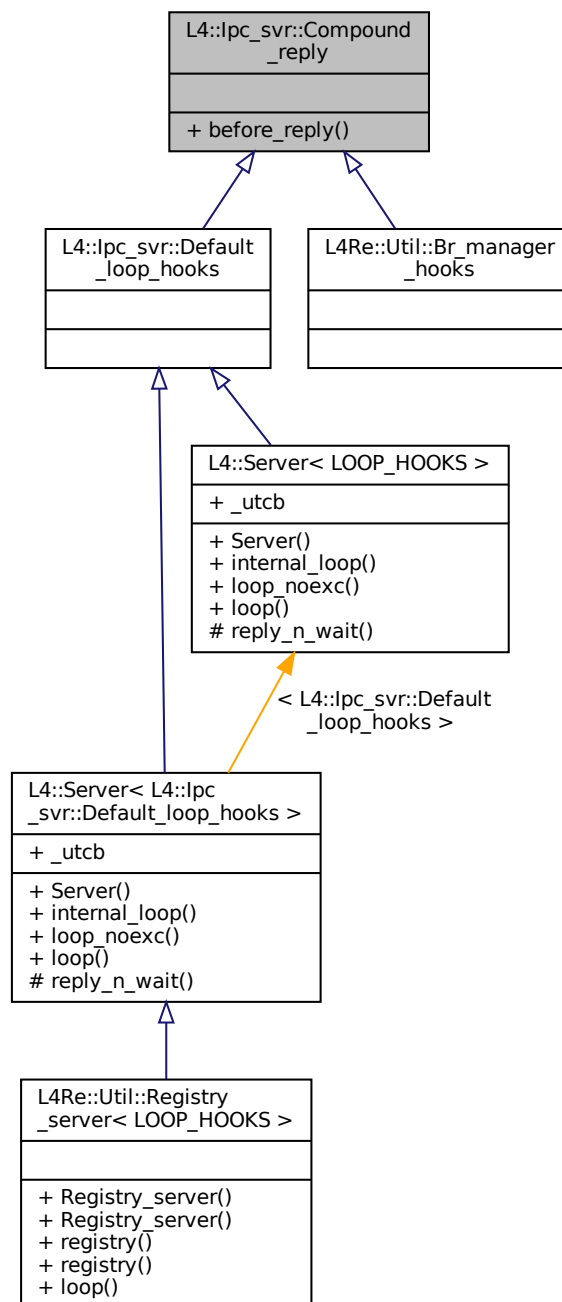
The documentation for this class was generated from the following file:

- l4/sys/cxx/lpc_server_loop

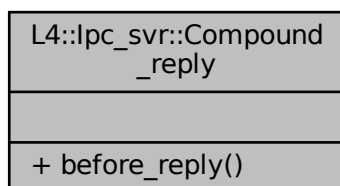
15.132 L4::lpc_svr::Compound_reply Struct Reference

Mix in for LOOP_HOOKS to always use compound reply and wait.

Inheritance diagram for L4::lpc_svr::Compound_reply:



Collaboration diagram for L4::lpc_svr::Compound_reply:



15.132.1 Detailed Description

Mix in for `LOOP_HOOKS` to always use compound reply and wait.

Definition at line 77 of file [ipc_server_loop](#).

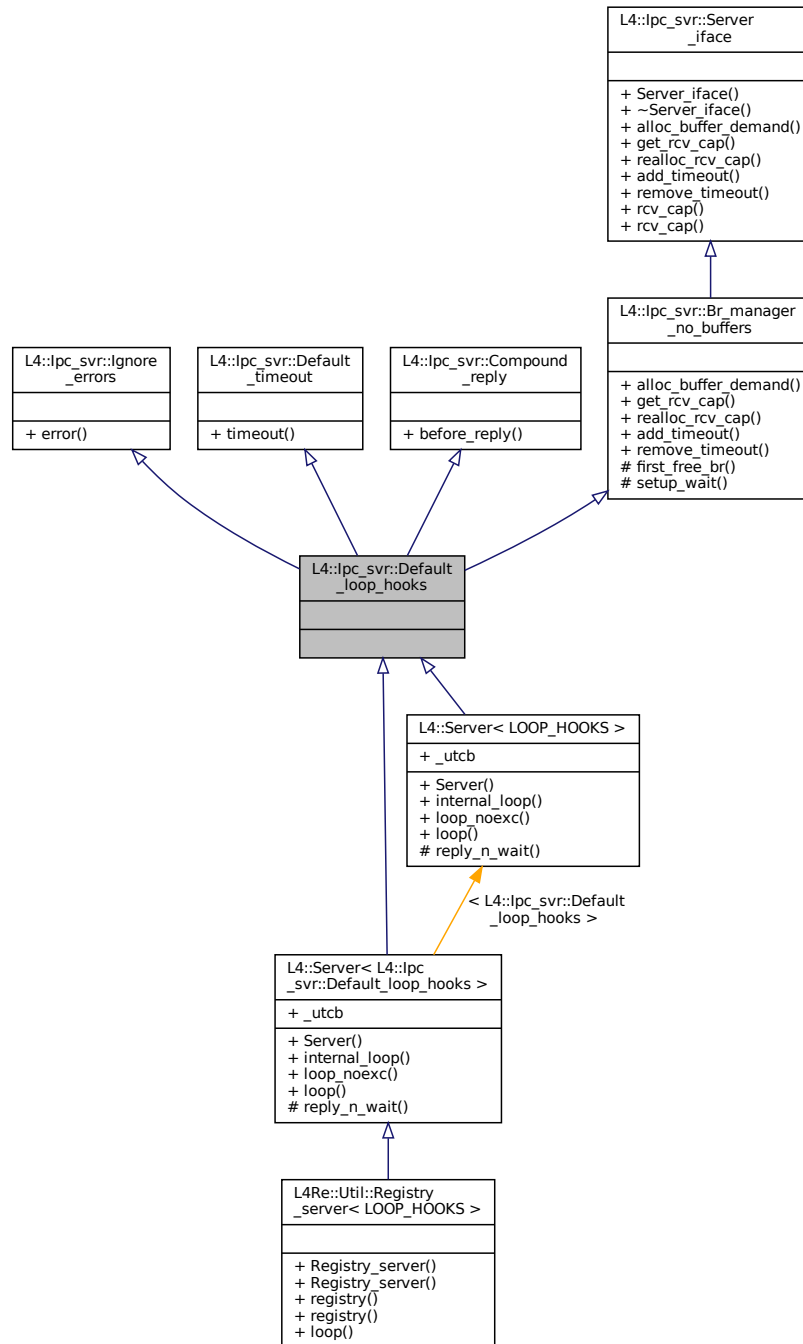
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_server_loop`

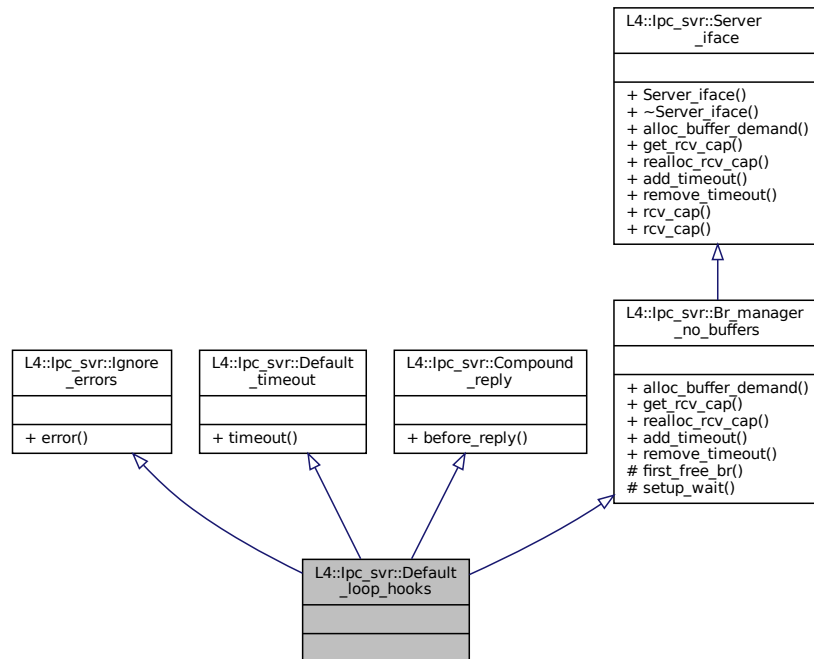
15.133 L4::lpc_svr::Default_loop_hooks Struct Reference

Default `LOOP_HOOKS`.

Inheritance diagram for L4::lpc_svr::Default_loop_hooks:



Collaboration diagram for L4::ipc_svr::Default_loop_hooks:



Additional Inherited Members

15.133.1 Detailed Description

Default LOOP_HOOKS.

Combination of [Ignore_errors](#), [Default_timeout](#), [Compound_reply](#), and [Br_manager_no_buffers](#).

Definition at line 276 of file [ipc_server_loop](#).

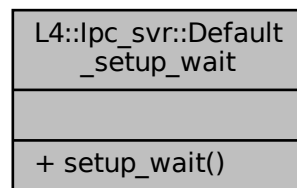
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_server_loop](#)

15.134 L4::ipc_svr::Default_setup_wait Struct Reference

Mix in for LOOP_HOOKS for setup_wait no op.

Collaboration diagram for L4::lpc_svr::Default_setup_wait:



15.134.1 Detailed Description

Mix in for LOOP_HOOKS for setup_wait no op.

Definition at line 88 of file [ipc_server_loop](#).

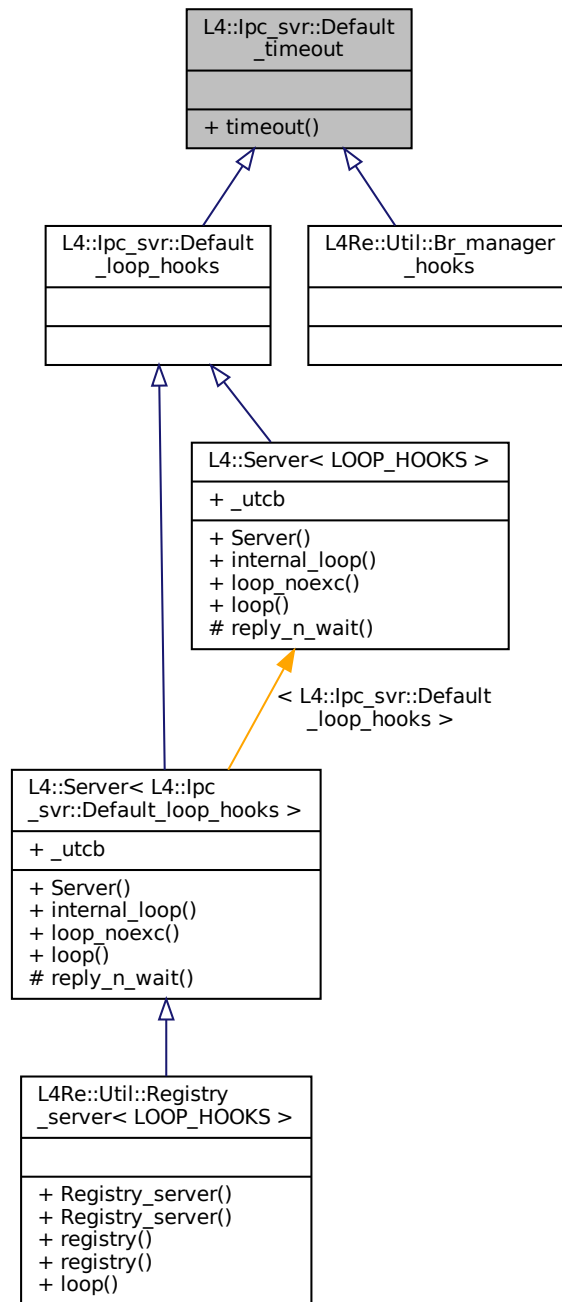
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_server_loop`

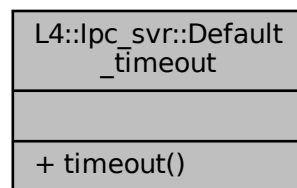
15.135 L4::lpc_svr::Default_timeout Struct Reference

Mix in for LOOP_HOOKS to use a 0 send and a infinite receive timeout.

Inheritance diagram for L4::lpc_svr::Default_timeout:



Collaboration diagram for L4::lpc_svr::Default_timeout:



15.135.1 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](#).

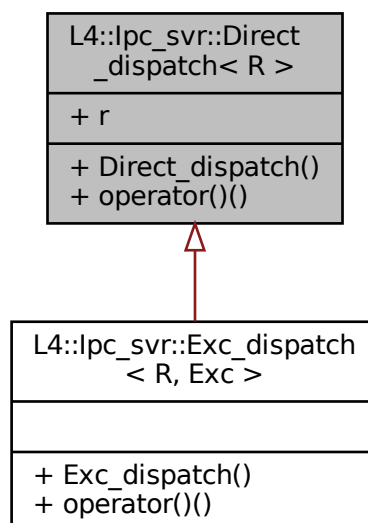
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_server_loop

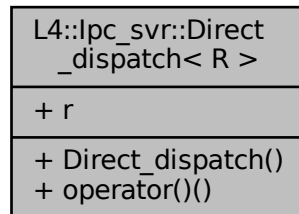
15.136 L4::lpc_svr::Direct_dispatch< R > Struct Template Reference

Direct dispatch helper, for forwarding dispatch calls to a registry *R*.

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

15.136.1 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 139 of file [ipc_server_loop](#).

The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_server_loop](#)

15.137 L4::lpc_svr::Direct_dispatch< R * > Struct Template Reference

Direct dispatch helper, for forwarding dispatch calls via a pointer to a registry *R*.

Collaboration diagram for L4::lpc_svr::Direct_dispatch< R * >:

L4::lpc_svr::Direct_dispatch< R * >
+ r
+ Direct_dispatch() + operator()

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

15.137.1 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 160 of file [ipc_server_loop](#).

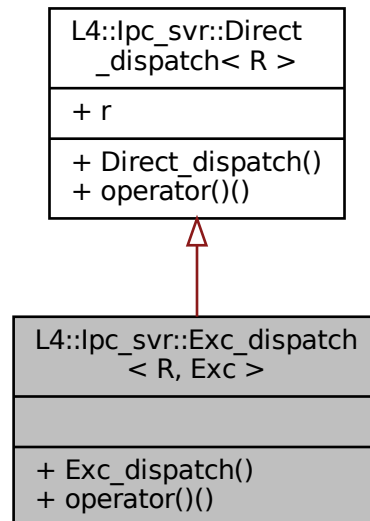
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_server_loop

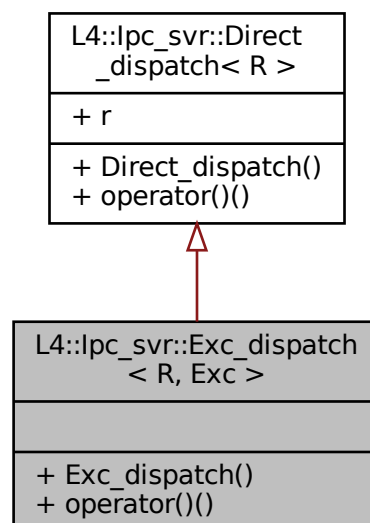
15.138 L4::lpc_svr::Exc_dispatch< R, Exc > Struct Template Reference

Dispatch helper wrapping try {} catch {} around the dispatch call.

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.

15.138.1 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 184 of file [ipc_server_loop](#).

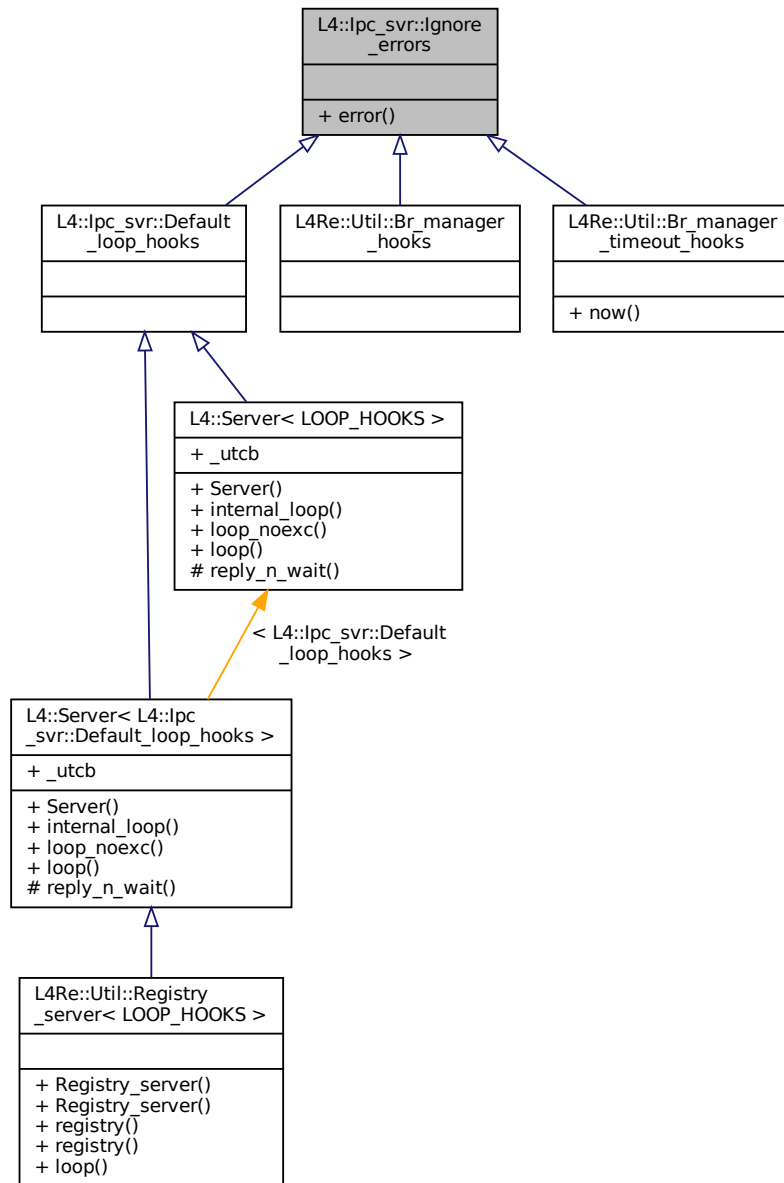
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_server_loop`

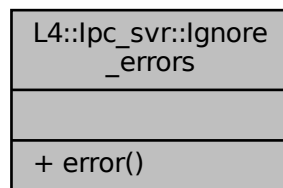
15.139 L4::lpc_svr::Ignore_errors Struct Reference

Mix in for `LOOP_HOOKS` to ignore IPC errors.

Inheritance diagram for L4::lpc_svr::Ignore_errors:



Collaboration diagram for L4::lpc_svr::Ignore_errors:



15.139.1 Detailed Description

Mix in for LOOP_HOOKS to ignore IPC errors.

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

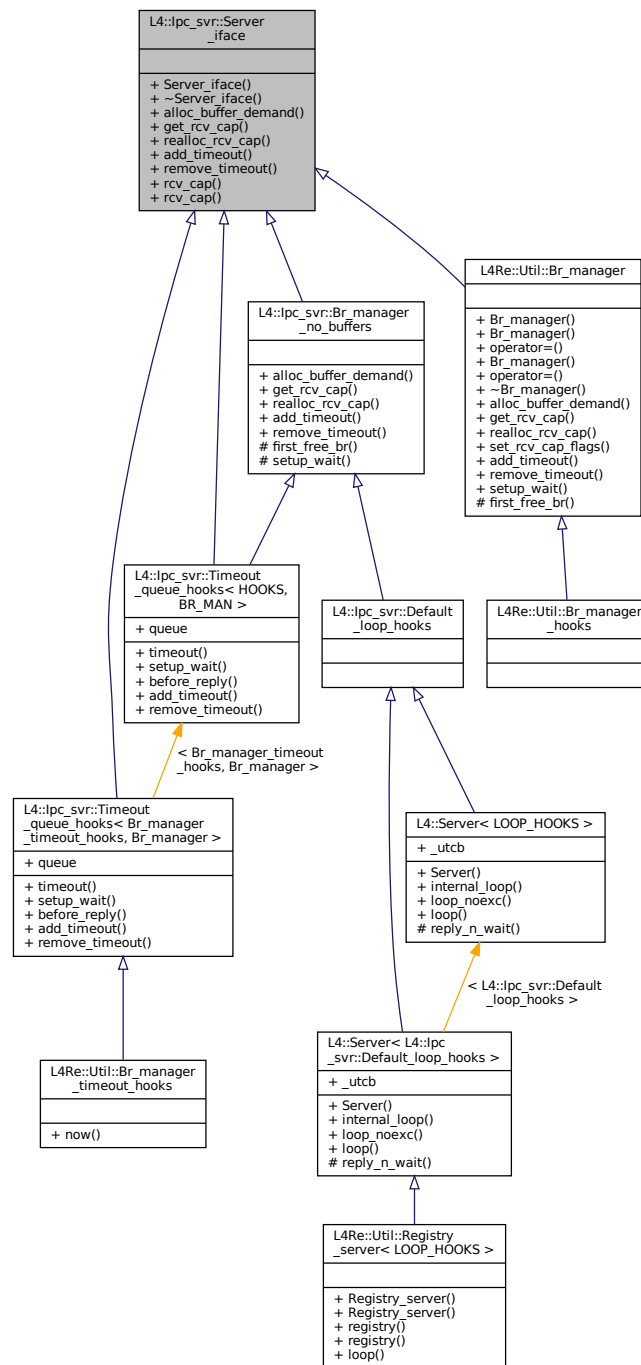
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_server_loop`

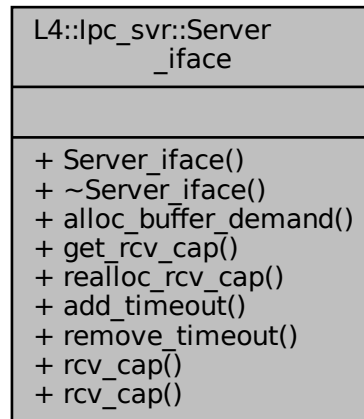
15.140 L4::lpc_svr::Server_iface Class Reference

Interface for server-loop related functions.

Inheritance diagram for L4::lpc_svr::Server_iface:



Collaboration diagram for L4::lpc_svr::Server_iface:



Public Types

- typedef [L4::Type_info::Demand](#) Demand
Data type expressing server-side demand for receive buffers.

Public Member Functions

- [Server_iface](#) ()
Make a server interface.
- virtual int [alloc_buffer_demand](#) ([Demand](#) const &demand)=0
Tells the server to allocate buffers for the given demand.
- virtual [L4::Cap](#)< void > [get_rcv_cap](#) (int index) const =0
Get capability slot allocated to the given receive buffer.
- virtual int [realloc_rcv_cap](#) (int index)=0
Allocate a new capability for the given receive buffer.
- virtual int [add_timeout](#) ([Timeout](#) *timeout, [l4_kernel_clock_t](#) time)=0
Add a timeout to the server internal timeout queue.
- virtual int [remove_timeout](#) ([Timeout](#) *timeout)=0
Remove the given timeout from the timer queue.
- template<typename T >
[L4::Cap](#)< T > [rcv_cap](#) (int index) const
Get given receive buffer as typed capability.
- [L4::Cap](#)< void > [rcv_cap](#) (int index) const
Get receive cap with the given index as generic (void) type.

15.140.1 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](#).

15.140.2 Member Function Documentation

15.140.2.1 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 [L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >](#), [L4::Ipc_svr::Timeout_queue_hooks< Br_manager_timeout_hooks, BR_MAN >](#), [L4::Ipc_svr::Br_manager_no_buffers](#), and [L4Re::Util::Br_manager](#).

15.140.2.2 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 [L4::lpc_svr::Br_manager_no_buffers](#), and [L4Re::Util::Br_manager](#).

15.140.2.3 get_rcv_cap()

```
virtual L4::Cap<void> L4::lpc_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\(\)](#)

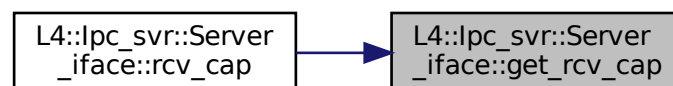
Returns

Capability slot currently allocated to the given receive buffer.

Implemented in [L4::lpc_svr::Br_manager_no_buffers](#), and [L4Re::Util::Br_manager](#).

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

Here is the caller graph for this function:



15.140.2.4 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.

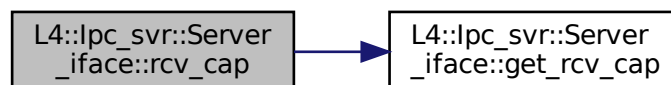
Note

This is a convenience wrapper for [get_rcv_cap\(\)](#) to avoid [L4::cap_cast<>\(\)](#).

Definition at line 122 of file [ipc_epiface](#).

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

Here is the call graph for this function:

**15.140.2.5 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.

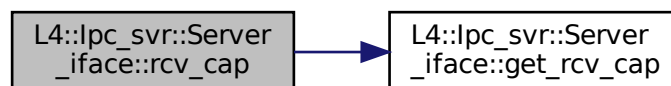
Note

This is a convenience wrapper for [get_rcv_cap\(\)](#).

Definition at line 134 of file [ipc_epiface](#).

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

Here is the call graph for this function:



15.140.2.6 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 [L4::lpc_svr::Br_manager_no_buffers](#), and [L4Re::Util::Br_manager](#).

15.140.2.7 remove_timeout()

```
virtual int L4::lpc_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 [L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >](#), [L4::lpc_svr::Timeout_queue_hooks< Br_manager_timeout, BR_MAN >](#), [L4::lpc_svr::Br_manager_no_buffers](#), and [L4Re::Util::Br_manager](#).

The documentation for this class was generated from the following file:

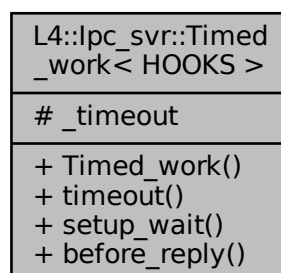
- l4/sys/cxx/ipc_epiface

15.141 L4::lpc_svr::Timed_work< HOOKS > Class Template Reference

DEPRECATED.

Inherits [HOOKS](#).

Collaboration diagram for [L4::lpc_svr::Timed_work< HOOKS >](#):



15.141.1 Detailed Description

```
template<typename HOOKS>  
class L4::lpc_svr::Timed_work< HOOKS >
```

DEPRECATED.

Deprecated Use [L4::lpc_svr::Timeout_queue_hooks](#)

Definition at line 97 of file [ipc_server_loop](#).

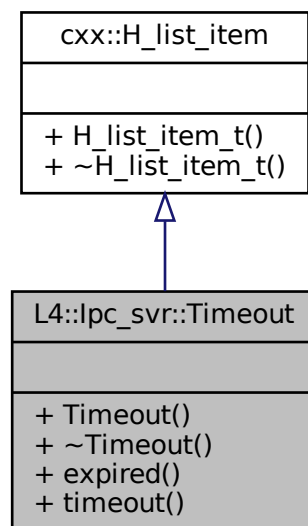
The documentation for this class was generated from the following file:

- [l4/sys/cxx/ipc_server_loop](#)

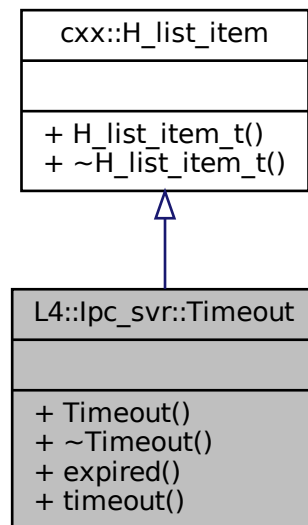
15.142 L4::lpc_svr::Timeout Class Reference

Callback interface for [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.

15.142.1 Detailed Description

Callback interface for `Timeout_queue`.

Definition at line 28 of file `ipc_timeout_queue`.

15.142.2 Member Function Documentation

15.142.2.1 expired()

```
virtual void L4::Ipc_svr::Timeout::expired ( ) [pure virtual]
```

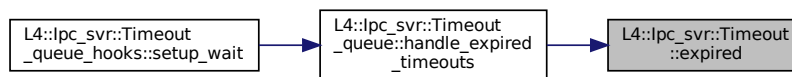
callback function to be called when timeout happened

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.

Referenced by [L4::lpc_svr::Timeout_queue::handle_expired_timeouts\(\)](#).

Here is the caller graph for this function:



15.142.2.2 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](#).

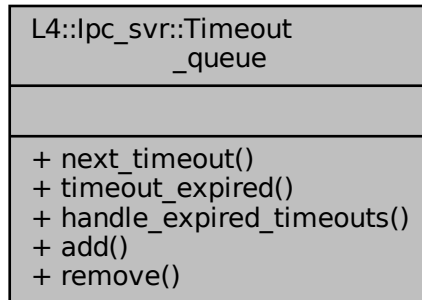
The documentation for this class was generated from the following file:

- [l4/cxx/ipc_timeout_queue](#)

15.143 L4::lpc_svr::Timeout_queue Class Reference

[Timeout](#) queue to be used in l4re server loop.

Collaboration diagram for L4::lpc_svr::Timeout_queue:



Public Types

- typedef [L4::lpc_svr::Timeout](#) Timeout
Provide a local definition of [Timeout](#) for backward compat.

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.

15.143.1 Detailed Description

[Timeout](#) queue to be used in l4re server loop.

Definition at line 65 of file [ipc_timeout_queue](#).

15.143.2 Member Function Documentation

15.143.2.1 add()

```
void L4::lpc_svr::Timeout_queue::add (
    Timeout * timeout,
    l4_kernel_clock_t time ) [inline]
```

Add a timeout to the queue.

Parameters

<i>timeout</i>	timeout object to add
<i>time</i>	the time when the timeout expires

Precondition

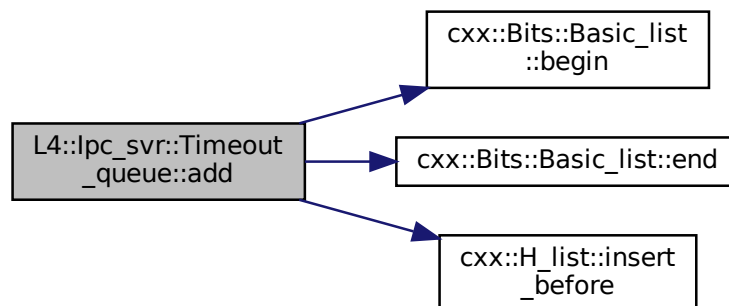
timeout must not be in any queue already

Definition at line 121 of file `ipc_timeout_queue`.

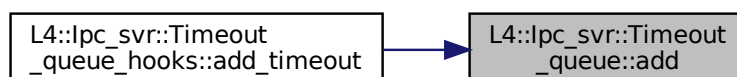
References `cxx::Bits::Basic_list< POLICY >::begin()`, `cxx::Bits::Basic_list< POLICY >::end()`, and `cxx::H_list< T, POLICY >::insert_`

Referenced by `L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >::add_timeout()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.143.2.2 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

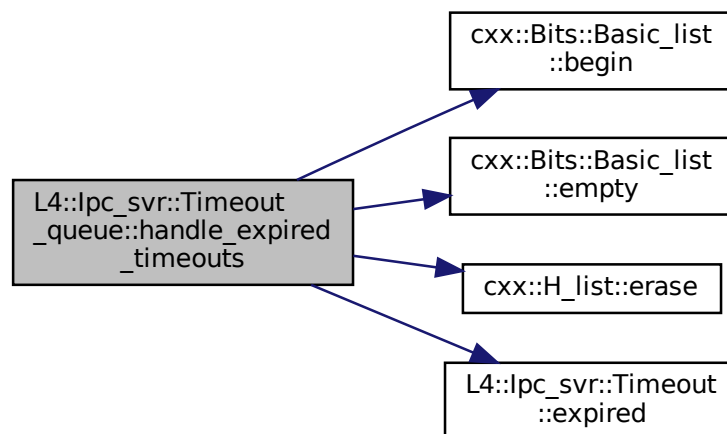
<i>now</i>	the current time.
------------	-------------------

Definition at line 101 of file [ipc_timeout_queue](#).

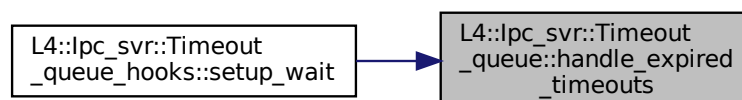
References [cxx::Bits::Basic_list< POLICY >::begin\(\)](#), [cxx::Bits::Basic_list< POLICY >::empty\(\)](#), [cxx::H_list< T, POLICY >::erase\(\)](#), and [L4::lpc_svr::Timeout::expired\(\)](#).

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:



15.143.2.3 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`.

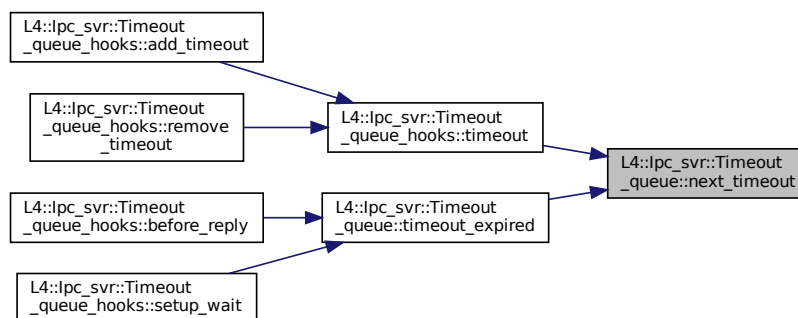
References `cxx::Bits::Basic_list< POLICY >::front()`.

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:



15.143.2.4 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

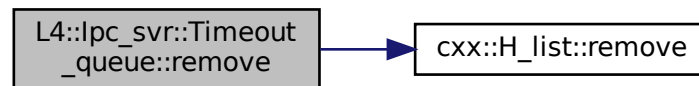
timeout must be in this queue

Definition at line 136 of file [ipc_timeout_queue](#).

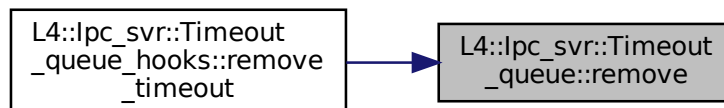
References [cxx::H_list< T, POLICY >::remove\(\)](#).

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:

**15.143.2.5 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

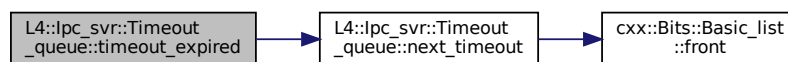
<i>true</i>	There is at least one expired timeout in the queue. <i>false</i> No expired timeout in the queue.
-------------	---

Definition at line 91 of file [ipc_timeout_queue](#).

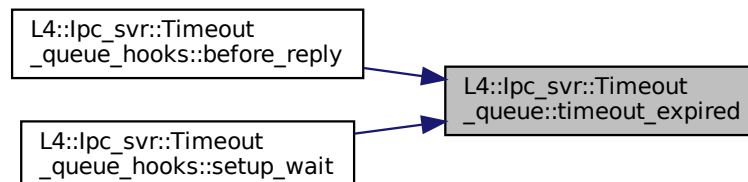
References [next_timeout\(\)](#).

Referenced by [L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >::before_reply\(\)](#), and [L4::lpc_svr::Timeout_queue_hooks<](#)

Here is the call graph for this function:



Here is the caller graph for this function:



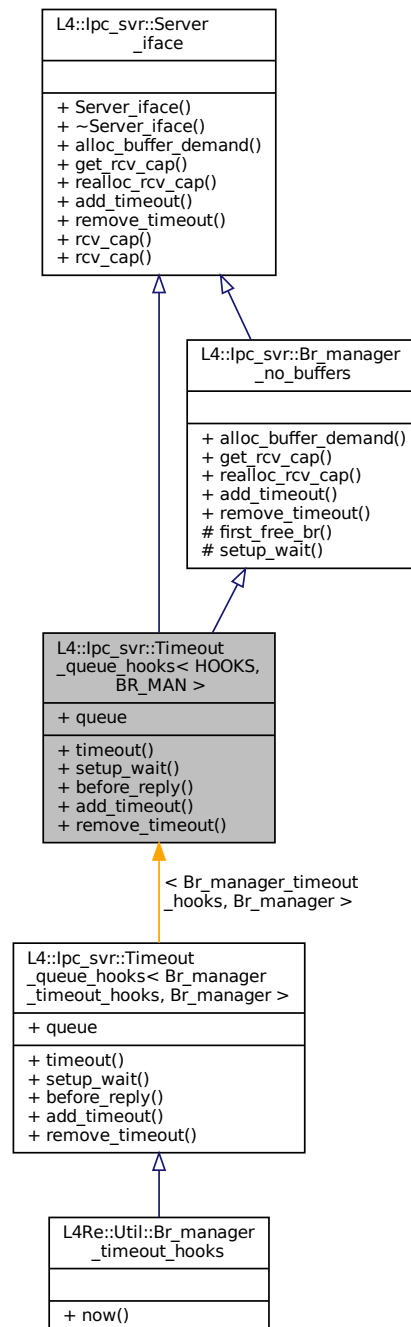
The documentation for this class was generated from the following file:

- `I4/cxx/ipc_timeout_queue`

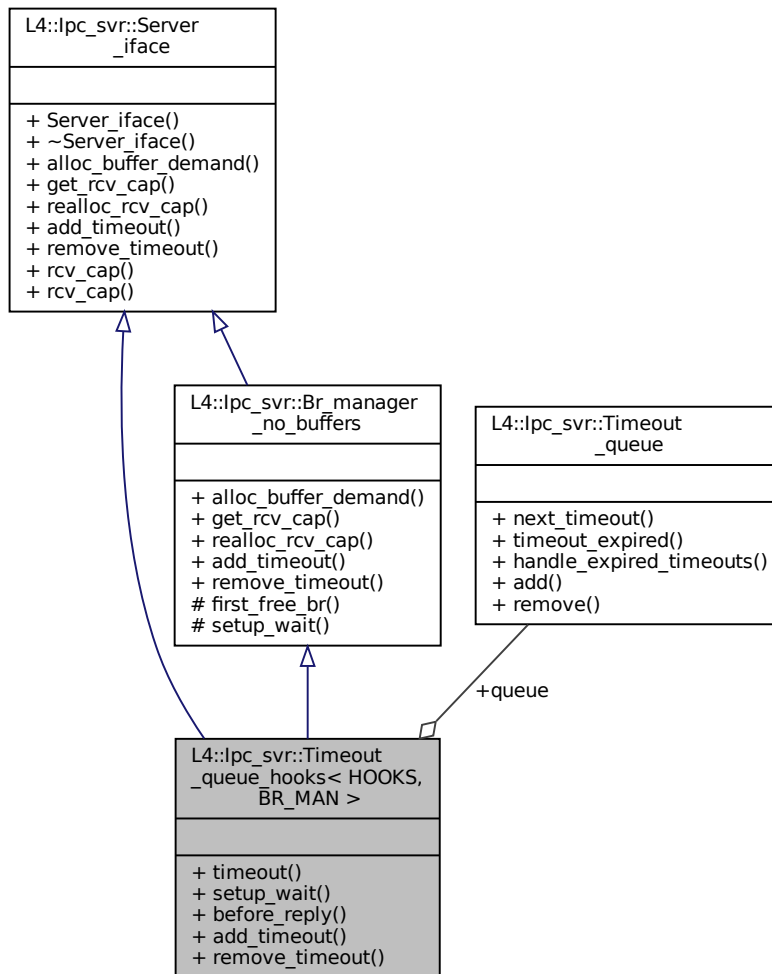
15.144 L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN > Class Template Reference

Loop hooks mixin for integrating a timeout queue into the server loop.

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)`
Add a timeout to the queue for time time.
- `int remove_timeout (Timeout *timeout)`
Remove timeout from the queue.

Data Fields

- `Timeout_queue queue`
Use this timeout queue.

Additional Inherited Members

15.144.1 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](#).

15.144.2 Member Function Documentation

15.144.2.1 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], [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

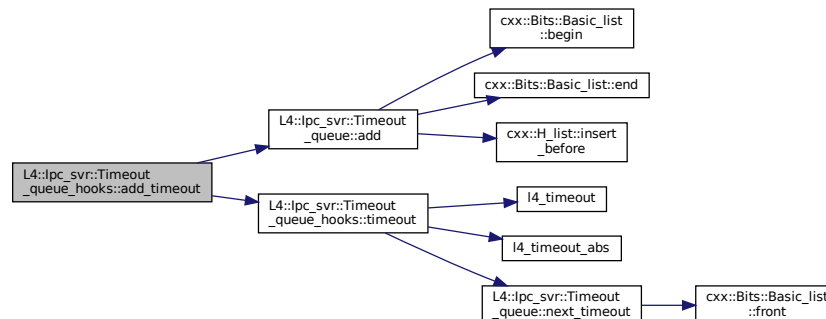
The timeout is automatically dequeued before the [Timeout::expired\(\)](#) function is called

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:



15.144.2.2 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], [virtual]
  
```

Remove timeout from the queue.

Parameters

<i>timeout</i>	The timeout object to be removed from the queue.
----------------	--

Note

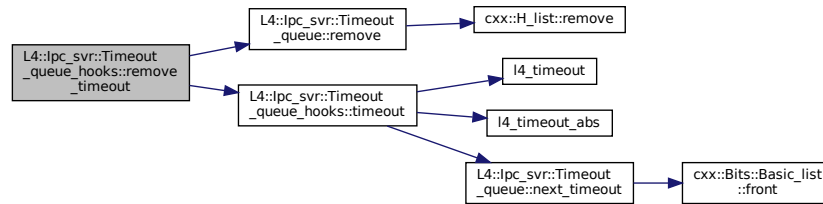
This function may be safely called even if the timeout is not currently enqueued.
in [Timeout::expired\(\)](#) the timeout is already dequeued!

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:



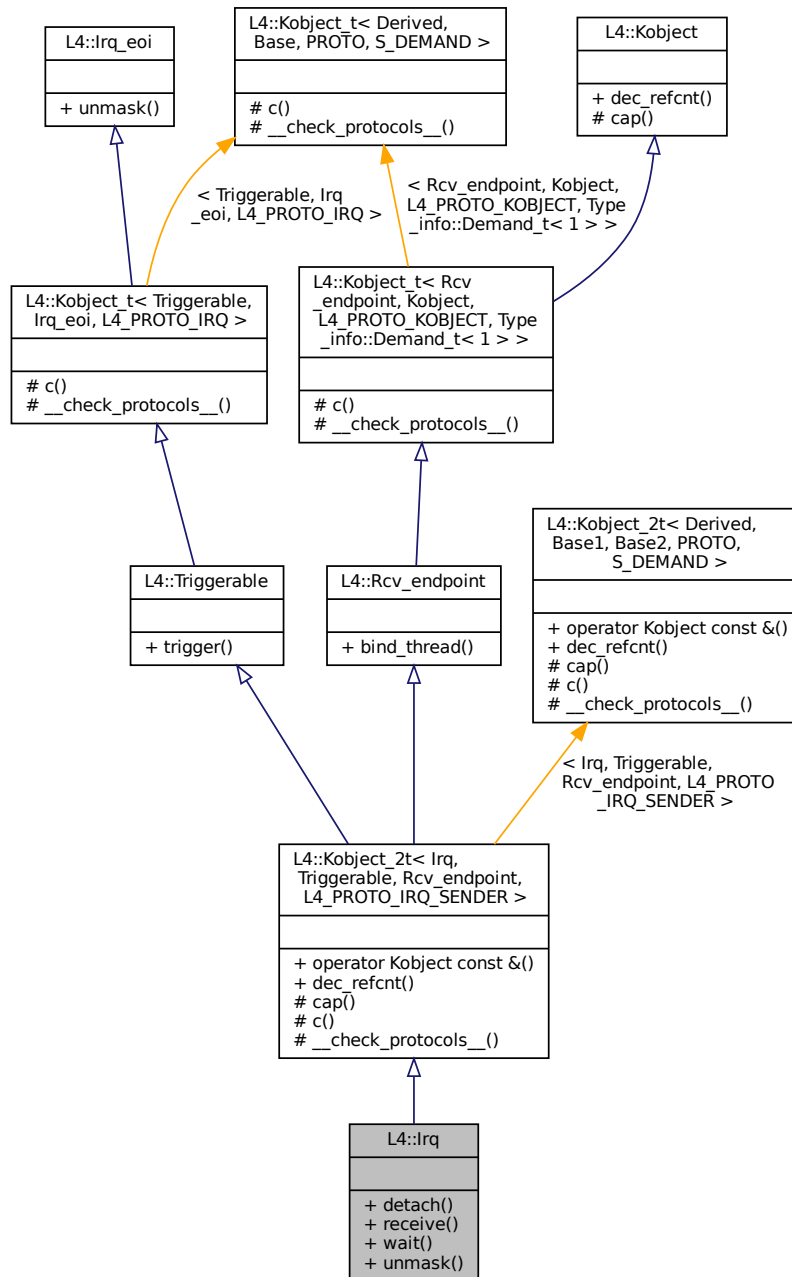
The documentation for this class was generated from the following file:

- `l4/cxx/ipc_timeout_queue`

15.145 L4::Irq Class Reference

C++ [Irq](#) interface.

Collaboration diagram for L4::Irq:



Public Member Functions

- `l4_msgtag_t detach (l4_utcb_t *utcb=l4_utcb()) noexcept`
Detach from this interrupt.
- `l4_msgtag_t receive (l4_timeout_t timeout=L4_IPC_NEVER, l4_utcb_t *utcb=l4_utcb()) noexcept`
Unmask and wait for this IRQ.
- `l4_msgtag_t wait (l4_umword_t *label, l4_timeout_t timeout=L4_IPC_NEVER, l4_utcb_t *utcb=l4_utcb()) noexcept`

Unmask IRQ and (open) wait for any message.

- `l4_msgtag_t unmask (l4_utcb_t *utcb=l4_utcb()) noexcept`

Unmask IRQ.

Additional Inherited Members

15.145.1 Detailed Description

C++ [Irq](#) 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)

[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 [117](#) of file [irq](#).

15.145.2 Member Function Documentation

15.145.2.1 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, usually the UTCB of the calling thread.
-------------	---

Returns

Syscall return tag

Definition at line 129 of file [irq](#).

References [l4_irq_detach_u\(\)](#).

Here is the call graph for this function:

**15.145.2.2 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, usually the UTCB of the calling thread.

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 144 of file [irq](#).

References [l4_irq_receive_u\(\)](#).

Here is the call graph for this function:



15.145.2.3 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, usually the UTCB of the calling thread.
-------------	---

Returns

Syscall return tag for a send-only operation, use [l4_ipc_error\(\)](#) to check for errors (**do not** use [l4_error\(\)](#)).

Note

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 errors, **do not** use [l4_error\(\)](#), because [l4_error\(\)](#) will always return an 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 180 of file [irq](#).

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

Here is the caller graph for this function:



15.145.2.4 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, usually the UTCB of the calling thread.

Returns

Syscall return tag

Definition at line 157 of file [irq](#).

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

Here is the call graph for this function:



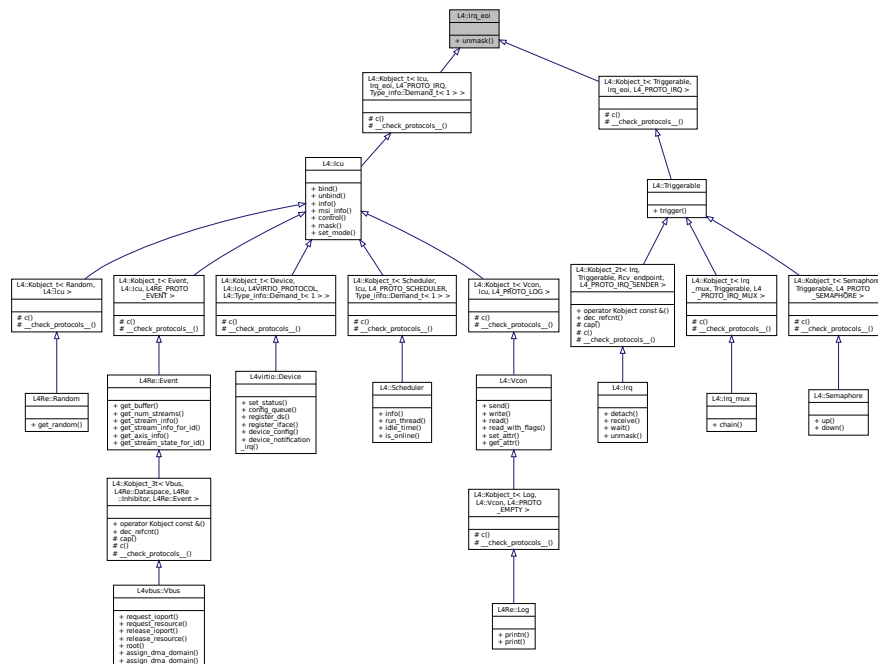
The documentation for this class was generated from the following file:

- [l4/sys/irq](#)

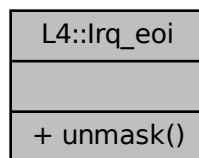
15.146 L4::Irq_eoi Class Reference

Interface for sending an acknowledge message to an object.

Inheritance diagram for L4::lrq_eoi:



Collaboration diagram for L4::lrq_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

Acknowledge the given interrupt line.

15.146.1 Detailed Description

Interface for sending an acknowledge message to an object.

The object is usually an ICU or an IRQ.

See also

L4::lcu, L4::lra

Definition at line 43 of file irq.

15.146.2 Member Function Documentation

15.146.2.1 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]
```

Acknowledge the given interrupt line.

Parameters

	<i>irqnum</i>	The interrupt line that shall be acknowledged.
out	<i>label</i>	If NULL this is a send-only unmask, if not NULL then 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, usually the UTCB of the calling thread.

Returns

Syscall return tag.

Note

If *label* is NULL this function is a send-only operation and there is no return value except for a failed send operation. In this case use [l4_ipc_error\(\)](#) to check for errors, **do not** use [l4_error\(\)](#), because [l4_error\(\)](#) will always return an error.

Definition at line 65 of file [irq](#).

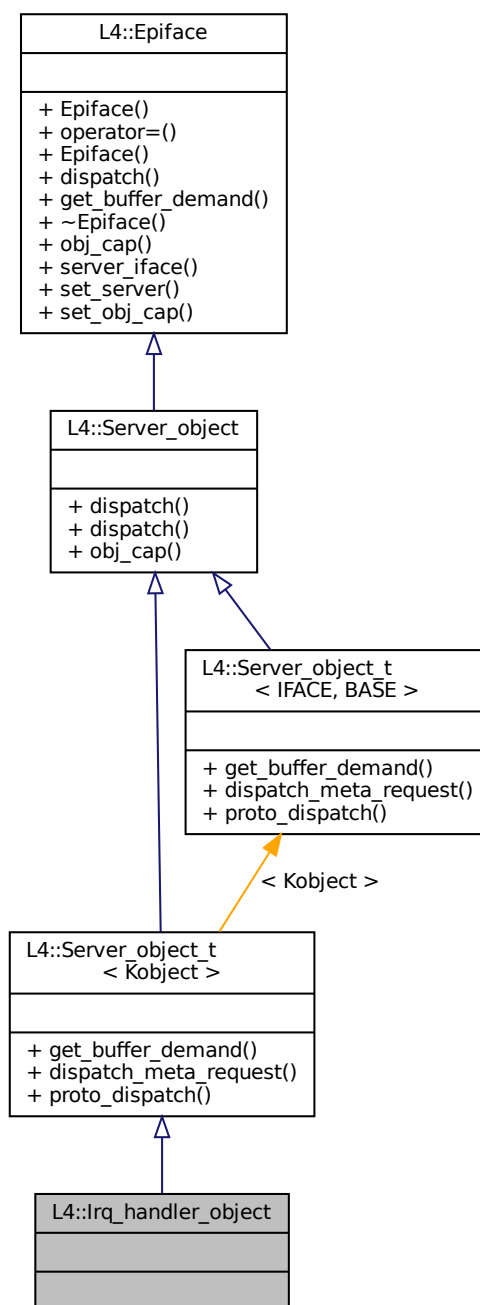
The documentation for this class was generated from the following file:

- [l4/sys/irq](#)

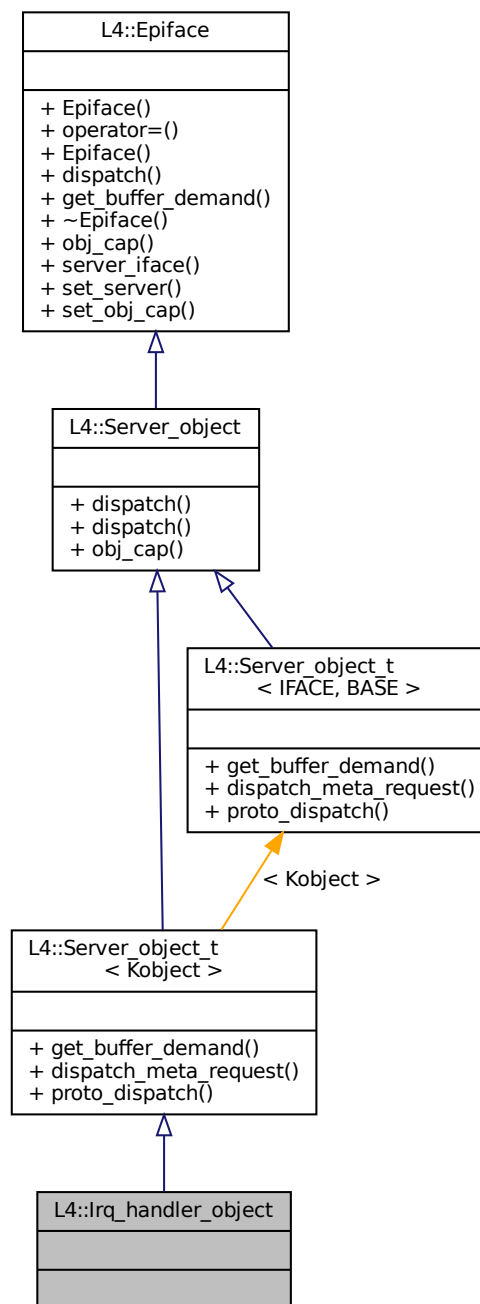
15.147 L4::Irq_handler_object Struct Reference

[Server](#) object base class for handling IRQ messages.

Inheritance diagram for L4::lrq_handler_object:



Collaboration diagram for L4::Irq_handler_object:



Additional Inherited Members

15.147.1 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](#).

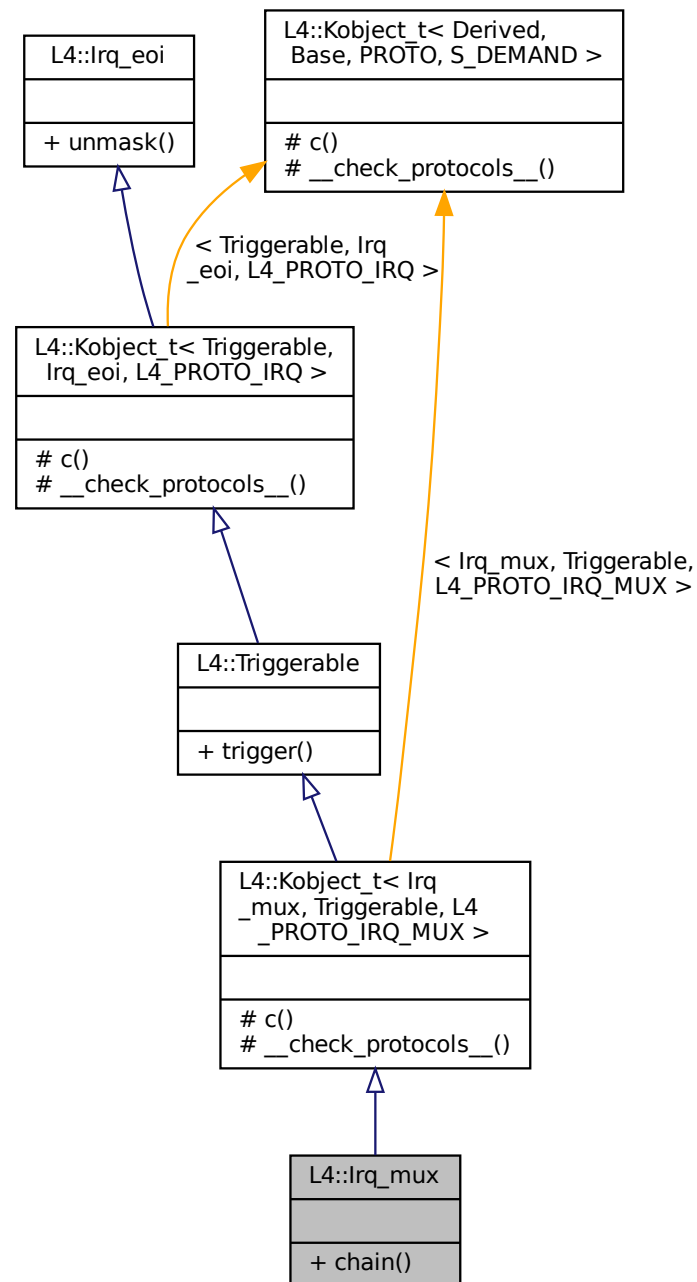
The documentation for this struct was generated from the following file:

- [l4/cxx/ipc_server](#)

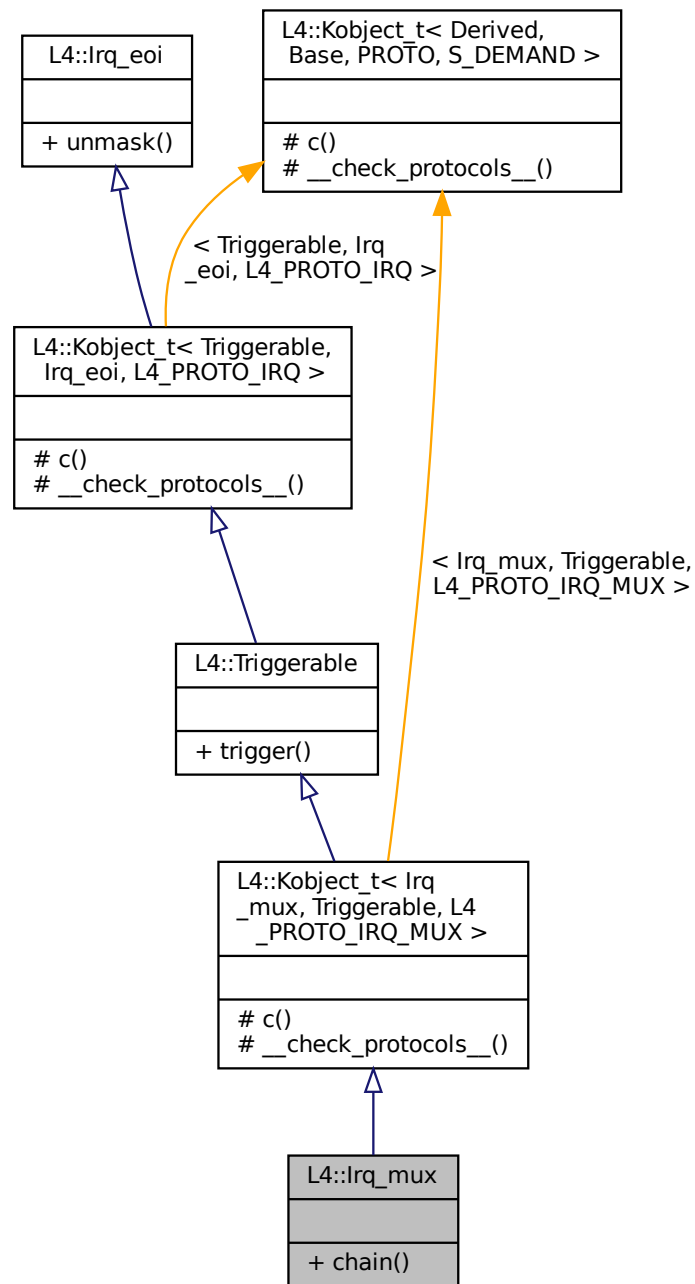
15.148 L4::Irq_mux Struct Reference

IRQ multiplexer for shared IRQs.

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.

Additional Inherited Members

15.148.1 Detailed Description

IRQ multiplexer for shared IRQs.

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 acknowledged.

Definition at line 197 of file [irq](#).

15.148.2 Member Function Documentation

15.148.2.1 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, usually the UTCB of the calling thread.

Returns

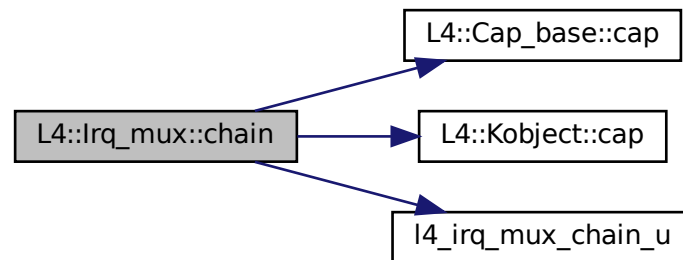
Syscall return tag

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 212 of file [irq](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4_irq_mux_chain_u\(\)](#).

Here is the call graph for this function:



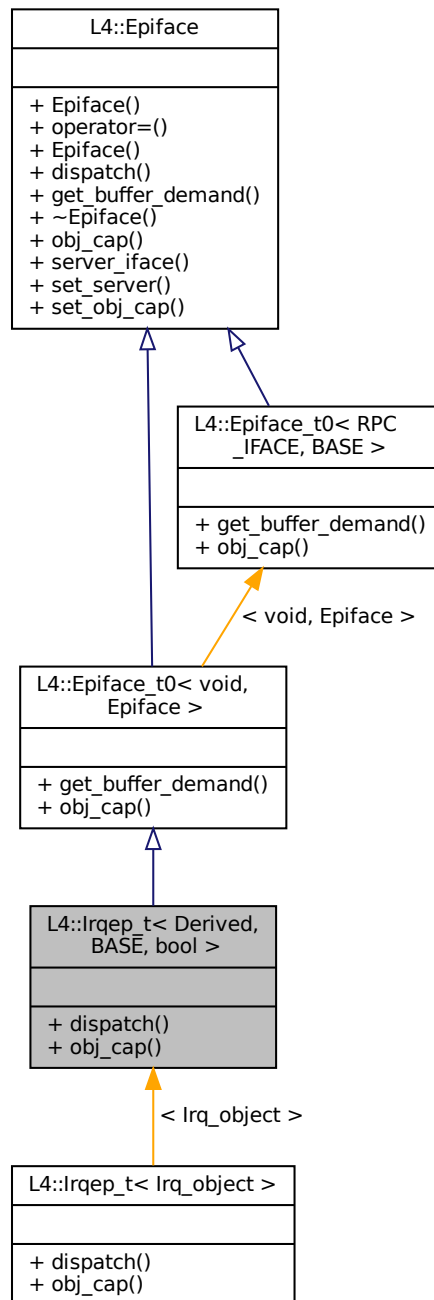
The documentation for this struct was generated from the following file:

- [l4/sys/irq](#)

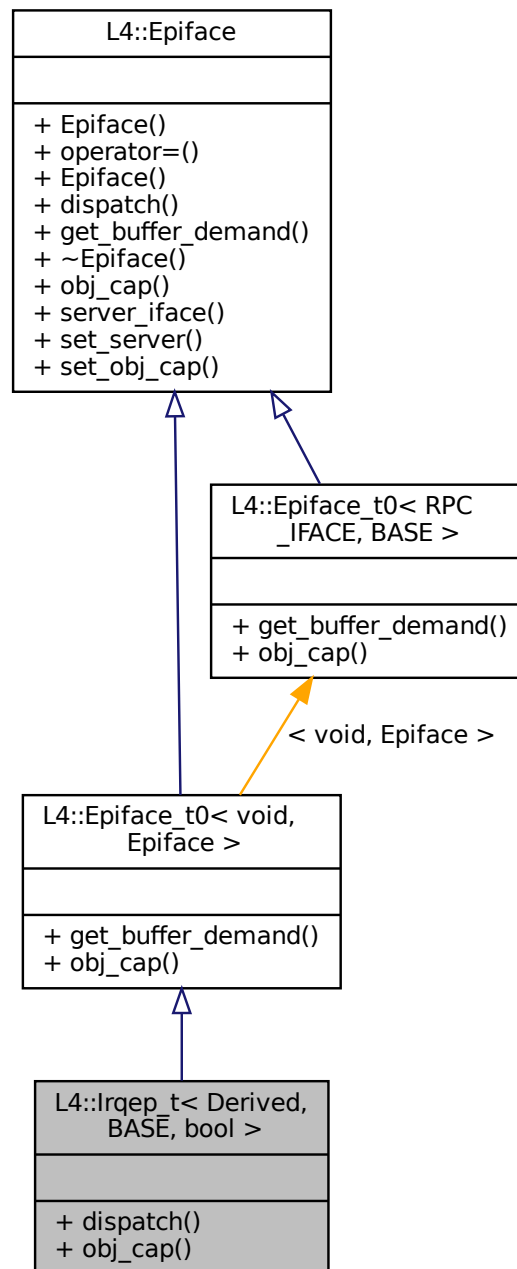
15.149 L4::lrqep_t< Derived, BASE, bool > Struct Template Reference

[Epiface](#) implementation for interrupt handlers.

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.

Additional Inherited Members

15.149.1 Detailed Description

```
template<typename Derived, typename BASE = Epiface, bool = cxx::is_polymorphic<BASE>::value>
struct L4::lrqep_t< Derived, BASE, bool >
```

[Epiface](#) implementation for interrupt handlers.

Template Parameters

<i>Derived</i>	lrq handler implementation class. The class must provide a single function <code>handle_irq()</code> .
<i>BASE</i>	Base Epiface class.

Definition at line 290 of file [ipc_epiface](#).

15.149.2 Member Function Documentation

15.149.2.1 `dispatch()`

```
template<typename Derived , typename BASE = Epiface, bool = cxx::is_polymorphic<BASE>::value>
L4_msgtag_t L4::lrqep_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 292 of file [ipc_epiface](#).

References [L4_ENOREPLY](#), and [l4_msgtag\(\)](#).

Here is the call graph for this function:



15.149.2.2 obj_cap()

```
template<typename Derived , typename BASE = Epiface, bool = cxx::is_polymorphic<BASE>::value>
Cap<L4::Irq> L4::lrqep_t< Derived, BASE, bool >::obj_cap ( ) const [inline]
```

Get the (typed) capability to this object.

Returns

[lrq](#) capability for the kernel object behind the server.

Definition at line [302](#) of file [ipc_epiface](#).

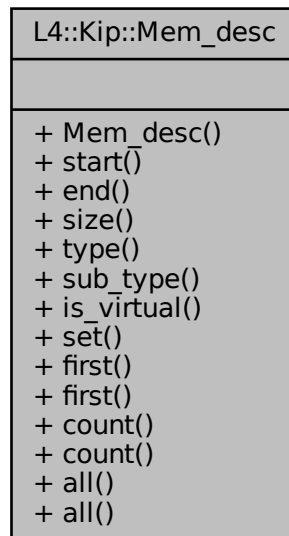
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_epiface](#)

15.150 L4::Kip::Mem_desc Class Reference

Memory descriptors stored in the kernel interface page.

Collaboration diagram for L4::Kip::Mem_desc:



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.

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.

15.150.1 Detailed Description

Memory descriptors stored in the kernel interface page.

Include File

```
#include <l4/sys/kip>
```

Definition at line 53 of file [kip](#).

15.150.2 Member Enumeration Documentation

15.150.2.1 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](#).

15.150.2.2 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](#).

15.150.3 Constructor & Destructor Documentation**15.150.3.1 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 166 of file [kip](#).

15.150.4 Member Function Documentation**15.150.4.1 all() [1/2]**

```
static cxx::static_vector<Mem_desc> L4::Kip::Mem_desc::all (
    void * kip ) [inline], [static]
```

Return enumerable list of memory descriptors.

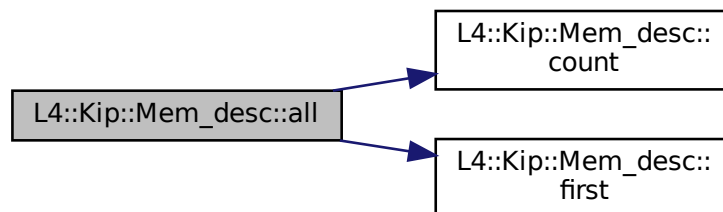
Parameters

<i>kip</i>	Pointer to the kernel info page.
------------	----------------------------------

Definition at line 150 of file [kip](#).

References [count\(\)](#), and [first\(\)](#).

Here is the call graph for this function:



15.150.4.2 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.

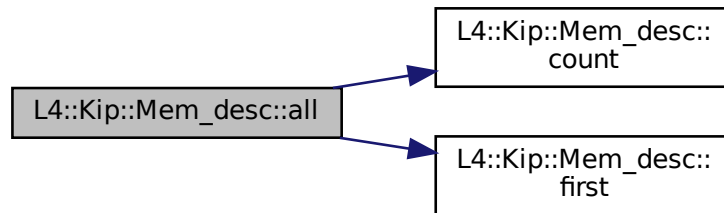
Parameters

<i>kip</i>	Pointer to the kernel info page.
------------	----------------------------------

Definition at line 139 of file [kip](#).

References [count\(\)](#), and [first\(\)](#).

Here is the call graph for this function:



15.150.4.3 `count()` [1/2]

```
static void L4::Kip::Mem_desc::count (
    void * kip,
    unsigned count ) [inline], [static], [noexcept]
```

Set number of memory descriptors.

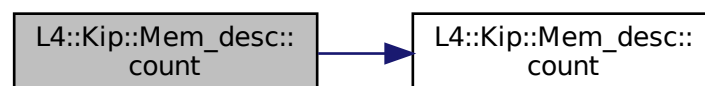
Parameters

<i>kip</i>	Pointer to the kernel info page
<i>count</i>	Number of memory descriptors

Definition at line 128 of file [kip](#).

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

Here is the call graph for this function:



15.150.4.4 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
------------	---------------------------------

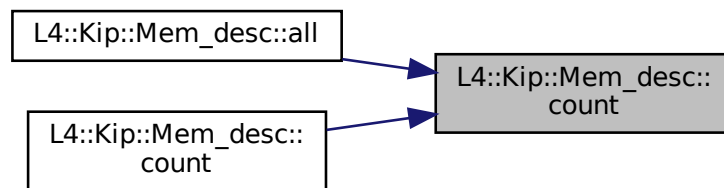
Returns

Number of memory descriptors in the kernel info page.

Definition at line 116 of file [kip](#).

Referenced by [all\(\)](#), and [count\(\)](#).

Here is the caller graph for this function:



15.150.4.5 end()

```
unsigned long L4::Kip::Mem_desc::end ( ) const [inline], [noexcept]
```

Return end address of memory descriptor.

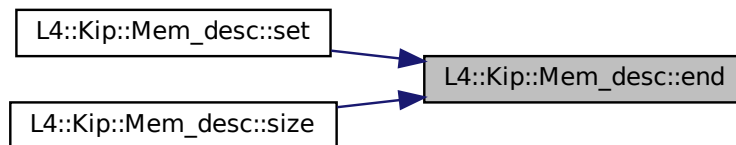
Returns

End address of memory descriptor

Definition at line 184 of file [kip](#).

Referenced by [set\(\)](#), and [size\(\)](#).

Here is the caller graph for this function:

**15.150.4.6 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
------------	---------------------------------

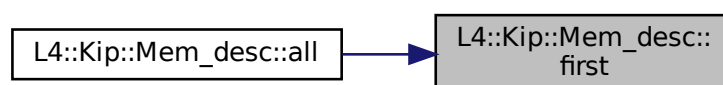
Returns

First memory descriptor stored in the kernel info page

Definition at line 97 of file [kip](#).

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

Here is the caller graph for this function:



15.150.4.7 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 213 of file [kip](#).

15.150.4.8 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.

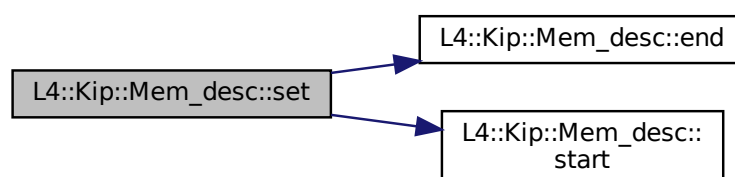
Parameters

<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 224 of file [kip](#).

References [end\(\)](#), and [start\(\)](#).

Here is the call graph for this function:



15.150.4.9 size()

```
unsigned long L4::Kip::Mem_desc::size ( ) const [inline], [noexcept]
```

Return size of region described by the memory descriptor.

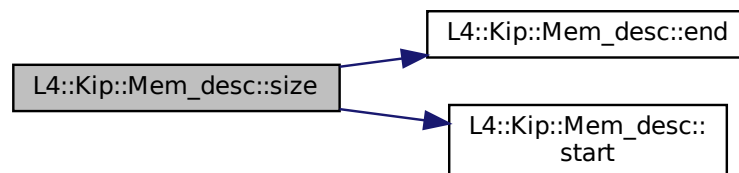
Returns

Size of the region described by the memory descriptor

Definition at line 191 of file [kip](#).

References [end\(\)](#), and [start\(\)](#).

Here is the call graph for this function:



15.150.4.10 start()

```
unsigned long L4::Kip::Mem_desc::start ( ) const [inline], [noexcept]
```

Return start address of memory descriptor.

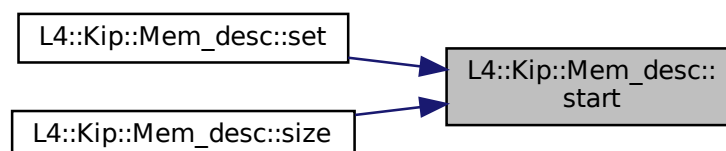
Returns

Start address of memory descriptor

Definition at line 177 of file [kip](#).

Referenced by [set\(\)](#), and [size\(\)](#).

Here is the caller graph for this function:



```
unsigned char L4::Kip::Mem_desc::sub_type ( ) const [inline], [noexcept]
```

Returns

Definition at line 205 of file kip.

```
Mem_type L4::Kip::Mem_desc::type ( ) const [inline], [noexcept]
```

Returns

Definition at line 198 of file kip.

- l4/sys/kip

Base class for all kinds of kernel objects and remote objects, referenced by capabilities.

[illegible]

L4::Kobject
+ dec_refcnt() # cap()

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.

15.151.1 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`.

15.151.2 Member Function Documentation

15.151.2.1 `cap()`

```
l4_cap_idx_t L4::Kobject::cap ( ) const [inline], [protected], [noexcept]
```

Return capability selector.

Returns

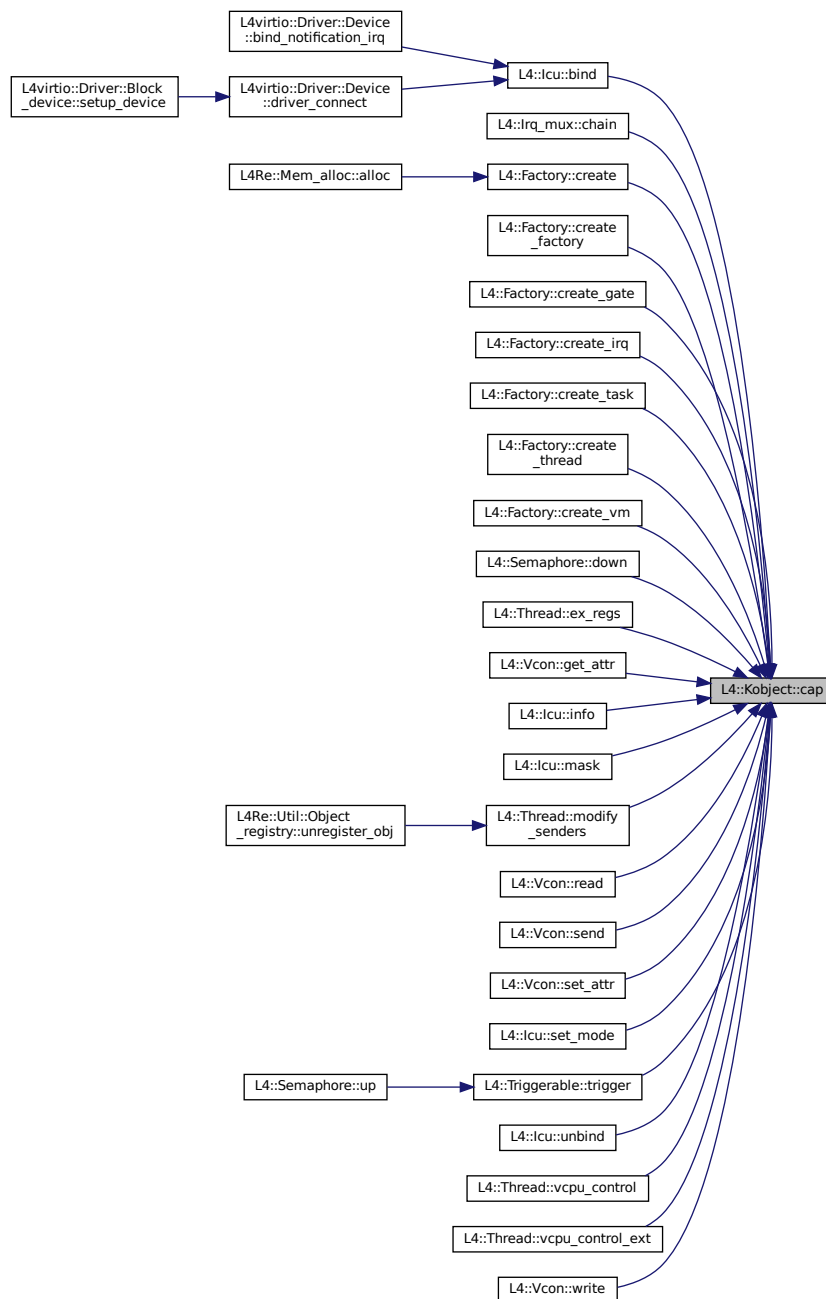
Capability selector.

This method is for derived classes to gain access to the actual capability selector.

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

Referenced by [L4::Icu::bind\(\)](#), [L4::Irq_mux::chain\(\)](#), [L4::Factory::create\(\)](#), [L4::Factory::create_factory\(\)](#), [L4::Factory::create_gate\(\)](#), [L4::Factory::create_irq\(\)](#), [L4::Factory::create_task\(\)](#), [L4::Factory::create_thread\(\)](#), [L4::Factory::create_vm\(\)](#), [L4::Semaphore::down\(\)](#), [L4::Thread::ex_regs\(\)](#), [L4::Vcon::get_attr\(\)](#), [L4::Icu::info\(\)](#), [L4::Icu::mask\(\)](#), [L4::Thread::modify_senders\(\)](#), [L4::Vcon::read\(\)](#), [L4::Vcon::send\(\)](#), [L4::Vcon::set_attr\(\)](#), [L4::Icu::set_mode\(\)](#), [L4::Triggerable::trigger\(\)](#), [L4::Icu::unbind\(\)](#), [L4::Thread::vcpu_control\(\)](#), [L4::Thread::vcpu_control_ext\(\)](#), and [L4::Vcon::write\(\)](#).

Here is the caller graph for this function:



15.151.2.2 `dec_refcnt()`

```
l4_msgtag_t L4::Kobject::dec_refcnt (
    l4_mword_t diff,
    l4_utcb_t * utcb = l4_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, usually the UTCB of the calling thread.

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.

Definition at line 104 of file [kobject](#).

The documentation for this class was generated from the following file:

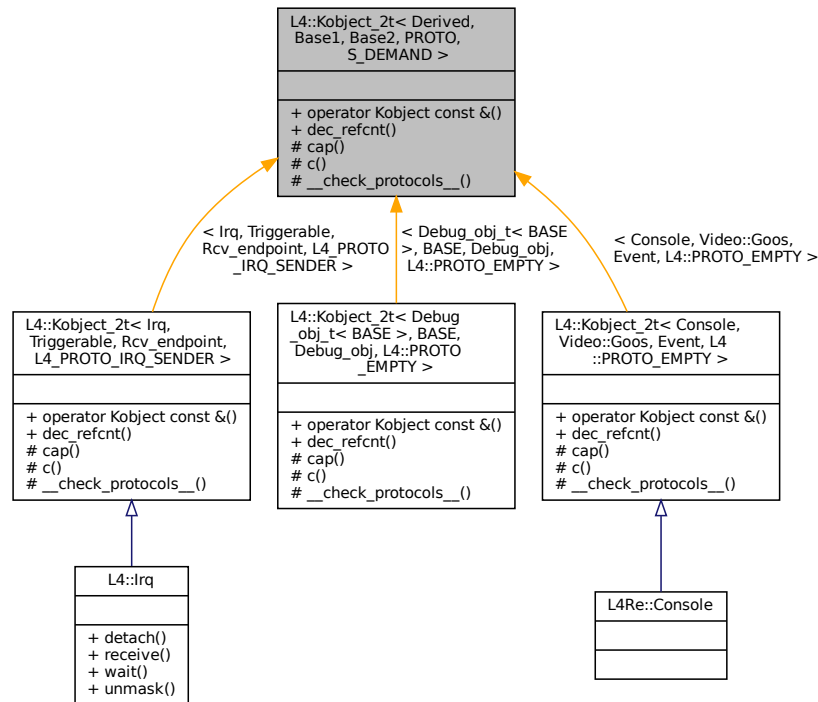
- [l4/sys/kobject](#)

15.152 `L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >` Class Template Reference

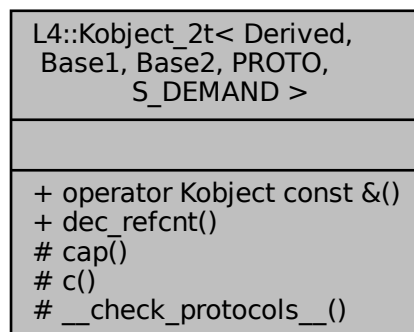
Helper class to create an [L4Re](#) interface class that is derived from two base classes (see [L4::Kobject_t](#)).

```
#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`

15.152.1 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 835 of file [__typeinfo.h](#).

15.152.2 Member Typedef Documentation

15.152.2.1 __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 841 of file [__typeinfo.h](#).

15.152.2.2 __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 849 of file [__typeinfo.h](#).

15.152.2.3 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 839 of file [__typeinfo.h](#).

15.152.3 Member Function Documentation

15.152.3.1 __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]
```

Definition at line 852 of file [__typeinfo.h](#).

15.152.3.2 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 871 of file [__typeinfo.h](#).

The documentation for this class was generated from the following file:

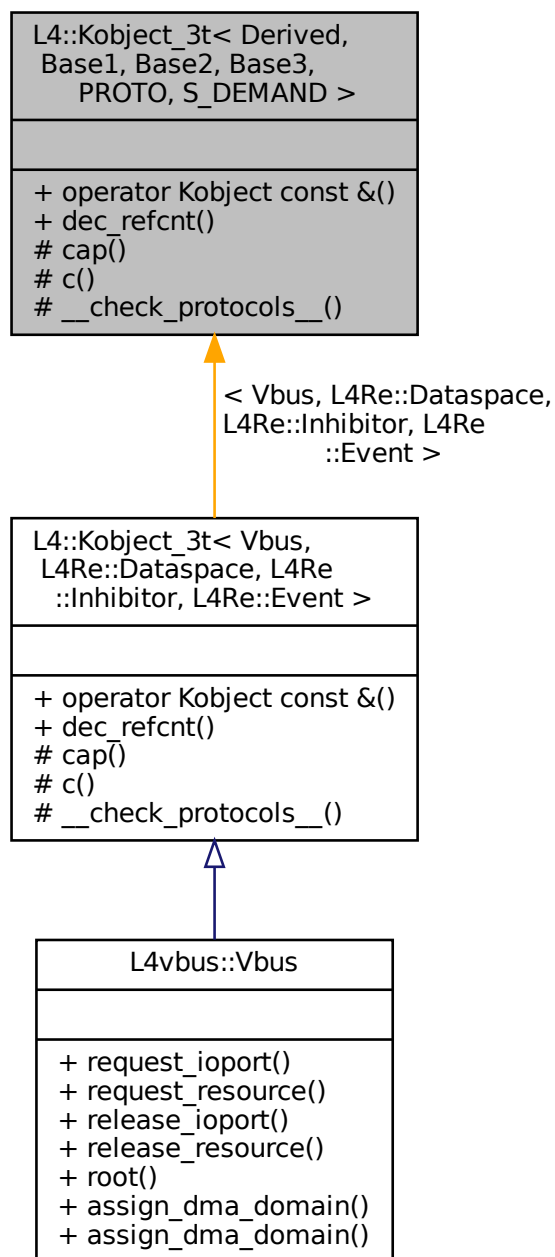
- [l4/sys/__typeinfo.h](#)

15.153 L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND > Struct Template Reference

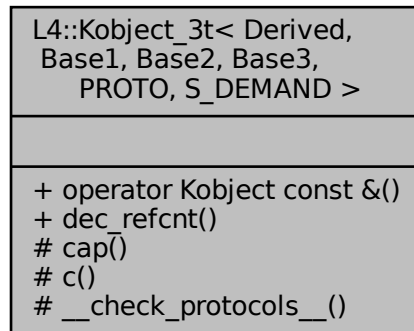
Helper class to create an [L4Re](#) interface class that is derived from three base classes (see [L4::Kobject_t](#)).

```
#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

15.153.1 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 automatically included.

See also

[L4::Kobject_t](#), [L4::Kobject_2t](#), [L4::Kobject_0t](#), [L4::Kobject_x](#)

Definition at line 936 of file [__typeinfo.h](#).

15.153.2 Member Typedef Documentation

15.153.2.1 __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 942 of file [__typeinfo.h](#).

15.153.2.2 __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 953 of file [__typeinfo.h](#).

15.153.2.3 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 940 of file [__typeinfo.h](#).

15.153.3 Member Function Documentation

15.153.3.1 __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]
```

Definition at line 956 of file [__typeinfo.h](#).

15.153.3.2 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 984 of file [__typeinfo.h](#).

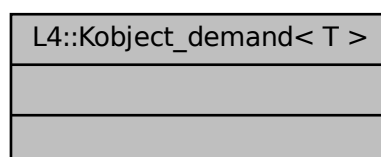
The documentation for this struct was generated from the following file:

- [l4/sys/__typeinfo.h](#)

15.154 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 >:



15.154.1 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 1035 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

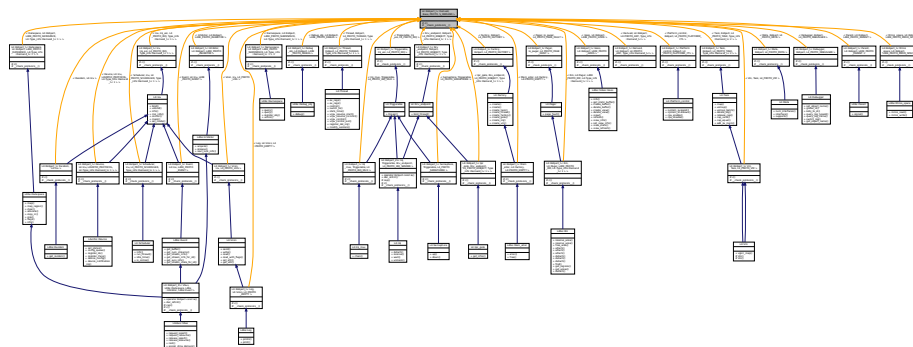
- [l4/sys/__typeinfo.h](#)

15.155 L4::Kobject_t< Derived, Base, PROTO, S_DEMAND > Class Template Reference

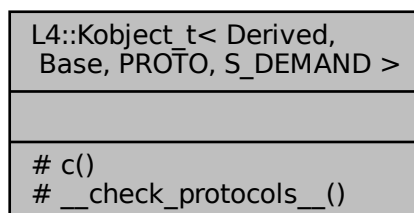
Helper class to create an [L4Re](#) interface class that is derived from a single base class.

```
#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::lface< PROTO, Derived > [__lface](#)
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::lface_list< [__lface](#) >, typename Base::__lface_list > [__lface_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.

15.155.1 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:

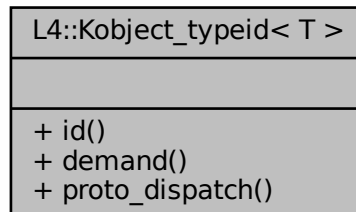
- [l4/sys/__typeinfo.h](#)

15.156 L4::Kobject_typeid< T > Struct Template Reference

[Meta](#) object for handling access to type information of Kobjects.

```
#include <__typeinfo.h>
```

Collaboration diagram for L4::Kobject_typeid< T >:



Public Types

- typedef T::__Kobject_typeid::Demand [Demand](#)
Data type expressing the static demand of receive buffers in a server.

Static Public Member Functions

- static [Type_info](#) const * [id](#) () noexcept
Get a pointer to teh [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.

15.156.1 Detailed Description

```
template<typename T>
struct L4::Kobject_typeid< T >
```

[Meta](#) object for handling access to type information of Kobjects.

Template Parameters

<i>T</i>	The data type derived from Kobject , usually using Kobject_t .
----------	--

Definition at line 620 of file [__typeinfo.h](#).

15.156.2 Member Typedef Documentation

15.156.2.1 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](#).

15.156.3 Member Function Documentation

15.156.3.1 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](#).

15.156.3.2 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:



15.156.3.3 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

Parameters

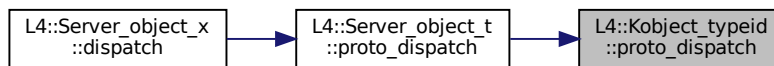
<i>a1</i>	The second argument passed to self->p_dispatch()
<i>a2</i>	The third argument passed to self->p_dispatch()

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:

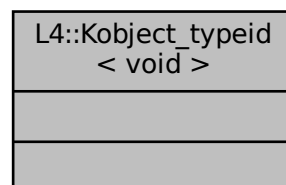
- [l4/sys/__typeinfo.h](#)

15.157 L4::Kobject_typeid< void > Struct Reference

Minimalistic ID for void interface.

```
#include <__typeinfo.h>
```

Collaboration diagram for L4::Kobject_typeid< void >:



15.157.1 Detailed Description

Minimalistic ID for `void` interface.

Definition at line 677 of file [__typeinfo.h](#).

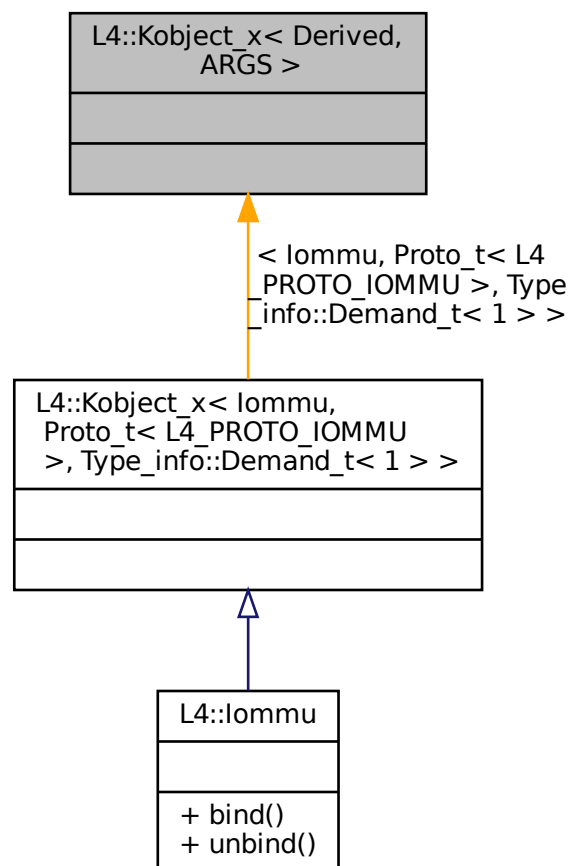
The documentation for this struct was generated from the following file:

- [l4/sys/__typeinfo.h](#)

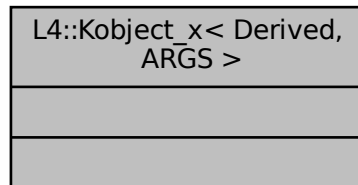
15.158 L4::Kobject_x< Derived, ARGS > Struct Template Reference

Generic [Kobject](#) inheritance template.

Inheritance diagram for L4::Kobject_x< Derived, ARGS >:



Collaboration diagram for L4::Kobject_x< Derived, ARGS >:



15.158.1 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 1198 of file [__typeinfo.h](#).

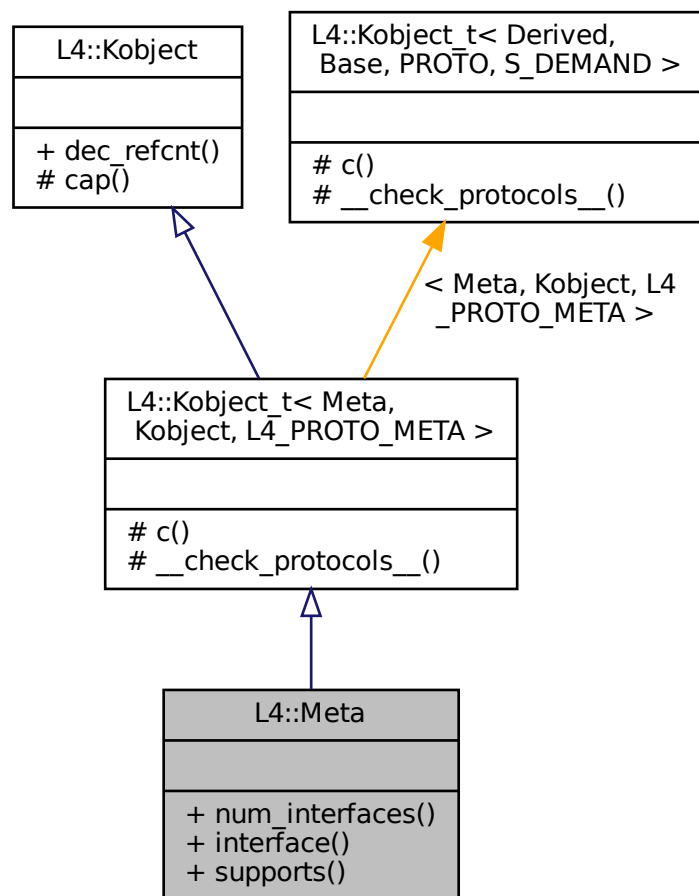
The documentation for this struct was generated from the following file:

- [l4/sys/__typeinfo.h](#)

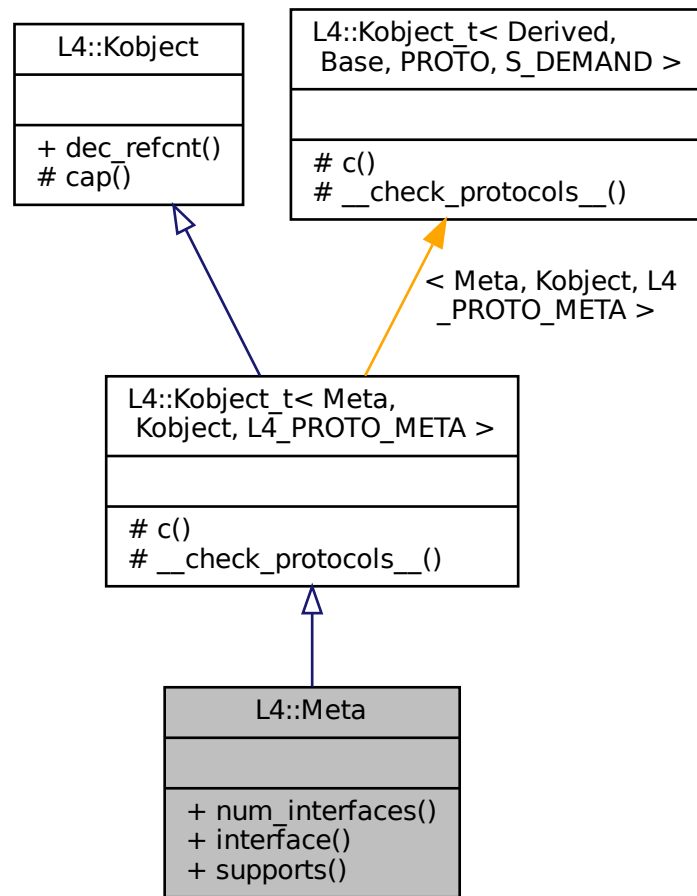
15.159 L4::Meta Class Reference

[Meta](#) interface that shall be implemented by each [L4Re](#) object and gives access to the dynamic type information for [L4Re](#) objects.

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).

Additional Inherited Members

15.159.1 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](#).

15.159.2 Member Function Documentation

15.159.2.1 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. <code>idx</code> must be ≥ 0 and $<$ the return value of num_interfaces() .
out	<i>proto</i>	The protocol number for interface <code>idx</code> .
out	<i>name</i>	The protocol name for interface <code>idx</code> .

Return values

l4_msgtag_t::label()	≥ 0 Successful; see <code>proto</code> and <code>name</code> .
l4_msgtag_t::label()	< 0 Error code.

15.159.2.2 num_interfaces()

```
l4_msgtag_t L4::Meta::num_interfaces ( )
```

Get the number of interfaces implemented by this object.

Return values

l4_msgtag_t::label()	≥ 0 The number of supported interfaces.
l4_msgtag_t::label()	< 0 Error code of the occurred error.

15.159.2.3 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:

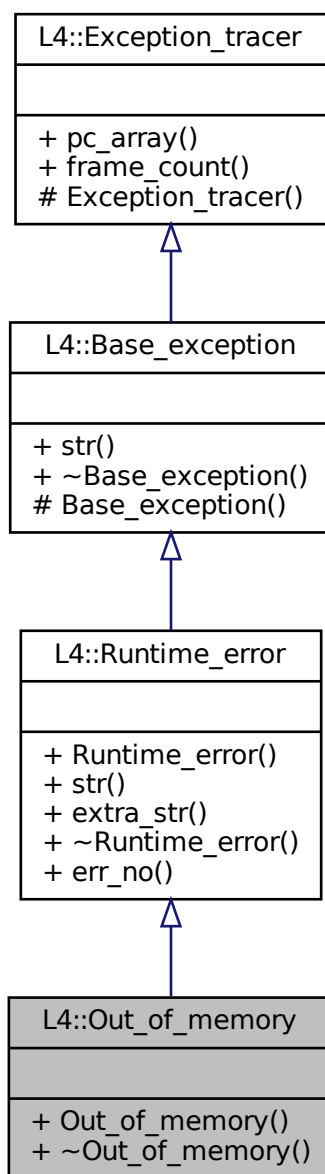
- [l4/sys/meta](#)

15.160 L4::Out_of_memory Class Reference

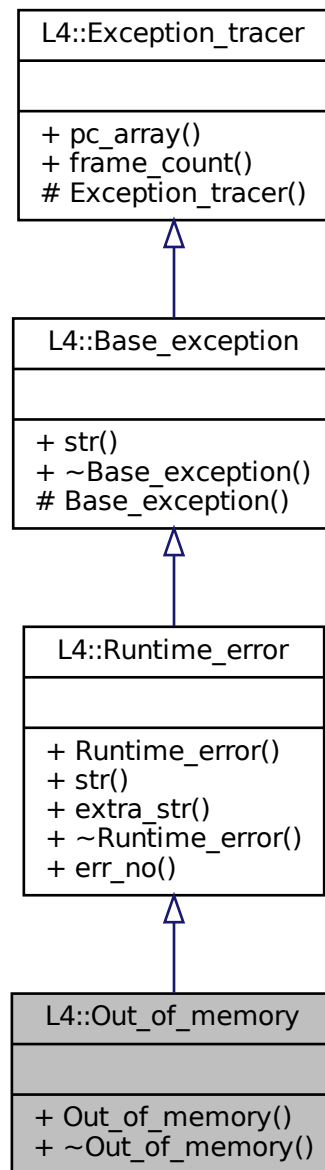
[Exception](#) signalling insufficient memory.

```
#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="") throw ()
Create an out-of-memory exception.
- [~Out_of_memory](#) () throw ()
Destruction.

Additional Inherited Members

15.160.1 Detailed Description

[Exception](#) signalling insufficient memory.

Definition at line [188](#) of file [exceptions](#).

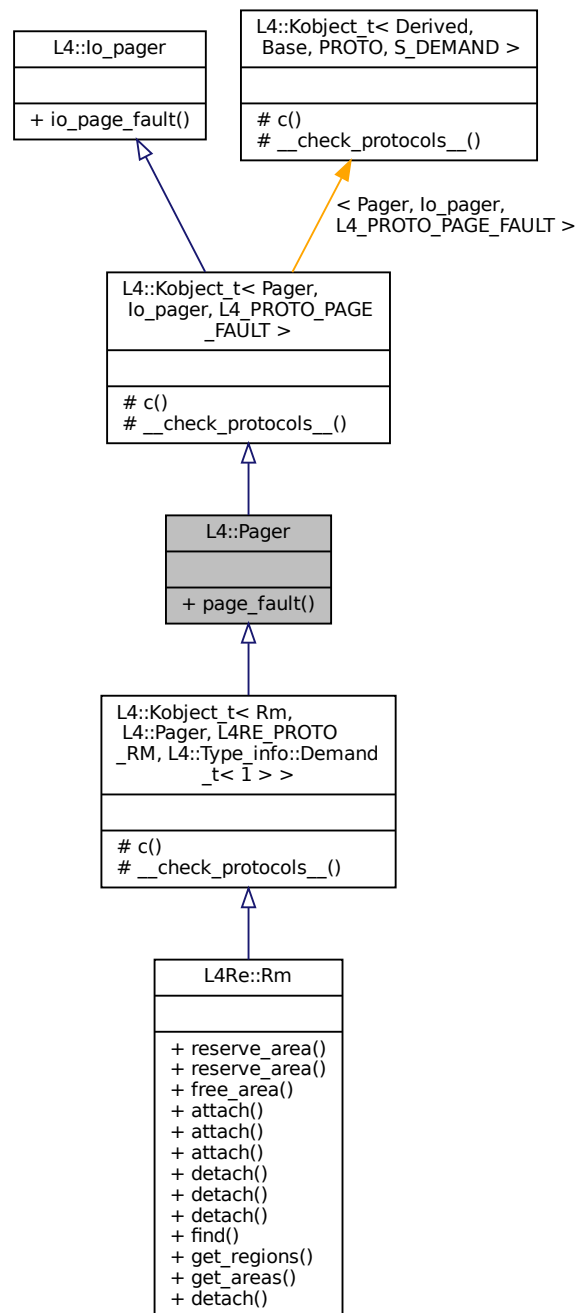
The documentation for this class was generated from the following file:

- [l4/cxx/exceptions](#)

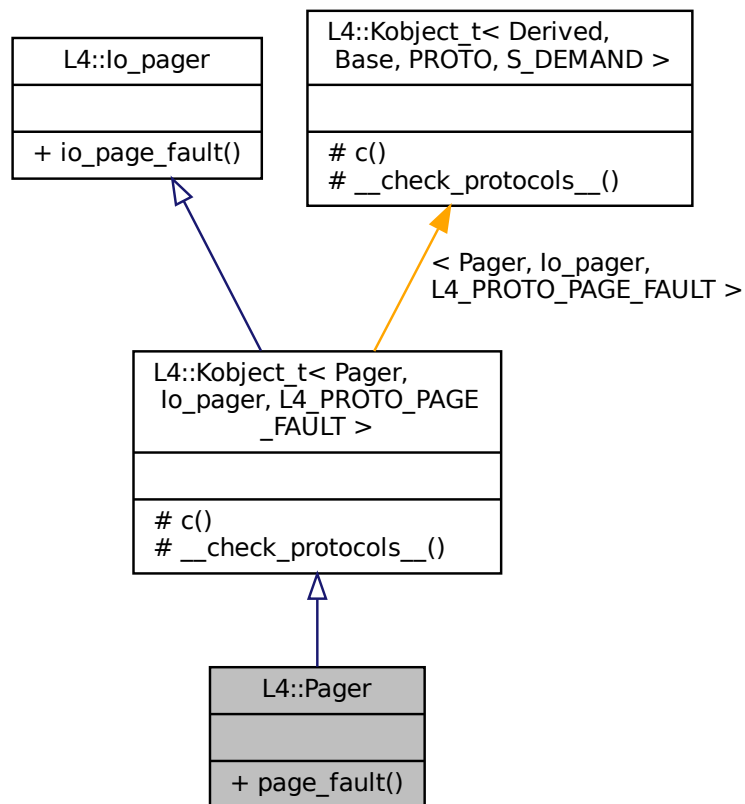
15.161 L4::Pager Class Reference

[Pager](#) interface including the [lo_pager](#) interface.

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.

Additional Inherited Members

15.161.1 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](#).

15.161.2 Member Function Documentation

15.161.2.1 `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.

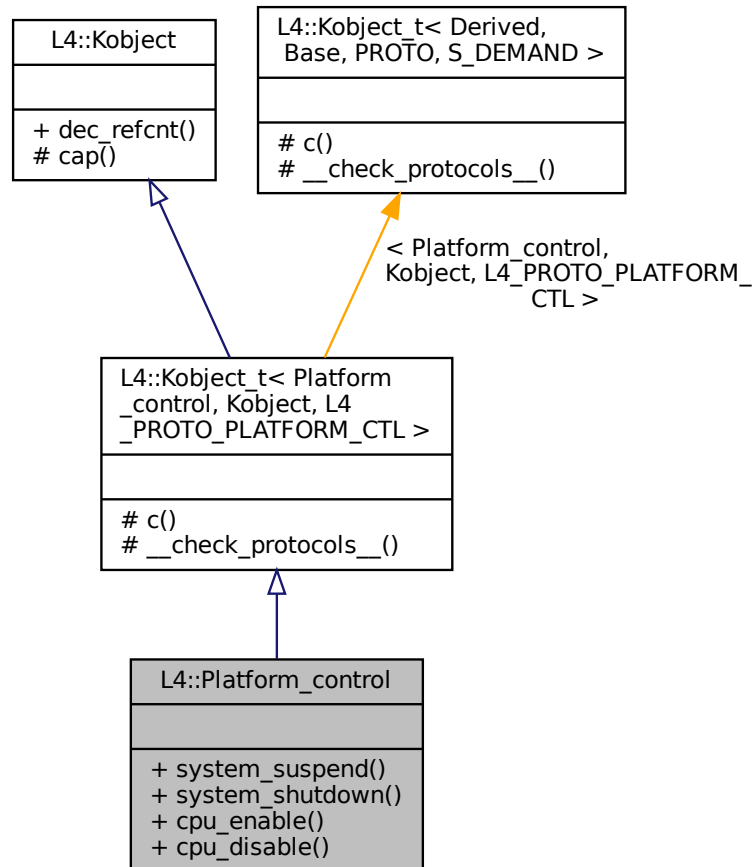
The documentation for this class was generated from the following file:

- [l4/sys/pager](#)

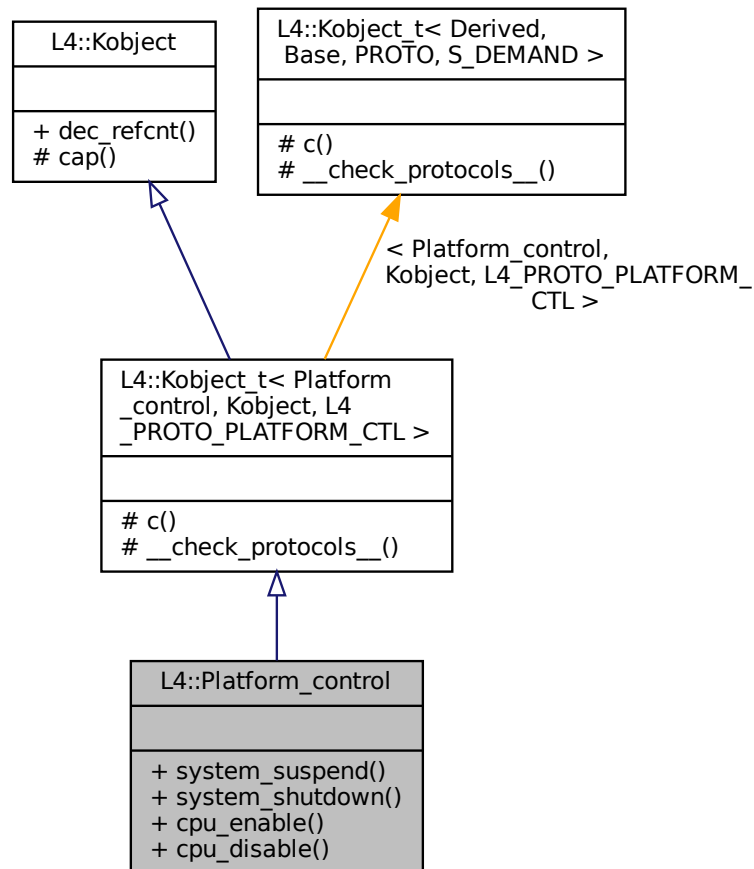
15.162 L4::Platform_control Class Reference

[L4](#) C++ interface for controlling platform-wide properties.

Inheritance diagram for L4::Platform_control:



Collaboration diagram for L4::Platform_control:



Public Types

- enum `Opcode` { `Suspend` = `L4_PLATFORM_CTL_SYS_SUSPEND_OP` , `Shutdown` = `L4_PLATFORM_CTL_SYS_SHUTDOWN_OP` , `Cpu_enable` = `L4_PLATFORM_CTL_CPU_ENABLE_OP` , `Cpu_disable` = `L4_PLATFORM_CTL_CPU_DISABLE_OP` }

Opcodes for platform-control object.

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_enable (l4_umword_t phys_id)`
Enable an offline CPU.
- `l4_msgtag_t cpu_disable (l4_umword_t phys_id)`
Disable an online CPU.

Additional Inherited Members

15.162.1 Detailed Description

[L4](#) C++ interface for controlling platform-wide properties.

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 [46](#) of file [platform_control](#).

15.162.2 Member Enumeration Documentation

15.162.2.1 Opcode

```
enum L4::Platform\_control::Opcode
```

Opcodes for platform-control object.

Enumerator

Suspend	Opcode for suspend to RAM.
Shutdown	Opcode for shutdown / reboot.
Cpu_enable	Opcode to enable a CPU.
Cpu_disable	Opcode to disable a CPU.

Definition at line [51](#) of file [platform_control](#).

15.162.3 Member Function Documentation

15.162.3.1 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

15.162.3.2 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

15.162.3.3 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
---------------	-------------------------------

15.162.3.4 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.
---------------	--

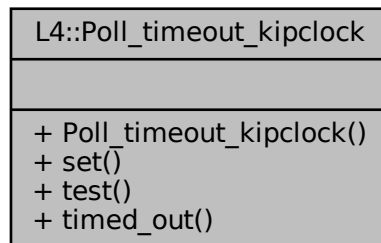
The documentation for this class was generated from the following file:

- [l4/sys/platform_control](#)

15.163 L4::Poll_timeout_kipclock Class Reference

A polling timeout based on the [L4Re](#) clock.

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.

15.163.1 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](#).

15.163.2 Constructor & Destructor Documentation

15.163.2.1 Poll_timeout_kipclock()

```
L4::Poll_timeout_kipclock::Poll_timeout_kipclock (
    unsigned poll_time_us ) [inline]
```

Initialise relative timeout in microseconds.

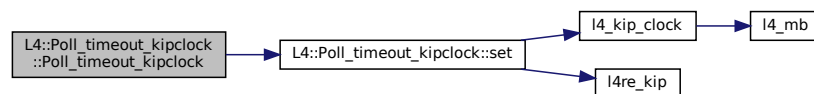
Parameters

<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:



15.163.3 Member Function Documentation

15.163.3.1 set()

```
void L4::Poll_timeout_kipclock::set (
    unsigned poll_time_us ) [inline]
```

(Re-)Set relative timeout in microseconds

Parameters

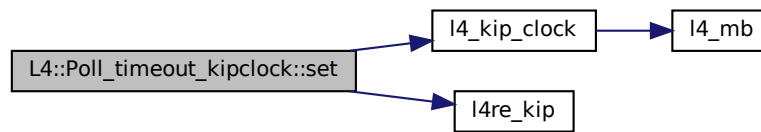
<i>poll_time_us</i>	Polling timeout in microseconds.
---------------------	----------------------------------

Definition at line 54 of file [poll_timeout_kipclock](#).

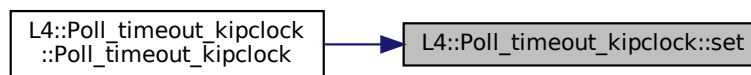
References [l4_kip_clock\(\)](#), and [l4re_kip\(\)](#).

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

Here is the call graph for this function:



Here is the caller graph for this function:



15.163.3.2 test()

```
bool L4::Poll_timeout_kipclock::test (
    bool expression = true ) [inline]
```

Test whether timeout has expired.

Parameters

<i>expression</i>	Optional expression.
-------------------	----------------------

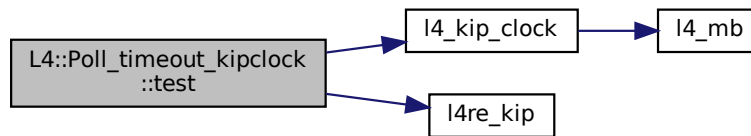
Return values

<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:



15.163.3.3 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:

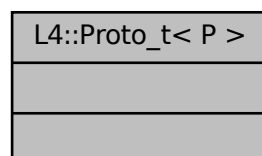
- `l4/re/util/poll_timeout_kipclock`

15.164 L4::Proto_t< P > Struct Template Reference

Data type for defining protocol numbers.

```
#include <__typeinfo.h>
```

Collaboration diagram for `L4::Proto_t< P >`:



15.164.1 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 1182 of file [__typeinfo.h](#).

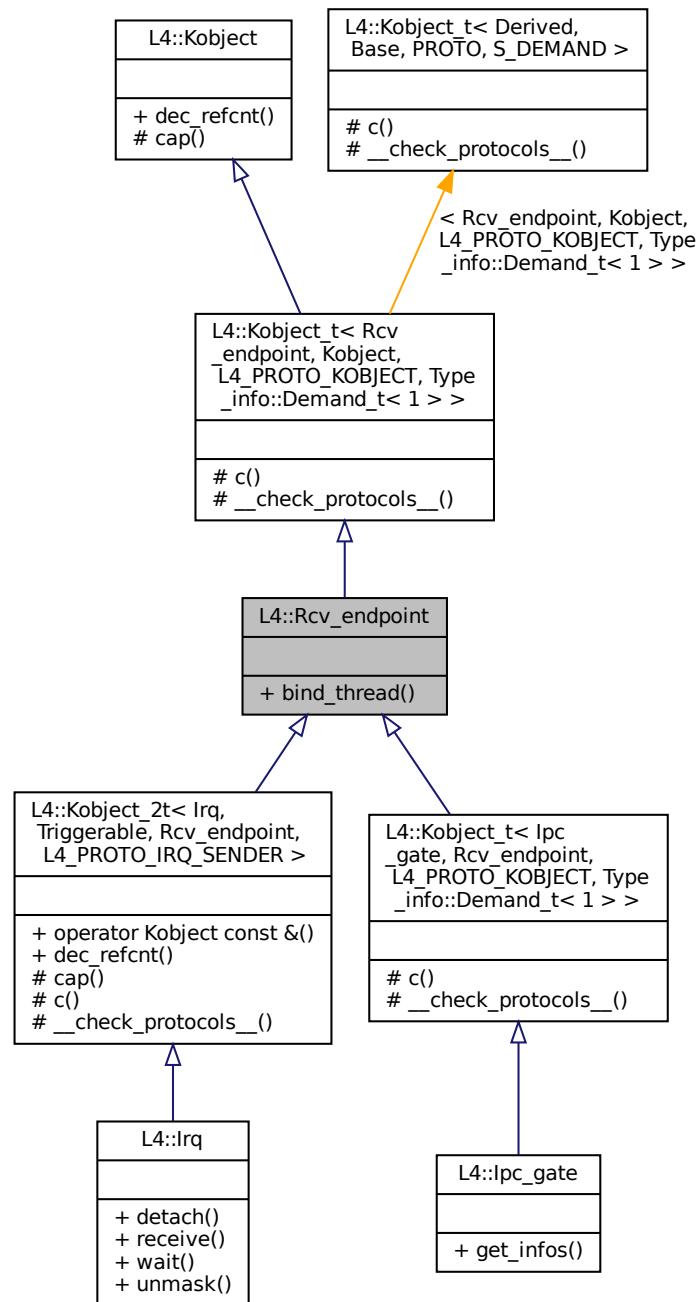
The documentation for this struct was generated from the following file:

- [l4/sys/__typeinfo.h](#)

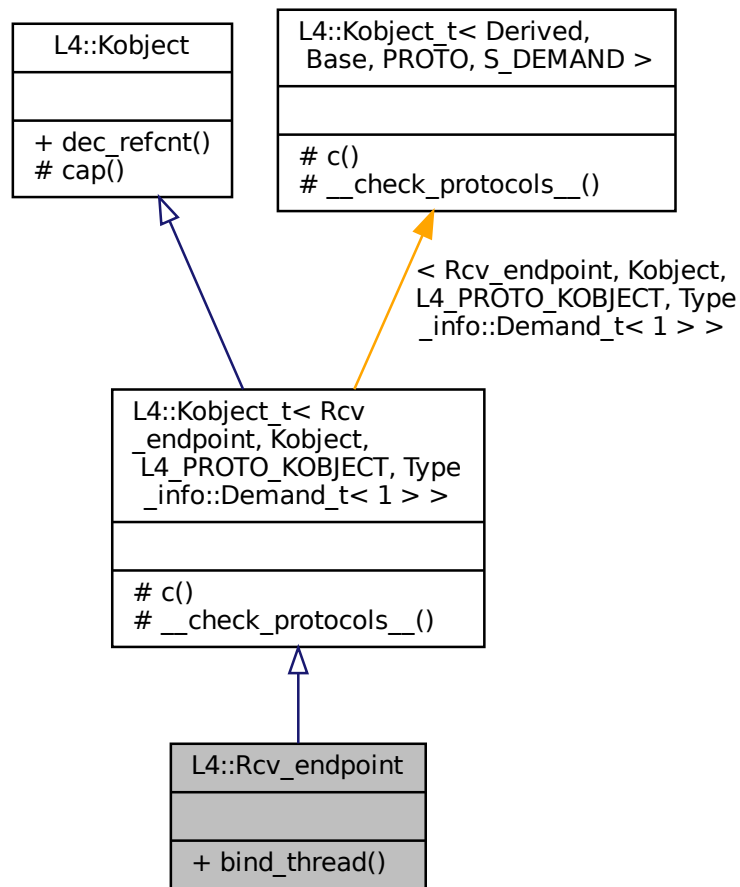
15.165 L4::Rcv_endpoint Class Reference

Interface for kernel objects that allow to receive IPC from them.

Inheritance diagram for L4::Rcv_endpoint:



Collaboration diagram for L4::Rcv_endpoint:



Public Member Functions

- `l4_msgtag_t bind_thread (lpc::Opt< lpc::Cap< Thread > > t, l4_umword_t label)`

Bind a thread to an IPC receive endpoint.

Additional Inherited Members

15.165.1 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 `lrc_sender`. 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`.

15.165.2 Member Function Documentation

15.165.2.1 bind_thread()

```
l4_msgtag_t L4::Rcv_endpoint::bind_thread (
    Ipc::Opt< 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

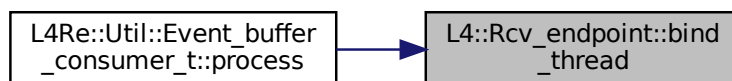
Syscall return tag containing one of the following return codes.

Return values

<i>L4_EOK</i>	Operation successful.
<i>-L4_EINVAL</i>	<code>t</code> is not a thread object or other arguments were malformed.
<i>-L4_EPERM</i>	No L4_CAP_FPAGE_S rights on <code>t</code> or the capability used to invoke this operation.

Referenced by [L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::process\(\)](#).

Here is the caller graph for this function:



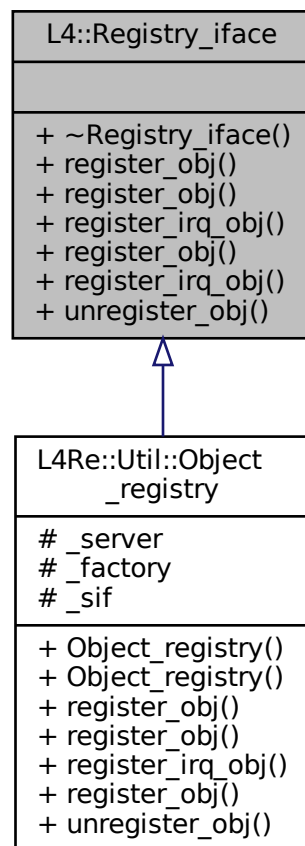
The documentation for this class was generated from the following file:

- [l4/sys/rcv_endpoint](#)

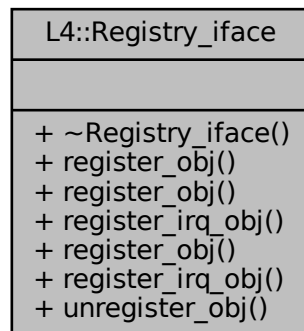
15.166 L4::Registry_iface Class Reference

Abstract interface for object registries.

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.

15.166.1 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 330 of file [ipc_epiface](#).

15.166.2 Member Function Documentation

15.166.2.1 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

<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.

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](#).

15.166.2.2 `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](#).

15.166.2.3 `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 <i>o</i> to as a server-side object.

Return values

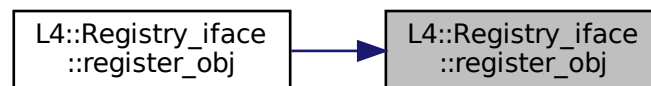
<i>L4::Cap<void></i>	The capability known as <i>service</i> on success.
<i>L4::Cap<void>::Invalid</i>	No capability with the given name found.

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](#).

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

Here is the caller graph for this function:



15.166.2.4 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

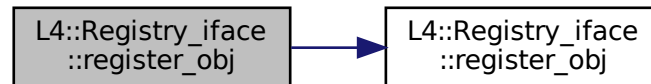
<i>L4::Cap<L4::Rcv_endpoint></i>	Capability <i>ep</i> on success.
<i>L4::Cap<L4::Rcv_endpoint>::Invalid</i>	The IRQ attach operation has failed.

After successful registration `o->obj_cap()` will be equal to `ep`.

Implemented in [L4Re::Util::Object_registry](#).

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

Here is the call graph for this function:



15.166.2.5 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 <code>o->obj_cap()</code> 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:

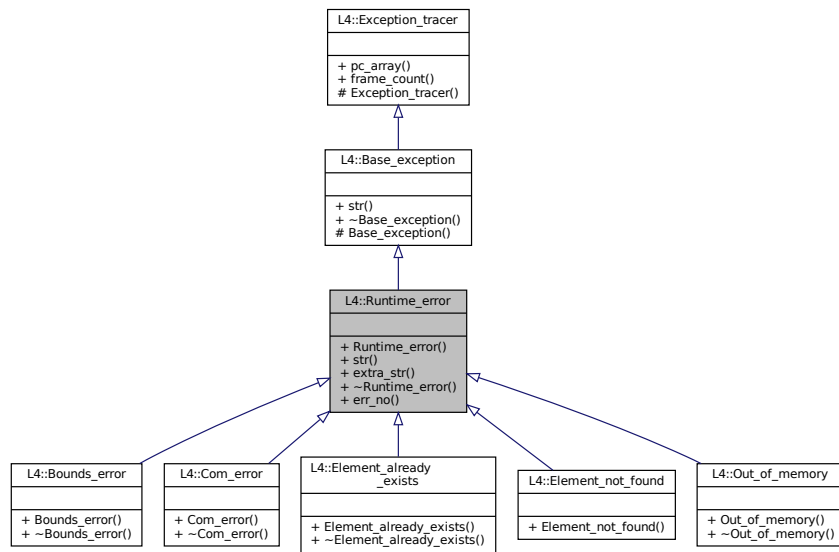
- `l4/sys/cxx/ipc_epiface`

15.167 L4::Runtime_error Class Reference

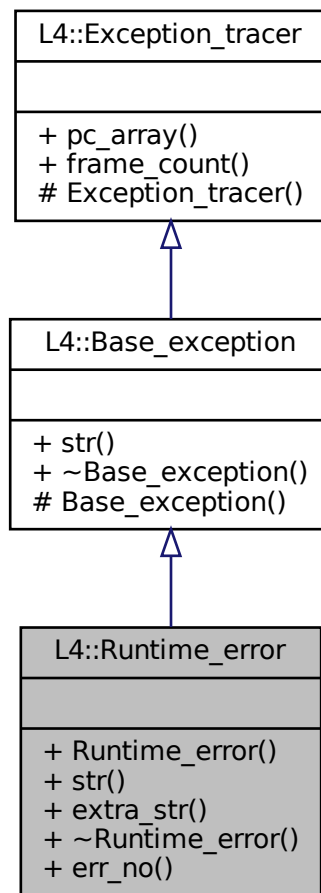
[Exception](#) for an abstract runtime error.

```
#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 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 throw ()
Get the error value for this runtime error.

Additional Inherited Members

15.167.1 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](#).

15.167.2 Constructor & Destructor Documentation

15.167.2.1 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](#).

15.167.3 Member Function Documentation

15.167.3.1 err_no()

```
long L4::Runtime_error::err_no ( ) const throw ( )    [inline]
```

Get the error value for this runtime error.

Returns

Error value.

Definition at line [181](#) of file [exceptions](#).

15.167.3.2 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](#).

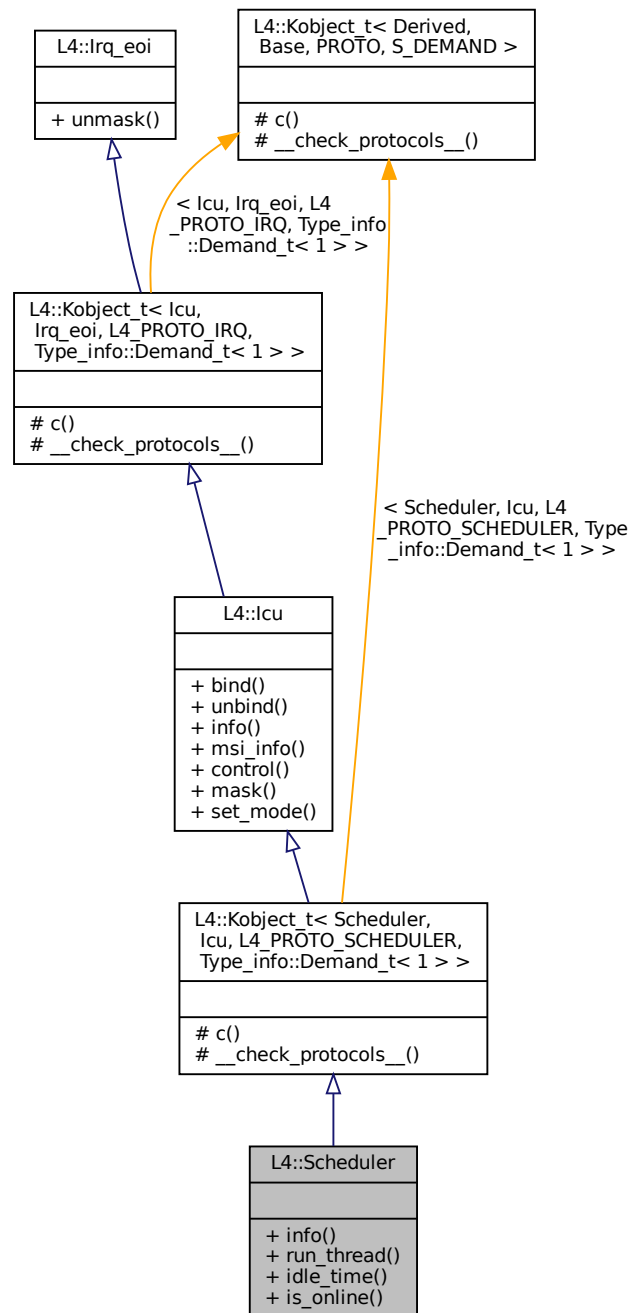
The documentation for this class was generated from the following file:

- [l4/cxx/exceptions](#)

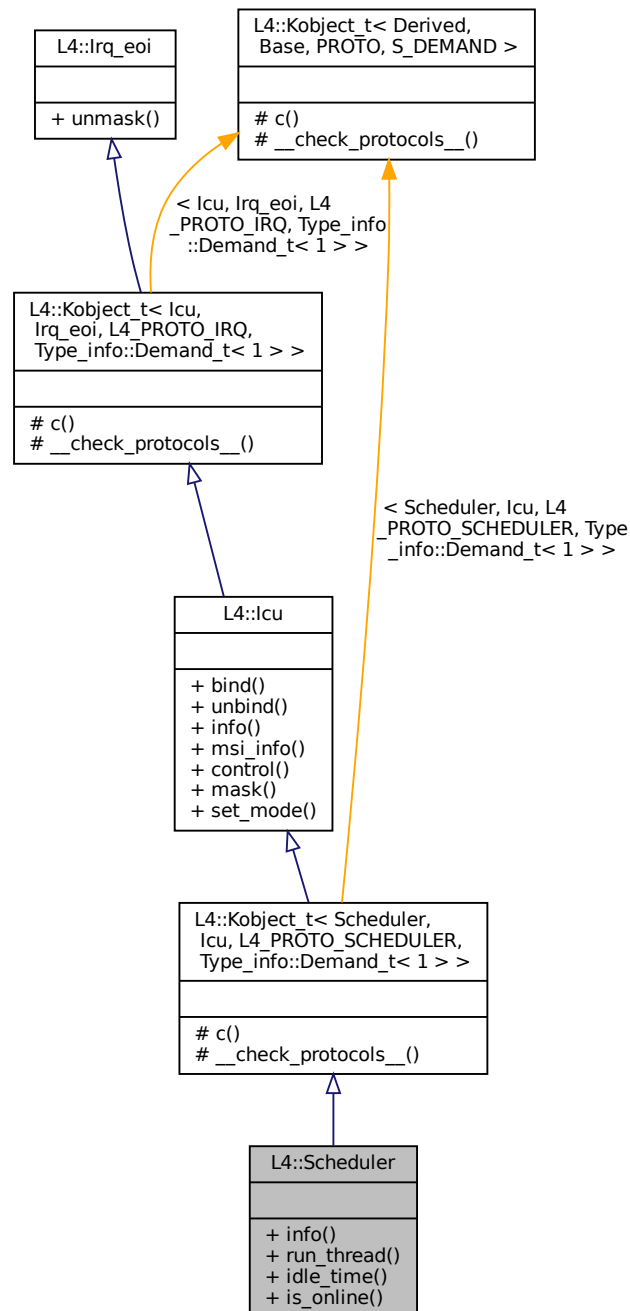
15.168 L4::Scheduler Class Reference

C++ interface of the [Scheduler](#) kernel object.

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_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.

Additional Inherited Members

15.168.1 Detailed Description

C++ interface of the [Scheduler](#) kernel object.

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.

Include File

```
#include <l4/sys/scheduler>
```

Definition at line 45 of file [scheduler](#).

15.168.2 Member Function Documentation

15.168.2.1 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 μ s) 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 \perp 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.

15.168.2.2 info()

```
l4_msgtag_t L4::Scheduler::info (
    l4_umword_t * cpu_max,
    l4_sched_cpu_set_t * cpus,
    l4_utcb_t * utcb = l4_utcb() ) const [inline], [noexcept]
```

Get scheduler information.

Parameters

out	<i>cpu_max</i>	Maximum number of CPUs ever available.
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.
	<i>utcb</i>	UTCB pointer of the calling thread. This defaults to the UTCB of the current thread.

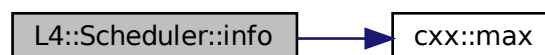
Return values

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

Definition at line 69 of file [scheduler](#).

References [l4_sched_cpu_set_t::gran_offset](#), [l4_sched_cpu_set_t::map](#), and [cxx::max\(\)](#).

Here is the call graph for this function:



15.168.2.3 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 pointer 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 [142](#) of file [scheduler](#).

15.168.2.4 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

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 physical CPUs.

- Two threads with a single identical CPU selected in the CPU set must be scheduled to the same physical CPU.

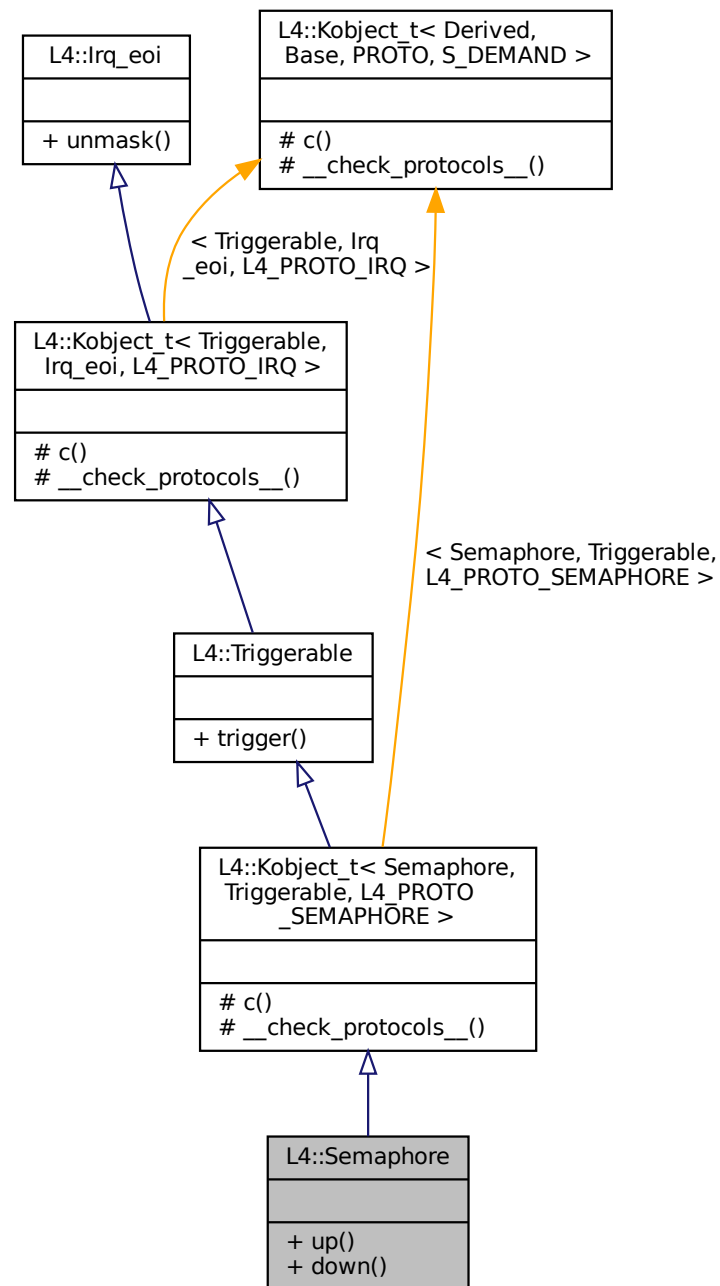
The documentation for this class was generated from the following file:

- [l4/sys/scheduler](#)

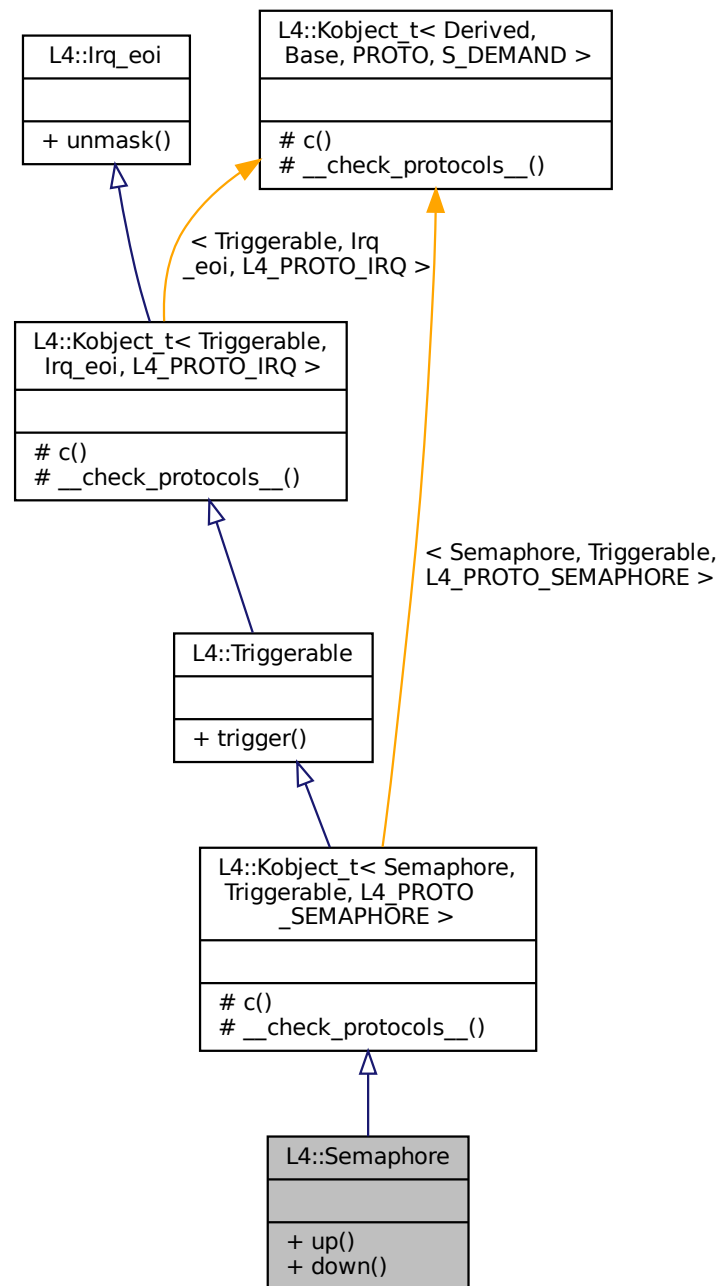
15.169 L4::Semaphore Struct Reference

Kernel-provided semaphore object.

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.

Additional Inherited Members

15.169.1 Detailed Description

Kernel-provided semaphore object.

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::irq](#) is to ensure that incoming trigger signals do not interfere with any open-wait operations, as used for example in a server loop.

Definition at line 51 of file [semaphore](#).

15.169.2 Member Function Documentation

15.169.2.1 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, usually the UTCB of the calling thread.

Returns

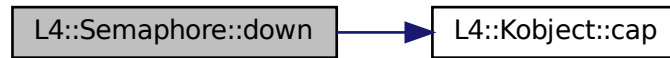
IPC message return tag. Use [l4_ipc_error\(\)](#) to check for a timeout or a cancel condition.

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 84 of file [semaphore](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.169.2.2 up()

```

l4_msgtag_t L4::Semaphore::up (
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
  
```

Semaphore up operation (wrapper for [trigger\(\)](#)).

Parameters

<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.
-------------	---

Returns

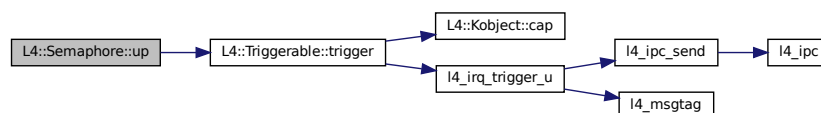
Send-only IPC message return tag. Use [l4_ipc_error\(\)](#) to check for errors, do **not** use [l4_error\(\)](#).

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 65 of file [semaphore](#).

References [L4::Triggerable::trigger\(\)](#).

Here is the call graph for this function:



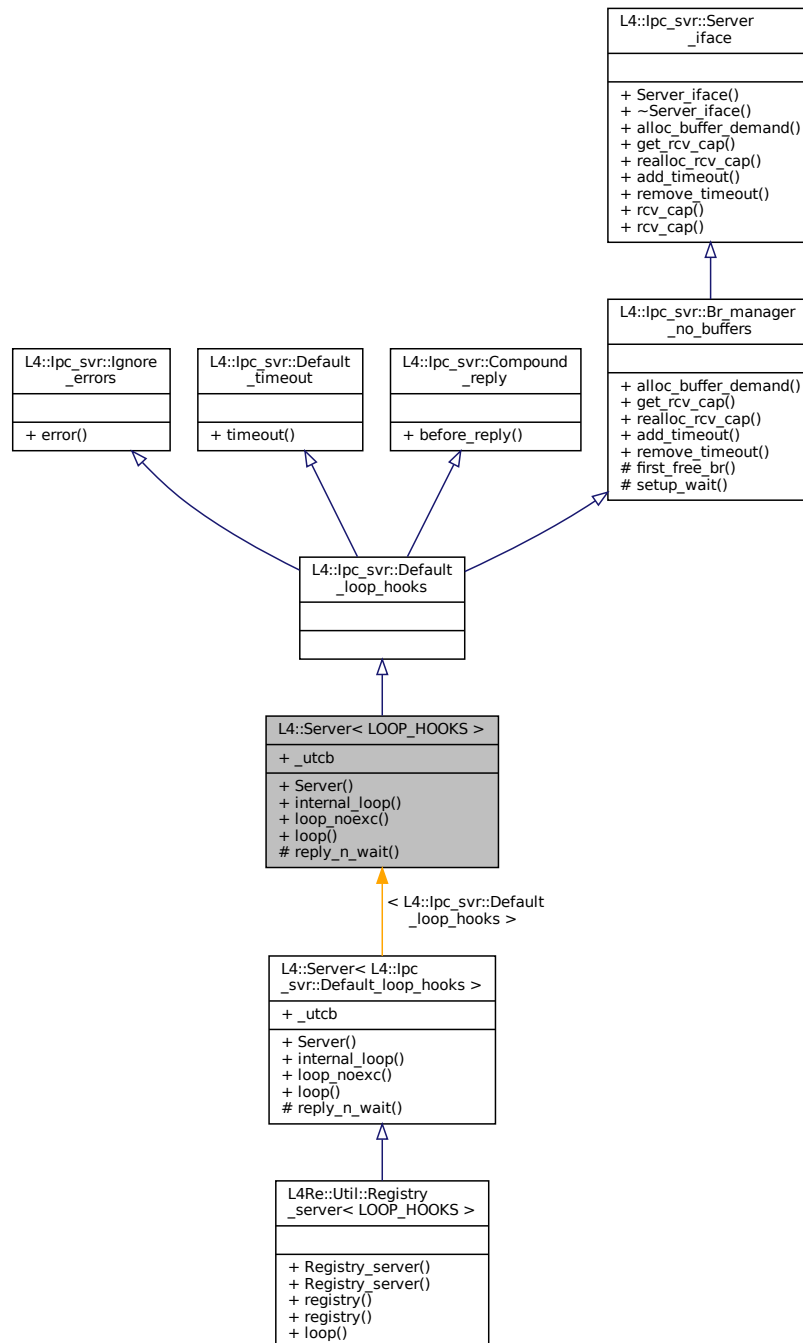
The documentation for this struct was generated from the following file:

- [l4/sys/semaphore](#)

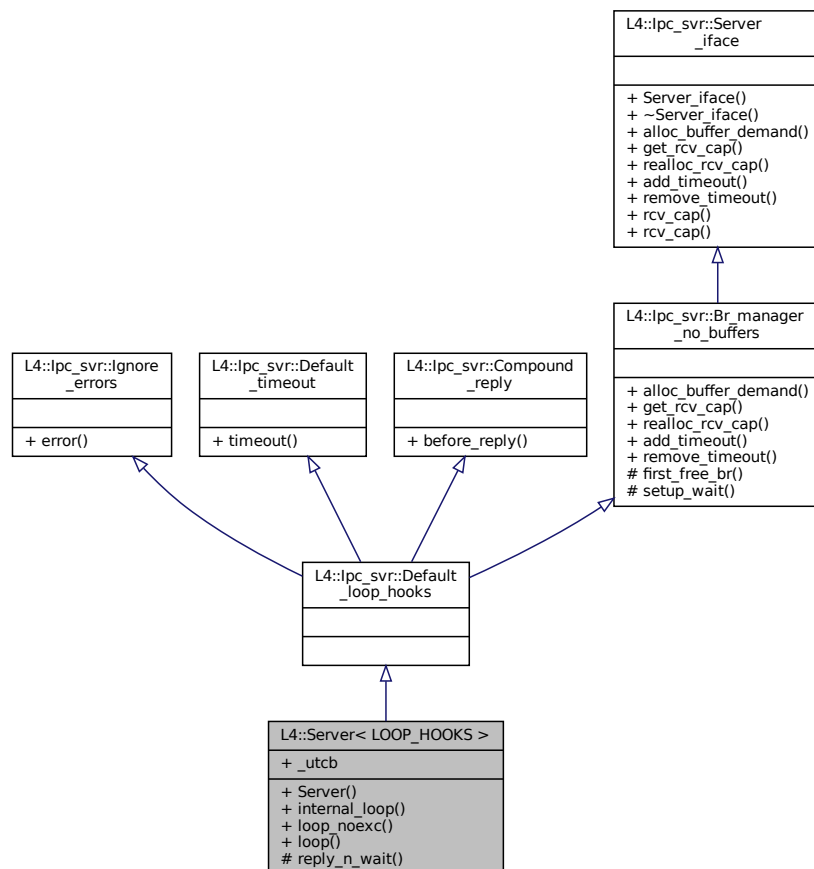
15.170 L4::Server< LOOP_HOOKS > Class Template Reference

Basic server loop for handling client requests.

Inheritance diagram for L4::Server< LOOP_HOOKS >:



Collaboration diagram for L4::Server< LOOP_HOOKS >:



Public Member Functions

- [Server](#) ([l4_utcb_t](#) *utcb)
Initializes the server loop.
- [template](#)<typename DISPATCH >
[L4_NORETURN](#) void [internal_loop](#) (DISPATCH dispatch)
The server loop.
- [template](#)<typename R >
[L4_NORETURN](#) void [loop_noexc](#) (R r)
Server loop without exception handling.
- [template](#)<typename EXC , typename R >
[L4_NORETURN](#) void [loop](#) (R r)
Server loop with internal exception handling.

Protected Member Functions

- [l4_msgtag_t](#) [reply_n_wait](#) ([l4_msgtag_t](#) reply, [l4_umword_t](#) *p)
Internal implementation for reply and wait.

Additional Inherited Members

15.170.1 Detailed Description

```
template<typename LOOP_HOOKS = ipc_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 ipc_svr::Default_loop_hooks , ipc_svr::Ignore_errors , ipc_svr::Default_timeout , ipc_svr::Compound_reply , and ipc_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 298 of file [ipc_server_loop](#).

15.170.2 Constructor & Destructor Documentation

15.170.2.1 Server()

```
template<typename LOOP_HOOKS = Ipc_svr::Default_loop_hooks>
L4::Server< LOOP_HOOKS >::Server (
    l4_utcb_t * utcb ) [inline], [explicit]
```

Initializes the server loop.

Parameters

<i>utcb</i>	The UTCB of the thread running the server loop.
-------------	---

Definition at line 306 of file [ipc_server_loop](#).

15.170.3 Member Function Documentation

15.170.3.1 internal_loop()

```
template<typename L >
template<typename DISPATCH >
```

```
L4_NORETURN void L4::Server< L >::internal_loop (
    DISPATCH dispatch ) [inline]
```

The server loop.

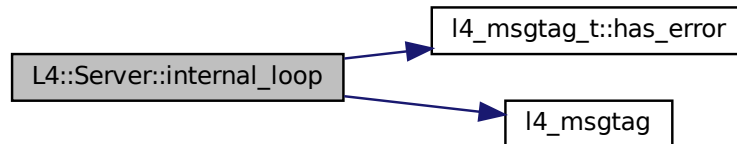
This function usually never returns, it waits for incoming messages calls the dispatcher, sends a reply and waits again.

Definition at line 368 of file [ipc_server_loop](#).

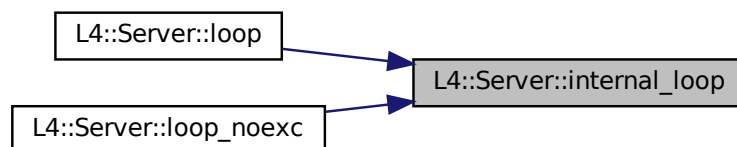
References [l4_msgtag_t::has_error\(\)](#), [L4_ENOREPLY](#), and [l4_msgtag\(\)](#).

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:



15.170.3.2 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 ) [inline]
```

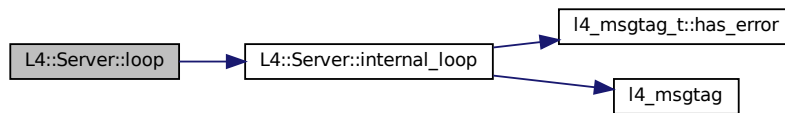
[Server](#) loop with internal exception handling.

This server loop translates [L4::Runtime_error](#) exceptions into negative error return codes sent to the caller.

Definition at line 331 of file [ipc_server_loop](#).

References [L4::Server< LOOP_HOOKS >::internal_loop\(\)](#).

Here is the call graph for this function:



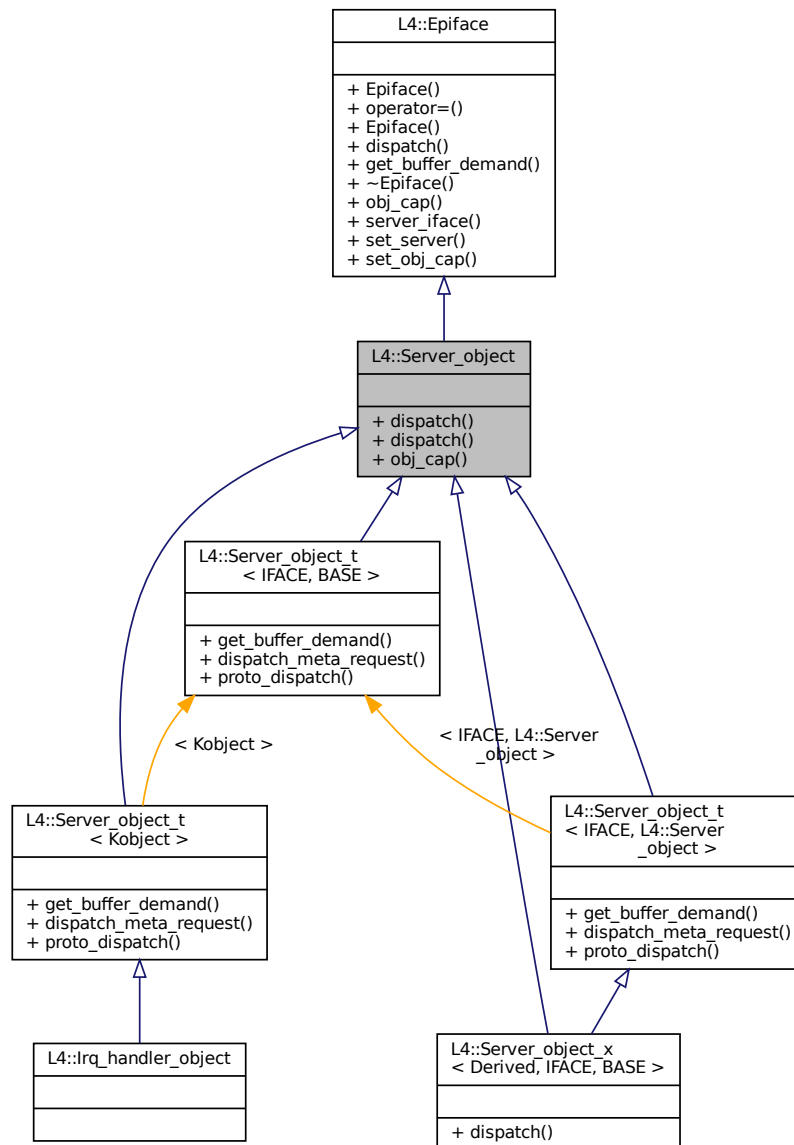
The documentation for this class was generated from the following file:

- `I4/sys/cxx/ipc_server_loop`

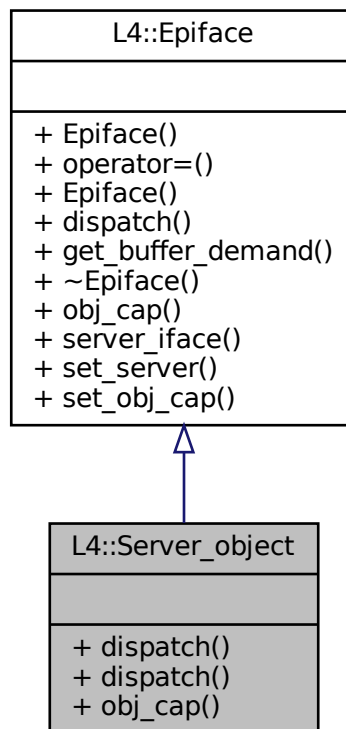
15.171 L4::Server_object Class Reference

Abstract server object to be used with [L4::Server](#) and [L4::Basic_registry](#).

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)
The abstract handler for client requests to the object.

Additional Inherited Members

15.171.1 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`.

15.171.2 Member Function Documentation

15.171.2.1 `dispatch()` [1/2]

```
l4_msgtag_t L4::Server_object::dispatch (
    l4_msgtag_t tag,
    unsigned rights,
    l4_utcb_t * utcb ) [inline], [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.

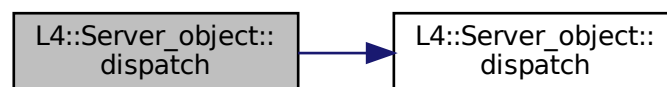
This function must be implemented by application specific server objects.

Implements [L4::Epiface](#).

Definition at line 71 of file [ipc_server](#).

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

Here is the call graph for this function:



15.171.2.2 `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 lpc::lostream 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.

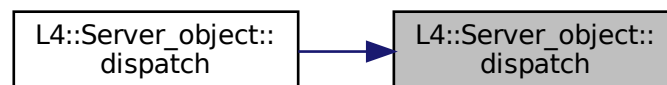
Implemented in [L4::Server_object_x< Derived, IFACE, BASE >](#).

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:



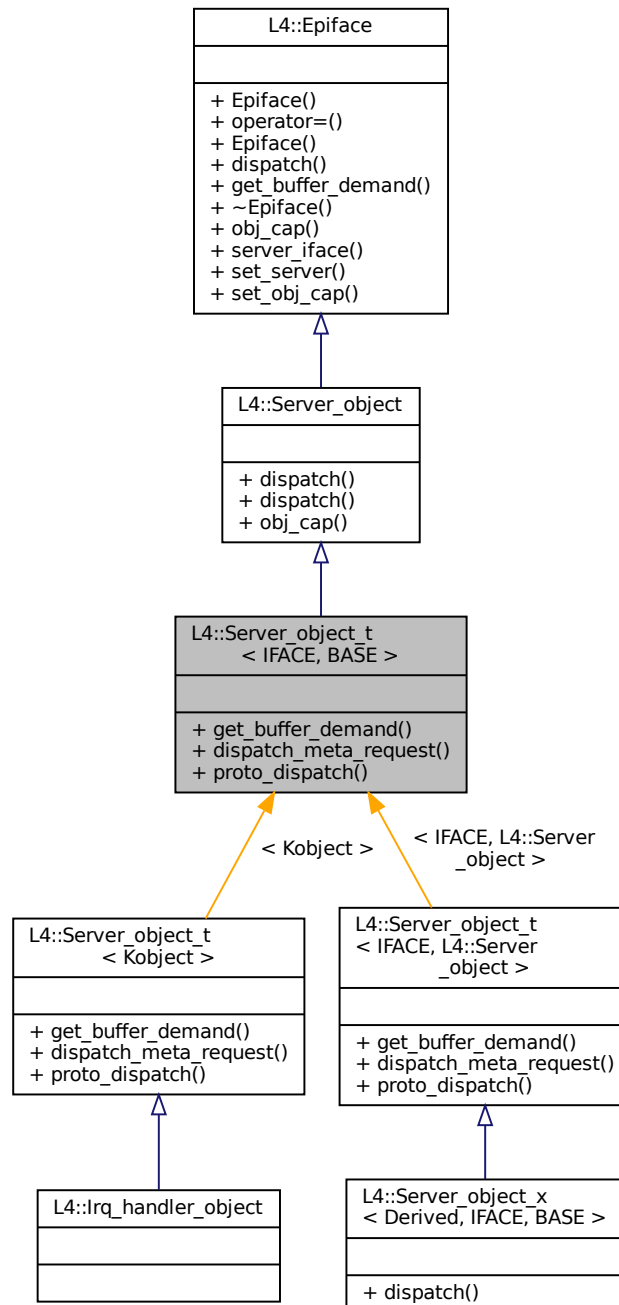
The documentation for this class was generated from the following file:

- [l4/cxx/ipc_server](#)

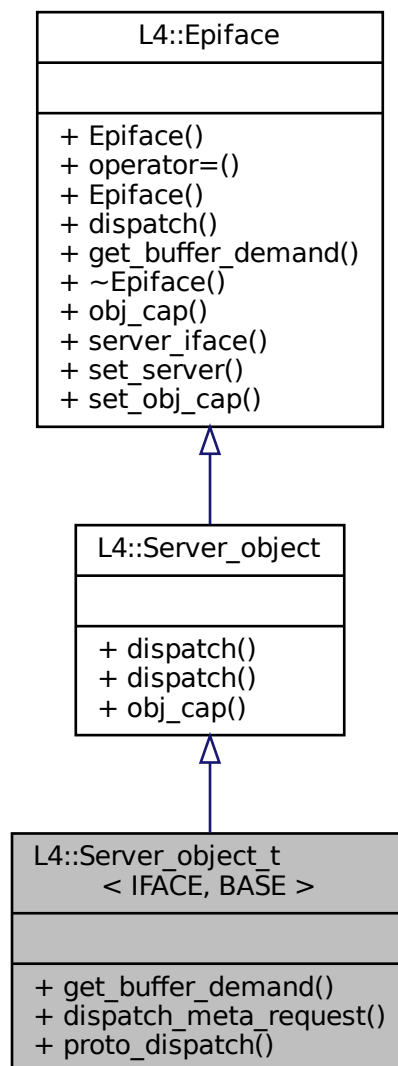
15.172 L4::Server_object_t< IFACE, BASE > Struct Template Reference

Base class (template) for server implementing server objects.

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 Member Functions

- BASE::Demand [get_buffer_demand](#) () const
- int [dispatch_meta_request](#) (L4::lpc::lostream &ios)
Implementation of the meta protocol based on IFACE.

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.

15.172.1 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

<i>IFACE</i>	The IPC interface class that defines the interface that shall be implemented.
<i>BASE</i>	The server object base class (usually L4::Server_object).

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](#).

15.172.2 Member Function Documentation

15.172.2.1 `dispatch_meta_request()`

```
template<typename IFACE , typename BASE = L4::Server_object>
int L4::Server\_object\_t< IFACE, BASE >::dispatch_meta_request (
    L4::lpc::lostream & 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](#).

15.172.2.2 get_buffer_demand()

```
template<typename IFACE , typename BASE = L4::Server_object>
BASE::Demand L4::Server_object_t< IFACE, BASE >::get_buffer_demand ( ) const [inline], [virtual]
```

Returns

the server-side buffer demand based in *IFACE*.

Implements [L4::Epiface](#).

Definition at line 97 of file [ipc_server](#).

15.172.2.3 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 p_dispatch()).
<i>ios</i>	The message stream for the incoming and the reply message.

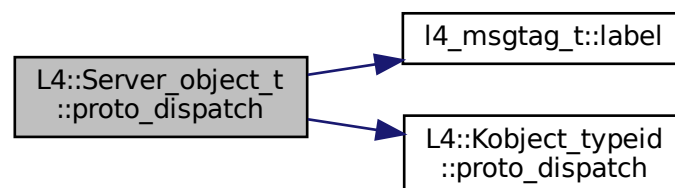
[Server](#) objects may call this function from their [dispatch\(\)](#) function. This function reads the protocol ID from the message tag and uses the [p_dispatch](#) code to dispatch to overloaded [p_dispatch](#) functions of self.

Definition at line 125 of file [ipc_server](#).

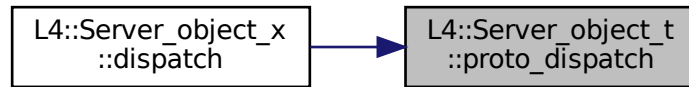
References [l4_msgtag_t::label\(\)](#), and [L4::Kobject_typeid< T >::proto_dispatch\(\)](#).

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:



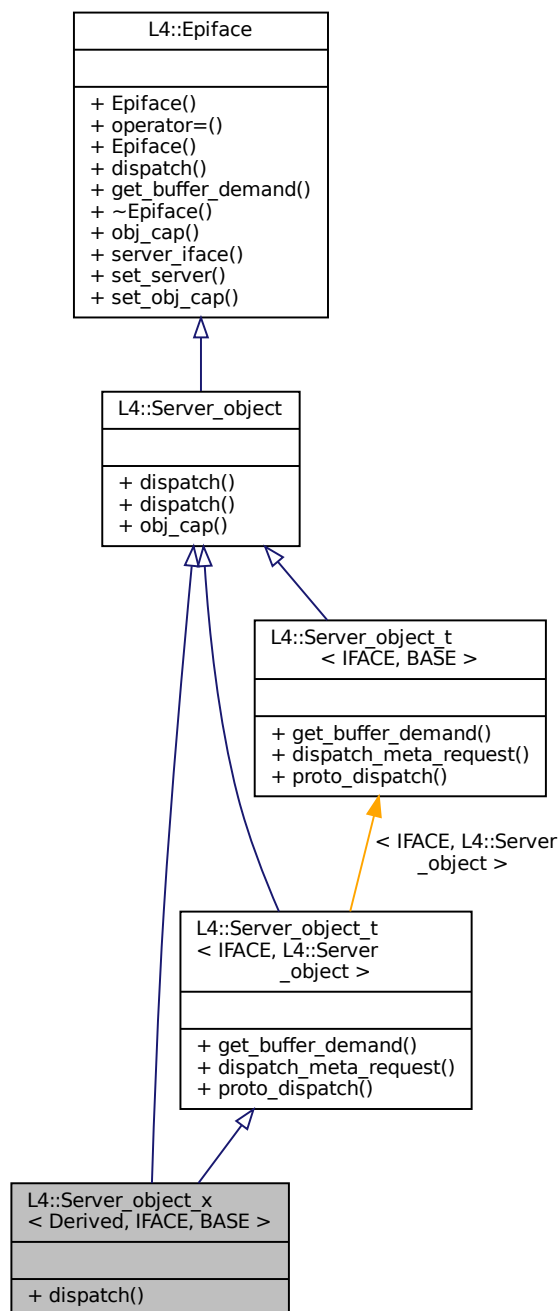
The documentation for this struct was generated from the following file:

- [l4/cxx/ipc_server](#)

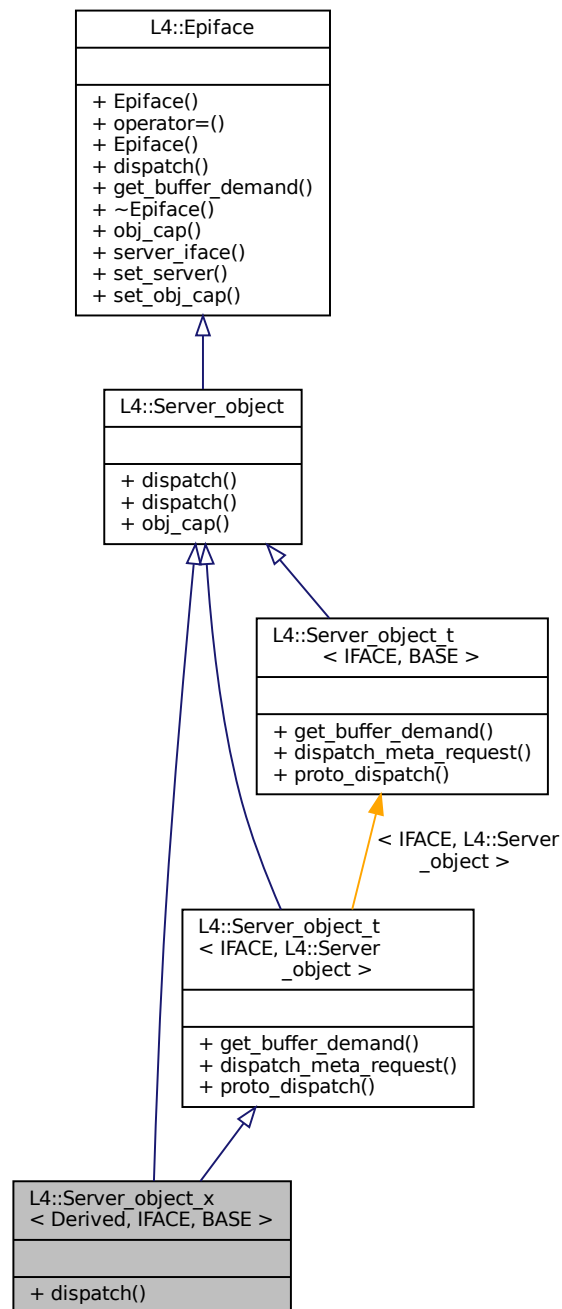
15.173 L4::Server_object_x< Derived, IFACE, BASE > Struct Template Reference

Helper class to implement p_dispatch based server objects.

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()`.

Additional Inherited Members

15.173.1 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](#).

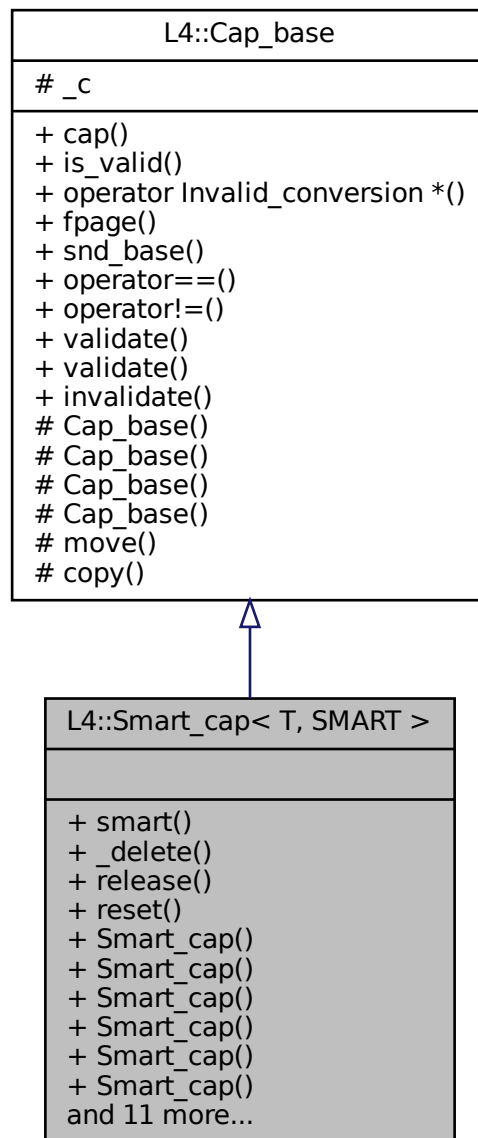
The documentation for this struct was generated from the following file:

- [l4/cxx/ipc_server](#)

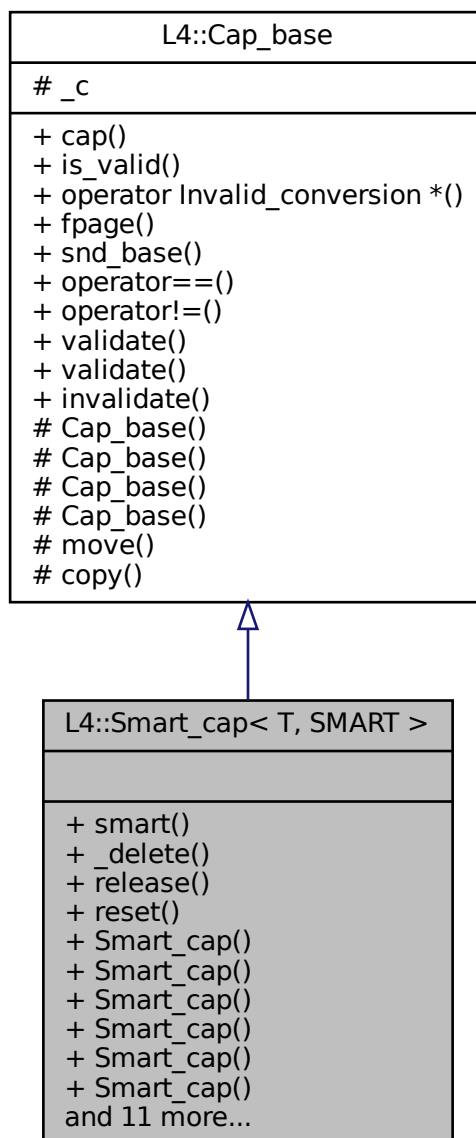
15.174 L4::Smart_cap< T, SMART > Class Template Reference

Smart capability class.

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`.

Additional Inherited Members

15.174.1 Detailed Description

```
template<typename T, typename SMART>
class L4::Smart_cap< T, SMART >
```

Smart capability class.

Definition at line 36 of file [smart_capability](#).

15.174.2 Constructor & Destructor Documentation

15.174.2.1 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:

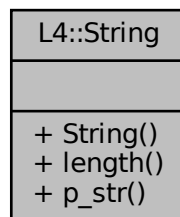
- [l4/sys/smart_capability](#)

15.175 L4::String Class Reference

A null-terminated string container class.

```
#include <string.h>
```

Collaboration diagram for L4::String:



15.175.1 Detailed Description

A null-terminated string container class.

Definition at line 33 of file [string.h](#).

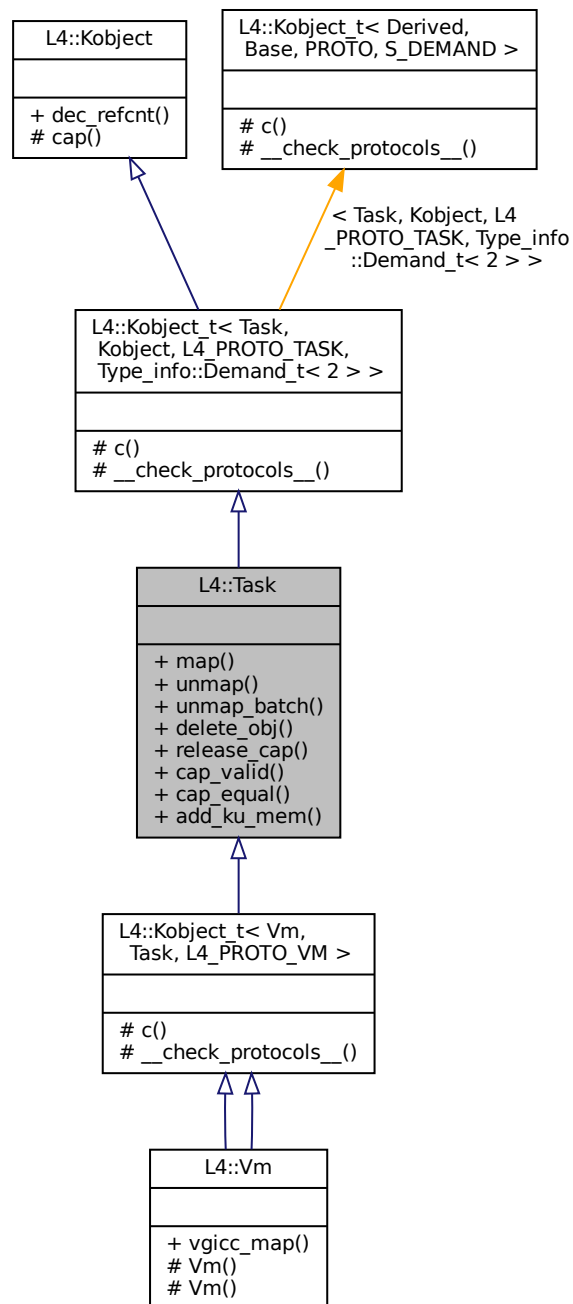
The documentation for this class was generated from the following file:

- [l4/cxx/string.h](#)

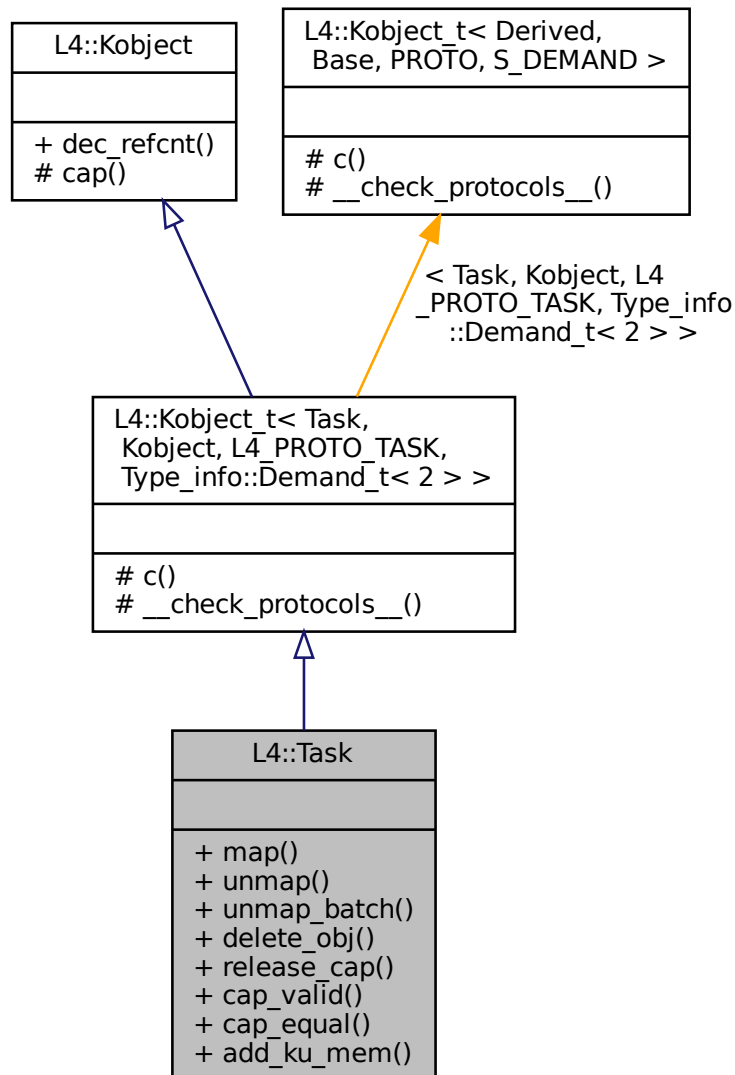
15.176 L4::Task Class Reference

C++ interface of the [Task](#) kernel object.

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 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.

Additional Inherited Members

15.176.1 Detailed Description

C++ interface of the [Task](#) kernel object.

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 43 of file [task](#).

15.176.2 Member Function Documentation

15.176.2.1 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>	UTCP pointer of the calling thread.

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.

Returns

Syscall return tag

Definition at line 202 of file [task](#).

15.176.2.2 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_a</i>	First capability selector to compare.
<i>cap_b</i>	Second capability selector to compare.
<i>utcb</i>	Optional: UTCB pointer of the calling thread.

Return values

<i>tag.label()</i>	= 1: <i>cap_a</i> and <i>cap_b</i> point to the same object.
<i>tag.label()</i>	= 0: The two caps do not point to the same object.

Definition at line 183 of file [task](#).

15.176.2.3 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, usually the UTCB of the calling thread.

Return values

<i>tag.label()</i>	> 0 Capability is present (refers to an object).
<i>tag.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 167 of file [task](#).

15.176.2.4 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 selector of the object to delete.
<i>utcb</i>	UTCB pointer of the calling thread.

Returns

Syscall return tag

The object will be deleted if the `obj` has sufficient rights. No error will be reported if the rights are insufficient, however, the capability is removed in all cases.

Definition at line 133 of file [task](#).

15.176.2.5 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.
<i>utcb</i>	UTCB pointer of the calling thread.

Returns

Syscall return tag.

This method allows for asynchronous rights delegation from one task to another. It can be used to share memory as well as to delegate access to objects. The destination task is the task referenced by the capability invoking map and the receive window is the whole address space of said task.

Definition at line 67 of file [task](#).

15.176.2.6 `release_cap()`

```
l4_msgtag_t L4::Task::release_cap (
    L4::Cap< void > cap,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
```

Release capability.

Parameters

<i>cap</i>	Capability selector to release.
<i>utcb</i>	UTCB pointer of the calling thread.

Returns

Syscall return tag.

This operation unmaps the capability from `this` task.

Definition at line 147 of file [task](#).

15.176.2.7 `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 the address space or object space of <code>this</code> task
<i>map_mask</i>	Unmap mask, see l4_unmap_flags_t
<i>utcb</i>	UTCB pointer of the calling thread.

Returns

Syscall return tag

This method allows to revoke rights from the destination task and from all the tasks that got the rights delegated from that task (i.e., this operation does a recursive rights revocation).

Note

If the capability possesses delete rights or if it is the last capability pointing to the object, calling this function might destroy the object itself.

Definition at line 90 of file [task](#).

15.176.2.8 unmap_batch()

```
l4_msgtag_t L4::Task::unmap_batch (
    l4_fpage_t const * fpages,
    unsigned num_fpages,
    l4_unword_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 the address or object 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 l4_unmap_flags_t .
<i>utcb</i>	UTCB pointer of the calling thread.

This method allows to revoke rights from the destination task and from all the tasks that got the rights delegated from that task (i.e., this operation does a recursive rights revocation).

Precondition

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

Note

If the capability possesses delete rights or if it is the last capability pointing to the object, calling this function might destroy the object itself.

Definition at line 115 of file [task](#).

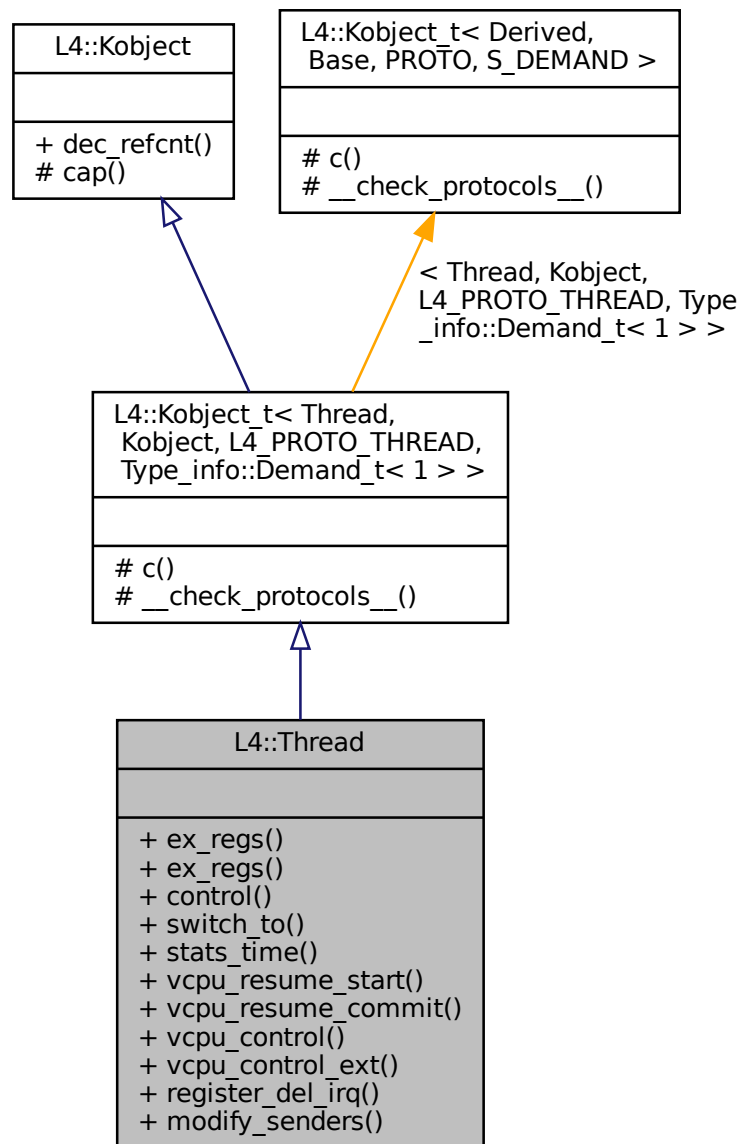
The documentation for this class was generated from the following file:

- [l4/sys/task](#)

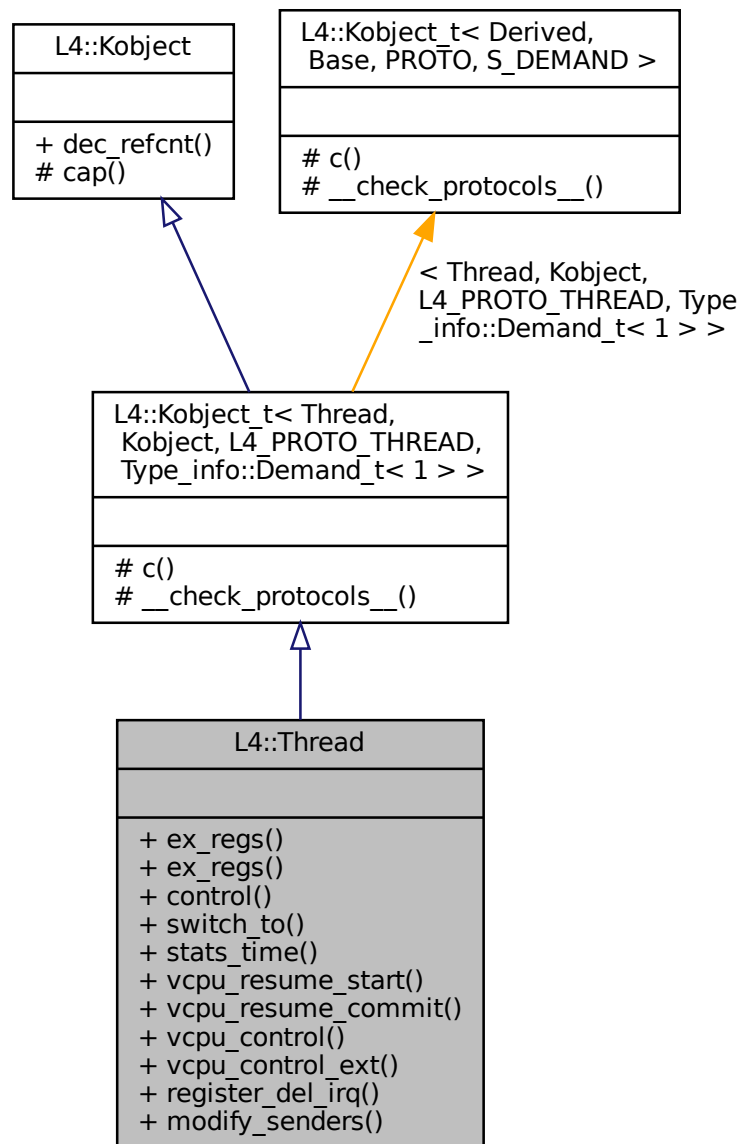
15.177 L4::Thread Class Reference

C++ [L4](#) kernel thread interface.

Inheritance diagram for L4::Thread:



Collaboration diagram for L4::Thread:



Data Structures

- class [Attr](#)
Thread attributes used for control_commit().
- class [Modify_senders](#)
Wrapper class for modifying senders.

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

- `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 control` (`Attr const &attr`) noexcept
Exchange basic thread registers and return previous values.
- `l4_msgtag_t switch_to` (`l4_utcb_t *utcb=l4_utcb()`) noexcept
Commit the given thread-attributes object.
- `l4_msgtag_t stats_time` (`l4_kernel_clock_t *us, l4_utcb_t *utcb=l4_utcb()`) noexcept
Switch execution to this thread.
- `l4_msgtag_t vcpu_resume_start` (`l4_utcb_t *utcb=l4_utcb()`) noexcept
Get consumed time of thread in us.
- `l4_msgtag_t vcpu_resume_commit` (`l4_msgtag_t tag, l4_utcb_t *utcb=l4_utcb()`) noexcept
vCPU resume, start.
- `l4_msgtag_t vcpu_control` (`l4_addr_t vcpu_state, l4_utcb_t *utcb=l4_utcb()`) noexcept
vCPU resume, commit.
- `l4_msgtag_t vcpu_control_ext` (`l4_addr_t ext_vcpu_state, l4_utcb_t *utcb=l4_utcb()`) noexcept
Enable or disable the vCPU feature for the thread.
- `l4_msgtag_t register_del_irq` (`Cap < Irq > irq, l4_utcb_t *u=l4_utcb()`) noexcept
Enable or disable the extended vCPU feature for the thread.
- `l4_msgtag_t modify_senders` (`Modify_senders const &todo`) noexcept
Register an IRQ that will trigger upon deletion events.
- `l4_msgtag_t modify_senders` (`Modify_senders const &todo`) noexcept
Apply sender modification rules.

Additional Inherited Members

15.177.1 Detailed Description

C++ [L4](#) kernel thread 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.

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

15.177.2 Member Function Documentation

15.177.2.1 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.
-------------	---

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

15.177.2.2 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 use for this operation.

Returns

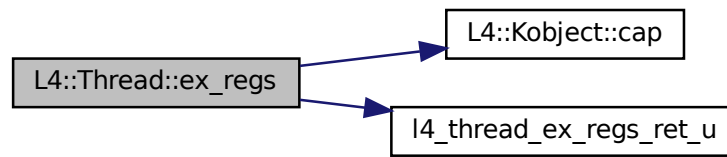
System call return tag. [out] parameters are only valid if the function returns successfully. Use [l4_error\(\)](#) to check.

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*).

Definition at line 111 of file [thread](#).

References [L4::Kobject::cap\(\)](#), and [l4_thread_ex_regs_ret_u\(\)](#).

Here is the call graph for this function:



15.177.2.3 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 use for this operation.

Returns

System call return tag

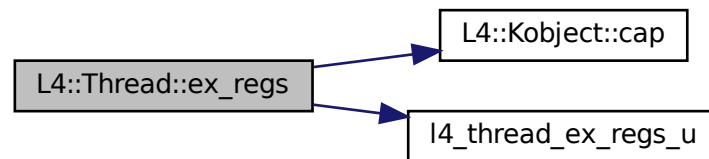
This method allows to manipulate a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see `flags`).

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 85 of file [thread](#).

References [L4::Kobject::cap\(\)](#), and [l4_thread_ex_regs_u\(\)](#).

Here is the call graph for this function:



15.177.2.4 modify_senders()

```
l4_msgtag_t L4::Thread::modify_senders (
    Modify_senders const & todo ) [inline], [noexcept]
```

Apply sender modification rules.

Parameters

<i>todo</i>	Prepared sender modification rules.
-------------	-------------------------------------

Returns

System call return tag.

The modification rules are applied to all IPCs to the thread (whether directly or by IPC gate) that are already in flight, that is that the sender is already blocking on.

See also

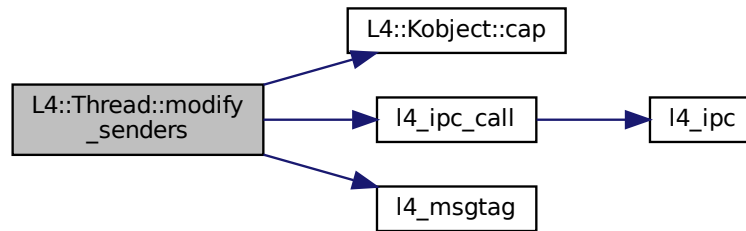
[l4_thread_modify_sender_commit\(\)](#)

Definition at line 394 of file [thread](#).

References [L4::Kobject::cap\(\)](#), [l4_ipc_call\(\)](#), [l4_msgtag\(\)](#), and [L4_PROTO_THREAD](#).

Referenced by [L4Re::Util::Object_registry::unregister_obj\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.177.2.5 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, usually the UTCB of the calling thread.

Returns

System call return tag containing the return code.

An example of a deletion event is the removal of an IPC gate that is bound to this thread.

See also

[l4_thread_register_del_irq](#)

Definition at line 324 of file [thread](#).

15.177.2.6 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 of the current thread.

Returns

Syscall return tag.

Definition at line 247 of file [thread](#).

15.177.2.7 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>	the UTCB of the current thread.
-------------	---------------------------------

Note

The current time slice is inherited to this thread.

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

15.177.2.8 vcpu_control()

```
l4_msgtag_t L4::Thread::vcpu_control (
    l4_addr_t vcpu_state,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
```

Enable or disable the vCPU feature for the thread.

Parameters

<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()).
<i>utcb</i>	UTCB to use for this operation.

Returns

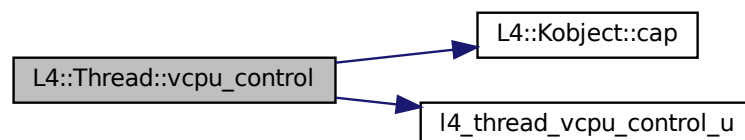
Syscall return tag.

This function enables the vCPU feature of this thread if *vcpu_state* is set to a valid kernel-user-memory address, or disables the vCPU feature if *vcpu_state* is 0. (Disable: optional, currently unsupported.)

Definition at line 283 of file [thread](#).

References [L4::Kobject::cap\(\)](#), and [l4_thread_vcpu_control_u\(\)](#).

Here is the call graph for this function:



15.177.2.9 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 or disable 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 use for this operation.

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 if `ext_vcpu_state` is set to a valid kernel-user-memory address, or disables the vCPU feature if `ext_vcpu_state` is 0.

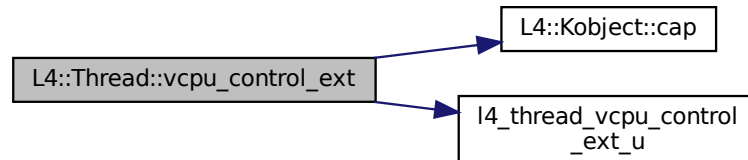
Note

The extended vCPU mode includes the normal vCPU mode.

Definition at line 307 of file [thread](#).

References [L4::Kobject::cap\(\)](#), and [l4_thread_vcpu_control_ext_u\(\)](#).

Here is the call graph for this function:



15.177.2.10 vcpu_resume_commit()

```

l4_msgtag_t L4::Thread::vcpu_resume_commit (
    l4_msgtag_t tag,
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
  
```

vCPU resume, commit.

See also

[l4_thread_vcpu_resume_commit](#)

Definition at line 264 of file [thread](#).

15.177.2.11 vcpu_resume_start()

```
l4_msgtag_t L4::Thread::vcpu_resume_start (
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
```

vCPU resume, start.

See also

[l4_thread_vcpu_resume_start](#)

Definition at line 256 of file [thread](#).

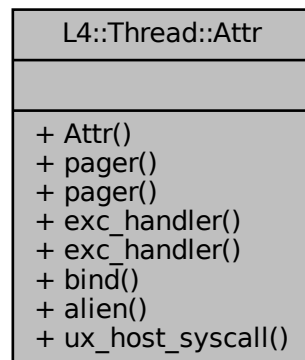
The documentation for this class was generated from the following file:

- [l4/sys/thread](#)

15.178 L4::Thread::Attr Class Reference

[Thread](#) attributes used for `control_commit()`.

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

Set the thread to alien mode.
- void [ux_host_syscall](#) (int on) noexcept

Allow host system calls on Fiasco-UX.

Friends

- class [L4::Thread](#)

15.178.1 Detailed Description

[Thread](#) attributes used for [control_commit\(\)](#).

This class is responsible for initializing various attributes of a thread in a UTCB for the [control_commit\(\)](#) 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 [129](#) of file [thread](#).

15.178.2 Constructor & Destructor Documentation

15.178.2.1 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_commit() function. Usually this is the UTCB of the calling thread.
-------------	--

Definition at line 142 of file [thread](#).

15.178.3 Member Function Documentation

15.178.3.1 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 L4_UTCB_OFFSET bytes of kernel-user memory.
<i>task</i>	The task the thread shall be bound to.

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.

Note

The UTCBs of different threads in the same task should not overlap in order to prevent data corruption.

Definition at line 202 of file [thread](#).

15.178.3.2 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 180 of file [thread](#).

15.178.3.3 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 171 of file [thread](#).

15.178.3.4 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 161 of file [thread](#).

15.178.3.5 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 152 of file [thread](#).

15.178.3.6 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 216 of file [thread](#).

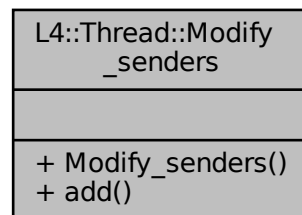
The documentation for this class was generated from the following file:

- [l4/sys/thread](#)

15.179 L4::Thread::Modify_senders Class Reference

Wrapper class for modifying senders.

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.

15.179.1 Detailed Description

Wrapper class for modifying senders.

Use the [add\(\)](#) function to add modification rules, and use [modify_senders\(\)](#) to commit. Do not use the UTCB in between as it is used by [add\(\)](#) and [modify_senders\(\)](#).

Definition at line 334 of file [thread](#).

15.179.2 Member Function Documentation

15.179.2.1 add()

```
int L4::Thread::Modify_senders::add (
    14_umword_t match_mask,
    14_umword_t match,
    14_umword_t del_bits,
    14_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 match_mask.
<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.

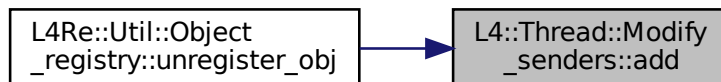
See also

[l4_thread_modify_sender_add\(\)](#)

Definition at line 367 of file [thread](#).

Referenced by [L4Re::Util::Object_registry::unregister_obj\(\)](#).

Here is the caller graph for this function:



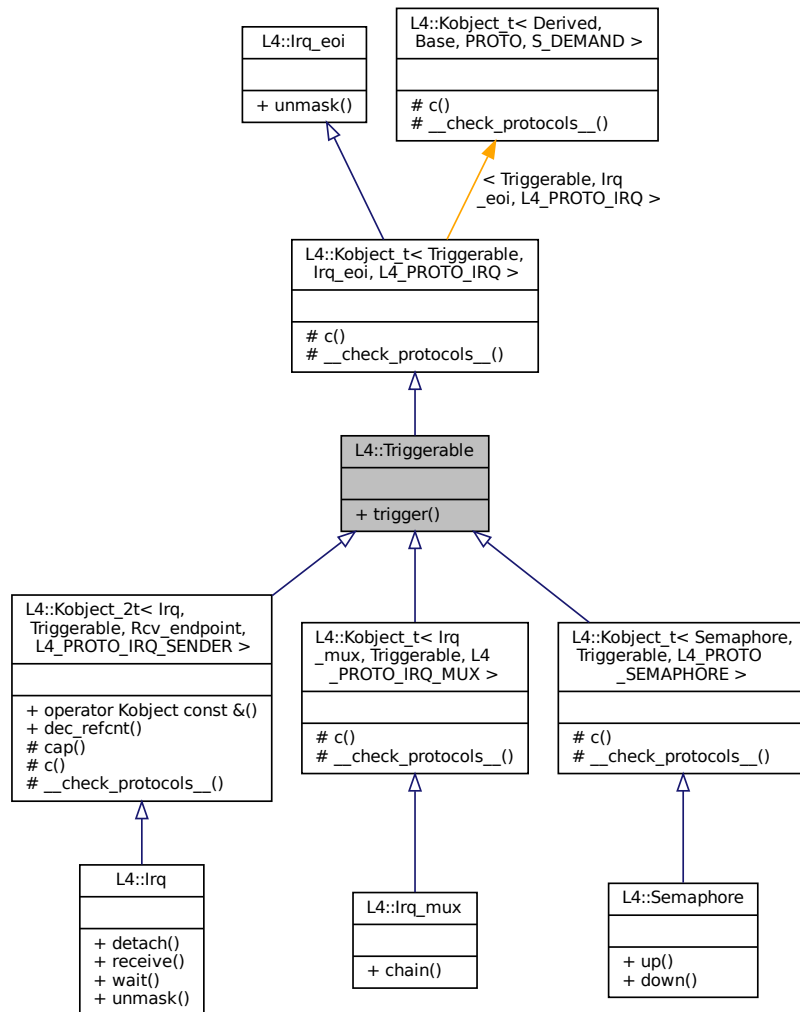
The documentation for this class was generated from the following file:

- [l4/sys/thread](#)

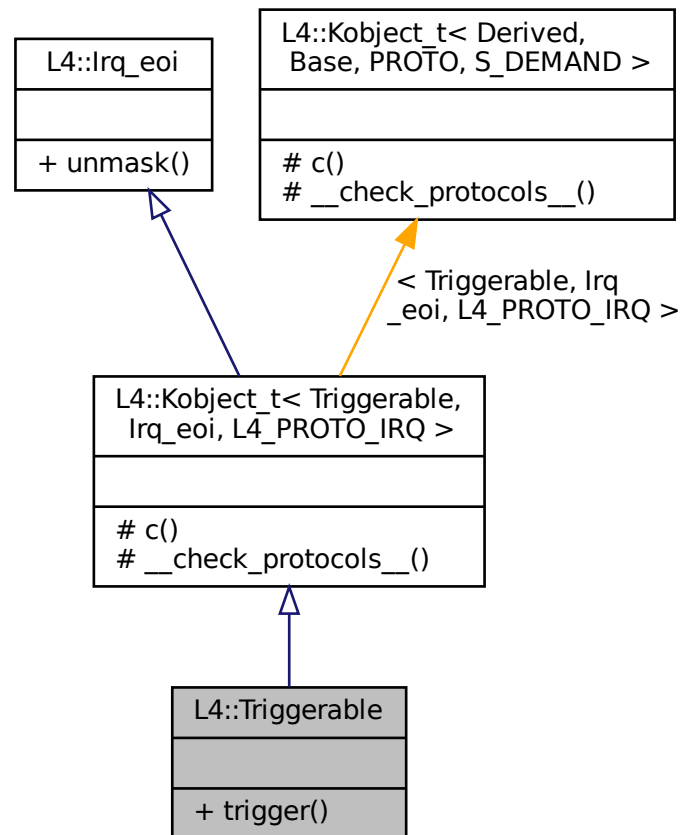
15.180 L4::Triggerable Struct Reference

Interface that allows an object to be triggered by some source.

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.

Additional Inherited Members

15.180.1 Detailed Description

Interface that allows an object to be triggered by some source.

This interface is usually used in conjunction with [L4::lcu](#).

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

15.180.2 Member Function Documentation

15.180.2.1 trigger()

```
l4_msgtag_t L4::Triggerable::trigger (
    l4_utcb_t * utcb = l4_utcb() ) [inline], [noexcept]
```

Trigger.

Parameters

<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.
-------------	---

Returns

Syscall return tag for a send-only operation, use [l4_ipc_error\(\)](#) to check for errors (**do not** use [l4_error\(\)](#)).

Note

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 errors, **do not** use [l4_error\(\)](#), because [l4_error\(\)](#) will always return an error.

Examples

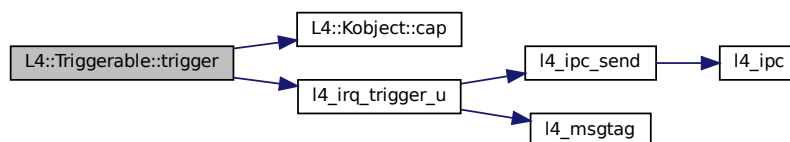
[examples/libs/l4re/c++/shared_ds/ds_clnt.cc](#).

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

References [L4::Kobject::cap\(\)](#), and [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:

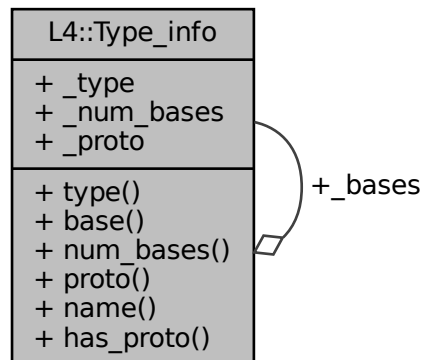
- [l4/sys/irq](#)

15.181 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.

15.181.1 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:

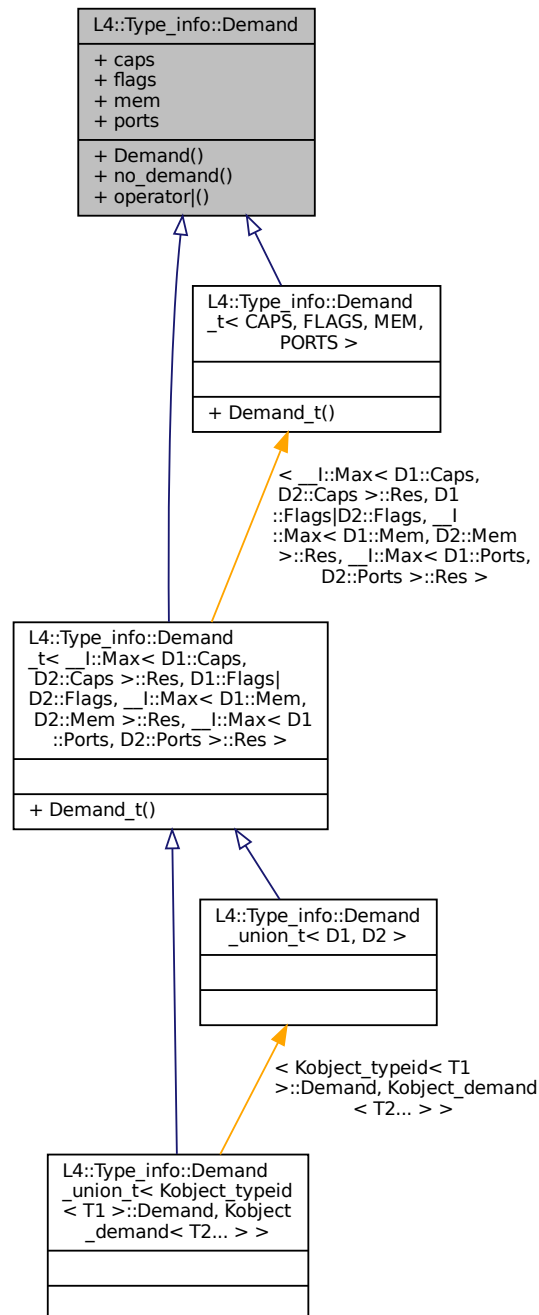
- [l4/sys/__typeinfo.h](#)

15.182 L4::Type_info::Demand Class Reference

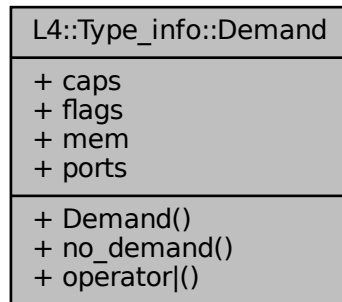
Data type for expressing the needed receive buffers at the server-side of an interface.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Type_info::Demand:



Collaboration diagram for L4::Type_info::Demand:



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.

15.182.1 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](#).

15.182.2 Constructor & Destructor Documentation

15.182.2.1 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](#).

15.182.3 Member Function Documentation

15.182.3.1 no_demand()

```
bool L4::Type_info::Demand::no_demand ( ) const [inline], [noexcept]
```

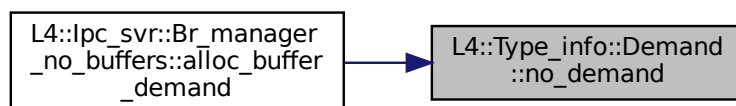
Returns

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:

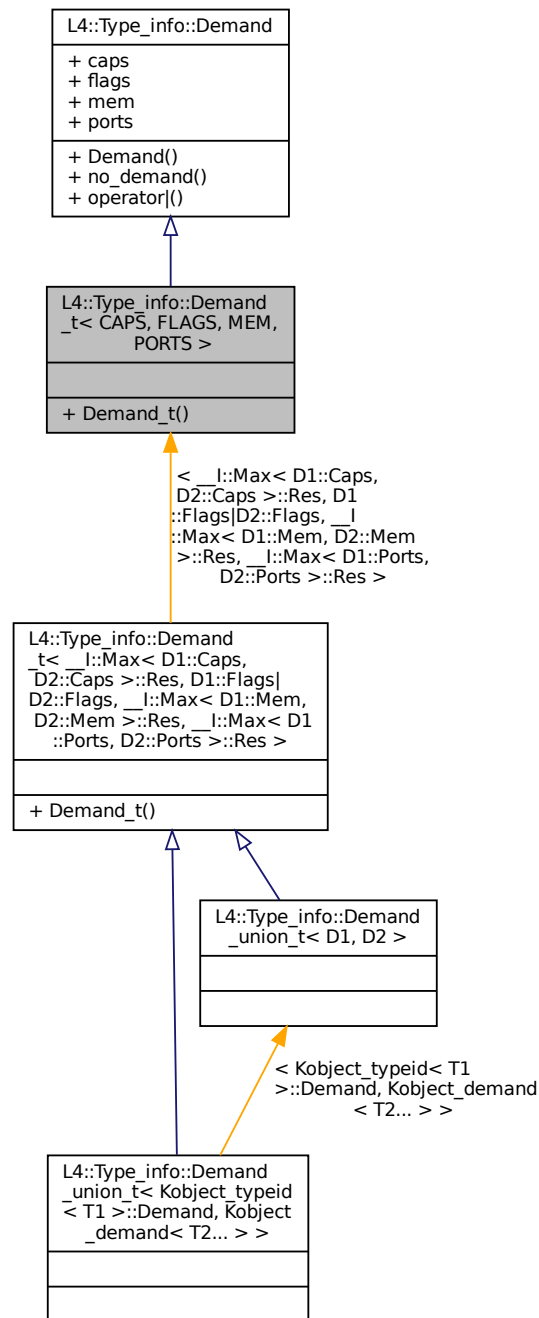
- [l4/sys/__typeinfo.h](#)

15.183 L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS > Struct Template Reference

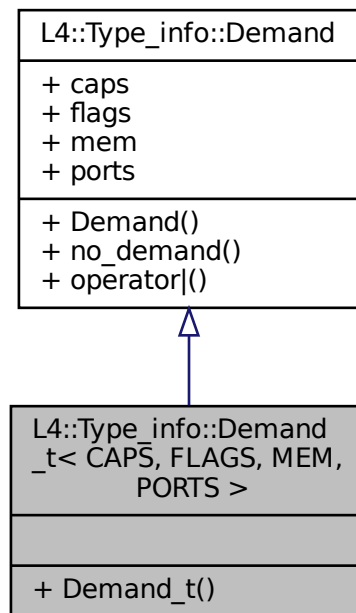
Template type statically describing demand of receive buffers.

```
#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

15.183.1 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 windwows needed.

Definition at line 563 of file [__typeinfo.h](#).

15.183.2 Member Enumeration Documentation

15.183.2.1 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:

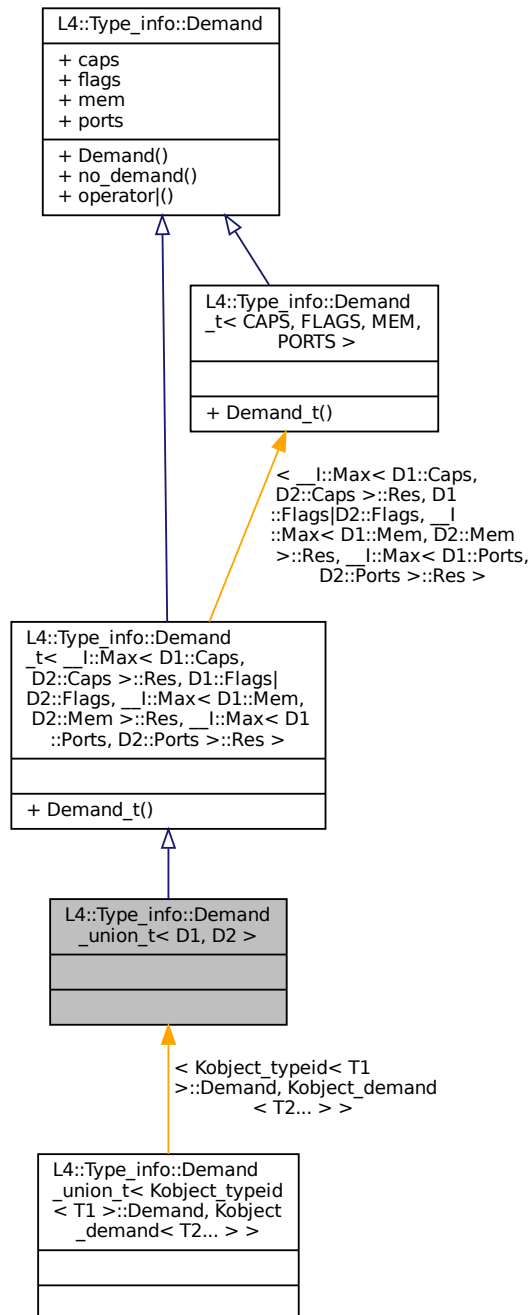
- [l4/sys/__typeinfo.h](#)

15.184 L4::Type_info::Demand_union_t< D1, D2 > Struct Template Reference

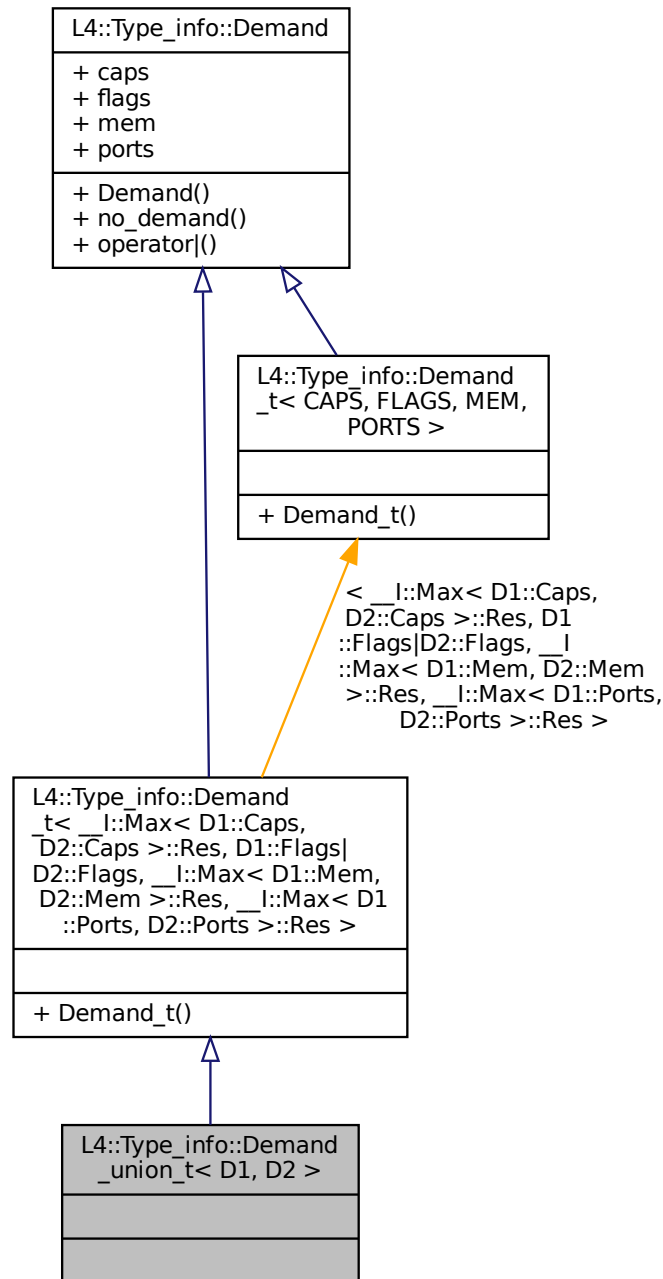
Template type statically describing the combination of two [Demand](#) object.

```
#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

15.184.1 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:

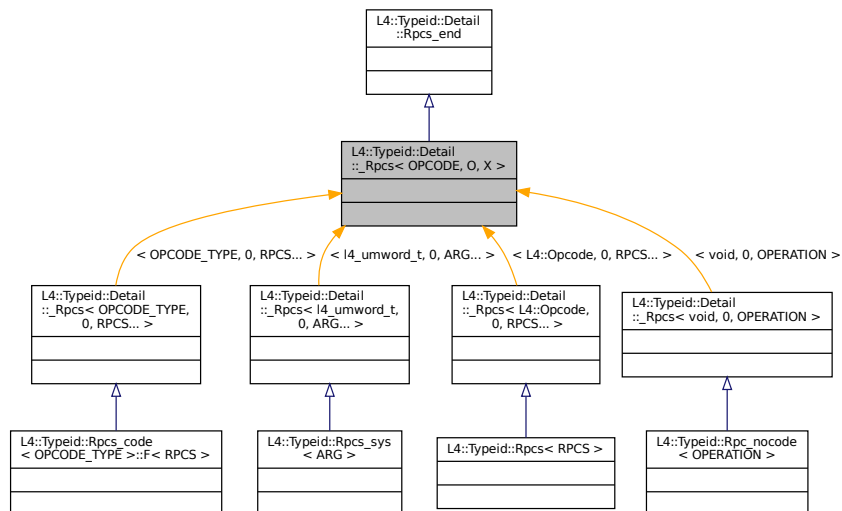
- [l4/sys/__typeinfo.h](#)

15.185 L4::Typeid::Detail::_Rpc< OPCODE, O, X > Struct Template Reference

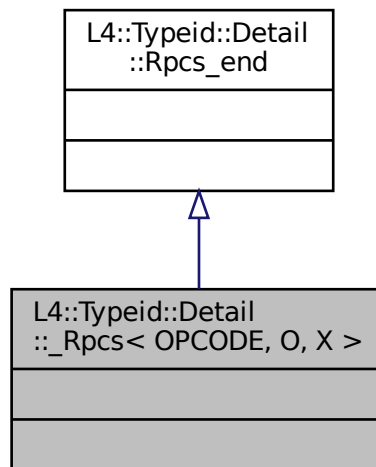
Empty list of RPCs.

```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Typeid::Detail::_Rpc< OPCODE, O, X >:



Collaboration diagram for L4::Typeid::Detail::_Rpc< OPCODE, O, X >:



15.185.1 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](#)

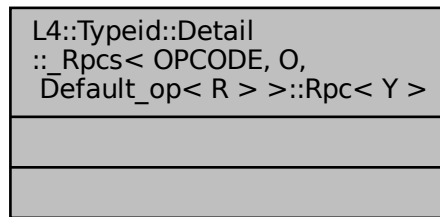
15.186 L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::_Rpc< Y > Struct Template Reference

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 >:



15.186.1 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:

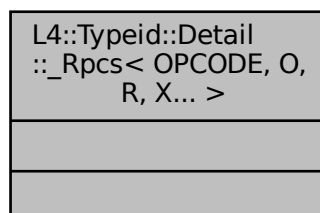
- [l4/sys/__typeinfo.h](#)

15.187 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 [_Rpcs 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 [_Rpcs](#)< OPCODE, _Get_opcode< R, O >::value+1, X... >::type next
The next RPC in the list or [Rpcs_end](#) if this is the last.

15.187.1 Detailed Description

```
template<typename OPCODE, unsigned O, typename R, typename ... X>
struct L4::Typeid::Detail::_Rpcs< 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](#)

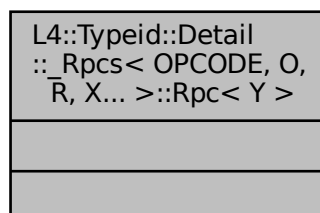
15.188 L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... >::Rpc< Y > Struct Template Reference

Find the given RPC in the list.

```
#include <__typeinfo.h>
```

Inherits L4::Typeid::Detail::Rpc< OP, RPCS >.

Collaboration diagram for L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... >::Rpc< Y >:



15.188.1 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:

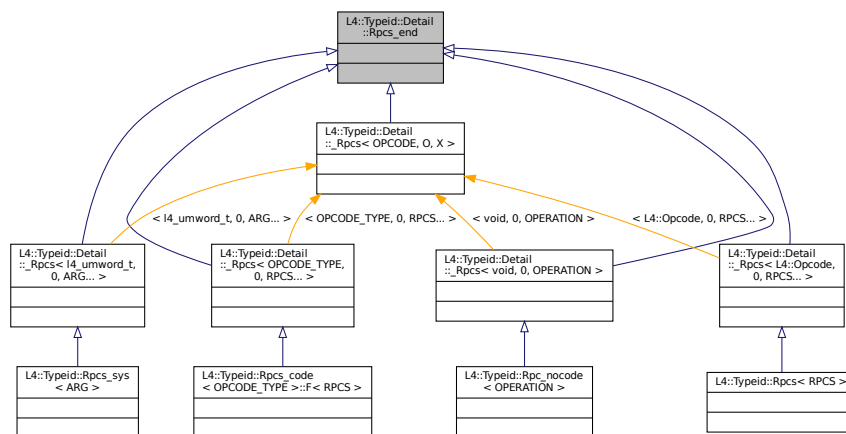
- [l4/sys/__typeinfo.h](#)

15.189 L4::Typeid::Detail::Rpc_end Struct Reference

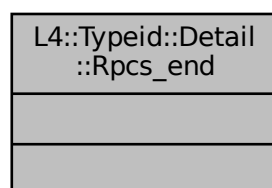
Internal end-of-list marker.

```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Typeid::Detail::Rpc_end:



Collaboration diagram for L4::Typeid::Detail::Rpc_end:



15.189.1 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](#)

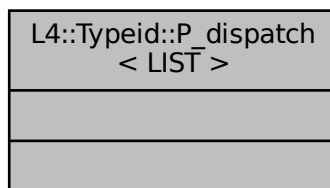
15.190 L4::Typeid::P_dispatch< LIST > Struct Template Reference

Use for protocol based dispatch stage.

```
#include <__typeinfo.h>
```

Inherits L4::Typeid::P_dispatch< LIST >.

Collaboration diagram for L4::Typeid::P_dispatch< LIST >:



15.190.1 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:

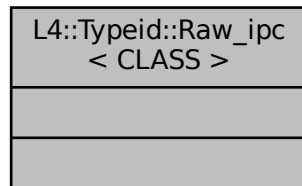
- [l4/sys/__typeinfo.h](#)

15.191 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 >:



15.191.1 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:

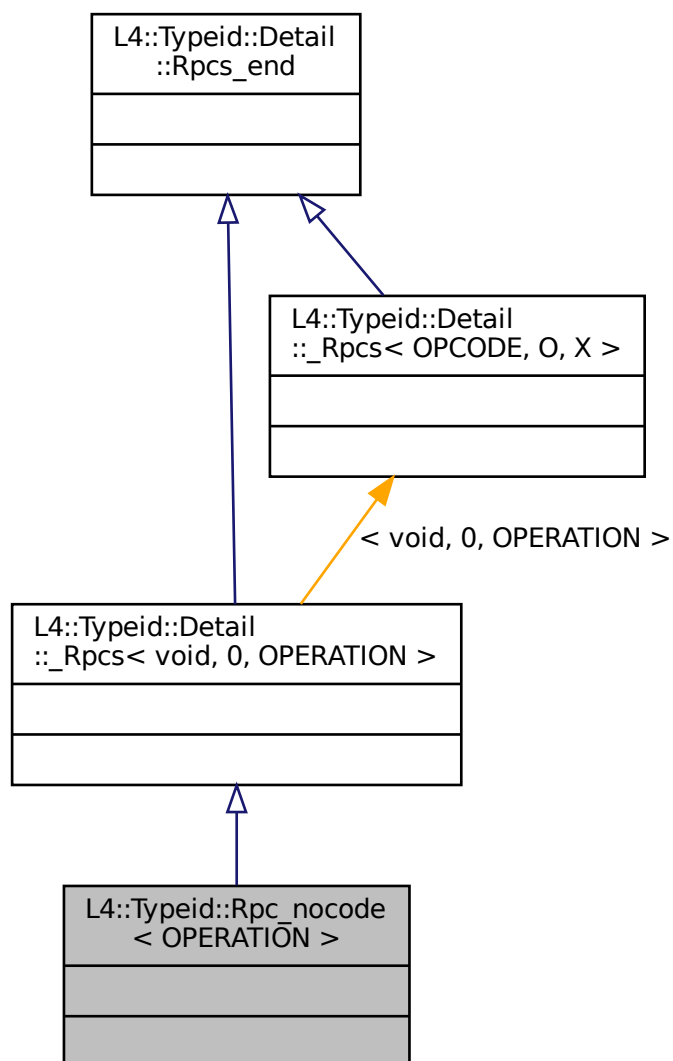
- [l4/sys/__typeinfo.h](#)

15.192 L4::Typeid::Rpc_nocode< OPERATION > Struct Template Reference

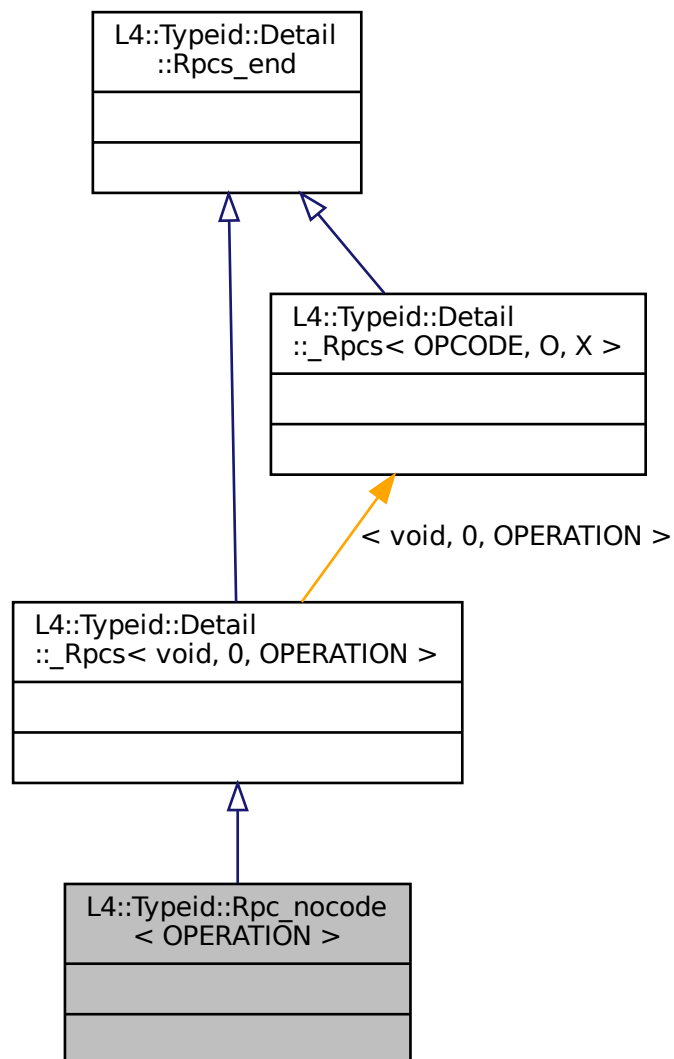
List of RPCs of an interface using a single operation without an opcode.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Typeid::Rpc_nocode < OPERATION >:



Collaboration diagram for L4::Typeid::Rpc_nocode< OPERATION >:



15.192.1 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:

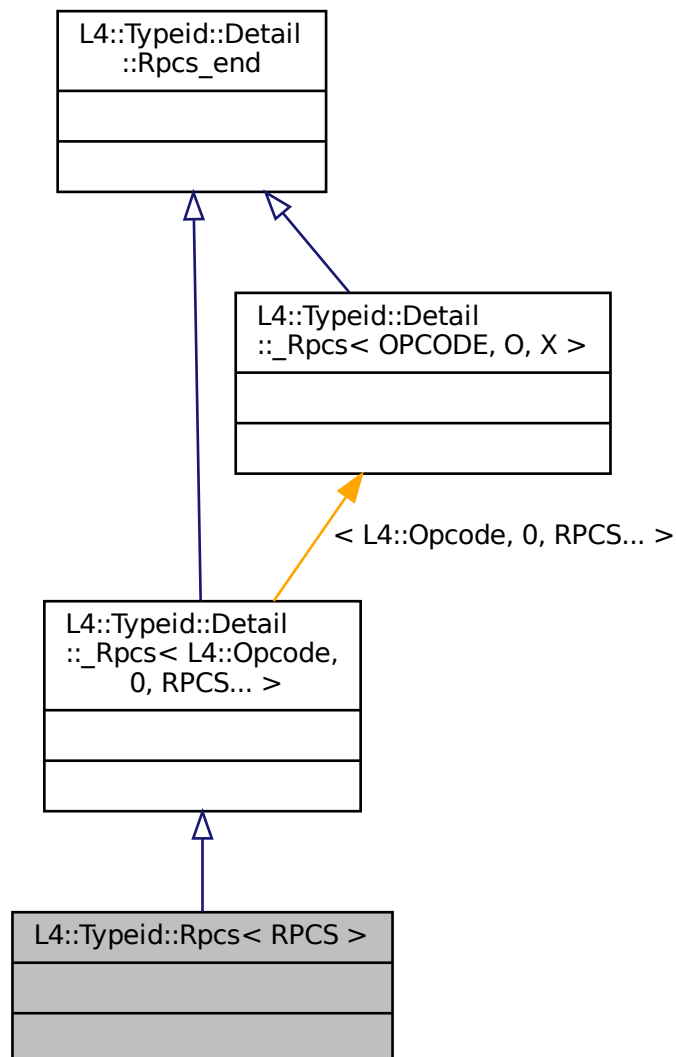
- [l4/sys/__typeinfo.h](#)

15.193 L4::Typeid::Rpc< RPCS > Struct Template Reference

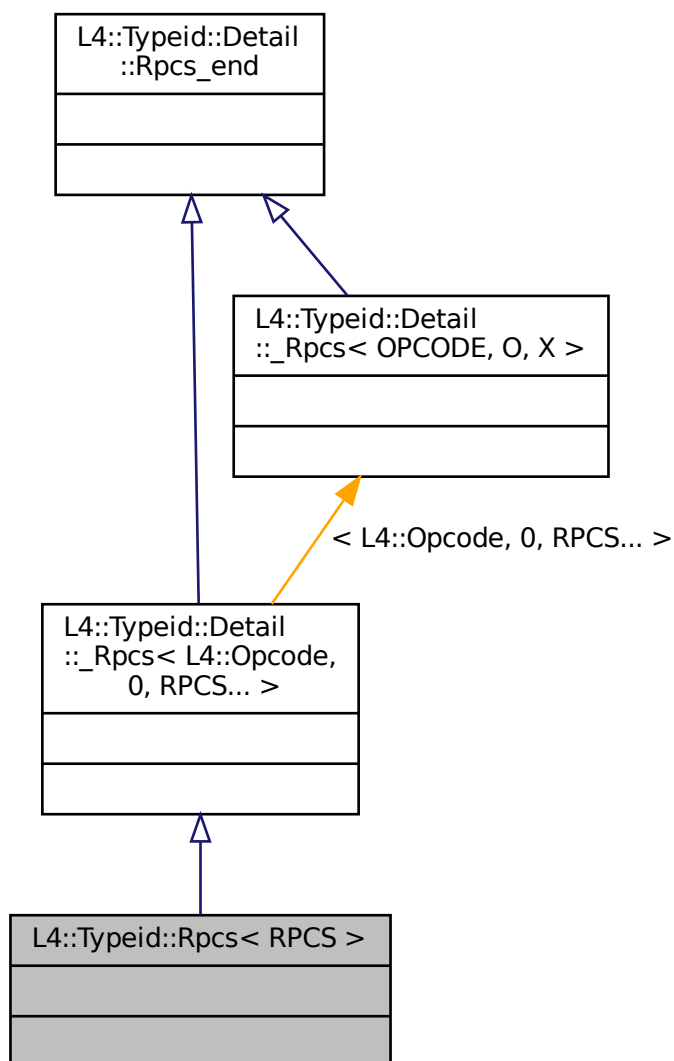
Standard list of RPCs of an interface.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Typeid::Rpc< RPCS >:



Collaboration diagram for L4::Typeid::Rpc< RPCS >:



15.193.1 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:

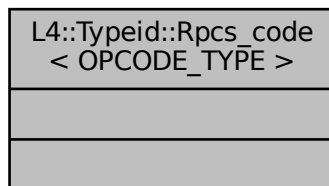
- [l4/sys/__typeinfo.h](#)

15.194 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](#)

15.194.1 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](#).

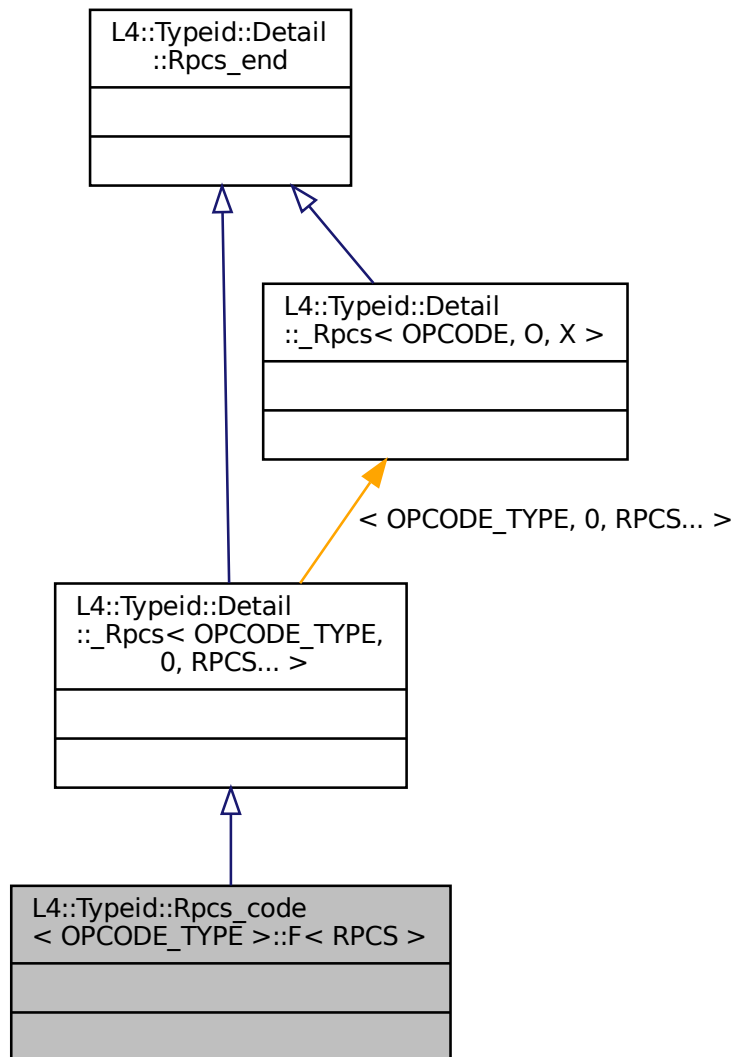
The documentation for this struct was generated from the following file:

- [l4/sys/__typeinfo.h](#)

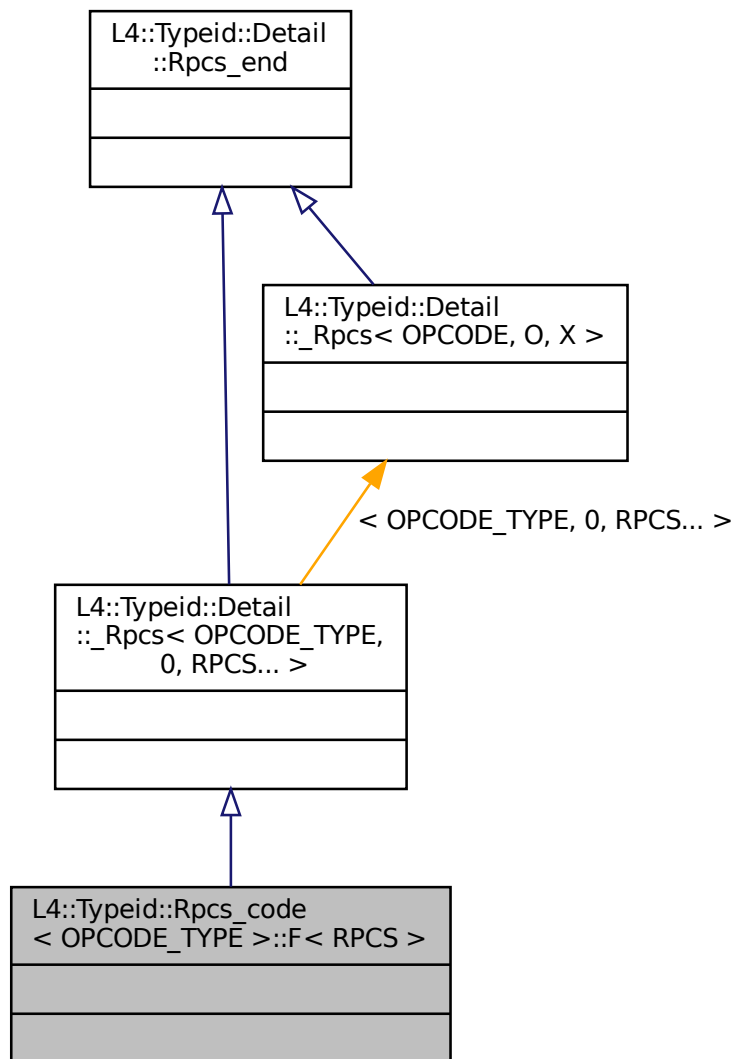
15.195 L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS > Struct Template Reference

```
#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 >:



15.195.1 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:

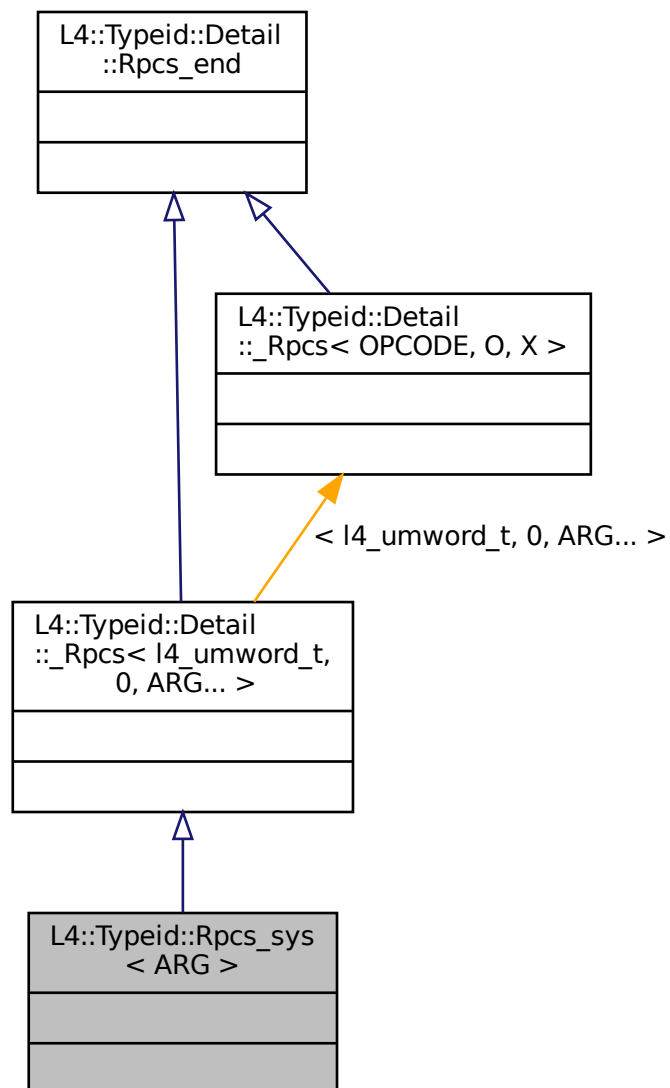
- [l4/sys/__typeinfo.h](#)

15.196 L4::Typeid::Rpcsys< ARG > Struct Template Reference

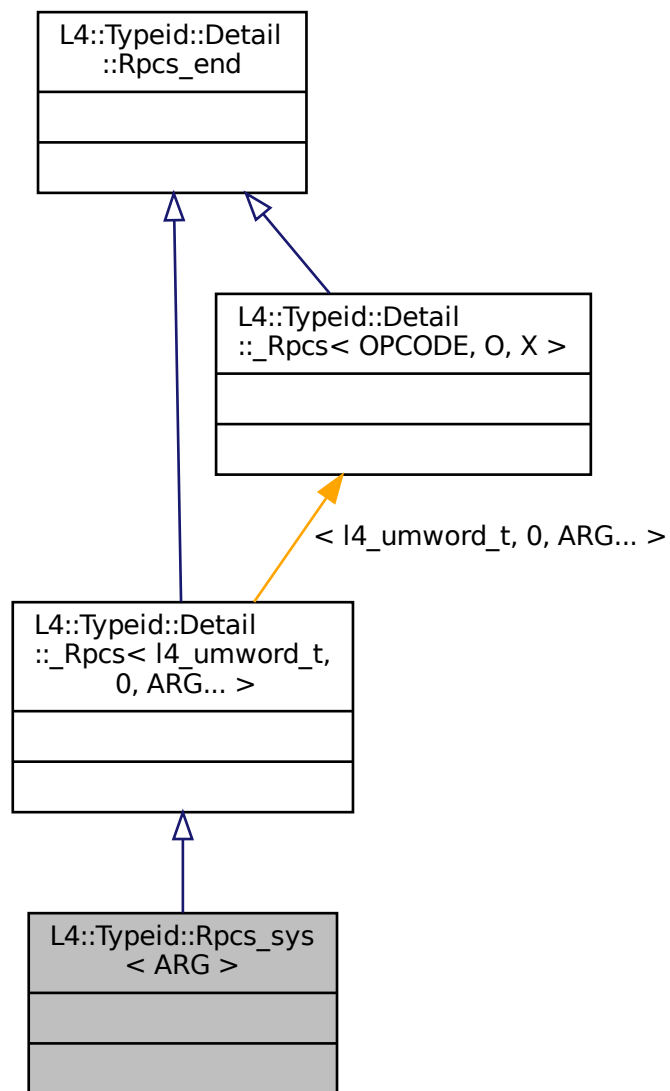
List of RPCs typically used for kernel interfaces.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Typeid::Rpcsys< ARG >:



Collaboration diagram for L4::Typeid::Rpcsys< ARG >:



15.196.1 Detailed Description

```
template<typename ... ARG>
struct L4::Typeid::Rpcsys_sys< 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](#).

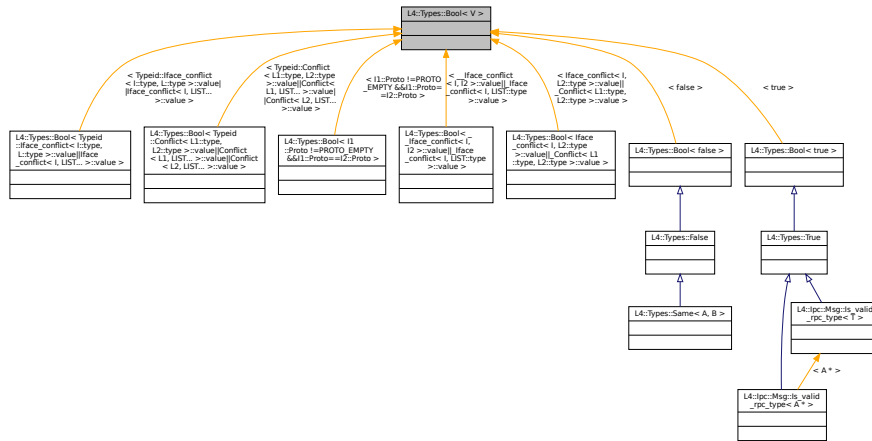
The documentation for this struct was generated from the following file:

- [I4/sys/__typeinfo.h](#)

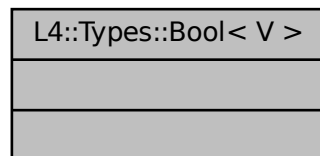
15.197 L4::Types::Bool< V > Struct Template Reference

Boolean meta type.

Inheritance diagram for `L4::Types::Bool< V >`:



Collaboration diagram for `L4::Types::Bool< V >`:



Public Types

- typedef `Bool< V > type`
The meta type itself.

15.197.1 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](#).

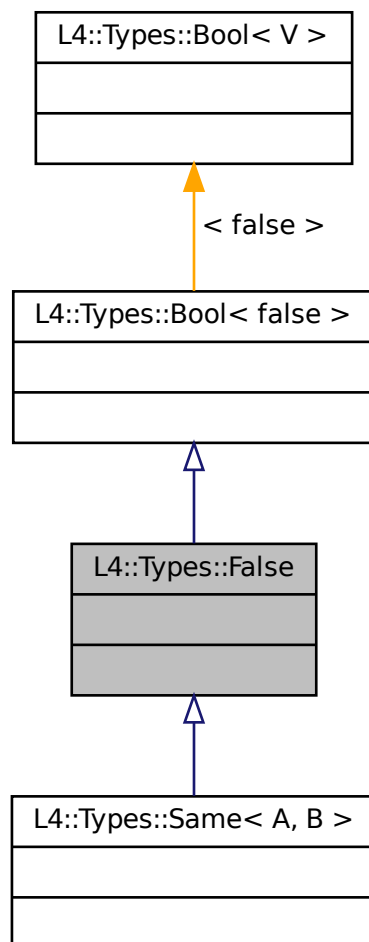
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/types](#)

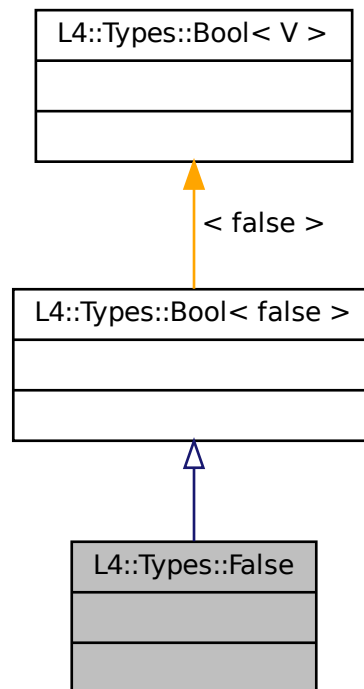
15.198 L4::Types::False Struct Reference

[False](#) meta value.

Inheritance diagram for L4::Types::False:



Collaboration diagram for L4::Types::False:



Additional Inherited Members

15.198.1 Detailed Description

[False](#) meta value.

Definition at line [308](#) of file [types](#).

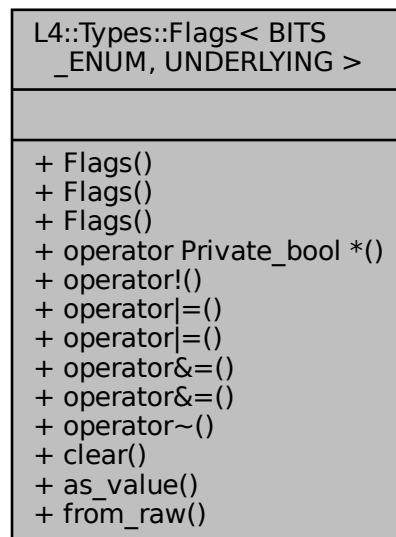
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/types](#)

15.199 L4::Types::Flags< BITS_ENUM, UNDERLYING > Class Template Reference

Template for defining typical [Flags](#) bitmaps.

Collaboration diagram for L4::Types::Flags< BITS_ENUM, UNDERLYING >:



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.

15.199.1 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](#).

15.199.2 Member Enumeration Documentation

15.199.2.1 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](#).

15.199.3 Constructor & Destructor Documentation

15.199.3.1 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](#).

15.199.3.2 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](#).

15.199.4 Member Function Documentation

15.199.4.1 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:



15.199.4.2 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](#).

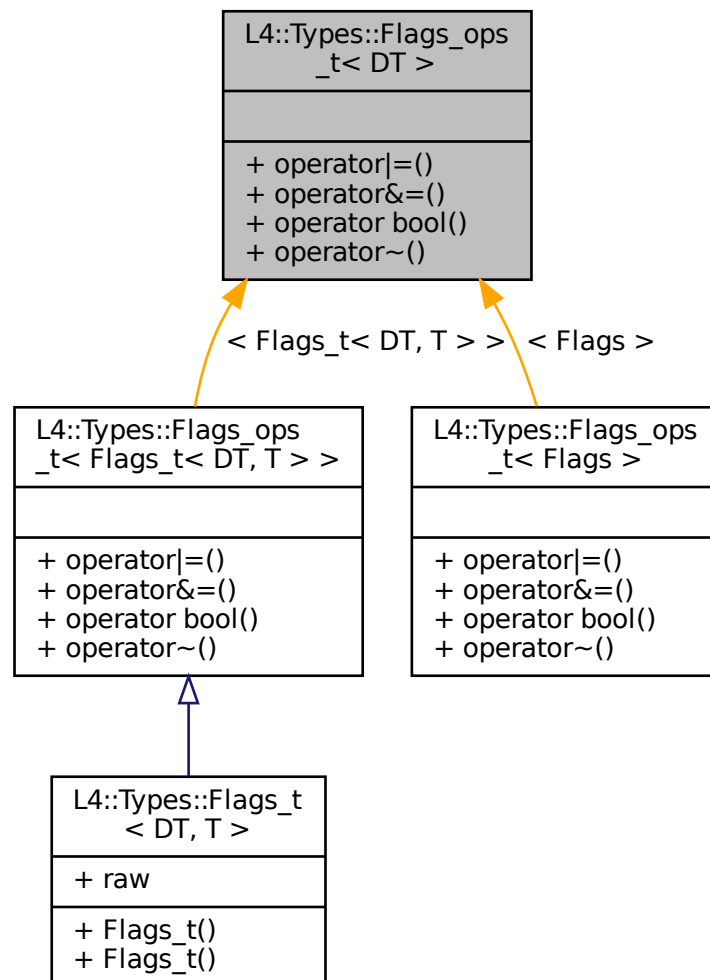
The documentation for this class was generated from the following file:

- [l4/sys/cxx/types](#)

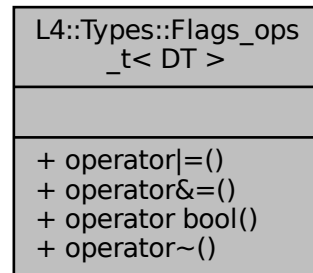
15.200 L4::Types::Flags_ops_t< DT > Struct Template Reference

Mixin class to define a set of friend bitwise operators on DT.

Inheritance diagram for L4::Types::Flags_ops_t< DT >:



Collaboration diagram for L4::Types::Flags_ops_t< DT >:



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 friend DT [operator|](#) (DT l, DT r)
bitwise or for DT
- constexpr friend DT [operator&](#) (DT l, DT r)
bitwise and for DT
- constexpr friend bool [operator==](#) (DT l, DT r)
equality for DT
- constexpr friend bool [operator!=](#) (DT l, DT r)
inequality for DT

15.200.1 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](#).

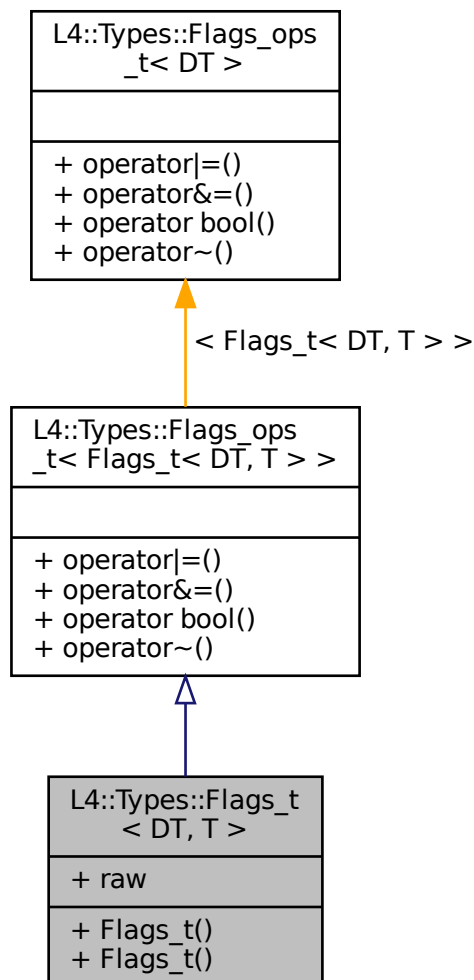
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/types](#)

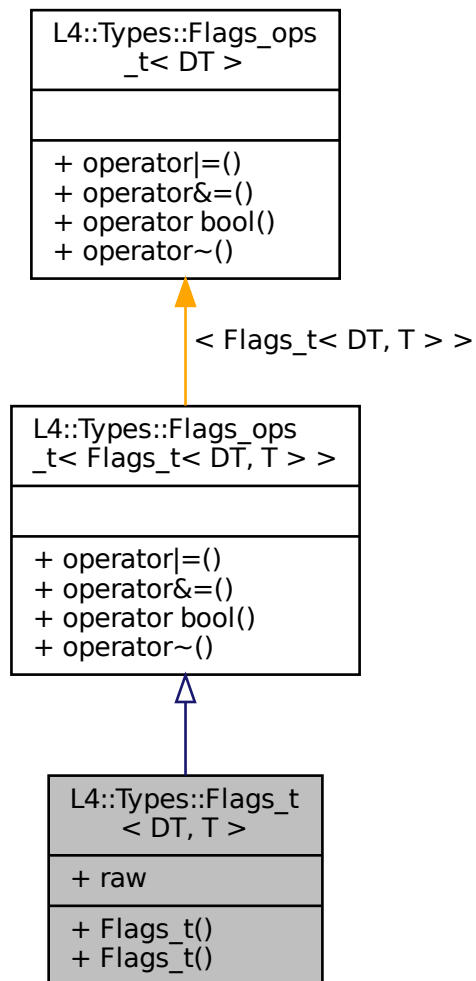
15.201 L4::Types::Flags_t< DT, T > Struct Template Reference

Template type to define a flags type with bitwise operations.

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.

Data Fields

- `T raw`
Raw integral value.

15.201.1 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](#).

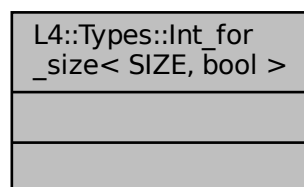
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/types](#)

15.202 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 >:



15.202.1 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](#).

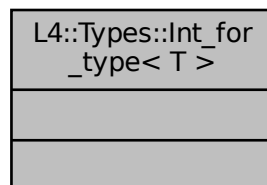
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/types](#)

15.203 L4::Types::Int_for_type< T > Struct Template Reference

Metafunction to get an integral type of the same size as T .

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 .

15.203.1 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

T	The type for which an unsigned integral type with the same size is needed.
-----	--

Definition at line 192 of file [types](#).

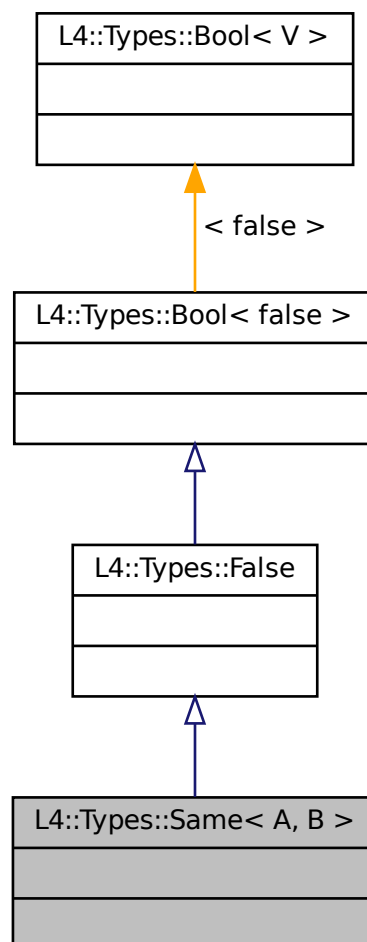
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/types](#)

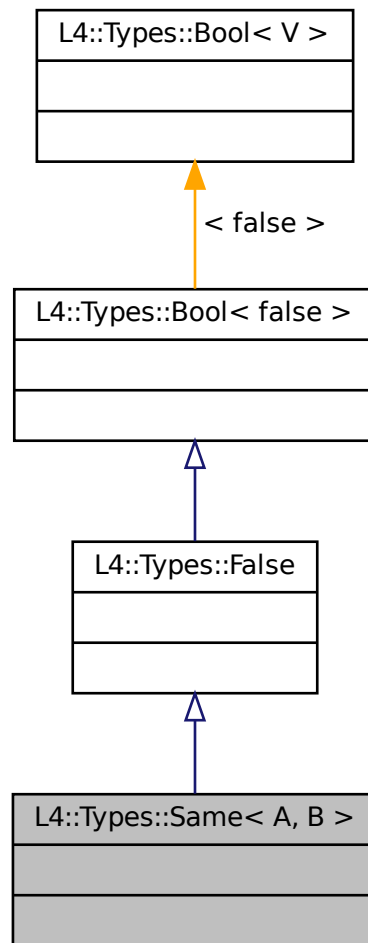
15.204 L4::Types::Same< A, B > Struct Template Reference

Compare two data types for equality.

Inheritance diagram for L4::Types::Same< A, B >:



Collaboration diagram for L4::Types::Same< A, B >:



Additional Inherited Members

15.204.1 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](#).

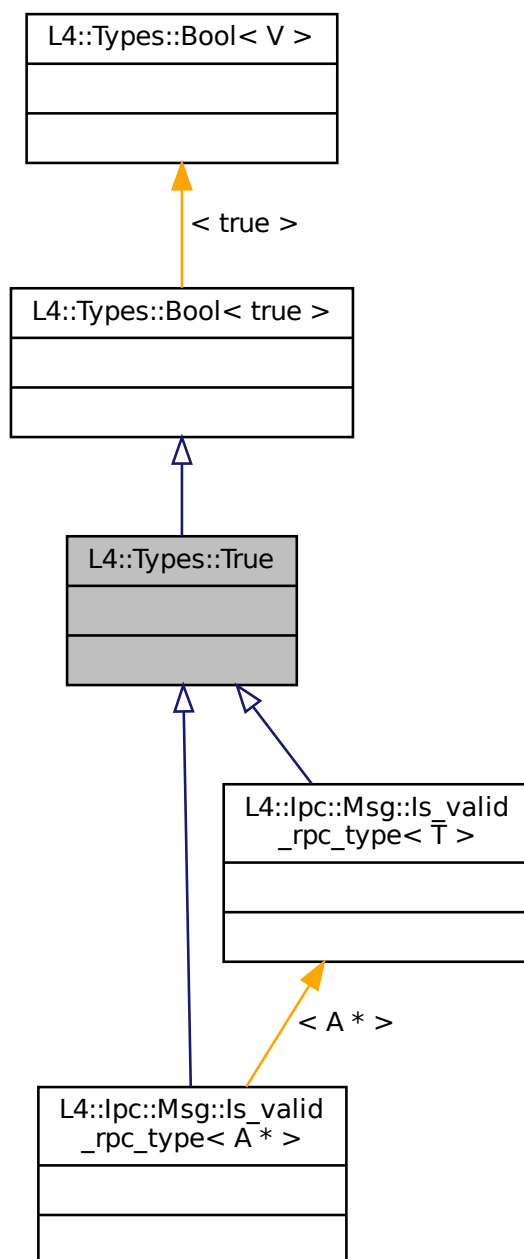
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/types](#)

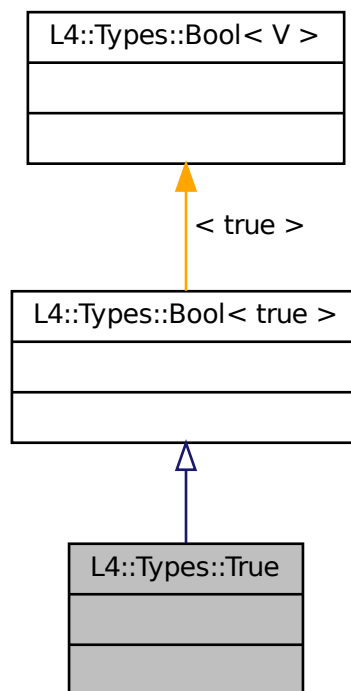
15.205 L4::Types::True Struct Reference

[True](#) meta value.

Inheritance diagram for L4::Types::True:



Collaboration diagram for L4::Types::True:



Additional Inherited Members

15.205.1 Detailed Description

`True` meta value.

Definition at line 312 of file `types`.

The documentation for this struct was generated from the following file:

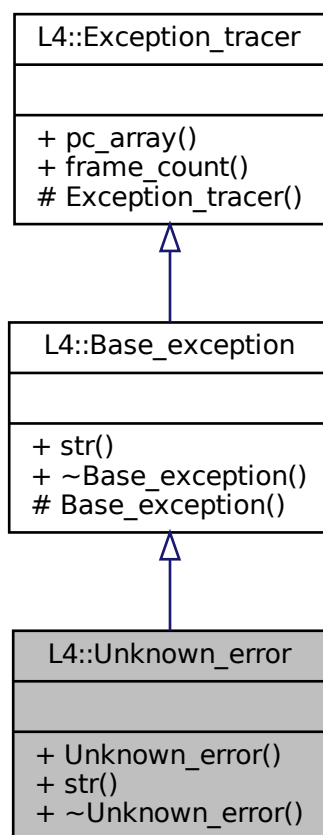
- `l4/sys/cxx/types`

15.206 L4::Unknown_error Class Reference

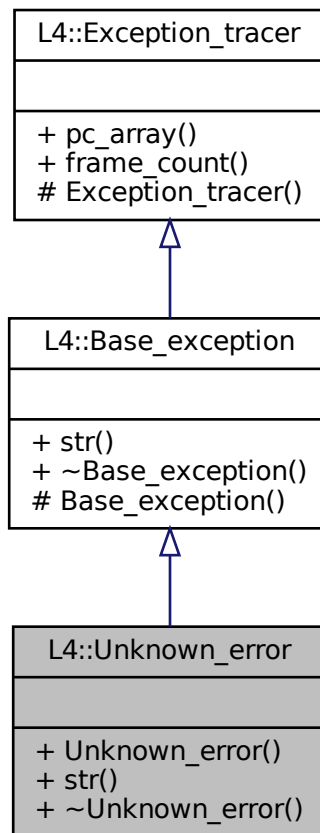
`Exception` for an unknown condition.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Unknown_error:



Collaboration diagram for L4::Unknown_error:



Public Member Functions

- `char const * str () const throw ()`
Return a human readable string for the exception.

Additional Inherited Members

15.206.1 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](#).

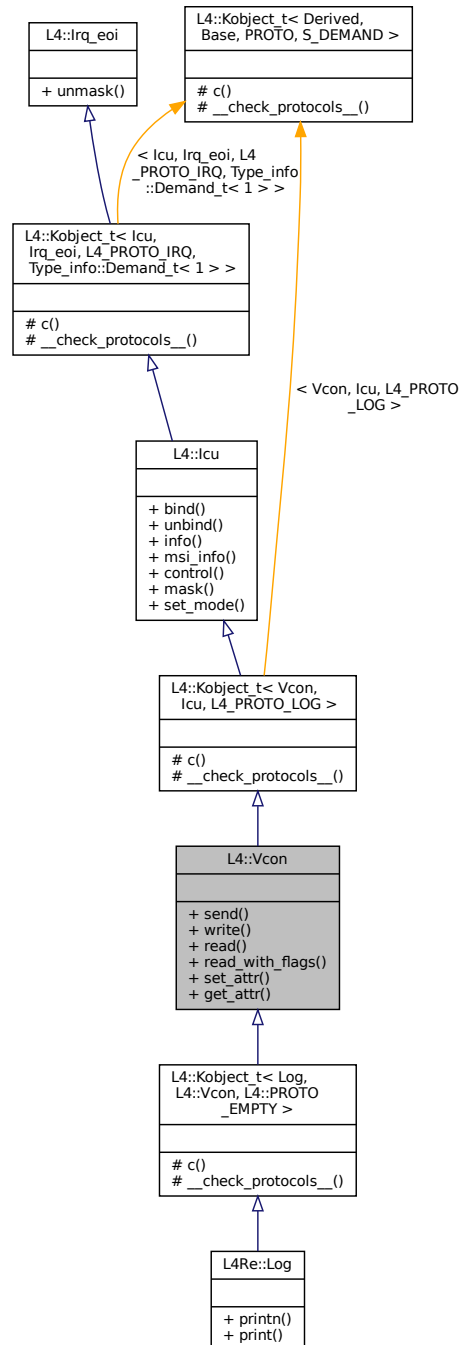
The documentation for this class was generated from the following file:

- `l4/cxx/exceptions`

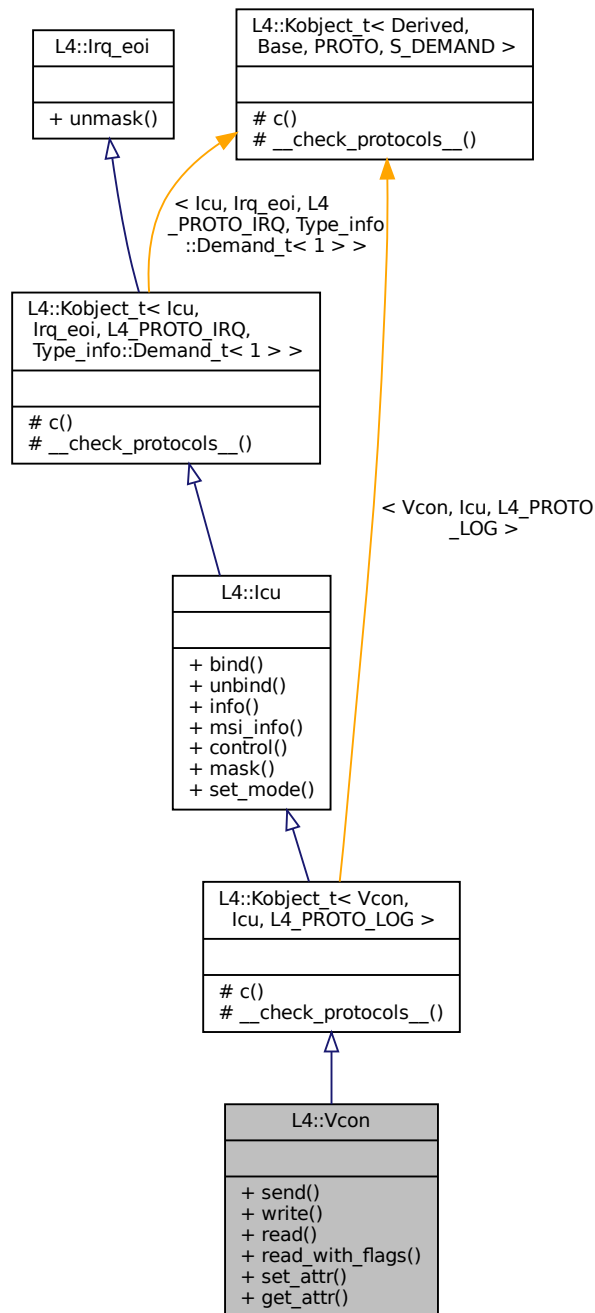
15.207 L4::Vcon Class Reference

C++ [L4 Vcon](#) interface.

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.

Additional Inherited Members

15.207.1 Detailed Description

C++ [L4 Vcon](#) 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.

Include File

```
#include <l4/sys/vcon>
```

See the [Virtual Console](#) for the C interface.

Definition at line 43 of file [vcon](#).

15.207.2 Member Function Documentation

15.207.2.1 [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	<i>attr</i>	Attribute structure. Contains the attributes after a successful call of this function.
	<i>utcb</i>	UTCB pointer of the calling thread.

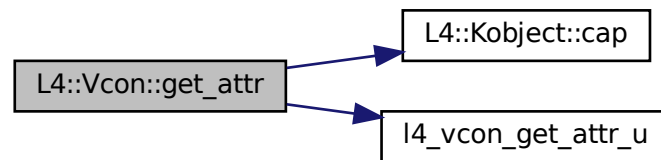
Returns

Syscall return tag.

Definition at line 145 of file [vcon](#).

References [L4::Kobject::cap\(\)](#), and [l4_vcon_get_attr_u\(\)](#).

Here is the call graph for this function:



15.207.2.2 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	<i>buf</i>	Pointer to data buffer.
	<i>size</i>	Size of the data buffer in bytes.
	<i>utcb</i>	UTCB pointer of the calling thread.

Return values

<0	Error code.
> <i>size</i>	More bytes to read, <i>size</i> bytes are in the buffer <i>buf</i> .
<= <i>size</i>	Number of bytes read.

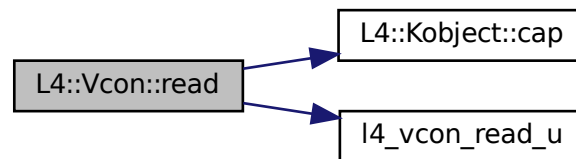
Note

Size must not exceed [L4_VCON_READ_SIZE](#).

Definition at line 94 of file [vcon](#).

References [L4::Kobject::cap\(\)](#), and [l4_vcon_read_u\(\)](#).

Here is the call graph for this function:



15.207.2.3 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 pointer of the calling thread.

Return values

<0	Error code.
> <i>size</i>	More bytes to read, <i>size</i> bytes are in the buffer <i>buf</i> .
<= <i>size</i>	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 [119](#) of file [vcon](#).

15.207.2.4 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>	UTBC pointer of the calling thread.

Returns

Syscall return tag

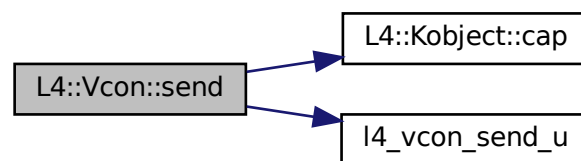
Note

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 63 of file [vcon](#).

References [L4::Kobject::cap\(\)](#), and [l4_vcon_send_u\(\)](#).

Here is the call graph for this function:



15.207.2.5 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 pointer of the calling thread.

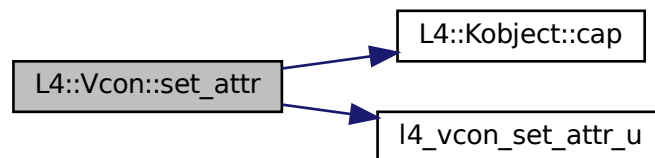
Returns

Syscall return tag.

Definition at line 132 of file [vcon](#).

References [L4::Kobject::cap\(\)](#), and [l4_vcon_set_attr_u\(\)](#).

Here is the call graph for this function:



15.207.2.6 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 pointer of the calling thread.

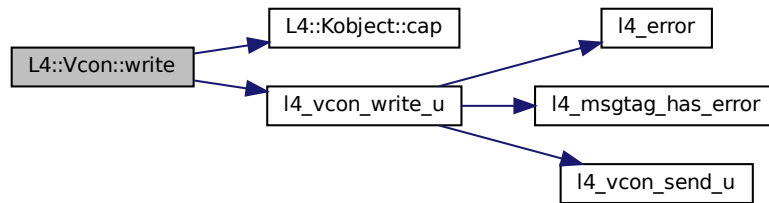
Return values

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

Definition at line 77 of file [vcon](#).

References [L4::Kobject::cap\(\)](#), and [l4_vcon_write_u\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

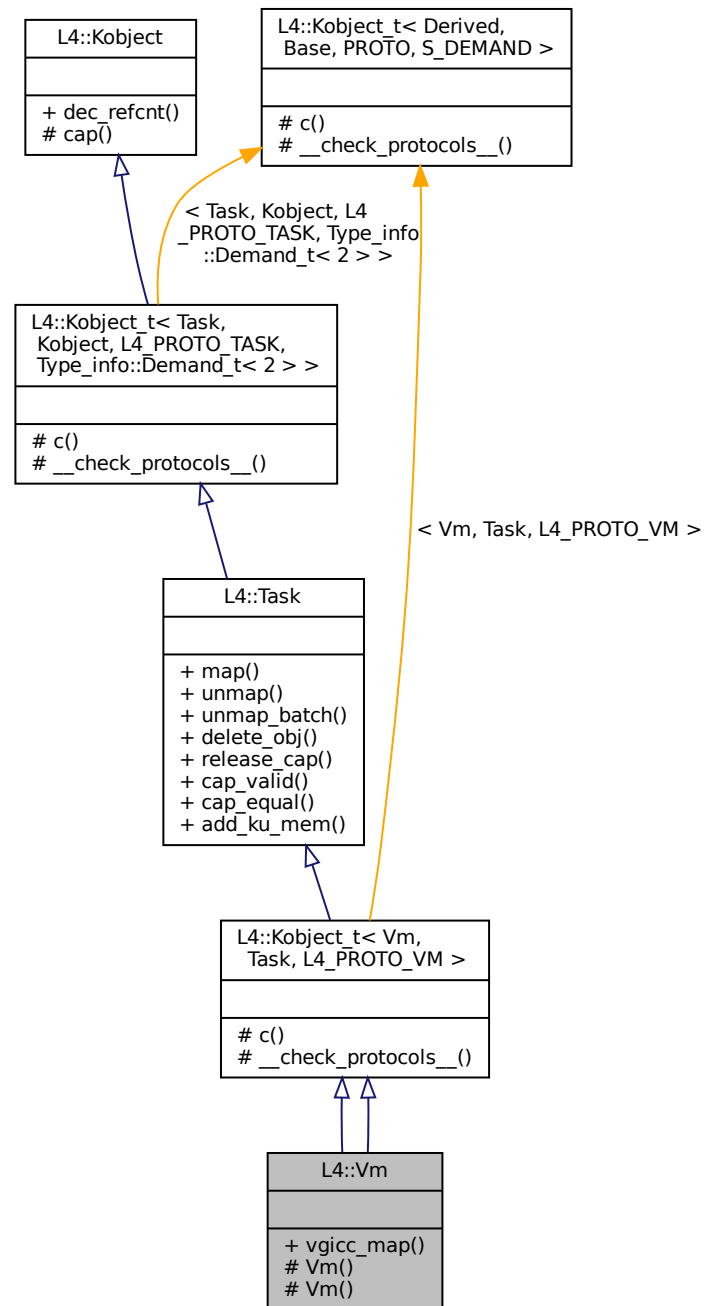
- [l4/sys/vcon](#)

15.208 L4::Vm Class Reference

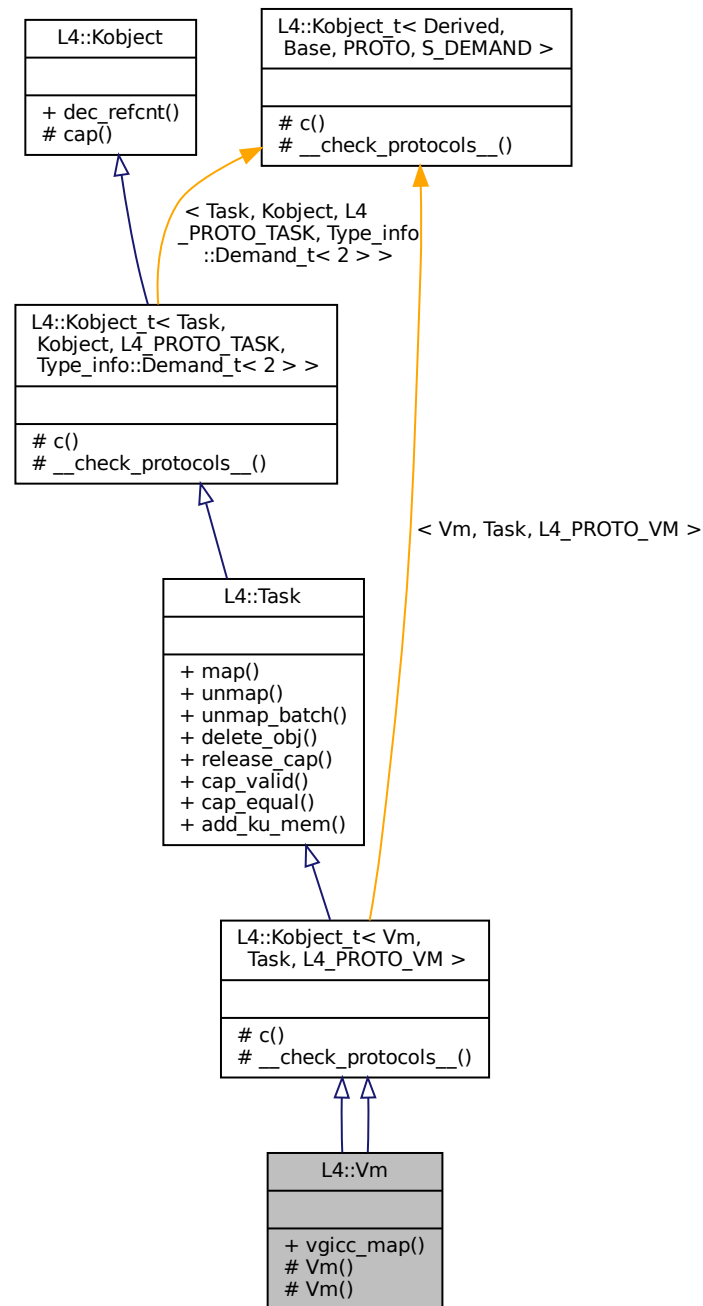
Virtual machine host address space.

```
#include <__vm-arm.h>
```

Inheritance diagram for L4::Vm:



Collaboration diagram for L4::Vm:



Additional Inherited Members

15.208.1 Detailed Description

Virtual machine host address space.

Virtual machine.

[L4::Vm](#) is a specialisation of [L4::Task](#), used for virtual machines. For Arm, it offers a call to make the virtual GICC area available to the VM.

[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.

Definition at line 36 of file [__vm-arm.h](#).

The documentation for this class was generated from the following files:

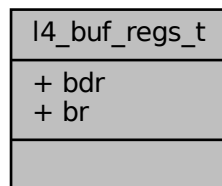
- [l4/sys/__vm-arm.h](#)
- [l4/sys/vm](#)

15.209 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.

15.209.1 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:

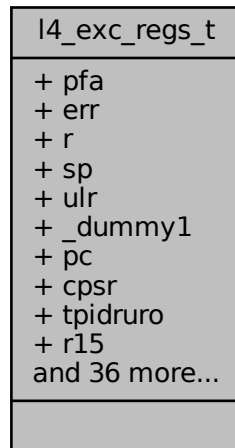
- [l4/sys/utcb.h](#)

15.210 l4_exc_regs_t Struct Reference

UTCB structure for exceptions.

```
#include <utcb.h>
```

Collaboration diagram for l4_exc_regs_t:



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 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

15.210.1 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](#).

15.210.2 Field Documentation

15.210.2.1 flags

[l4_umword_t](#) l4_exc_regs_t::flags

rflags

eflags

Definition at line 80 of file [utcb.h](#).

15.210.2.2 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 file:

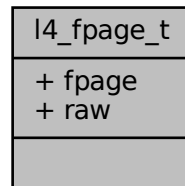
- [arm/l4/sys/utcb.h](#)

15.211 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.

15.211.1 Detailed Description

[L4](#) flexpage type.

Definition at line 83 of file [__l4_fpage.h](#).

The documentation for this union was generated from the following file:

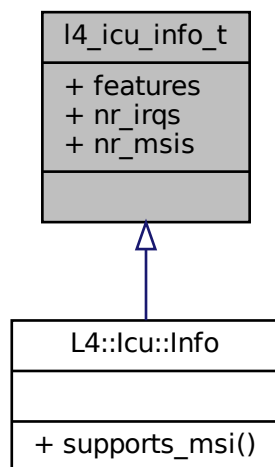
- l4/sys/__l4_fpage.h

15.212 l4_icu_info_t Struct Reference

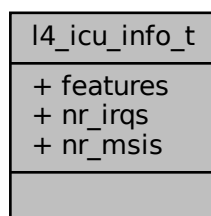
Info structure for an ICU.

```
#include <icu.h>
```


Inheritance diagram for l4_icu_info_t:



Collaboration diagram for l4_icu_info_t:



Data Fields

- unsigned `features`
Feature flags.
- unsigned `nr_irqs`
The number of IRQ lines supported by the ICU,.
- unsigned `nr_msis`
The number of MSI vectors supported by the ICU,.

15.212.1 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 [159](#) of file [icu.h](#).

15.212.2 Field Documentation

15.212.2.1 features

```
unsigned l4_icu_info_t::features
```

Feature flags.

If [L4_ICU_FLAG_MSI](#) is set the ICU supports MSIs.

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

Referenced by [L4::Icu::Info::supports_msi\(\)](#).

The documentation for this struct was generated from the following file:

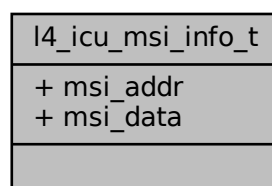
- [l4/sys/icu.h](#)

15.213 l4_icu_msi_info_t Struct Reference

Info to use for a specific MSI.

```
#include <icu.h>
```

Collaboration diagram for [l4_icu_msi_info_t](#):



Data Fields

- [l4_uint64_t msi_addr](#)
Value to use as address when sending this MSI.
- [l4_uint32_t msi_data](#)
Value to use as data written to msi_addr, when sending this MSI.

15.213.1 Detailed Description

Info to use for a specific MSI.

Definition at line 180 of file [icu.h](#).

The documentation for this struct was generated from the following file:

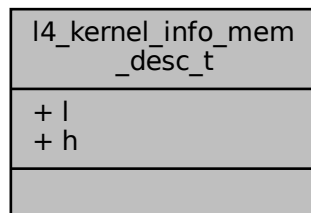
- [l4/sys/icu.h](#)

15.214 l4_kernel_info_mem_desc_t Struct Reference

Memory descriptor data structure.

```
#include <memdesc.h>
```

Collaboration diagram for l4_kernel_info_mem_desc_t:



15.214.1 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 74 of file [memdesc.h](#).

The documentation for this struct was generated from the following file:

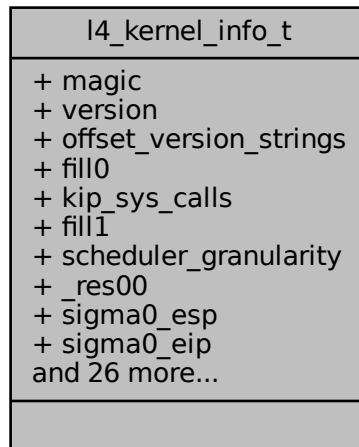
- [l4/sys/memdesc.h](#)

15.215 l4_kernel_info_t Struct Reference

[L4 Kernel Interface Page](#).

```
#include <__kip-32bit.h>
```

Collaboration diagram for l4_kernel_info_t:



Data Fields

- [l4_uint32_t magic](#)
Kernel Info Page identifier ("L4μK").
- [l4_uint32_t version](#)
Kernel version.
- [l4_uint8_t offset_version_strings](#)
offset to version string
- [l4_uint8_t fill0](#) [3]
reserved
- [l4_uint8_t kip_sys_calls](#)
pointer to system calls
- [l4_uint8_t fill1](#) [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

15.215.1 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:

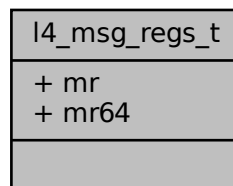
- [l4/sys/__kip-32bit.h](#)
- [l4/sys/__kip-64bit.h](#)

15.216 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.

15.216.1 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](#)

15.217 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

15.217.1 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/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 [159](#) of file [types.h](#).

15.217.2 Member Function Documentation

15.217.2.1 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 [177](#) of file [types.h](#).

References [raw](#).

The documentation for this struct was generated from the following file:

- [l4/sys/types.h](#)

15.218 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.

15.218.1 Detailed Description

CPU sets.

Examples

[examples/sys/migrate/thread_migrate.cc](#).

Definition at line [44](#) of file [scheduler.h](#).

15.218.2 Member Function Documentation

15.218.2.1 granularity()

```
unsigned char l4_sched_cpu_set_t::granularity ( ) const [inline]
```

Returns

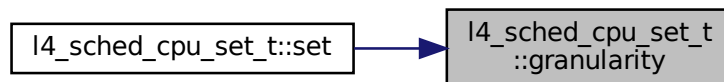
Get granularity value

Definition at line 67 of file [scheduler.h](#).

References [gran_offset](#).

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

Here is the caller graph for this function:



15.218.2.2 offset()

```
unsigned l4_sched_cpu_set_t::offset ( ) const [inline]
```

Returns

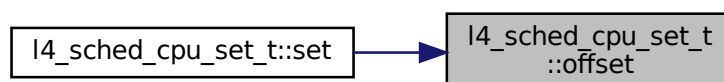
Get offset value

Definition at line 69 of file [scheduler.h](#).

References [gran_offset](#).

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

Here is the caller graph for this function:



15.218.3 Field Documentation

15.218.3.1 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 58 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:

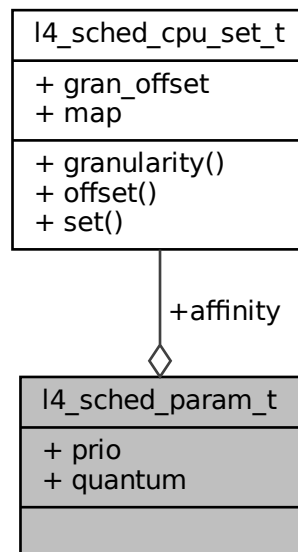
- [l4/sys/scheduler.h](#)

15.219 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.

15.219.1 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 120 of file [scheduler.h](#).

The documentation for this struct was generated from the following file:

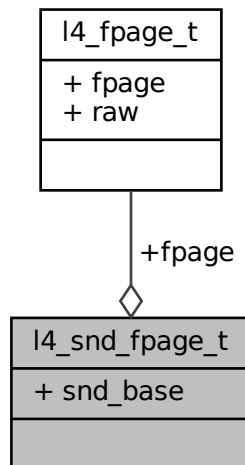
- [l4/sys/scheduler.h](#)

15.220 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.

15.220.1 Detailed Description

Send-flex-page types.

Definition at line 100 of file [__l4_fpage.h](#).

The documentation for this struct was generated from the following file:

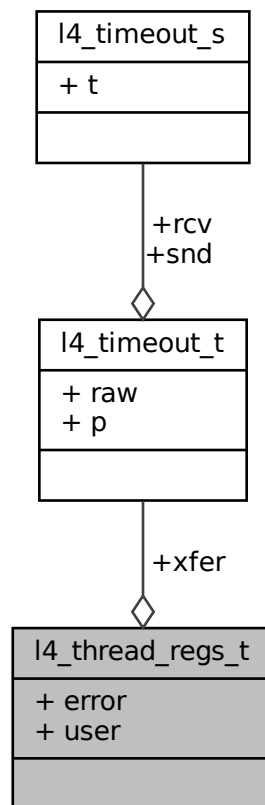
- `l4/sys/__l4_fpage.h`

15.221 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_timeout_t xfer](#)
Message transfer timeout.
- [l4_umword_t user](#) [3]
User values (ignored and preserved by the kernel)

15.221.1 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:

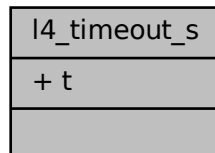
- [l4/sys/utcb.h](#)

15.222 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

15.222.1 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:

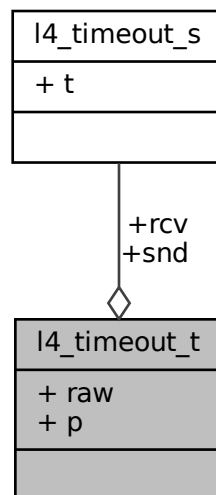
- [l4/sys/__timeout.h](#)

15.223 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*

### 15.223.1 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](#)



## 15.224 l4\_vcon\_attr\_t Struct Reference

Vcon attribute structure.

```
#include <vcon.h>
```

Collaboration diagram for l4\_vcon\_attr\_t:



### Data Fields

- [l4\\_umword\\_t i\\_flags](#)  
*input flags*
- [l4\\_umword\\_t o\\_flags](#)  
*output flags*
- [l4\\_umword\\_t l\\_flags](#)  
*local flags*

### 15.224.1 Detailed Description

Vcon attribute structure.

Definition at line 178 of file [vcon.h](#).

The documentation for this struct was generated from the following file:

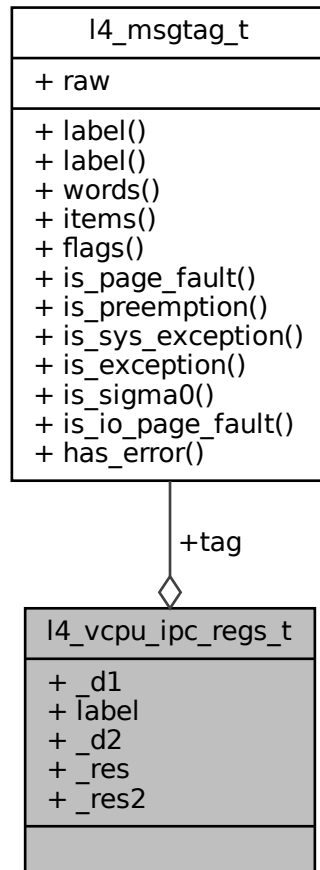
- [l4/sys/vcon.h](#)

## 15.225 l4\_vcpu\_ipc\_regs\_t Struct Reference

vCPU message registers.

```
#include <__vcpu-arch.h>
```

Collaboration diagram for l4\_vcpu\_ipc\_regs\_t:



### 15.225.1 Detailed Description

vCPU message registers.

Definition at line 63 of file [\\_\\_vcpu-arch.h](#).

The documentation for this struct was generated from the following file:

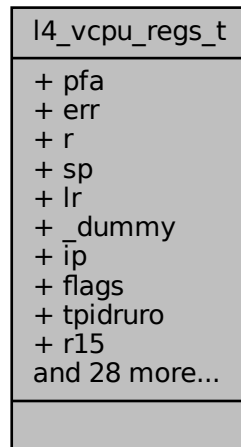
- arm/l4/sys/\_\_vcpu-arch.h

## 15.226 l4\_vcpu\_regs\_t Struct Reference

vCPU registers.

```
#include <__vcpu-arch.h>
```

Collaboration diagram for l4\_vcpu\_regs\_t:



### 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 tpidruro](#)  
*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 rdi](#)  
*rdi register*
- [l4\\_umword\\_t rsi](#)  
*rsi register*
- [l4\\_umword\\_t rbp](#)  
*rbp register*
- [l4\\_umword\\_t rbx](#)  
*rbx register*
- [l4\\_umword\\_t rdx](#)  
*rdx register*
- [l4\\_umword\\_t rcx](#)  
*rcx register*
- [l4\\_umword\\_t rax](#)  
*rax register*
- [l4\\_umword\\_t trapno](#)  
*trap number*
- [l4\\_umword\\_t cs](#)  
*dummy*
- [l4\\_umword\\_t ss](#)  
*ss register*
- [l4\\_umword\\_t es](#)  
*gs register*
- [l4\\_umword\\_t ds](#)  
*fs register*
- [l4\\_umword\\_t gs](#)  
*gs register*
- [l4\\_umword\\_t fs](#)  
*fs register*
- [l4\\_umword\\_t dummy1](#)  
*dummy*

### 15.226.1 Detailed Description

vCPU registers.

Definition at line 39 of file [\\_\\_vcpu-arch.h](#).

### 15.226.2 Field Documentation

### 15.226.2.1 ax

`l4_umword_t l4_vcpu_regs_t::ax`

rax register

eax register

Definition at line 68 of file [\\_\\_vcpu-arch.h](#).

### 15.226.2.2 bp

`l4_umword_t l4_vcpu_regs_t::bp`

rbp register

ebp register

Definition at line 63 of file [\\_\\_vcpu-arch.h](#).

### 15.226.2.3 bx

`l4_umword_t l4_vcpu_regs_t::bx`

rbx register

ebx register

Definition at line 65 of file [\\_\\_vcpu-arch.h](#).

### 15.226.2.4 cx

`l4_umword_t l4_vcpu_regs_t::cx`

rcx register

ecx register

Definition at line 67 of file [\\_\\_vcpu-arch.h](#).

#### 15.226.2.5 di

`l4_umword_t l4_vcpu_regs_t::di`

rdi register

edi register

Definition at line 61 of file [\\_\\_vcpu-arch.h](#).

#### 15.226.2.6 dx

`l4_umword_t l4_vcpu_regs_t::dx`

rdx register

edx register

Definition at line 66 of file [\\_\\_vcpu-arch.h](#).

#### 15.226.2.7 si

`l4_umword_t l4_vcpu_regs_t::si`

rsi register

esi register

Definition at line 62 of file [\\_\\_vcpu-arch.h](#).

The documentation for this struct was generated from the following file:

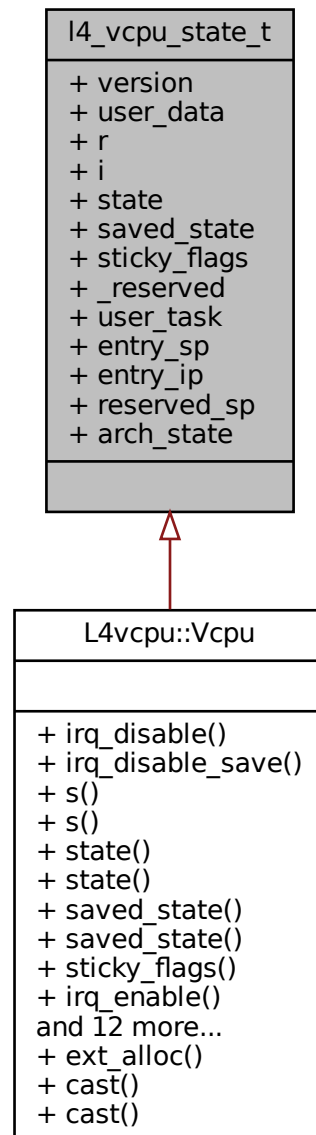
- [arm/l4/sys/\\_\\_vcpu-arch.h](#)

## 15.227 l4\_vcpu\_state\_t Struct Reference

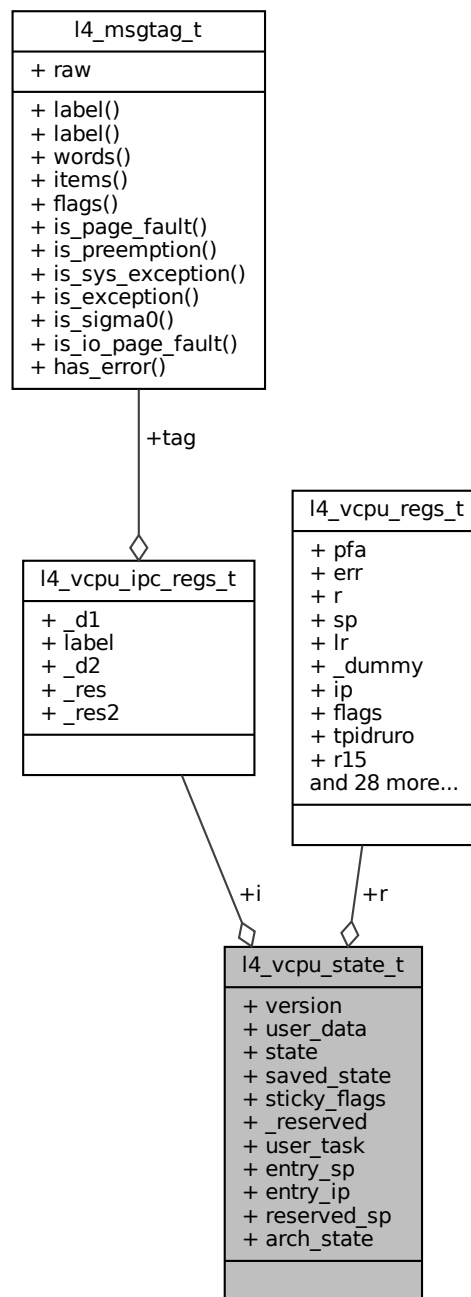
State of a vCPU.

```
#include <vcpu.h>
```

Inheritance diagram for l4\_vcpu\_state\_t:



Collaboration diagram for `l4_vcpu_state_t`:



## Data Fields

- [l4\\_vcpu\\_regs\\_t r](#)  
*Register state.*
- [l4\\_vcpu\\_ipc\\_regs\\_t i](#)  
*IPC state.*
- [l4\\_uint16\\_t state](#)



- Current vCPU state.*
  - [l4\\_uint16\\_t saved\\_state](#)
    - Saved vCPU state.*
  - [l4\\_uint16\\_t sticky\\_flags](#)
    - Pending 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.*

### 15.227.1 Detailed Description

State of a vCPU.

Definition at line 47 of file [vcpu.h](#).

The documentation for this struct was generated from the following file:

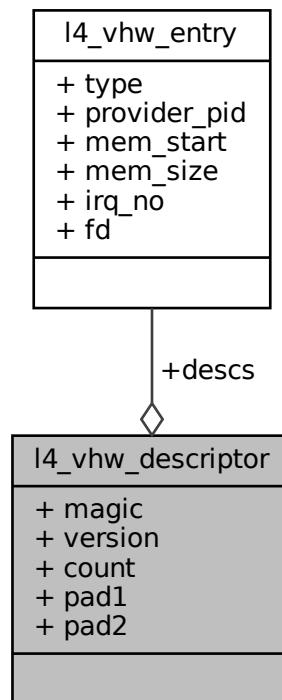
- [l4/sys/vcpu.h](#)

## 15.228 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.*

### 15.228.1 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:

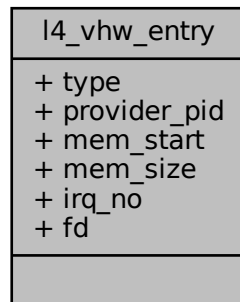
- [l4/sys/vhw.h](#)

## 15.229 l4\_vhw\_entry Struct Reference

Description of a device.

```
#include <vhw.h>
```

Collaboration diagram for l4\_vhw\_entry:



### 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.*

### 15.229.1 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](#)

## 15.230 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:

| l4_vm_svm_vmcb_control_area                                                                                                                                                                                                                                                       |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <div>+ intercept_rd_crX<br/>+ intercept_wr_crX<br/>+ intercept_rd_drX<br/>+ intercept_wr_drX<br/>+ intercept_exceptions<br/>+ intercept_instruction0<br/>+ intercept_instruction1<br/>+ _reserved0<br/>+ pause_filter_threshold<br/>+ pause_filter_count<br/>and 18 more...</div> |
|                                                                                                                                                                                                                                                                                   |

### 15.230.1 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:

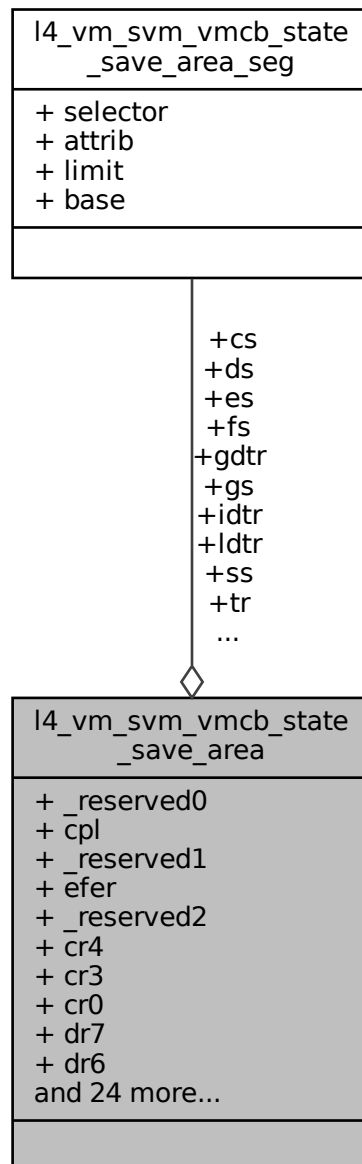
- [l4/sys/\\_\\_vm-svm.h](#)

## 15.231 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:



### 15.231.1 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

## 15.232 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:

| l4_vm_svm_vmcb_state<br>_save_area_seg      |
|---------------------------------------------|
| + selector<br>+ attrib<br>+ limit<br>+ base |
|                                             |

### 15.232.1 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:

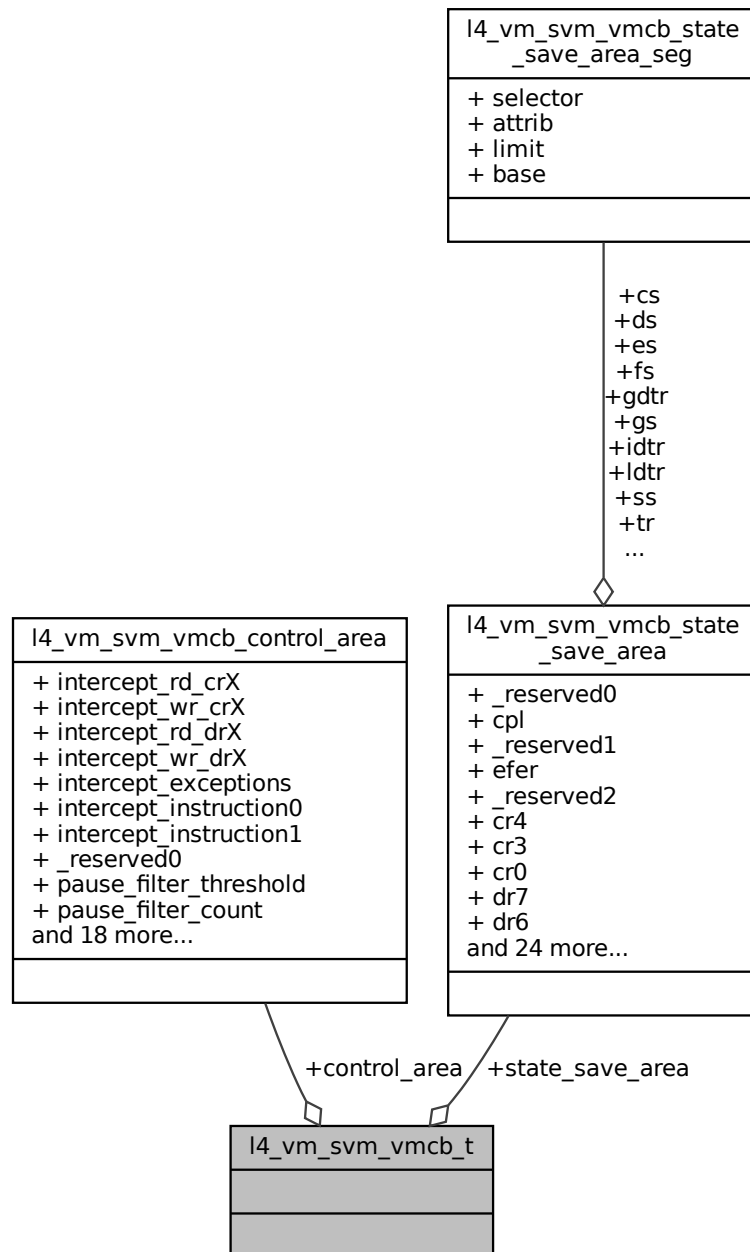
- l4/sys/\_\_vm-svm.h

## 15.233 l4\_vm\_svm\_vmcb\_t Struct Reference

Control structure for SVM VMs.

```
#include <__vm-svm.h>
```

Collaboration diagram for l4\_vm\_svm\_vmcb\_t:



### 15.233.1 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](#)

## 15.234 l4\_vm\_tz\_state Struct Reference

state structure for TrustZone VMs

```
#include <vm.h>
```

Collaboration diagram for l4\_vm\_tz\_state:

| l4_vm_tz_state                                                                                                                                                                                                             |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>+ r</li> <li>+ sp_usr</li> <li>+ lr_usr</li> <li>+ irq</li> <li>+ r_fiq</li> <li>+ fiq</li> <li>+ abt</li> <li>+ und</li> <li>+ svc</li> <li>+ pc</li> <li>and 8 more...</li> </ul> |
|                                                                                                                                                                                                                            |

### 15.234.1 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](#)

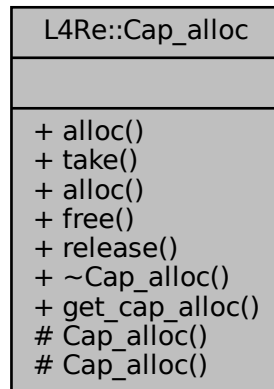


## 15.235 L4Re::Cap\_alloc Class Reference

Capability allocator interface.

Inherited by L4Re::Cap\_alloc\_t< ALLOC >.

Collaboration diagram for L4Re::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.*

#### 15.235.1 Detailed Description

Capability allocator interface.

Definition at line 41 of file [cap\\_alloc](#).

## 15.235.2 Member Function Documentation

### 15.235.2.1 `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:



### 15.235.2.2 `alloc()` [2/2]

```
virtual L4::Cap<void> L4Re::Cap_alloc::alloc () [pure virtual], [noexcept]
```

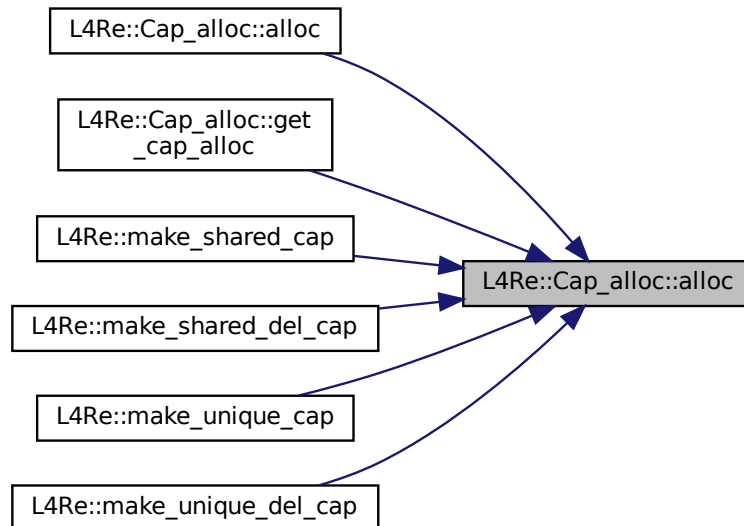
Allocate a capability.

## Returns

Capability of type void

Referenced by [alloc\(\)](#), [get\\_cap\\_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:



## 15.235.2.3 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]

```

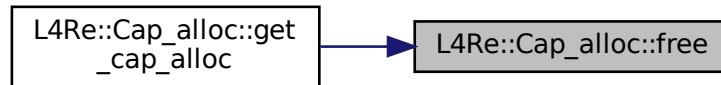
Free a capability.

## 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> . |

Referenced by [get\\_cap\\_alloc\(\)](#).

Here is the caller graph for this function:



#### 15.235.2.4 get\_cap\_alloc()

```

template<typename CAP_ALLOC >
static L4Re::Cap_alloc* L4Re::Cap_alloc::get_cap_alloc (
 CAP_ALLOC & ca) [inline], [static]

```

Construct an instance of a capability allocator.

##### Parameters

|           |                      |
|-----------|----------------------|
| <i>ca</i> | Capability allocator |
|-----------|----------------------|

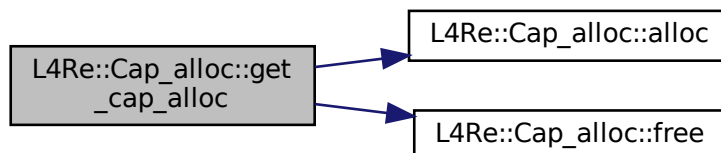
##### Returns

Instance of a capability allocator.

Definition at line 90 of file [cap\\_alloc](#).

References [alloc\(\)](#), [free\(\)](#), [L4\\_FP\\_ALL\\_SPACES](#), and [L4\\_INVALID\\_CAP](#).

Here is the call graph for this function:



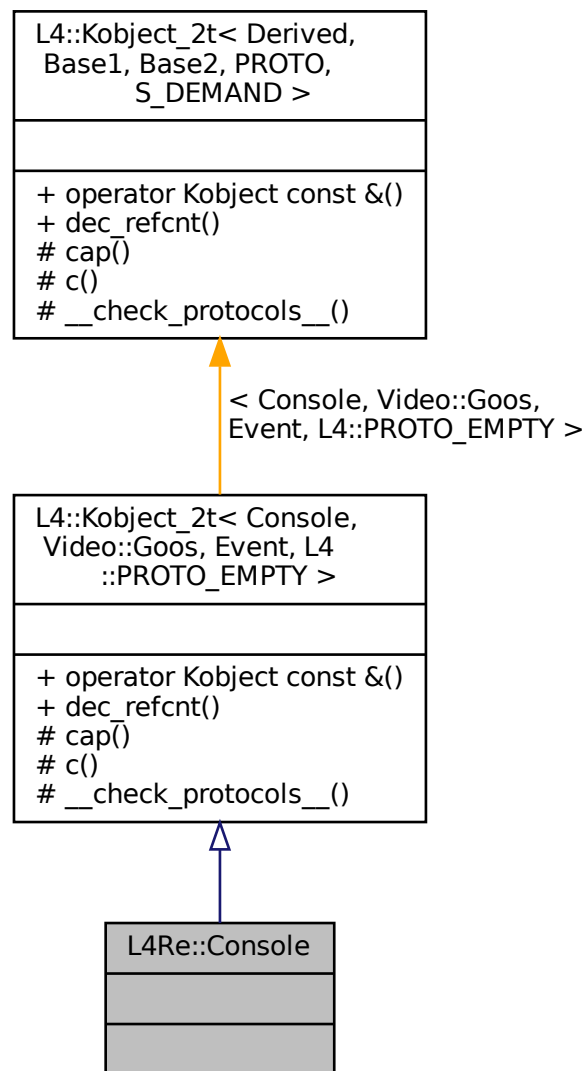
The documentation for this class was generated from the following file:

- [l4/re/cap\\_alloc](#)

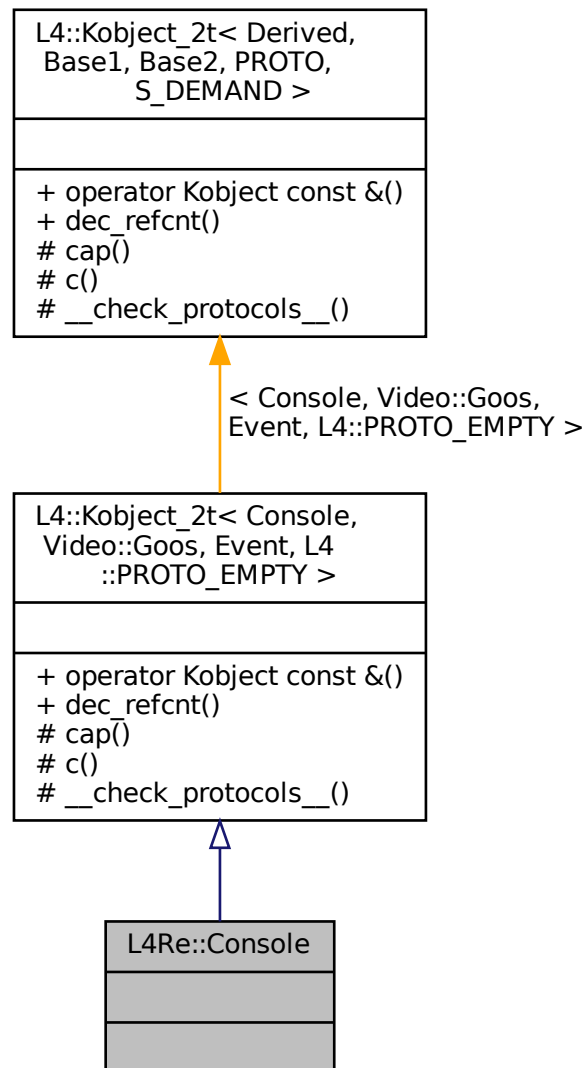
## 15.236 L4Re::Console Class Reference

[Console](#) class.

Inheritance diagram for L4Re::Console:



Collaboration diagram for L4Re::Console:



## Additional Inherited Members

### 15.236.1 Detailed Description

[Console](#) class.

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

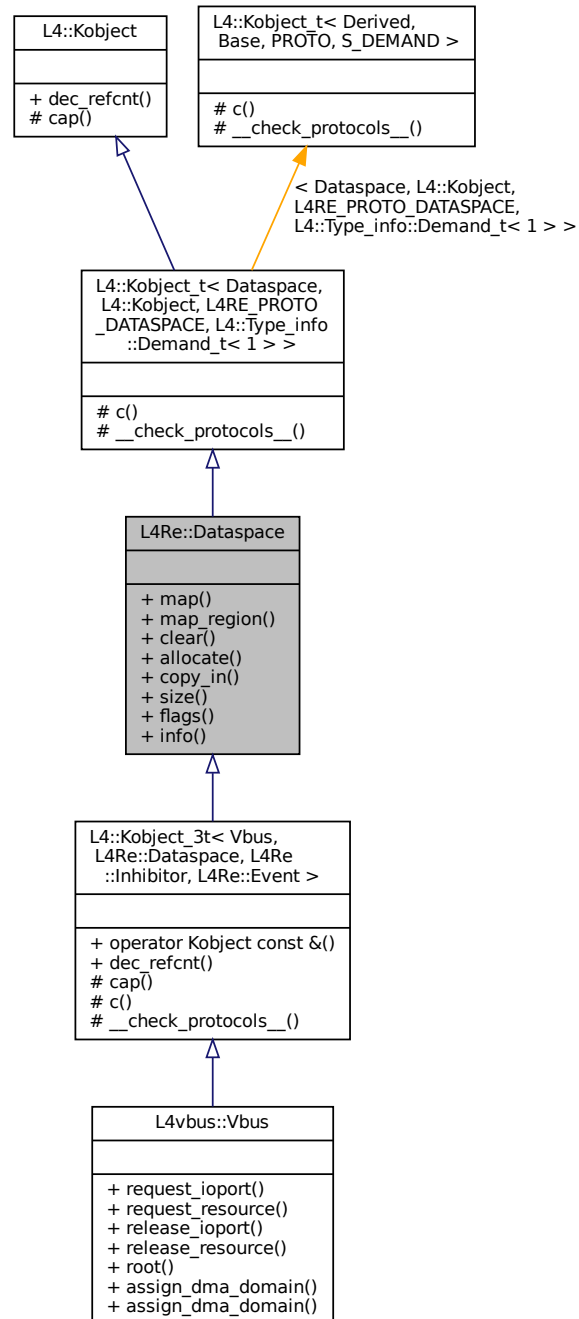
The documentation for this class was generated from the following file:

- [l4/re/console](#)

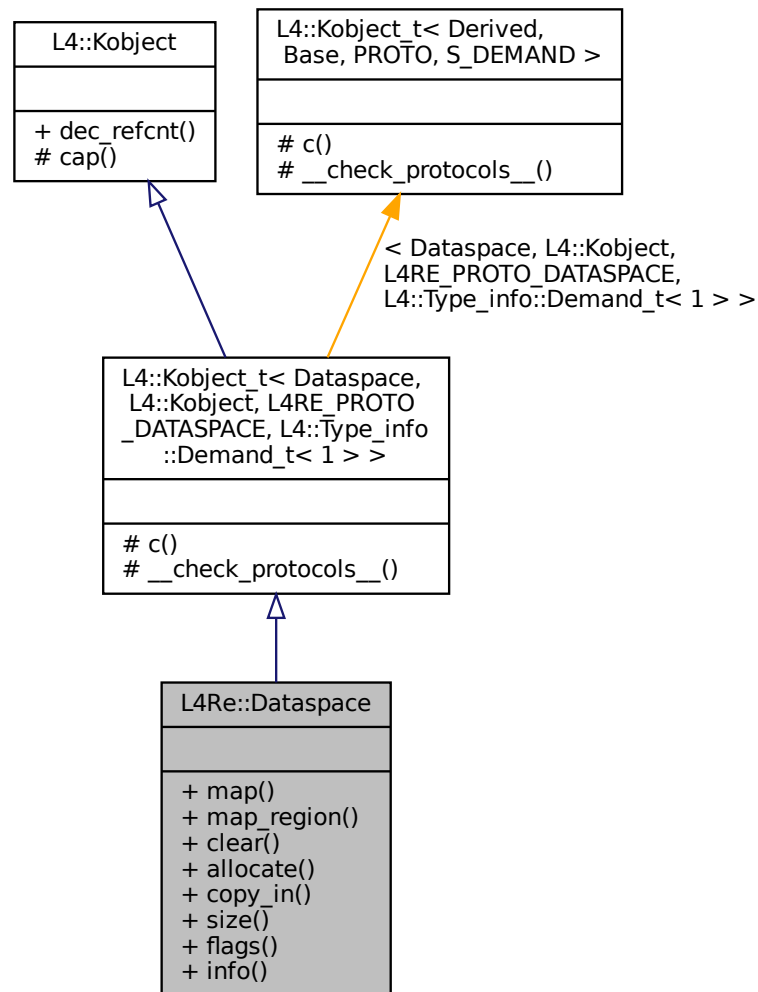
## 15.237 L4Re::Dataspace Class Reference

Interface for memory-like objects.

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.*

## Additional Inherited Members

### 15.237.1 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](#).

### 15.237.2 Member Function Documentation

#### 15.237.2.1 [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>&lt;0</i>      | IPC errors                                                                                                         |

On success, at least the given range is guaranteed to be allocated. The dataspace manager may also allocate more memory due to page granularity.

The memory is allocated with the same rights as the dataspace capability.

**15.237.2.2 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

|                    |                                                                                                                    |
|--------------------|--------------------------------------------------------------------------------------------------------------------|
| <i>&gt;=0</i>      | Success.                                                                                                           |
| <i>-L4_ERANGE</i>  | Given range is outside the dataspace. (A dataspace provider may also silently ignore areas outside the dataspace.) |
| <i>-L4_EACCESS</i> | <a href="#">Dataspace</a> is read-only.                                                                            |
| <i>&lt;0</i>       | IPC errors                                                                                                         |

Zeroes out the memory. Depending on the type of memory the memory could also be deallocated and replaced by a shared zero-page.

**15.237.2.3 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

|                    |                                     |
|--------------------|-------------------------------------|
| <i>L4_EOK</i>      | Success                             |
| <i>-L4_EACCESS</i> | Destination dataspace not writable. |
| <i>-L4_EINVAL</i>  | Invalid parameter supplied.         |
| <i>&lt;0</i>       | IPC errors                          |

The copy operation may use copy-on-write mechanisms. The operation may also fail if both dataspaces are not from the same dataspace manager or the dataspace managers do not cooperate.

**15.237.2.4 flags()**

```
Dataspace::Flags L4Re::Dataspace::flags () const [noexcept]
```

Get flags of the dataspace.

## Return values

|               |                        |
|---------------|------------------------|
| <i>&gt;=0</i> | Flags of the dataspace |
| <i>&lt;0</i>  | IPC errors             |

## See also

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

Definition at line 119 of file [dataspace\\_impl.h](#).

Referenced by [L4vbus::Vbus::assign\\_dma\\_domain\(\)](#).

Here is the caller graph for this function:



### 15.237.2.5 info()

```
long L4Re::Dataspace::info (
 Stats * stats)
```

Get information on the dataspace.

#### Parameters

|     |       |                       |
|-----|-------|-----------------------|
| out | stats | Dataspace information |
|-----|-------|-----------------------|

#### Return values

|    |            |
|----|------------|
| 0  | Success    |
| <0 | IPC errors |

### 15.237.2.6 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 <a href="#">L4Re::Dataspace::F::Flags</a> . |
| <i>local_addr</i> | Local address to map to.                                         |
| <i>min_addr</i>   | Defines start of receive window. (Rounded down to page size.)    |
| <i>max_addr</i>   | Defines end of receive window. (Rounded up to page size.)        |

#### 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](#).

### 15.237.2.7 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>    | <a href="#">Dataspace</a> flags, see <a href="#">L4Re::Dataspace::F::Flags</a> . |
| <i>min_addr</i> | (Inclusive) start of the receive area.                                           |
| <i>max_addr</i> | (Exclusive) end of receive area.                                                 |

#### 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>&lt;0</i>      | IPC errors                                                |

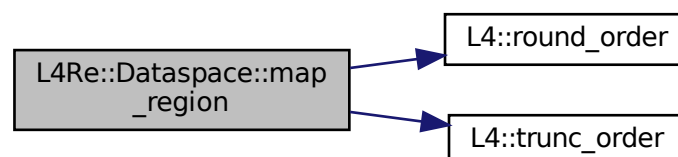
This is a convenience function which maps flex-pages consecutively into the given memory area in the local task. The area is expected to be filled completely. If the dataspace is not large enough to provide the mappings for the entire size of the area, then an error is returned. Mappings may or may not 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:



### 15.237.2.8 size()

```
Dataspace::Size L4Re::Dataspace::size () const [noexcept]
```

Get size of a dataspace.

#### Returns

Size of the dataspace in bytes.

#### Examples

[examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#).

Definition at line 109 of file [dataspace\\_impl.h](#).

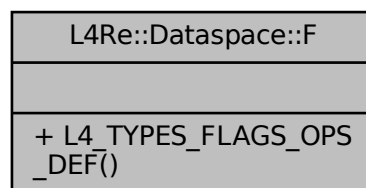
The documentation for this class was generated from the following files:

- [l4re/dataspace](#)
- [l4re/impl/dataspace\\_impl.h](#)

## 15.238 L4Re::Dataspace::F Struct Reference

[Dataspace](#) flags definitions.

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.*

### 15.238.1 Detailed Description

[Dataspace](#) flags definitions.

Definition at line 67 of file [dataspace](#).

### 15.238.2 Member Enumeration Documentation

#### 15.238.2.1 anonymous enum

anonymous enum

Enumerator

|               |                               |
|---------------|-------------------------------|
| Caching_shift | shift value for caching flags |
|---------------|-------------------------------|

Definition at line 69 of file [dataspace](#).

#### 15.238.2.2 Flags

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

Flags for map operations.

Enumerator

|              |                                              |
|--------------|----------------------------------------------|
| R            | Request read-only mapping.                   |
| Ro           | Request read-only mapping.                   |
| RW           | Request writable mapping (R + W)             |
| W            | Request write-only memory.                   |
| 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 77 of file [dataspace](#).

The documentation for this struct was generated from the following file:

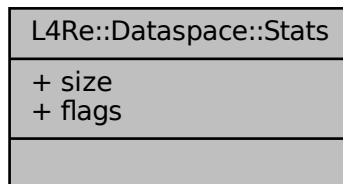
- [l4/re/dataspace](#)



## 15.239 L4Re::Dataspace::Stats Struct Reference

Information about the dataspace.

Collaboration diagram for L4Re::Dataspace::Stats:



### Data Fields

- Size [size](#)  
*size*
- Flags [flags](#)  
*flags*

### 15.239.1 Detailed Description

Information about the dataspace.

Definition at line [129](#) of file [dataspace](#).

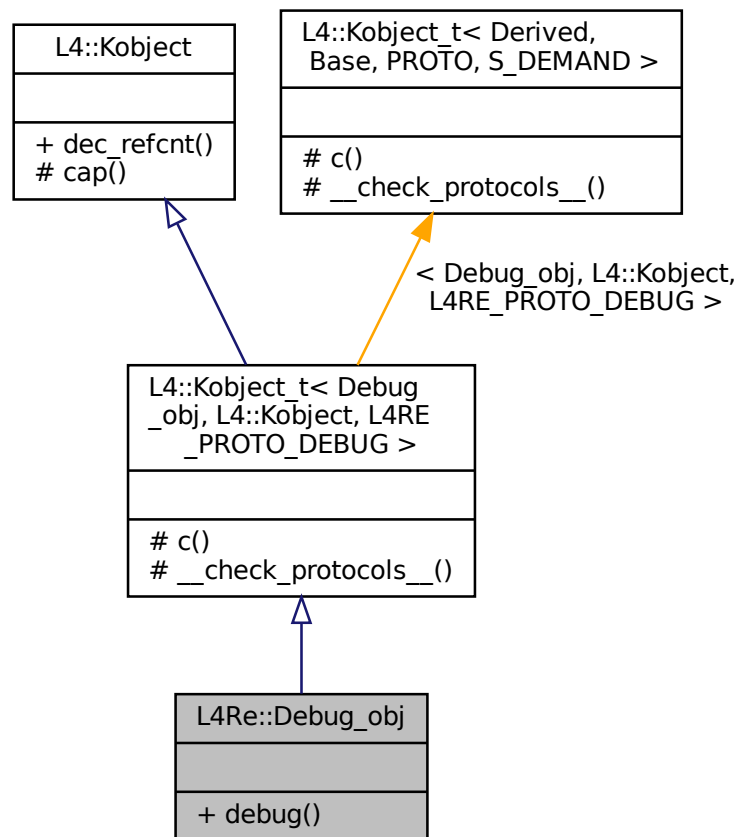
The documentation for this struct was generated from the following file:

- [l4/re/dataspace](#)

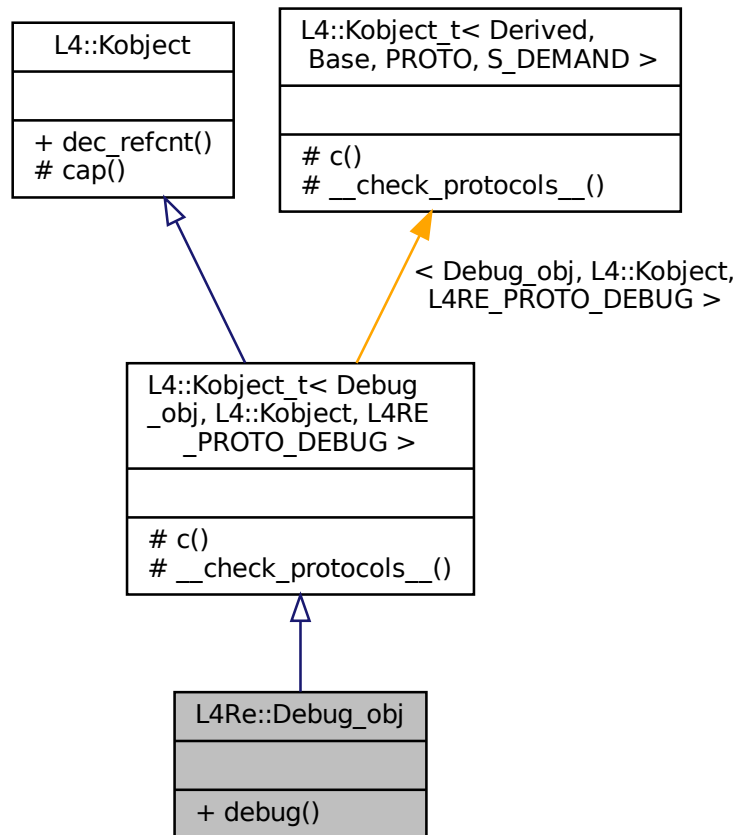
## 15.240 L4Re::Debug\_obj Class Reference

Debug interface.

Inheritance diagram for L4Re::Debug\_obj:



Collaboration diagram for L4Re::Debug\_obj:



## Public Member Functions

- long [debug](#) (unsigned long function)  
*Debug call.*

## Additional Inherited Members

### 15.240.1 Detailed Description

Debug interface.

See also

[Debugging API](#) .

Definition at line 51 of file [debug](#).

## 15.240.2 Member Function Documentation

### 15.240.2.1 debug()

```
long L4Re::Debug_obj::debug (
 unsigned long function)
```

Debug call.

#### Parameters

|                 |                   |
|-----------------|-------------------|
| <i>function</i> | Function to call. |
|-----------------|-------------------|

#### Returns

- L4\_EOK
- IPC errors

The documentation for this class was generated from the following file:

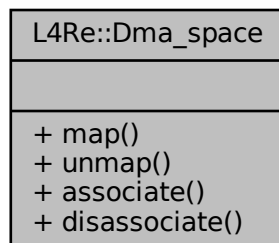
- [l4/re/debug](#)

## 15.241 L4Re::Dma\_space Class Reference

DMA Address 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 DMA task for a device to this [Dma\\_space](#).*
- long [disassociate](#) ()  
*Disassociate the DMA task from this [Dma\\_space](#).*

### 15.241.1 Detailed Description

DMA Address Space.

A [Dma\\_space](#) represents the DMA address space of a device. Whenever a device needs direct access to parts of an [L4Re::Dataspace](#), that part of the data space must be mapped to the DMA address space that is assigned to that device. After the DMA accesses to the memory are finished the memory must be unmapped from the device's DMA address space.

Mapping to a 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 57 of file [dma\\_space](#).

### 15.241.2 Member Typedef Documentation

### 15.241.2.1 Attributes

```
typedef L4::Types::Flags<Attribute> L4Re::Dma_space::Attributes
```

Attributes for DMA mappings.

See also

[Attribute](#)

Definition at line 102 of file [dma\\_space](#).

## 15.241.3 Member Enumeration Documentation

### 15.241.3.1 Attribute

```
enum L4Re::Dma_space::Attribute
```

Attributes used for the memory region during the transfer.

See also

[Attributes](#)

Enumerator

|         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| No_sync | Do not sync the memory hierarchy. When this flag is <i>not set</i> (default) the memory region shall be made coherent to the point-of-coherency of the device associated with this <a href="#">Dma_space</a> . When using this attribute the client is responsible for syncing the memory hierarchy for DMA. This can either be done using the cache API or by another <a href="#">map()</a> or <a href="#">unmap()</a> operation of the same part of the data space (without the <a href="#">No_sync</a> attribute). |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Definition at line 81 of file [dma\\_space](#).

### 15.241.3.2 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 69 of file [dma\\_space](#).

### 15.241.3.3 Space\_attrb

enum [L4Re::Dma\\_space::Space\\_attrb](#)

Attributes assigned to the DMA space when associated with a specific device.

See also

[Space\\_attrbs](#)

#### Enumerator

|            |                                                                                                                                                                               |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Coherent   | The device is connected coherently with the cache. This means that the <a href="#">map()</a> and <a href="#">unmap()</a> 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 109 of file [dma\\_space](#).

## 15.241.4 Member Function Documentation

### 15.241.4.1 associate()

```
long L4Re::Dma_space::associate (
 L4::Ipc::Opt< L4::Ipc::Cap< L4::Task > > dma_task,
 Space_attrbs attr)
```

Associate a DMA task for a device to this [Dma\\_space](#).

#### Precondition

requires capability rights: {RW}

#### Parameters

|    |                 |                                                                                                                                                                                                                                                                                |
|----|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in | <i>dma_task</i> | The DMA task used for the device that shall be associated with this DMA space. The <i>dma_task</i> might be an invalid capability when <a href="#">L4Re::Dma_space::Phys_space</a> is set in <i>attr</i> , in this case the CPUs physical memory is used as DMA address space. |
| in | <i>attr</i>     | Attributes for this DMA space. See <a href="#">L4Re::Dma_space::Space_attrb</a> .                                                                                                                                                                                              |

**15.241.4.2 disassociate()**

```
long L4Re::Dma_space::disassociate ()
```

Disassociate the DMA task from this [Dma\\_space](#).

**Precondition**

requires capability rights: {RW}

**15.241.4.3 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.

**Precondition**

requires capability rights: {R}

**Parameters**

|         |                 |                                                                                                                                                                                                                                                                                                                 |
|---------|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in      | <i>src</i>      | Source data space (that describes the memory). Caller needs write rights 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 <a href="#">map()</a> again with a new offset and the remaining size. |
| in      | <i>attrs</i>    | The attributes used for this DMA mapping (a combination of <a href="#">Dma_space::Attribute</a> values).                                                                                                                                                                                                        |
| in      | <i>dir</i>      | The direction of the DMA transfer issued with this mapping. The same value must later be passed to <a href="#">unmap()</a> .                                                                                                                                                                                    |
| out     | <i>dma_addr</i> | The DMA address to use for DMA with the associated device.                                                                                                                                                                                                                                                      |

**Returns**

0 in the case of success, a negative error code otherwise.



#### 15.241.4.4 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.

##### Precondition

requires capability rights: {R}

##### Parameters

|                 |                                                                  |
|-----------------|------------------------------------------------------------------|
| <i>dma_addr</i> | The DMA address (returned by <a href="#">Dma_space::map()</a> ). |
| <i>size</i>     | The size (bytes) of the memory region to unmap.                  |
| <i>attrs</i>    | The attributes for the unmap (currently none).                   |
| <i>dir</i>      | The direction of the finished DMA operation.                     |

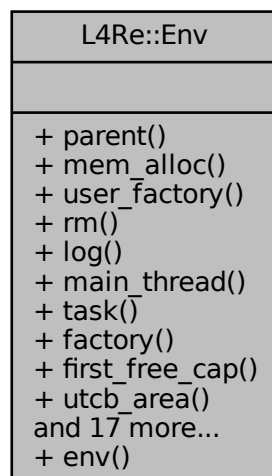
The documentation for this class was generated from the following file:

- [l4/re/dma\\_space](#)

## 15.242 L4Re::Env Class Reference

C++ interface of the initial environment that is provided to an [L4](#) task.

Collaboration diagram for L4Re::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](#) utcbs) 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.*

### 15.242.1 Detailed Description

C++ interface of the initial environment that is provided to an [L4](#) task.

The initial environment is provided to each [L4](#) task that is started by an [L4Re](#) conform loader, such as the Moe root task. The initial environment provides access to a set of initial capabilities and some additional information about the available resources, such as free UTCBs (see [Virtual Registers](#) ) and available entries in capability table (provided by the micro kernel).

Each of the initial capabilities is stored at a fixed index in the task's capability table and the [L4](#) runtime environment provides convenience functions to retrieve the capabilities. See the table below for an comprehensive overview.

| Name         | Object Type                   | Convenience Function                      |
|--------------|-------------------------------|-------------------------------------------|
| parent       | <a href="#">L4Re::Parent</a>  | <a href="#">L4Re::Env::parent()</a>       |
| user_factory | <a href="#">L4::Factory</a>   | <a href="#">L4Re::Env::user_factory()</a> |
| log          | <a href="#">L4Re::Log</a>     | <a href="#">L4Re::Env::log()</a>          |
| main_thread  | <a href="#">L4::Thread</a>    | <a href="#">L4Re::Env::main_thread()</a>  |
| rm           | <a href="#">L4Re::Rm</a>      | <a href="#">L4Re::Env::rm()</a>           |
| factory      | <a href="#">L4::Factory</a>   | <a href="#">L4Re::Env::factory()</a>      |
| task         | <a href="#">L4::Task</a>      | <a href="#">L4Re::Env::task()</a>         |
| scheduler    | <a href="#">L4::Scheduler</a> | <a href="#">L4Re::Env::scheduler()</a>    |

Additional information found in the initial environment is:

- First free entry in capability table
- The [UTCB](#) area (as flex page)
- First free UTCB (address in the UTCB area)

**Include File**

```
#include <l4/re/env>
```

For an explanation of the default task capabilities see [l4\\_default\\_caps\\_t](#).

For the C interface refer to [Initial Environment](#).

Definition at line 85 of file [env](#).

**15.242.2 Member Function Documentation****15.242.2.1 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>()

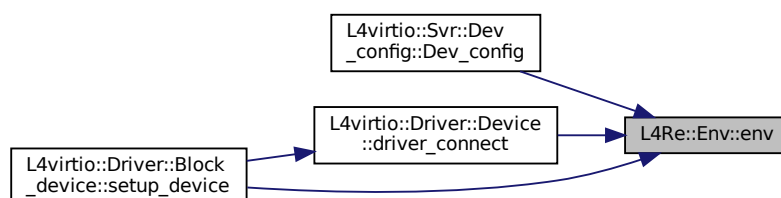
**Examples**

[examples/clntsrv/client.cc](#), [examples/libs/l4re/c++/mem\\_alloc/marm.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 103 of file [env](#).

Referenced by [L4virtio::Svr::Dev\\_config::Dev\\_config\(\)](#), [L4virtio::Driver::Device::driver\\_connect\(\)](#), and [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#).

Here is the caller graph for this function:



**15.242.2.2 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](#).

**15.242.2.3 factory()** [2/2]

```
void L4Re::Env::factory (
 L4::Cap< L4::Factory > const & c) [inline], [noexcept]
```

Set factory object-capability.

**Parameters**

|          |                           |
|----------|---------------------------|
| <b>c</b> | Factory object-capability |
|----------|---------------------------|

Definition at line 256 of file [env](#).

References [l4re\\_env\\_t::factory](#).

**15.242.2.4 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](#).

**15.242.2.5 first\_free\_cap()** [2/2]

```
void L4Re::Env::first_free_cap (
 l4_cap_idx_t c) [inline], [noexcept]
```

Set first available capability selector.

**Parameters**

|                |                                                         |
|----------------|---------------------------------------------------------|
| <code>c</code> | First capability selector available to the application. |
|----------------|---------------------------------------------------------|

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

References [l4re\\_env\\_t::first\\_free\\_cap](#).

**15.242.2.6 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](#).

**15.242.2.7 first\_free\_utcb()** [2/2]

```
void L4Re::Env::first_free_utcb (
 l4_addr_t u) [inline], [noexcept]
```

Set first free UTCB.

**Parameters**

|                |                                                  |
|----------------|--------------------------------------------------|
| <code>u</code> | First UTCB available for the application to use. |
|----------------|--------------------------------------------------|

Definition at line 274 of file [env](#).

References [l4re\\_env\\_t::first\\_free\\_utcb](#).

**15.242.2.8 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:

15.242.2.9 `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`.

**15.242.2.10 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](#).

**15.242.2.11 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](#).

**15.242.2.12 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](#).

**15.242.2.13 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](#).

**15.242.2.14 log()** [2/2]

```
void L4Re::Env::log (
 L4::Cap< Log > const & c) [inline], [noexcept]
```

Set log object-capability.

## Parameters

|          |                                       |
|----------|---------------------------------------|
| <i>c</i> | <a href="#">Log</a> object-capability |
|----------|---------------------------------------|

Definition at line 244 of file [env](#).

References [l4re\\_env\\_t::log](#).

**15.242.2.15 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](#).

**15.242.2.16 main\_thread() [2/2]**

```
void L4Re::Env::main_thread (
 L4::Cap< L4::Thread > const & c) [inline], [noexcept]
```

Set object-capability of first user thread.

**Parameters**

|          |                                  |
|----------|----------------------------------|
| <b>c</b> | First thread's object-capability |
|----------|----------------------------------|

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

References [l4re\\_env\\_t::main\\_thread](#).

**15.242.2.17 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](#).

**15.242.2.18 mem\_alloc() [2/2]**

```
void L4Re::Env::mem_alloc (
 L4::Cap< Mem_alloc > const & c) [inline], [noexcept]
```

Set memory allocator object-capability.

## Parameters

|                |                                    |
|----------------|------------------------------------|
| <code>c</code> | Memory allocator object-capability |
|----------------|------------------------------------|

Definition at line 232 of file [env](#).

References [l4re\\_env\\_t::mem\\_alloc](#).

**15.242.2.19 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](#).

**15.242.2.20 parent()** [2/2]

```
void L4Re::Env::parent (
 L4::Cap< Parent > const & c) [inline], [noexcept]
```

Set parent object-capability.

## Parameters

|                |                                          |
|----------------|------------------------------------------|
| <code>c</code> | <a href="#">Parent</a> object-capability |
|----------------|------------------------------------------|

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

References [l4re\\_env\\_t::parent](#).

**15.242.2.21 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](#).

**15.242.2.22 rm() [2/2]**

```
void L4Re::Env::rm (
 L4::Cap< Rm > const & c) [inline], [noexcept]
```

Set region map object-capability.

**Parameters**

|   |                              |
|---|------------------------------|
| c | Region map object-capability |
|---|------------------------------|

Definition at line 238 of file [env](#).

References [l4re\\_env\\_t::rm](#).

**15.242.2.23 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](#).

**15.242.2.24 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](#).

#### 15.242.2.25 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](#).

#### 15.242.2.26 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](#).

#### 15.242.2.27 utcb\_area() [2/2]

```
void L4Re::Env::utcb_area (
 l4_fpage_t utcbs) [inline], [noexcept]
```

Set UTCB area of the task.

**Parameters**

|               |           |
|---------------|-----------|
| <i>utcb</i> s | UTCB area |
|---------------|-----------|

Definition at line 268 of file [env](#).

References [l4re\\_env\\_t::utcb\\_area](#).

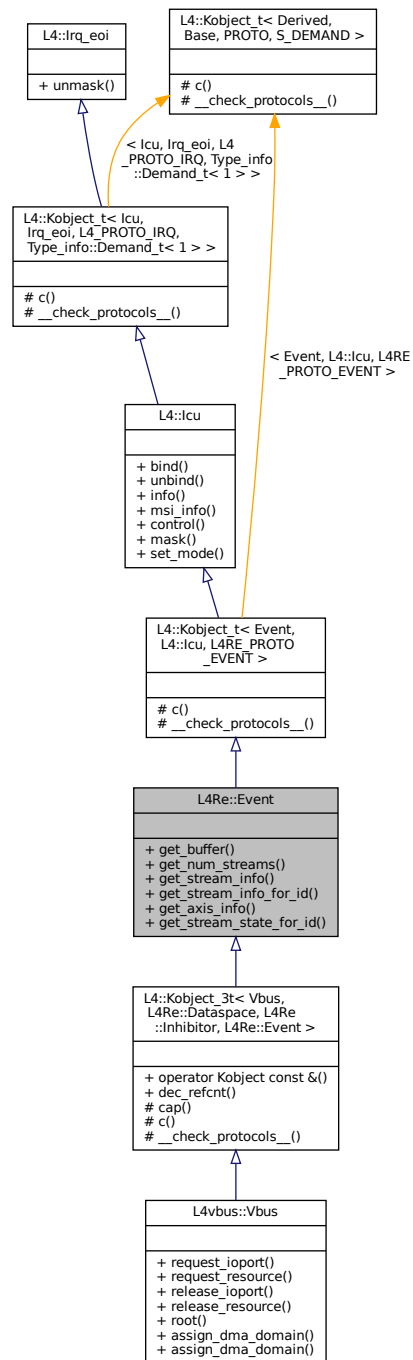
The documentation for this class was generated from the following file:

- [l4/re/env](#)

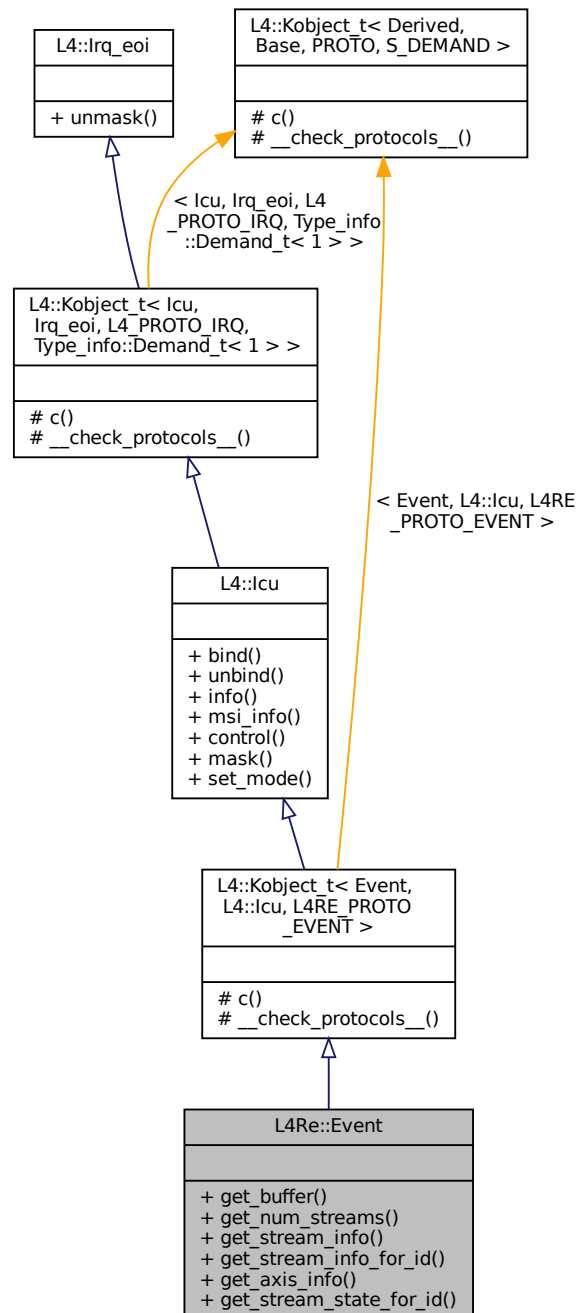
## 15.243 L4Re::Event Class Reference

[Event](#) class.

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.



## Additional Inherited Members

### 15.243.1 Detailed Description

[Event](#) class.

See also

[L4Re Event API](#)

Definition at line 144 of file [event](#).

### 15.243.2 Member Function Documentation

#### 15.243.2.1 get\_buffer()

```
long L4Re::Event::get_buffer (
 L4::Ipc::Out< L4::Cap< Dataspace > > ds)
```

Get event signal buffer.

Return values

|           |                               |
|-----------|-------------------------------|
| <i>ds</i> | <a href="#">Event</a> buffer. |
|-----------|-------------------------------|

Returns

0 on success, negative error code otherwise.

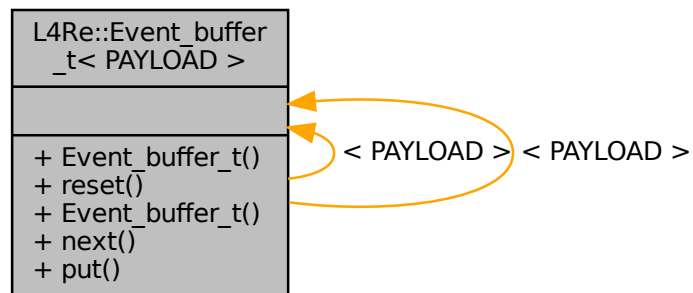
The documentation for this class was generated from the following file:

- [l4/re/event](#)

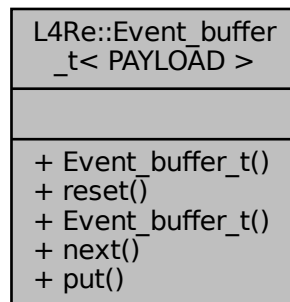
## 15.244 L4Re::Event\_buffer\_t< PAYLOAD > Class Template Reference

[Event](#) buffer class.

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.*

### 15.244.1 Detailed Description

```
template<typename PAYLOAD = Default_event_payload>
class L4Re::Event_buffer_t< PAYLOAD >
```

[Event](#) buffer class.

Definition at line 198 of file [event](#).

### 15.244.2 Constructor & Destructor Documentation

#### 15.244.2.1 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 245 of file [event](#).

### 15.244.3 Member Function Documentation

#### 15.244.3.1 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 255 of file [event](#).

References [L4Re::Event\\_buffer\\_t< PAYLOAD >::Event::time](#).

### 15.244.3.2 put()

```
template<typename PAYLOAD = Default_event_payload>
bool L4Re::Event_buffer_t< PAYLOAD >::put (
 Event const & ev) [inline], [noexcept]
```

Put event into buffer at current position.

#### Parameters

|           |                                               |
|-----------|-----------------------------------------------|
| <i>ev</i> | <a href="#">Event</a> to put into the buffer. |
|-----------|-----------------------------------------------|

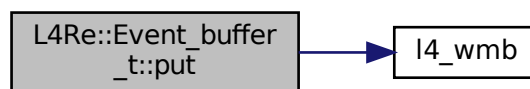
#### Returns

false if buffer is full and entry could not be added.

Definition at line 272 of file [event](#).

References [l4\\_wmb\(\)](#), and [L4Re::Event\\_buffer\\_t< PAYLOAD >::Event::time](#).

Here is the call graph for this function:



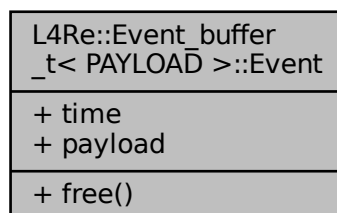
The documentation for this class was generated from the following file:

- `l4/re/event`

## 15.245 L4Re::Event\_buffer\_t< PAYLOAD >::Event Struct Reference

[Event](#) structure used in buffer.

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.*

### 15.245.1 Detailed Description

```
template<typename PAYLOAD = Default_event_payload>
struct L4Re::Event_buffer_t< PAYLOAD >::Event
```

[Event](#) structure used in buffer.

Definition at line [205](#) of file [event](#).

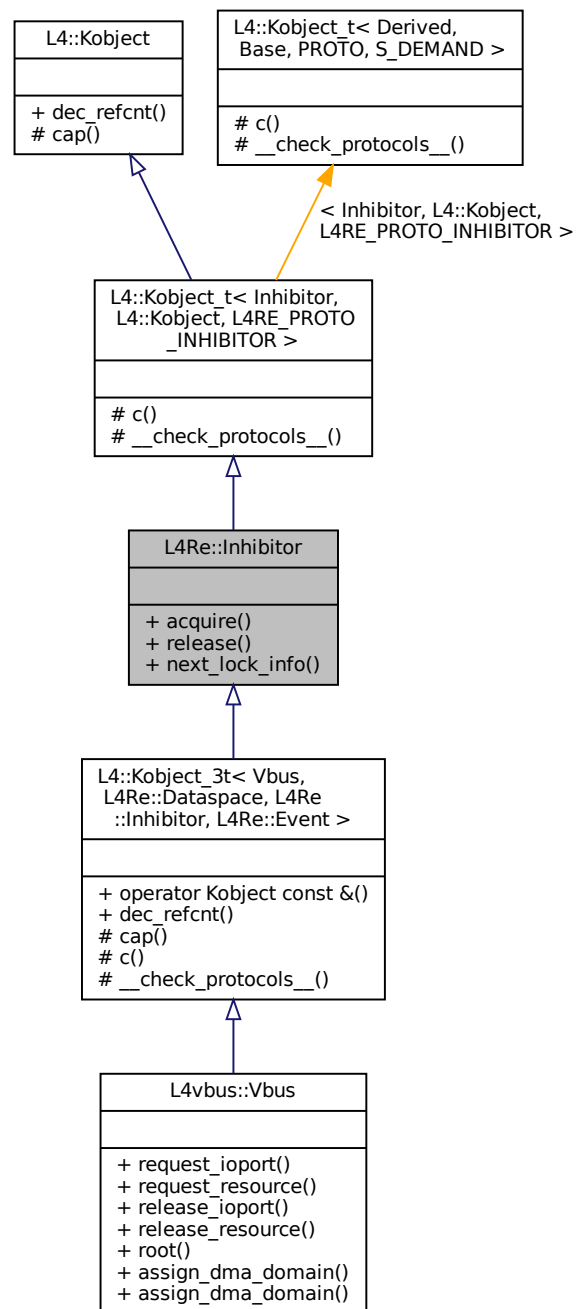
The documentation for this struct was generated from the following file:

- [l4/re/event](#)

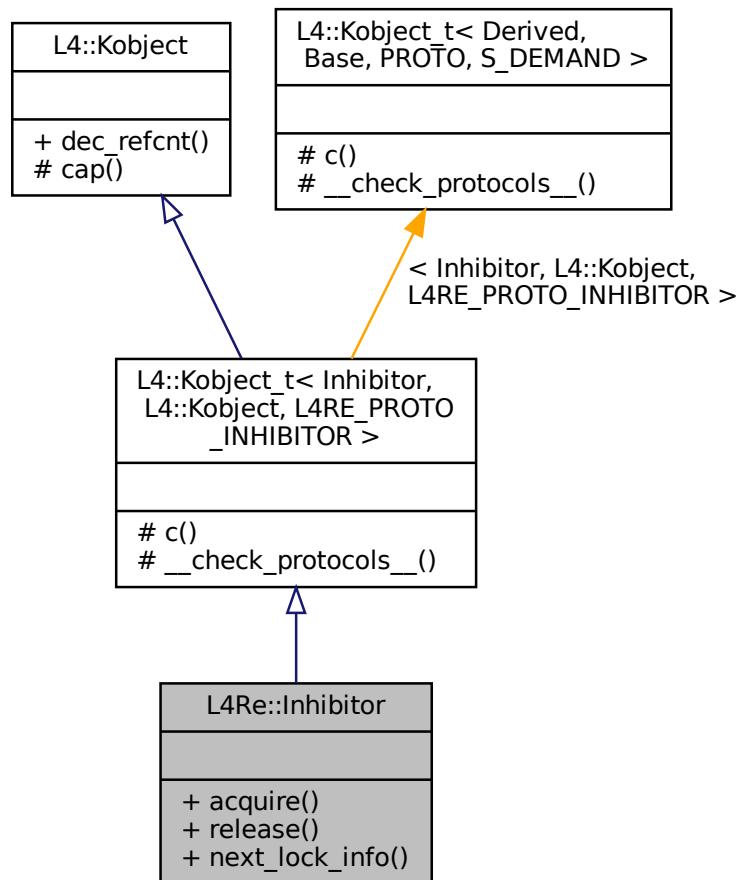
## 15.246 L4Re::Inhibitor Class Reference

Set of inhibitor locks, which inhibit specific actions when held.

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.*

## Additional Inherited Members

### 15.246.1 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](#).

### 15.246.2 Member Enumeration Documentation

#### 15.246.2.1 anonymous enum

anonymous enum

Enumerator

|          |                                      |
|----------|--------------------------------------|
| Name_max | The maximum length of a lock's name. |
|----------|--------------------------------------|

Definition at line 44 of file [inhibitor](#).

### 15.246.3 Member Function Documentation

#### 15.246.3.1 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

|                   |                                         |
|-------------------|-----------------------------------------|
| <i>0</i>          | Success                                 |
| <i>-L4_ENODEV</i> | The specified <i>id</i> does not exist. |

## 15.246.3.2 next\_lock\_info()

```
long L4Re::Inhibitor::next_lock_info (
 char * name,
 unsigned len,
 l4_mword_t current_id = -1,
 l4_utcb_t * utcb = l4_utcb()) [inline]
```

Get information for the next available inhibitor lock.

## Parameters

|                   |                                                                               |
|-------------------|-------------------------------------------------------------------------------|
| <i>name</i>       | A pointer to a buffer for the name of the lock.                               |
| <i>len</i>        | The length of the available buffer (usually <a href="#">Name_max</a> is used) |
| <i>current_id</i> | The ID of the last available lock, use -1 to get the first lock.              |
| <i>utcb</i>       | The UTCB to use for the message.                                              |

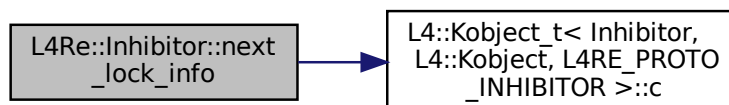
## Return values

|                   |                                                                                                                            |
|-------------------|----------------------------------------------------------------------------------------------------------------------------|
| <i>&gt;0</i>      | 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). |
| <i>-L4_ENODEV</i> | 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:



### 15.246.3.3 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. |

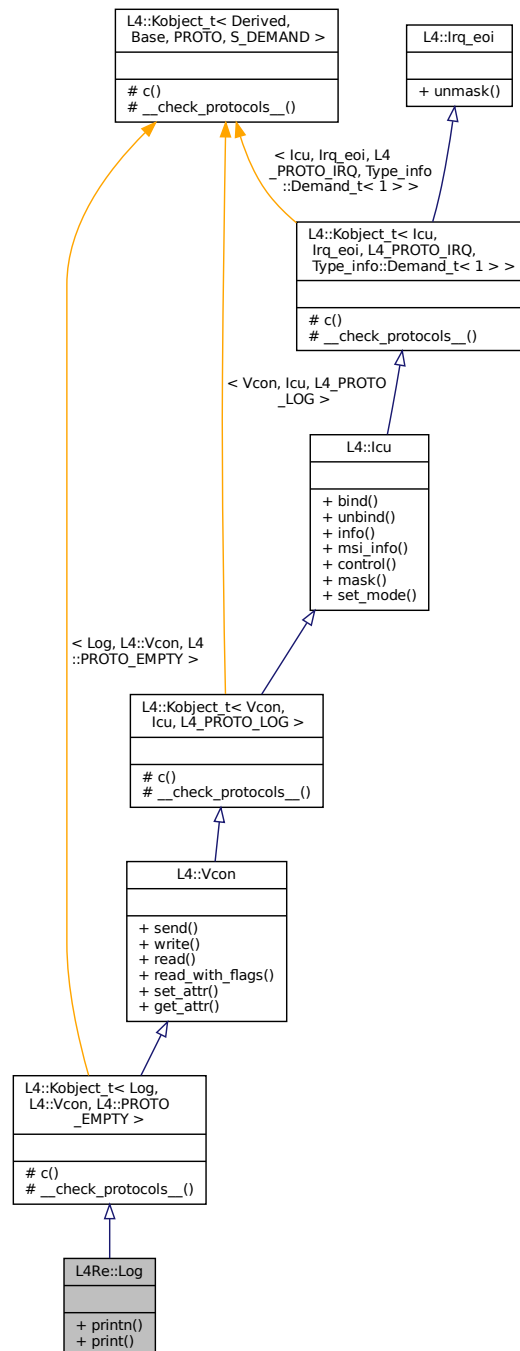
The documentation for this class was generated from the following file:

- l4/re/inhibitor

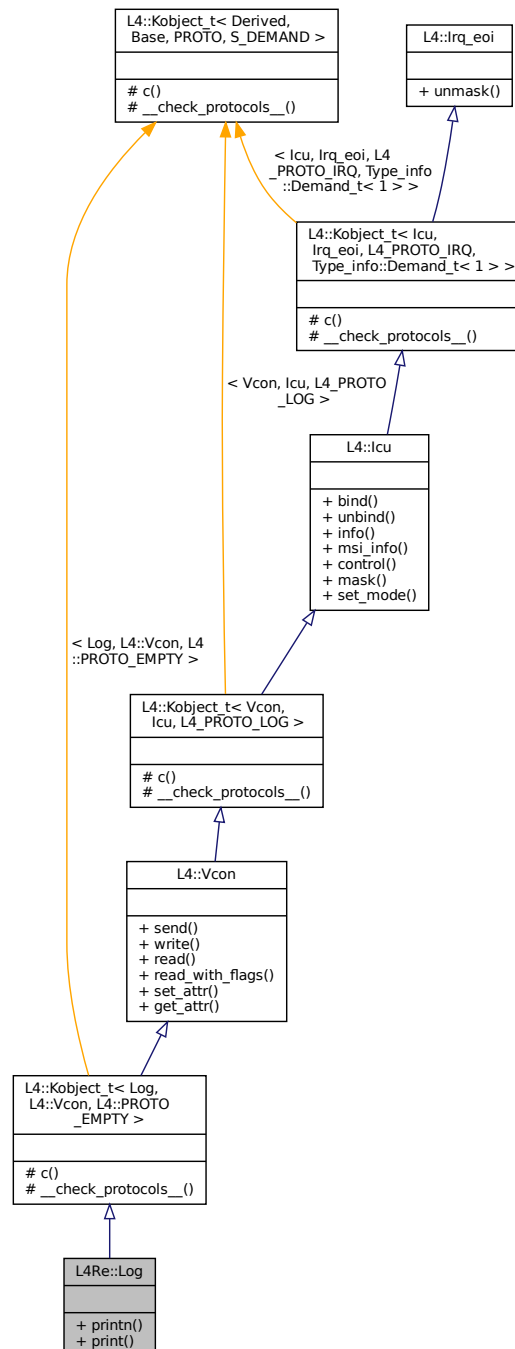
## 15.247 L4Re::Log Class Reference

[Log](#) interface class.

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.

## Additional Inherited Members

### 15.247.1 Detailed Description

[Log](#) interface class.

Definition at line 44 of file [log](#).

### 15.247.2 Member Function Documentation

#### 15.247.2.1 print()

```
void L4Re::Log::print (
 char const * string) const [noexcept]
```

Print NULL-terminated string.

##### Parameters

|               |                 |
|---------------|-----------------|
| <i>string</i> | string to print |
|---------------|-----------------|

#### 15.247.2.2 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 |

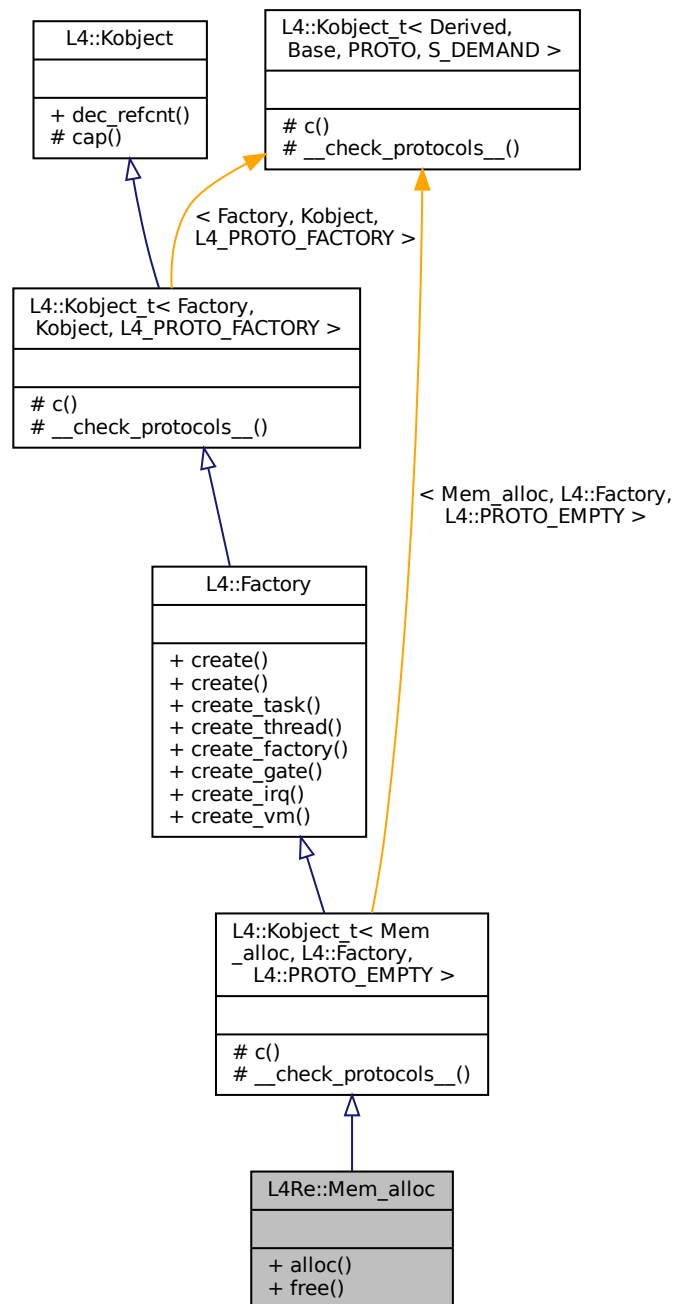
The documentation for this class was generated from the following file:

- [l4/re/log](#)

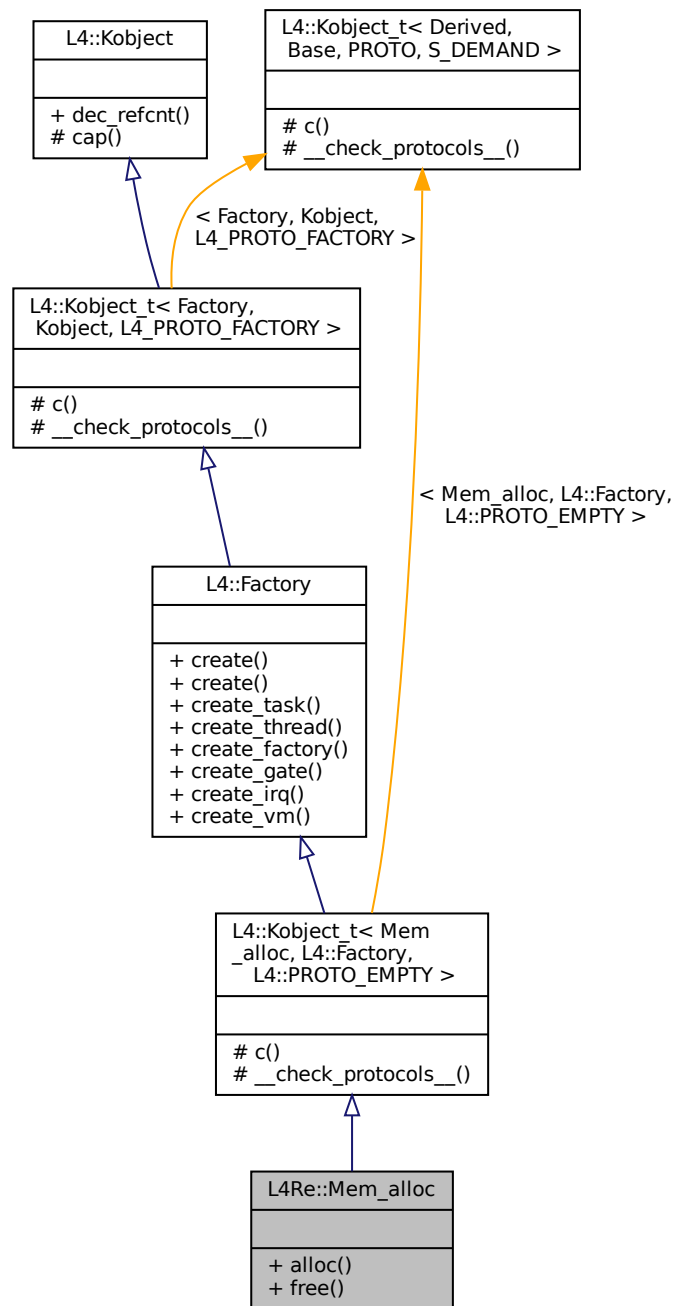
## 15.248 L4Re::Mem\_alloc Class Reference

Memory allocation interface.

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.*
- long `free` (`L4::Cap< Dataspace >` mem) const noexcept  
*Free dataspace.*

## Additional Inherited Members

### 15.248.1 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`.

### 15.248.2 Member Enumeration Documentation

#### 15.248.2.1 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 <code>L4Re::Dma_space</code> instead. |
| Super_pages | Allocate super pages.                                 |

Definition at line 71 of file `mem_alloc`.



### 15.248.3 Member Function Documentation

#### 15.248.3.1 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 <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. |
| out | <i>mem</i>   | Capability slot where the capability to the dataspace is received.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|     | <i>flags</i> | Special dataspace properties, see <a href="#">Mem_alloc_flags</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|     | <i>align</i> | Log2 alignment of dataspace if supported by allocator, will be at least <code>L4_PAGESHIFT</code> , with <code>Super_pages</code> flag set at least <code>L4_SUPERPAGESHIFT</code>                                                                                                                                                                                                                                                                                                                                                                                                  |

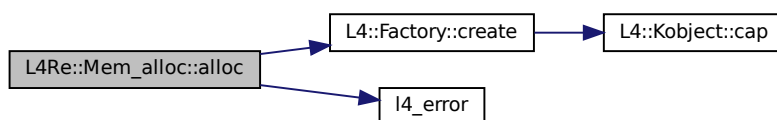
##### Return values

|            |                              |
|------------|------------------------------|
| 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::Factory::create\(\)](#), and [l4\\_error\(\)](#).

Here is the call graph for this function:



**15.248.3.2 free()**

```
long L4Re::Mem_alloc::free (
 L4::Cap< Dataspace > mem) const [noexcept]
```

Free dataspace.

**Parameters**

|            |                           |
|------------|---------------------------|
| <i>mem</i> | Dataspace to be released. |
|------------|---------------------------|

**Return values**

|   |  |
|---|--|
| 0 |  |
|---|--|

**Deprecated** This function is an empty stub which remains here for backward compatibility only. Use `L4::Task↔::unmap(mem, L4_FP_DELETE_OBJ)` or other similar means to remove a dataspace.

Definition at line 47 of file [mem\\_alloc\\_impl.h](#).

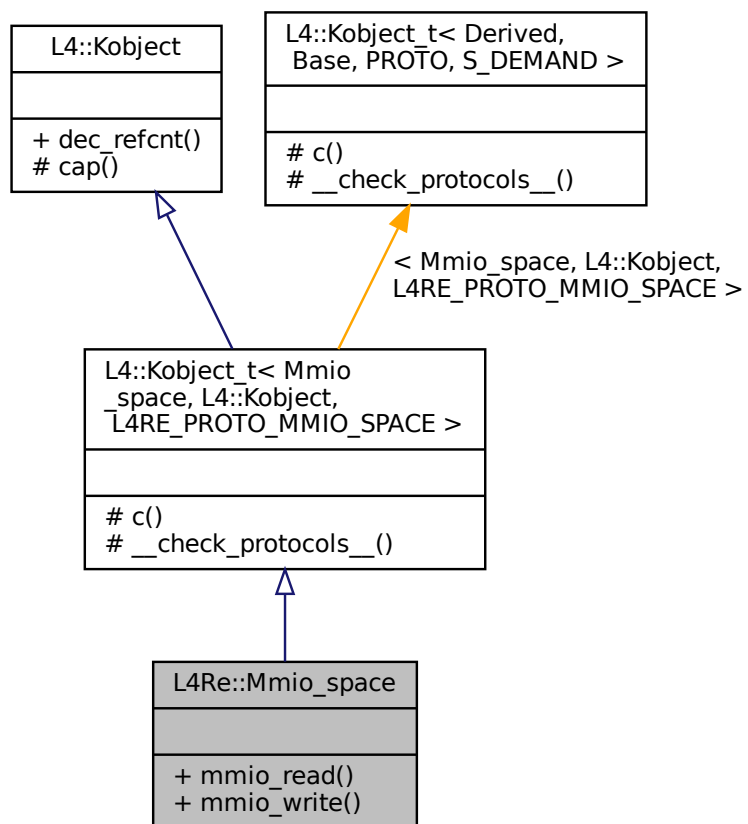
The documentation for this class was generated from the following files:

- [l4/re/mem\\_alloc](#)
- [l4/re/impl/mem\\_alloc\\_impl.h](#)

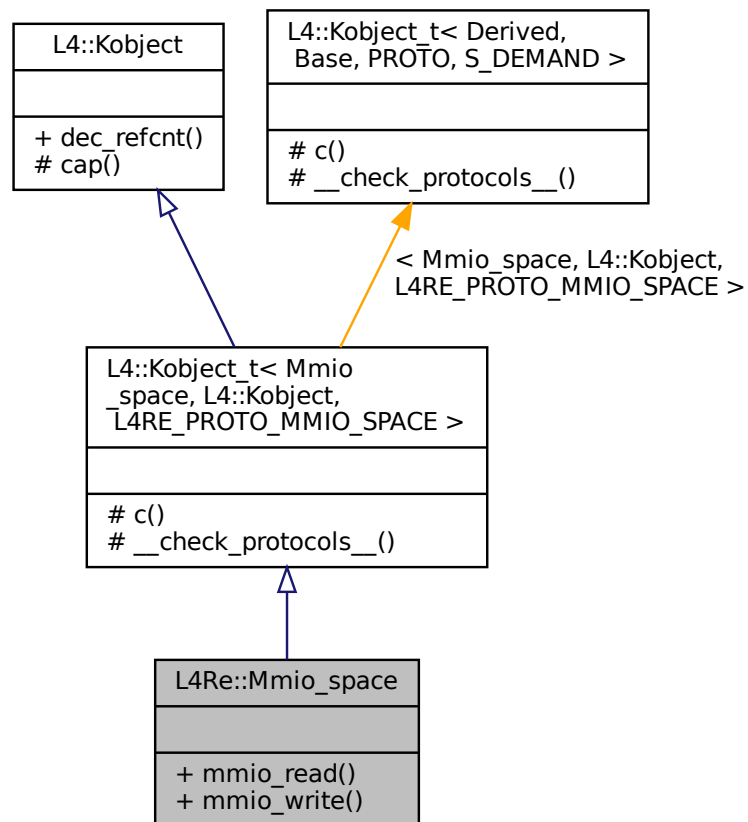
**15.249 L4Re::Mmio\_space Struct Reference**

Interface for memory-like address space accessible via IPC.

Inheritance diagram for L4Re::Mmio\_space:



Collaboration diagram for L4Re::Mmio\_space:



## Public Types

- enum `Access_width` { `Wd_8bit` = 0 , `Wd_16bit` = 1 , `Wd_32bit` = 2 , `Wd_64bit` = 3 }
- Actual size of the value to read or write.*
- typedef `l4_uint64_t` `Addr`
- Device address.*

## Public Member Functions

- long `mmio_read` (`Addr` addr, char width, `l4_uint64_t` \*value)
- Read a value from the given address.*
- long `mmio_write` (`Addr` addr, char width, `l4_uint64_t` value)
- Write a value to the given address.*

## Additional Inherited Members

### 15.249.1 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 42 of file [mmio\\_space](#).

### 15.249.2 Member Enumeration Documentation

#### 15.249.2.1 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 46 of file [mmio\\_space](#).

### 15.249.3 Member Function Documentation

### 15.249.3.1 mmio\_read()

```
long L4Re::Mmio_space::mmio_read (
 Addr addr,
 char width,
 l4_uint64_t * value)
```

Read a value from the given address.

#### Parameters

|     |              |                                                                                                |
|-----|--------------|------------------------------------------------------------------------------------------------|
|     | <i>addr</i>  | Device virtual address to read from. The address must be aligned relative to the access width. |
|     | <i>width</i> | Access width of value to be read, see <a href="#">Access_width</a> .                           |
| out | <i>value</i> | Return value. If width is smaller than 64 bit, the upper bits are guaranteed to be 0.          |

#### Return values

|                        |                                                                    |
|------------------------|--------------------------------------------------------------------|
| <a href="#">L4_EOK</a> | Success.                                                           |
| -L4_EPERM              | Insufficient read rights.                                          |
| -L4_EINVAL             | Address does not exist or cannot be accessed with the given width. |

### 15.249.3.2 mmio\_write()

```
long L4Re::Mmio_space::mmio_write (
 Addr addr,
 char width,
 l4_uint64_t value)
```

Write a value to the given address.

#### Parameters

|              |                                                                                               |
|--------------|-----------------------------------------------------------------------------------------------|
| <i>addr</i>  | Device virtual address to write to. The address must be aligned relative to the access width. |
| <i>width</i> | Access width of value to write, see <a href="#">Access_width</a> .                            |
| <i>value</i> | Value to write. If width is smaller than 64 bit, the upper bits are ignored.                  |

#### Return values

|                        |                                                                    |
|------------------------|--------------------------------------------------------------------|
| <a href="#">L4_EOK</a> | Success.                                                           |
| -L4_EPERM              | Insufficient write rights.                                         |
| -L4_EINVAL             | Address does not exist or cannot be accessed with the given width. |

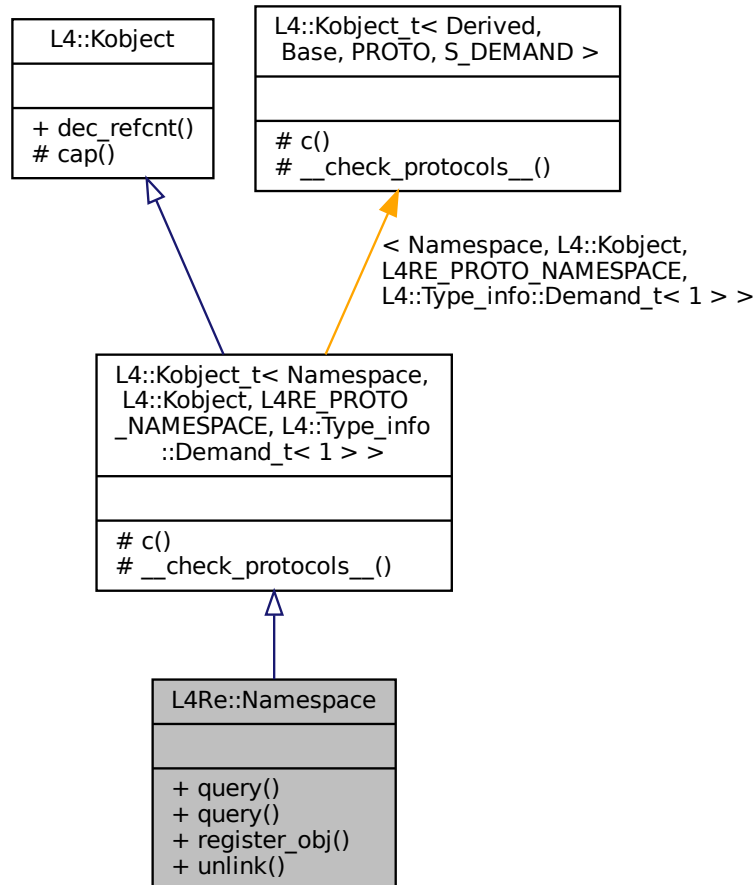
The documentation for this struct was generated from the following file:

- l4/re/mmio\_space

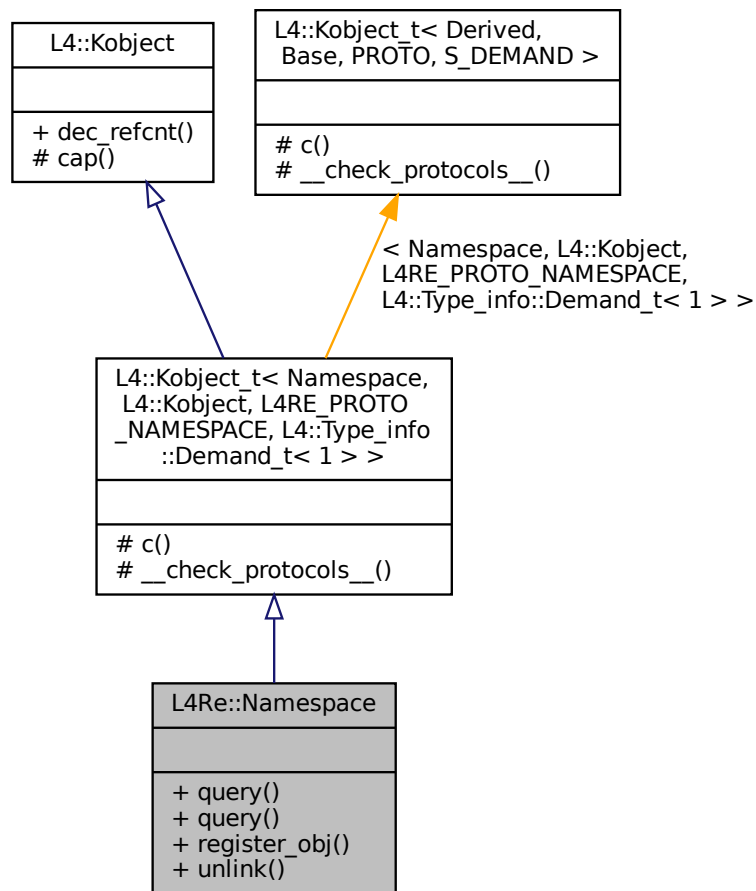
## 15.250 L4Re::Namespace Class Reference

Name-space interface.

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.*

## 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.*

## Additional Inherited Members

### 15.250.1 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](#).

### 15.250.2 Member Enumeration Documentation

#### 15.250.2.1 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](#).

#### 15.250.2.2 Register\_flags

```
enum L4Re::Namespace::Register_flags
```

Flags for registering name spaces.

## Enumerator

|           |                                        |
|-----------|----------------------------------------|
| Ro        | Read-only.                             |
| 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](#).

### 15.250.3 Member Function Documentation

#### 15.250.3.1 query() [1/2]

```
long L4Re::Namespace::query (
 char const * name,
 L4::Cap< void > const & cap,
 int timeout = To_default,
 l4_umword_t * local_id = 0,
 bool iterate = true) const [noexcept]
```

Query the name space for a named object.

## Parameters

|     |                 |                                                                                                                                                                                                                  |
|-----|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in  | <i>name</i>     | String to query (without any leading slashes).                                                                                                                                                                   |
| out | <i>cap</i>      | Capability slot where the received capability will be put.                                                                                                                                                       |
| in  | <i>timeout</i>  | Timeout of query in milliseconds. The client will only wait if a name has already been registered with the server but no object has yet been attached.                                                           |
| out | <i>local_id</i> | If given, <a href="#">L4_RCV_ITEM_LOCAL_ID</a> will be set for the IPC from the name space, so that if the capability that was received is a local item, the capability ID will be returned with this parameter. |
| in  | <i>iterate</i>  | If true, the client will try to resolve names by iteratively calling the name spaces until the name is fully resolved.                                                                                           |

## Return values

|            |                                                                                     |
|------------|-------------------------------------------------------------------------------------|
| 0          | Name could be fully resolved.                                                       |
| >0         | Name could only be partly resolved. The number of remaining characters is returned. |
| -L4_ENOENT | Entry could not be found.                                                           |
| -L4_EAGAIN | Entry exists but no object is yet attached. Try again later.                        |
| <0         | IPC errors, see <a href="#">l4_error_code_t</a> .                                   |

Definition at line 118 of file [namespace\\_impl.h](#).

### 15.250.3.2 query() [2/2]

```
long L4Re::Namespace::query (
 char const * name,
 unsigned len,
 L4::Cap< void > const & cap,
 int timeout = To_default,
 l4_umword_t * local_id = 0,
 bool iterate = true) const [noexcept]
```

Query the name space for a named object.

The query string does not necessarily need to be null-terminated.

#### Parameters

|     |                 |                                                                                                                                                                                                                  |
|-----|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in  | <i>len</i>      | Length of the string to query without any terminating null characters.                                                                                                                                           |
| in  | <i>name</i>     | String to query (without any leading slashes).                                                                                                                                                                   |
| out | <i>cap</i>      | Capability slot where the received capability will be put.                                                                                                                                                       |
| in  | <i>timeout</i>  | Timeout of query in milliseconds. The client will only wait if a name has already been registered with the server but no object has yet been attached.                                                           |
| out | <i>local_id</i> | If given, <a href="#">L4_RCV_ITEM_LOCAL_ID</a> will be set for the IPC from the name space, so that if the capability that was received is a local item, the capability ID will be returned with this parameter. |
| in  | <i>iterate</i>  | If true, the client will try to resolve names by iteratively calling the name spaces until the name is fully resolved.                                                                                           |

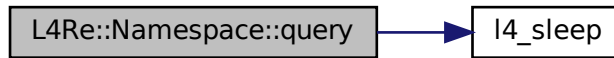
#### Return values

|            |                                                                                     |
|------------|-------------------------------------------------------------------------------------|
| 0          | Name could be fully resolved.                                                       |
| >0         | Name could only be partly resolved. The number of remaining characters is returned. |
| -L4_ENOENT | Entry could not be found.                                                           |
| -L4_EAGAIN | Entry exists but no object is yet attached. Try again later.                        |
| <0         | IPC errors, see <a href="#">l4_error_code_t</a> .                                   |

Definition at line 77 of file [namespace\\_impl.h](#).

References [L4\\_EAGAIN](#), [L4\\_EINVAL](#), [l4\\_sleep\(\)](#), and [L4\\_UNLIKELY](#).

Here is the call graph for this function:



### 15.250.3.3 register\_obj()

```

long L4Re::Namespace::register_obj (
 char const * name,
 L4::Ipc::Cap< void > obj,
 unsigned flags = Rw) const [inline], [noexcept]

```

Register an object with a name.

#### Parameters

|              |                                                                                                                                                                                                                                                                                                 |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>name</i>  | Name under which the object should be registered.                                                                                                                                                                                                                                               |
| <i>obj</i>   | Capability to object to register. An invalid capability may be given to only reserve the name for later use.                                                                                                                                                                                    |
| <i>flags</i> | Flags to assign to the entry, see <a href="#">L4Re::Namespace::Register_flags</a> . Note that the rights that are assigned to a capability are not only determined by the rights given in these flags but also by the rights with which the <i>obj</i> 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 doesn't have necessary permissions.            |
| <i>-L4_ENOMEM</i> | Server has insufficient resources.                    |
| <i>-L4_EINVAL</i> | Invalid parameter.                                    |
| <i>&lt;0</i>      | IPC errors, see <a href="#">l4_error_code_t</a> .     |

#### Precondition

requires capability rights: {RW}

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

### 15.250.3.4 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 write permission.            |
| -L4_EACCESS | Name cannot be removed.                           |
| <0          | IPC errors, see <a href="#">l4_error_code_t</a> . |

#### Precondition

requires capability rights: {RW}

Definition at line 202 of file [namespace](#).

The documentation for this class was generated from the following files:

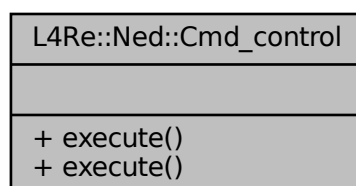
- [l4/re/namespace](#)
- [l4/re/impl/namespace\\_impl.h](#)

## 15.251 L4Re::Ned::Cmd\_control Class Reference

Direct control interface for Ned.

Inherits [L4::Kobject\\_0t](#)< [Derived](#), [PROTO](#), [S\\_DEMAND](#) >.

Collaboration diagram for L4Re::Ned::Cmd\_control:



## Public Member Functions

- long [execute](#) (L4::Ipc::String<> cmd) noexcept  
*Execute the given Lua code.*
- long [execute](#) (L4::Ipc::String<> cmd, L4::Ipc::String< char > \*result) noexcept  
*Execute the given Lua code.*

### 15.251.1 Detailed Description

Direct control interface for Ned.

Definition at line 20 of file [cmd\\_control](#).

### 15.251.2 Member Function Documentation

#### 15.251.2.1 [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](#).

#### 15.251.2.2 [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](#).

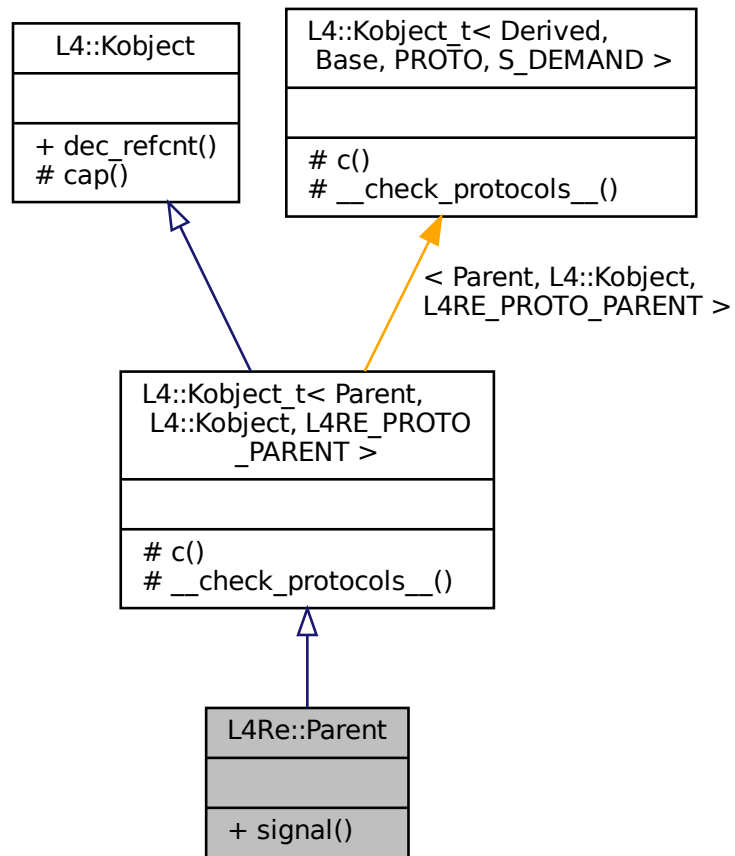
The documentation for this class was generated from the following file:

- [pkg/l4re-core/ned/lib/include/cmd\\_control](#)

## 15.252 L4Re::Parent Class Reference

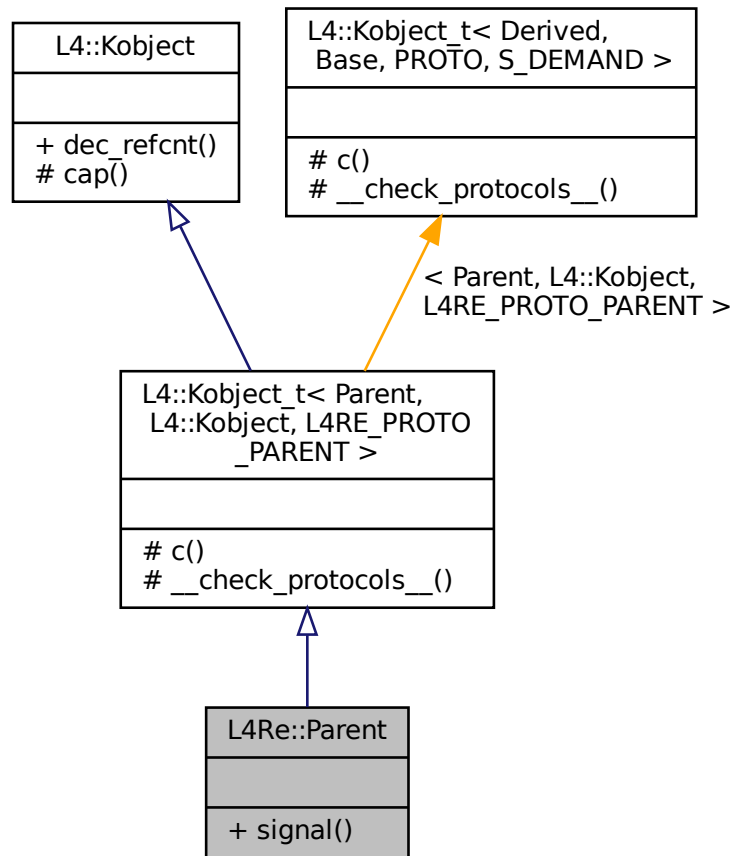
[Parent](#) interface.

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.*

## Additional Inherited Members

### 15.252.1 Detailed Description

[Parent](#) interface.

See also

[Parent API](#) for more details about the purpose.

Definition at line 51 of file [parent](#).

## 15.252.2 Member Function Documentation

### 15.252.2.1 `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 |

#### Returns

0 on success, <0 on error

- [-L4\\_ENOREPLY](#)
- IPC errors

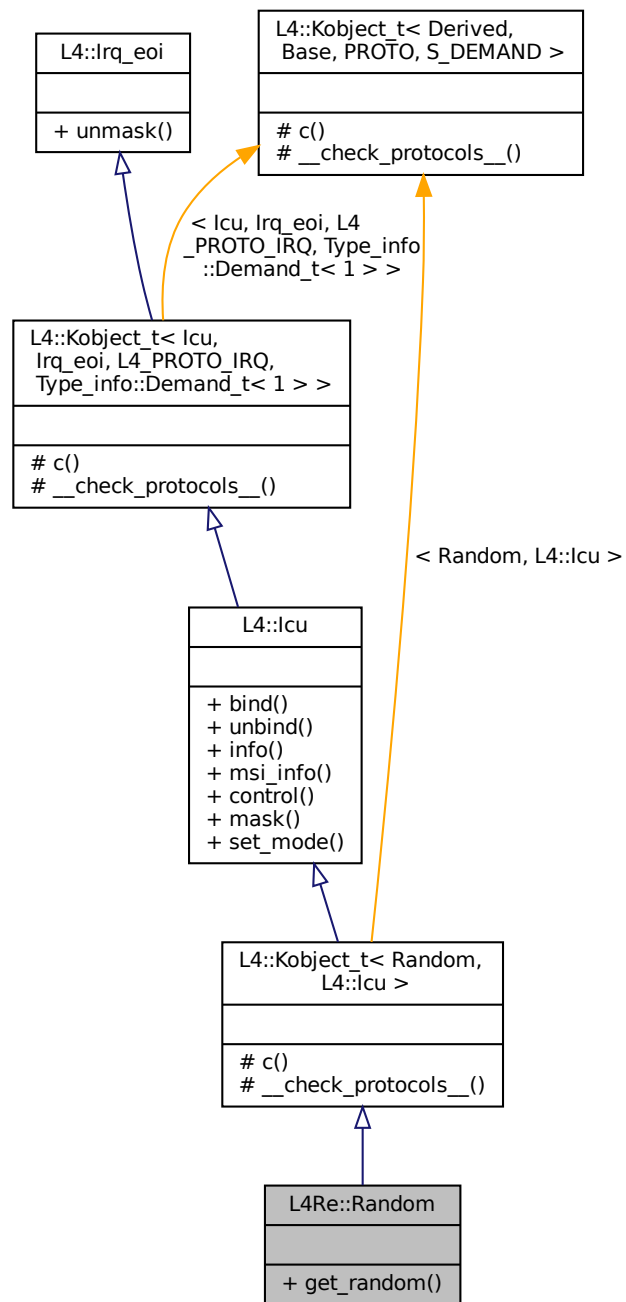
The documentation for this class was generated from the following file:

- [l4/re/parent](#)

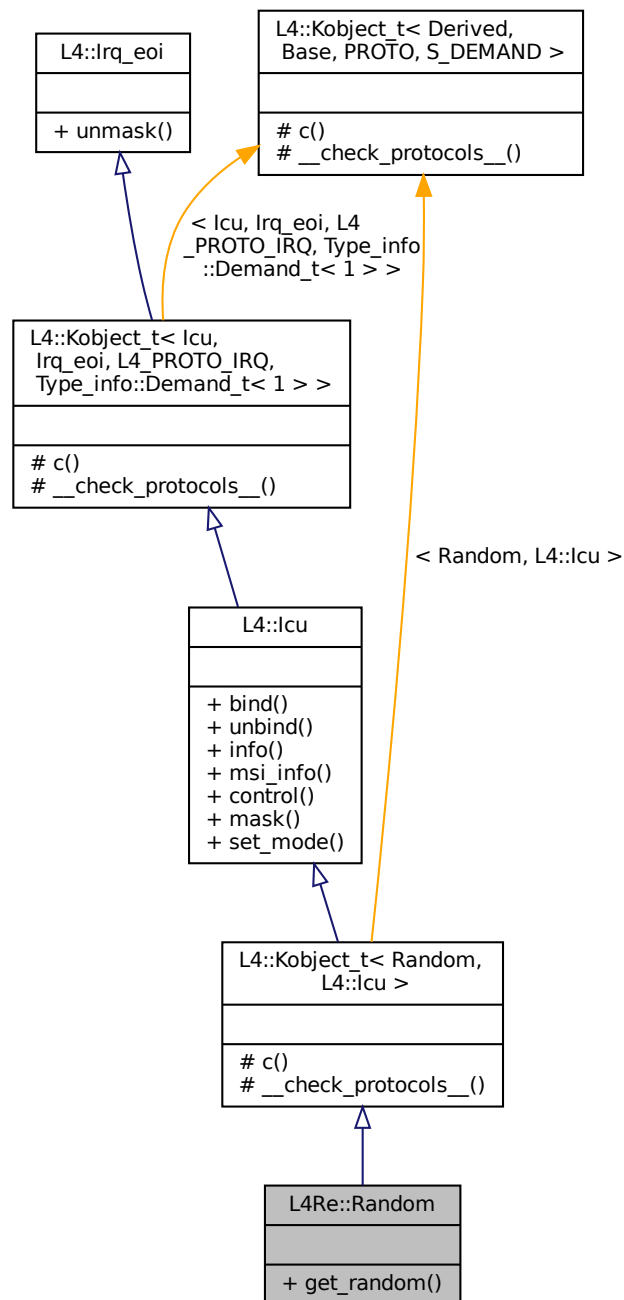
## 15.253 L4Re::Random Struct Reference

Low-bandwidth interface for random number generators.

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.*

## Additional Inherited Members

### 15.253.1 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 29 of file [random](#).

### 15.253.2 Member Function Documentation

#### 15.253.2.1 [get\\_random\(\)](#)

```
long L4Re::Random::get_random (
 l4_size_t size,
 L4::Ipc::Array< char, unsigned long > * buffer)
```

Get a random number.

#### Parameters

|     |               |                                                                                             |
|-----|---------------|---------------------------------------------------------------------------------------------|
|     | <i>size</i>   | Number of bytes of entropy requested.                                                       |
| out | <i>buffer</i> | Buffer containing the random number. Each byte in the buffer contains 8 bits of randomness. |

#### Return values

|            |                                                                                                                                                                      |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $\geq 0$   | Actual size of the returned random number in bytes. This may be less than the requested size. The return value may also be 0 if temporarily no entropy is available. |
| $-L4\_EIO$ | 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:

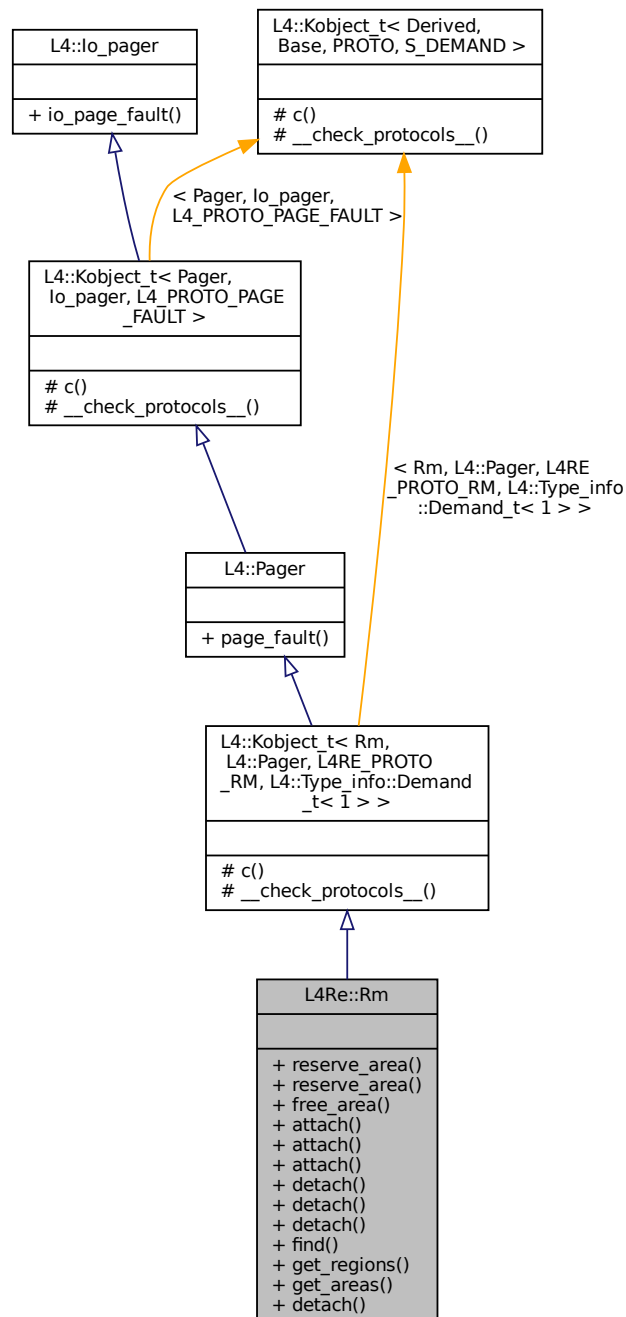
- l4/re/random

## 15.254 L4Re::Rm Class Reference

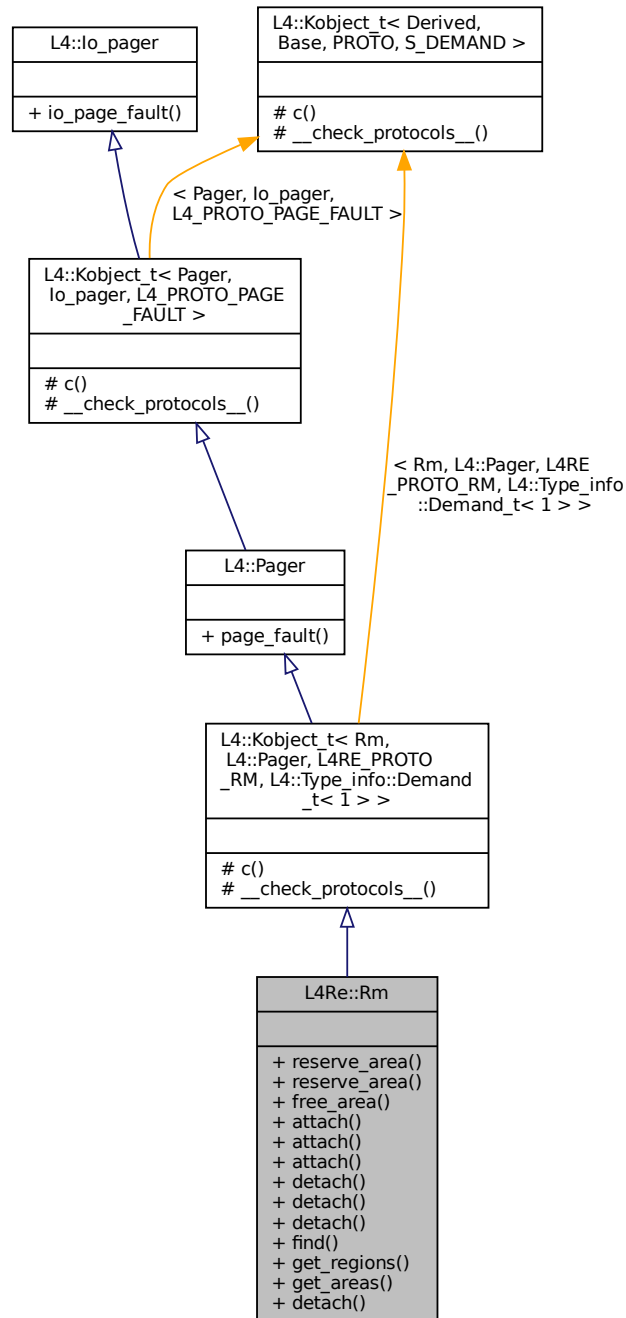
Region map.

```
#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.*

## 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 }
- 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 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 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 all regions of the specified interval.*
- int [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.*



## Additional Inherited Members

### 15.254.1 Detailed Description

Region map.

See also

[Region map API](#) .

Definition at line 73 of file [rm](#).

### 15.254.2 Member Typedef Documentation

#### 15.254.2.1 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 592 of file [rm](#).

#### 15.254.2.2 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 584 of file [rm](#).

### 15.254.3 Member Enumeration Documentation

#### 15.254.3.1 Detach\_flags

```
enum L4Re::Rm::Detach_flags
```

Flags for detach operation.

**Enumerator**

|                |                                                                                  |
|----------------|----------------------------------------------------------------------------------|
| Detach_exact   | Do an unmap of the exact region given.                                           |
| Detach_overlap | Do an unmap of all overlapping regions.                                          |
| Detach_keep    | Do not free the detached data space, ignore the <a href="#">F::Detach_free</a> . |

Definition at line 200 of file [rm](#).

**15.254.3.2 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 81 of file [rm](#).

**15.254.3.3 Region\_flag\_shifts**

```
enum L4Re::Rm::Region_flag_shifts
```

**Enumerator**

|               |                                         |
|---------------|-----------------------------------------|
| Caching_shift | Start of <a href="#">Rm</a> cache bits. |
|---------------|-----------------------------------------|

Definition at line 92 of file [rm](#).

**15.254.4 Member Function Documentation****15.254.4.1 attach() [1/2]**

```
long L4Re::Rm::attach (
 14_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. If <a href="#">L4Re::Rm::F::Search_addr</a> is given this value is used as the start address to search for a free virtual memory region and the resulting address is returned here. If <a href="#">L4Re::Rm::F::In_area</a> is given the value is used as a selector for the area (see <a href="#">L4Re::Rm::reserve_area</a> ) to attach the data space to.                  |
|                | <i>size</i>  | Size of the data space to attach (in bytes)                                                                                                                                                                                                                                                                                                                                                                                                               |
|                | <i>flags</i> | The flags control how and with which rights the dataspace is attached to the region. See <a href="#">L4Re::Rm::F::Attach_flags</a> and <a href="#">L4Re::Rm::F::Region_flags</a> . The caller must specify the desired rights of the attached region explicitly. The default set of rights is empty. If the <a href="#">F::Eager_map</a> flag is set this function may also return <a href="#">L4Re::Dataspace::map</a> error codes if the mapping fails. |
|                | <i>mem</i>   | Data space                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|                | <i>offs</i>  | Offset into the data space to use                                                                                                                                                                                                                                                                                                                                                                                                                         |
|                | <i>align</i> | Alignment of the virtual region, log2-size, default: a page ( <a href="#">L4_PAGESHIFT</a> ). This is only meaningful if the <a href="#">L4Re::Rm::F::Search_addr</a> flag is used.                                                                                                                                                                                                                                                                       |

#### Return values

|                   |                                                                    |
|-------------------|--------------------------------------------------------------------|
| 0                 | Success                                                            |
| -L4_ENOENT        | No area could be found (see <a href="#">L4Re::Rm::F::In_area</a> ) |
| -L4_EPERM         | Operation not allowed.                                             |
| -L4_EINVAL        |                                                                    |
| -L4_EADDRNOTAVAIL | The given address is not available.                                |
| <0                | IPC errors                                                         |

Makes the whole or parts of a data space visible in the virtual memory of the corresponding task. The corresponding region in the virtual address space is backed with the contents of the dataspace.

#### Note

When searching for a free place in the virtual address space, the space between *start* and the end of the virtual address space is searched.

There is no region object created, instead the region is defined by a virtual address within this range (see [L4Re::Rm::find](#)).

Definition at line 43 of file [rm\\_impl.h](#).

**15.254.4.2 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. If <a href="#">L4Re::Rm::F::Search_addr</a> is given this value is used as the start address to search for a free virtual memory region and the resulting address is returned here. If <a href="#">L4Re::Rm::F::In_area</a> is given the value is used as a selector for the area (see <a href="#">L4Re::Rm::reserve_area</a> ) to attach the data space to.                  |
|                | <i>size</i>  | Size of the data space to attach (in bytes)                                                                                                                                                                                                                                                                                                                                                                                                               |
|                | <i>flags</i> | The flags control how and with which rights the dataspace is attached to the region. See <a href="#">L4Re::Rm::F::Attach_flags</a> and <a href="#">L4Re::Rm::F::Region_flags</a> . The caller must specify the desired rights of the attached region explicitly. The default set of rights is empty. If the <a href="#">F::Eager_map</a> flag is set this function may also return <a href="#">L4Re::Dataspace::map</a> error codes if the mapping fails. |
|                | <i>mem</i>   | Data space                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|                | <i>offs</i>  | Offset into the data space to use                                                                                                                                                                                                                                                                                                                                                                                                                         |
|                | <i>align</i> | Alignment of the virtual region, log2-size, default: a page ( <a href="#">L4_PAGESHIFT</a> ). This is only meaningful if the <a href="#">L4Re::Rm::F::Search_addr</a> flag is used.                                                                                                                                                                                                                                                                       |

**Return values**

|                          |                                                                    |
|--------------------------|--------------------------------------------------------------------|
| <i>0</i>                 | Success                                                            |
| <i>-L4_ENOENT</i>        | No area could be found (see <a href="#">L4Re::Rm::F::In_area</a> ) |
| <i>-L4_EPERM</i>         | Operation not allowed.                                             |
| <i>-L4_EINVAL</i>        |                                                                    |
| <i>-L4_EADDRNOTAVAIL</i> | The given address is not available.                                |
| <i>&lt;0</i>             | IPC errors                                                         |

Makes the whole or parts of a data space visible in the virtual memory of the corresponding task. The corresponding region in the virtual address space is backed with the contents of the dataspace.

**Note**

When searching for a free place in the virtual address space, the space between *start* and the end of the virtual address space is searched.

There is no region object created, instead the region is defined by a virtual address within this range (see [L4Re::Rm::find](#)).

Definition at line 369 of file [rm](#).

**15.254.4.3 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 a region from the address space.

**Parameters**

|     |             |                                                                                                                                                                    |
|-----|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|     | <i>addr</i> | Virtual address of region, any address within the region is valid.                                                                                                 |
| out | <i>mem</i>  | <a href="#">Dataspace</a> that is affected. Give 0 if not interested.                                                                                              |
|     | <i>task</i> | This argument specifies the task where the pages are unmapped. Provide <a href="#">L4::Cap&lt;L4::Task&gt;::Invalid</a> for none. The default is the current task. |

**Return values**

|                               |                  |
|-------------------------------|------------------|
| <a href="#">Detach_result</a> | On success.      |
| <a href="#">-L4_ENOENT</a>    | No region found. |
| <0                            | IPC errors       |

Frees a region in the virtual address space given by *addr* (address type). The corresponding part of the address space is now available again.

Definition at line 637 of file [rm](#).

**15.254.4.4 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 all regions of the specified interval.

**Parameters**

|     |              |                                                                                                                                                                    |
|-----|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|     | <i>start</i> | Start of area to detach, must be within region.                                                                                                                    |
|     | <i>size</i>  | Size of of area to detach (in bytes).                                                                                                                              |
| out | <i>mem</i>   | <a href="#">Dataspace</a> that is affected. Give 0 if not interested.                                                                                              |
|     | <i>task</i>  | This argument specifies the task where the pages are unmapped. Provide <a href="#">L4::Cap&lt;L4::Task&gt;::Invalid</a> for none. The default is the current task. |

**Return values**

|                               |                  |
|-------------------------------|------------------|
| <a href="#">Detach_result</a> | On success.      |
| <a href="#">-L4_ENOENT</a>    | No region found. |

## Return values

|    |            |
|----|------------|
| <0 | 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 647 of file [rm](#).

**15.254.4.5 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 a region from the address space.

## Parameters

|     |             |                                                                                                                                                                    |
|-----|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|     | <i>addr</i> | Virtual address of region, any address within the region is valid.                                                                                                 |
| out | <i>mem</i>  | <a href="#">Dataspace</a> that is affected. Give 0 if not interested.                                                                                              |
|     | <i>task</i> | This argument specifies the task where the pages are unmapped. Provide <a href="#">L4::Cap&lt;L4::Task&gt;::Invalid</a> for none. The default is the current task. |

## Return values

|                               |                  |
|-------------------------------|------------------|
| <a href="#">Detach_result</a> | On success.      |
| <a href="#">-L4_ENOENT</a>    | No region found. |
| <0                            | IPC errors       |

Frees a region in the virtual address space given by addr (address type). The corresponding part of the address space is now available again.

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

**15.254.4.6 find()**

```
int L4Re::Rm::find (
 L4_addr_t * addr,
 unsigned long * size,
 Offset * offset,
 L4Re::Rm::Flags * flags,
 L4::Cap< Dataspace > * m) [inline], [noexcept]
```

Find a region given an address and size.

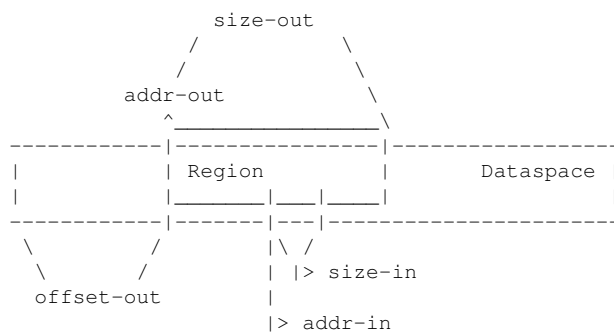
## Parameters

|         |               |                                                                                           |
|---------|---------------|-------------------------------------------------------------------------------------------|
| in, out | <i>addr</i>   | Address to look for. Returns the start address of the found region.                       |
| in, out | <i>size</i>   | Size of the area to look for (in bytes). Returns the size of the found region (in bytes). |
| out     | <i>offset</i> | Offset at the beginning of the region within the associated dataspace.                    |
| out     | <i>flags</i>  | Region flags, see <a href="#">F::Region_flags</a> (and <a href="#">F::ln_area</a> ).      |
| out     | <i>m</i>      | Associated dataspace or paging service.                                                   |

## Return values

|            |                        |
|------------|------------------------|
| 0          | Success                |
| -L4_EPERM  | Operation not allowed. |
| -L4_ENOENT | No region found.       |
| <0         | IPC errors             |

This function returns the properties of the region that contains the area described by the *addr* and *size* parameter. If no such region is found but a reserved area, the area is returned and [F::ln\\_area](#) is set in *flags*. Note, in the case of an area the *offset* and *m* return values are invalid.



## Note

The value of the size input parameter should be 1 to assure that a region can be determined unambiguously.

Definition at line 559 of file [rm](#).

## 15.254.4.7 free\_area()

```
long L4Re::Rm::free_area (
 l4_addr_t addr)
```

Free an area from the region map.

## Parameters

|             |                                     |
|-------------|-------------------------------------|
| <i>addr</i> | An address within the area to free. |
|-------------|-------------------------------------|

## Return values

|            |                |
|------------|----------------|
| 0          | Success        |
| -L4_ENOENT | No area found. |
| <0         | IPC errors     |

## Note

The data spaces that are attached to that area are not detached by this operation.

## See also

[reserve\\_area\(\)](#) for more information about areas.

**15.254.4.8 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.

**15.254.4.9 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

|          |                                                               |
|----------|---------------------------------------------------------------|
| $\geq 0$ | 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.

15.254.4.10 `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

|         |              |                                                                                                                             |
|---------|--------------|-----------------------------------------------------------------------------------------------------------------------------|
| in, out | <i>start</i> | The virtual start address of the area to reserve. Returns the start address of the area.                                    |
|         | <i>size</i>  | The size of the area to reserve (in bytes).                                                                                 |
|         | <i>flags</i> | Flags for the reserved area (see <a href="#">L4Re::Rm::F::Region_flags</a> and <a href="#">L4Re::Rm::F::Attach_flags</a> ). |
|         | <i>align</i> | Alignment of area if searched as bits (log2 value).                                                                         |

## Return values

|                   |                                    |
|-------------------|------------------------------------|
| 0                 | Success                            |
| -L4_EADDRNOTAVAIL | The given area cannot be reserved. |
| $< 0$             | IPC errors                         |

This function reserves an area within the virtual address space implemented by the region map. There are two kinds of areas available:

- Reserved areas (*flags* = [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 [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* = [F::Search\\_addr](#)), the space between *start* and the end of the virtual address space is searched.

Definition at line [258](#) of file [rm](#).

**15.254.4.11 reserve\_area()** [2/2]

```
template<typename T >
long L4Re::Rm::reserve_area (
 T ** start,
 unsigned long size,
 Flags flags = Flags(0),
 unsigned char align = L4_PAGESHIFT) const [inline], [noexcept]
```

Reserve the given area in the region map.

**Parameters**

|                |              |                                                                                                         |
|----------------|--------------|---------------------------------------------------------------------------------------------------------|
| <i>in, out</i> | <i>start</i> | The virtual start address of the area to reserve. Returns the start address of the area.                |
|                | <i>size</i>  | The size of the area to reserve (in bytes).                                                             |
|                | <i>flags</i> | Flags for the reserved area (see <a href="#">F::Region_flags</a> and <a href="#">F::Attach_flags</a> ). |
|                | <i>align</i> | Alignment of area if searched as bits (log2 value).                                                     |

**Return values**

|                          |                                    |
|--------------------------|------------------------------------|
| <i>0</i>                 | Success                            |
| <i>-L4_EADDRNOTAVAIL</i> | The given area cannot be reserved. |
| <i>&lt;0</i>             | IPC errors                         |

For more information, please refer to the analogous function

**See also**

[L4Re::Rm::reserve\\_area](#).

Definition at line [284](#) of file [rm](#).

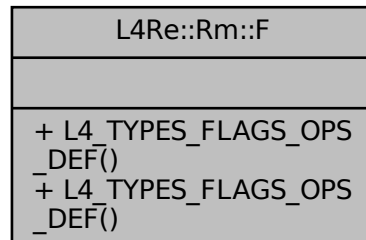
The documentation for this class was generated from the following files:

- [l4/re/rm](#)
- [l4/re/impl/rm\\_impl.h](#)

## 15.255 L4Re::Rm::F Struct Reference

[Rm](#) flags definitions.

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 }

### 15.255.1 Detailed Description

[Rm](#) flags definitions.

Definition at line 99 of file [rm](#).

### 15.255.2 Member Enumeration Documentation

#### 15.255.2.1 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 102 of file [rm](#).

### 15.255.2.2 Region\_flags

```
enum L4Re::Rm::F::Region_flags : l4_uint16_t
```

## Enumerator

|                   |                                                   |
|-------------------|---------------------------------------------------|
| Rights_mask       | Region rights.                                    |
| Detach_free       | Free the portion of the data space after detach.  |
| Pager             | Region has a pager.                               |
| Reserved          | Region is reserved (blocked)                      |
| Caching_mask      | Mask of all <a href="#">Rm</a> cache bits.        |
| Cache_normal      | Cache bits for normal cacheable memory.           |
| Cache_buffered    | Cache bits for buffered (write combining) memory. |
| Cache_uncached    | Cache bits for uncached memory.                   |
| Ds_map_mask       | Mask for all bits for cache options and rights.   |
| Region_flags_mask | Mask of all region flags.                         |

Definition at line 116 of file [rm](#).

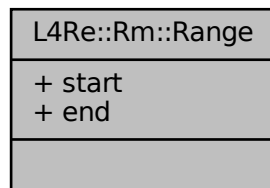
The documentation for this struct was generated from the following file:

- [l4/re/rm](#)

## 15.256 L4Re::Rm::Range Struct Reference

A range of virtual addresses.

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.*

## 15.256.1 Detailed Description

A range of virtual addresses.

Definition at line 571 of file [rm](#).

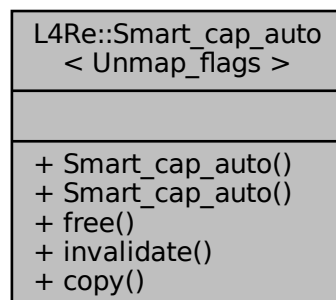
The documentation for this struct was generated from the following file:

- [l4/re/rm](#)

## 15.257 L4Re::Smart\_cap\_auto< Unmap\_flags > Class Template Reference

Helper for Unique\_cap and Unique\_del\_cap.

Collaboration diagram for L4Re::Smart\_cap\_auto< Unmap\_flags >:



### 15.257.1 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](#).

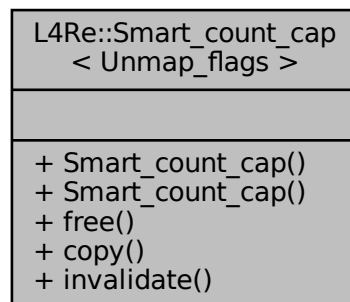
The documentation for this class was generated from the following file:

- [l4/re/cap\\_alloc](#)

## 15.258 L4Re::Smart\_count\_cap< Unmap\_flags > Class Template Reference

Helper for Ref\_cap and Ref\_del\_cap.

Collaboration diagram for L4Re::Smart\_count\_cap< Unmap\_flags >:



### 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](#).*

### 15.258.1 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](#).

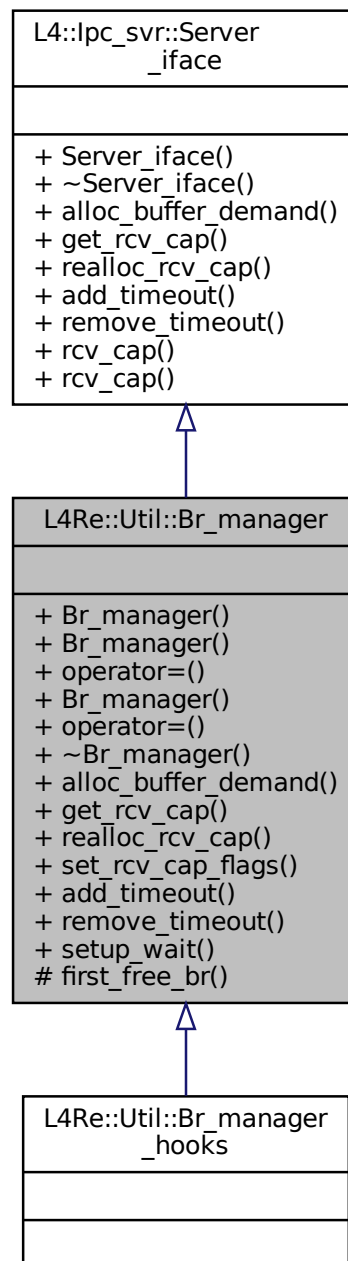
The documentation for this class was generated from the following file:

- [l4/re/cap\\_alloc](#)

## 15.259 L4Re::Util::Br\_manager Class Reference

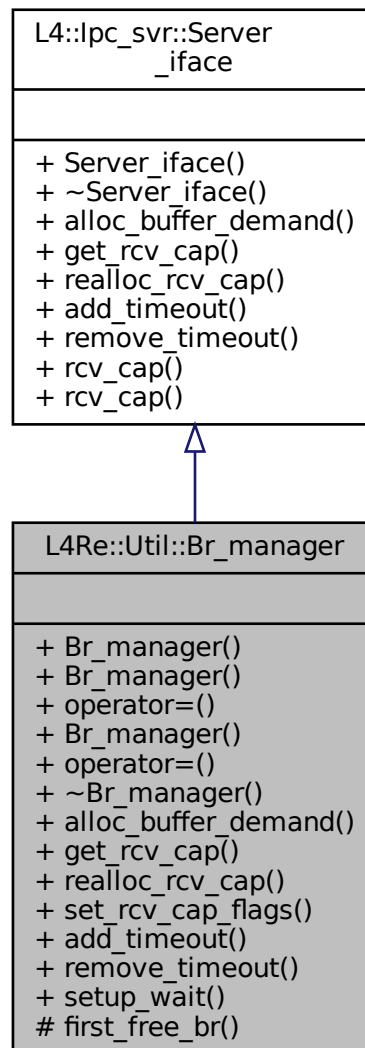
Buffer-register (BR) manager for [L4::Server](#).

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)  
*Tells the server to allocate buffers for the given demand.*
- [L4::Cap](#)< void > [get\\_rcv\\_cap](#) (int i) const  
*Get capability slot allocated to the given receive buffer.*
- int [realloc\\_rcv\\_cap](#) (int i)  
*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](#))  
*No timeouts handled by us.*
- int [remove\\_timeout](#) ([L4::lpc\\_svr::Timeout](#) \*)  
*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](#))*

## Protected Member Functions

- unsigned [first\\_free\\_br](#) () const  
*Used for assigning BRs for a timeout.*

## Additional Inherited Members

### 15.259.1 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](#).

### 15.259.2 Member Function Documentation

#### 15.259.2.1 [alloc\\_buffer\\_demand\(\)](#)

```
int L4Re::Util::Br_manager::alloc_buffer_demand (
 Demand const & demand) [inline], [virtual]
```

Tells the server to allocate buffers for the given demand.

#### Parameters

|                               |                                                                                                            |
|-------------------------------|------------------------------------------------------------------------------------------------------------|
| <i><a href="#">demand</a></i> | The total server-side demand of receive buffers needed for a given interface, see <a href="#">Demand</a> . |
|-------------------------------|------------------------------------------------------------------------------------------------------------|

This function is not called by user applications directly. Usually the server implementation or the registry implementation calls this function whenever a new object is registered at the server.

Implements [L4::lpc\\_svr::Server\\_iface](#).

Definition at line 64 of file [br\\_manager](#).

### 15.259.2.2 get\_rcv\_cap()

```
L4::Cap<void> L4Re::Util::Br_manager::get_rcv_cap (
 int index) const [inline], [virtual]
```

Get capability slot allocated to the given receive buffer.

#### Parameters

|              |                                                                                                                                                             |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>index</i> | The receive buffer index of the expected capability argument ( $0 \leq \text{index} < \text{caps}$ registered with <a href="#">alloc_buffer_demand()</a> ). |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Precondition

$0 \leq \text{index} < \text{caps}$  registered with [alloc\\_buffer\\_demand\(\)](#)

#### Returns

Capability slot currently allocated to the given receive buffer.

Implements [L4::lpc\\_svr::Server\\_iface](#).

Definition at line 95 of file [br\\_manager](#).

References [L4\\_CAP\\_MASK](#).

### 15.259.2.3 realloc\_rcv\_cap()

```
int L4Re::Util::Br_manager::realloc_rcv_cap (
 int index) [inline], [virtual]
```

Allocate a new capability for the given receive buffer.

#### Parameters

|              |                                                                                                                                                             |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>index</i> | The receive buffer index of the expected capability argument ( $0 \leq \text{index} < \text{caps}$ registered with <a href="#">alloc_buffer_demand()</a> ). |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Precondition

$0 \leq \text{index} < \text{caps}$  registered with [alloc\\_buffer\\_demand\(\)](#)

#### Returns

0 on success,  $< 0$  on error.

Implements [L4::lpc\\_svr::Server\\_iface](#).

Definition at line 103 of file [br\\_manager](#).

#### 15.259.2.4 `set_rcv_cap_flags()`

```
void L4Re::Util::Br_manager::set_rcv_cap_flags (
 unsigned long flags) [inline]
```

Set the receive flags for the buffers.

##### Precondition

Must be called before any handlers are registered.

##### Parameters

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| <i>flags</i> | New receive capability flags, see <a href="#">l4_msg_item_consts_t</a> . |
|--------------|--------------------------------------------------------------------------|

Definition at line [127](#) of file [br\\_manager](#).

References [l4\\_assert](#).

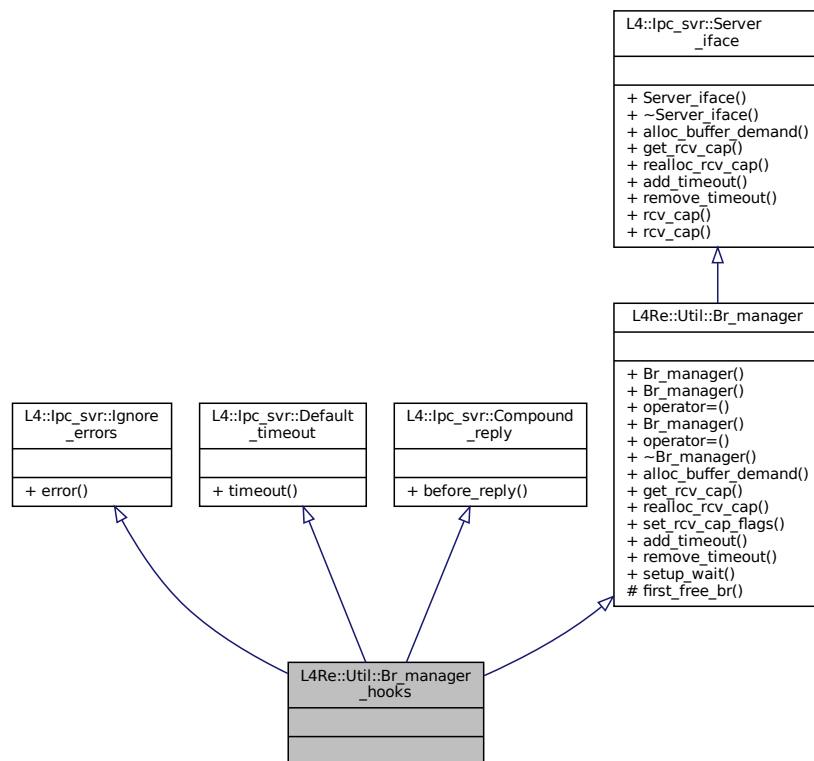
The documentation for this class was generated from the following file:

- `l4/re/util/br_manager`

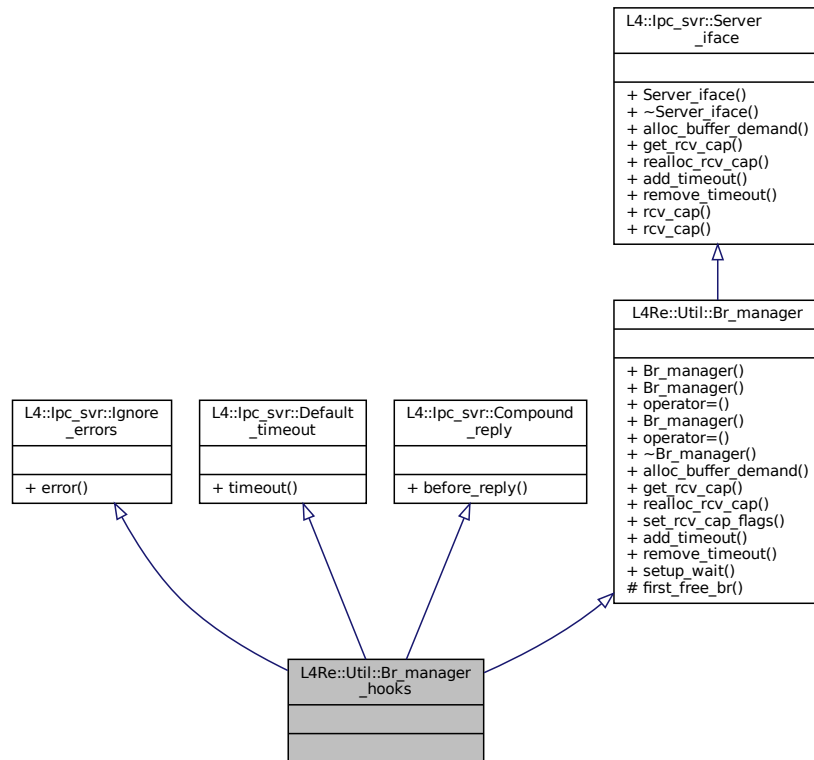
## 15.260 L4Re::Util::Br\_manager\_hooks Struct Reference

Predefined server-loop hooks for a server loop using the [Br\\_manager](#).

Inheritance diagram for L4Re::Util::Br\_manager\_hooks:



Collaboration diagram for L4Re::Util::Br\_manager\_hooks:



## Additional Inherited Members

### 15.260.1 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 174 of file [br\\_manager](#).

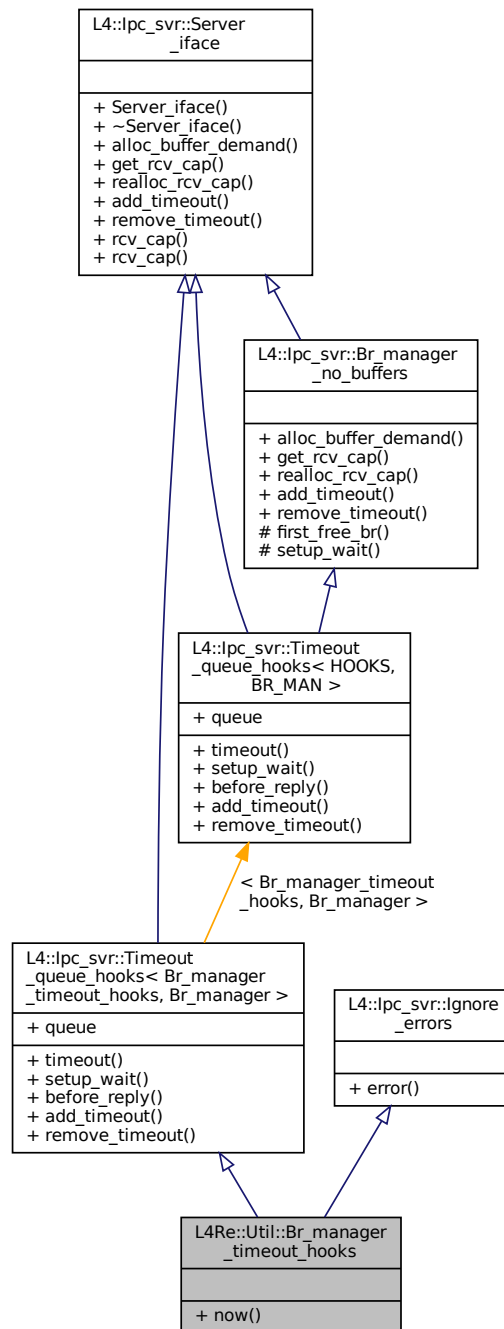
The documentation for this struct was generated from the following file:

- [l4/re/util/br\\_manager](#)

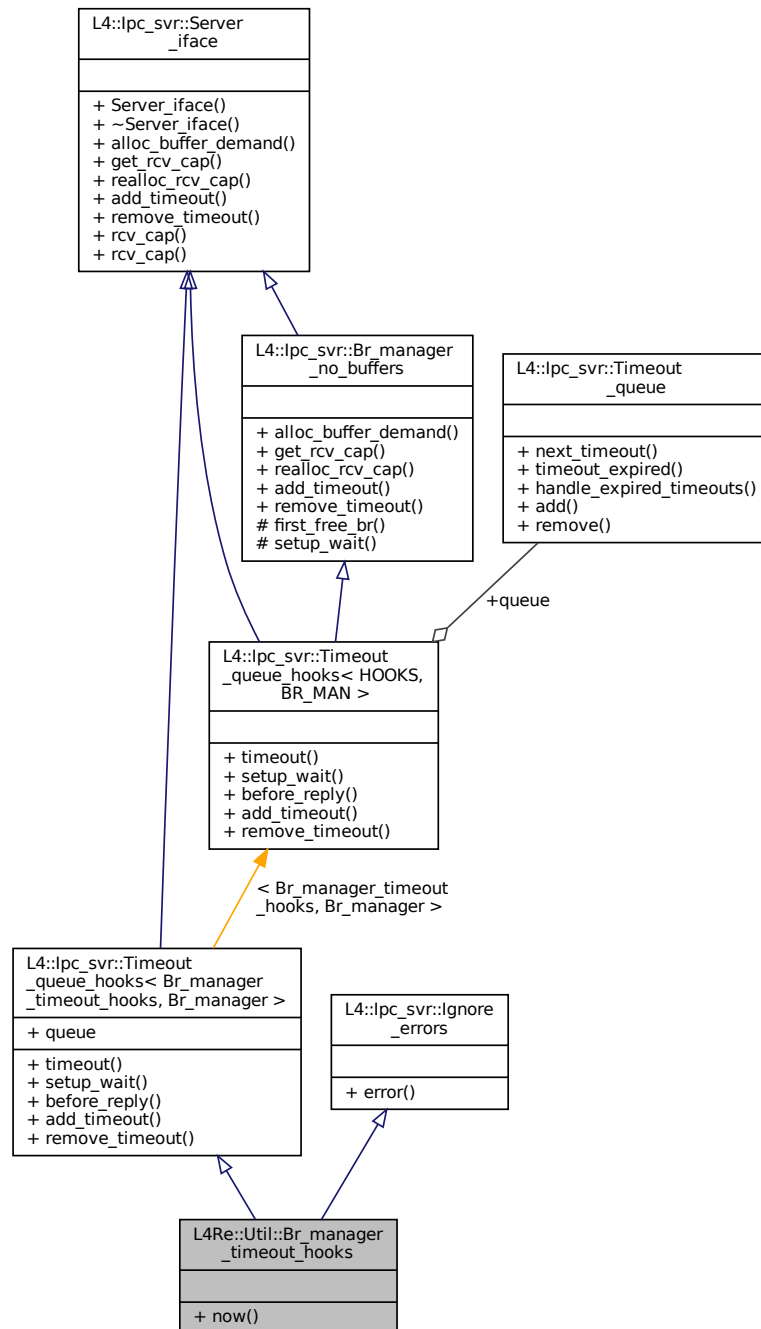
## 15.261 L4Re::Util::Br\_manager\_timeout\_hooks Struct Reference

Predefined server-loop hooks for a server with using the [Br\\_manager](#) and a timeout queue.

Inheritance diagram for L4Re::Util::Br\_manager\_timeout\_hooks:



Collaboration diagram for L4Re::Util::Br\_manager\_timeout\_hooks:



## Additional Inherited Members

### 15.261.1 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 188 of file [br\\_manager](#).

The documentation for this struct was generated from the following file:

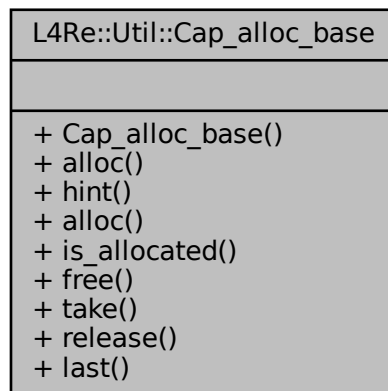
- [l4/re/util/br\\_manager](#)

## 15.262 L4Re::Util::Cap\_alloc\_base Class Reference

Capability allocator.

Inherited by L4Re::Util::Cap\_alloc< Size >.

Collaboration diagram for L4Re::Util::Cap\_alloc\_base:



### 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.*

### 15.262.1 Detailed Description

Capability allocator.

Definition at line 38 of file [bitmap\\_cap\\_alloc](#).

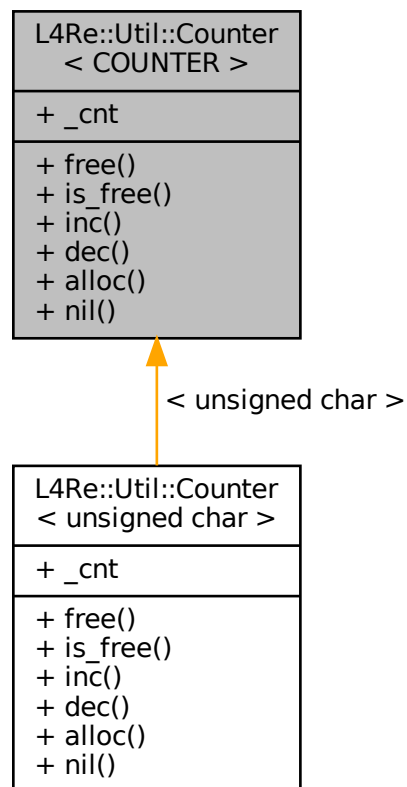
The documentation for this class was generated from the following file:

- [l4/re/util/bitmap\\_cap\\_alloc](#)

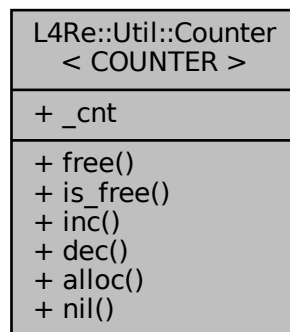
## 15.263 L4Re::Util::Counter< COUNTER > Struct Template Reference

[Counter](#) for [Counting\\_cap\\_alloc](#) with variable data width.

Inheritance diagram for L4Re::Util::Counter< COUNTER >:



Collaboration diagram for L4Re::Util::Counter< COUNTER >:



### 15.263.1 Detailed Description

```
template<typename COUNTER = unsigned char>
struct L4Re::Util::Counter< COUNTER >
```

[Counter](#) for [Counting\\_cap\\_alloc](#) with variable data width.

Definition at line 36 of file [counting\\_cap\\_alloc](#).

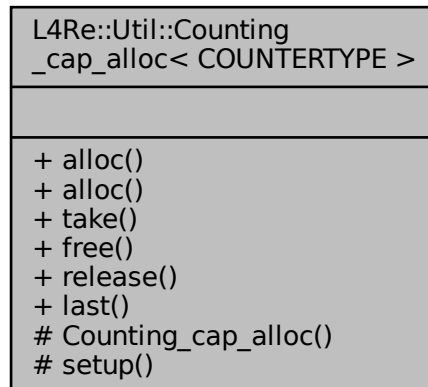
The documentation for this struct was generated from the following file:

- [l4/re/util/counting\\_cap\\_alloc](#)

## 15.264 L4Re::Util::Counting\_cap\_alloc< COUNTERTYPE > Class Template Reference

Internal reference-counting cap allocator.

Collaboration diagram for L4Re::Util::Counting\_cap\_alloc< COUNTERTYPE >:



## 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.*

### 15.264.1 Detailed Description

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
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 75 of file [counting\\_cap\\_alloc](#).

### 15.264.2 Constructor & Destructor Documentation

#### 15.264.2.1 Counting\_cap\_alloc()

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
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 104 of file [counting\\_cap\\_alloc](#).

### 15.264.3 Member Function Documentation

### 15.264.3.1 alloc() [1/2]

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
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.

#### Examples

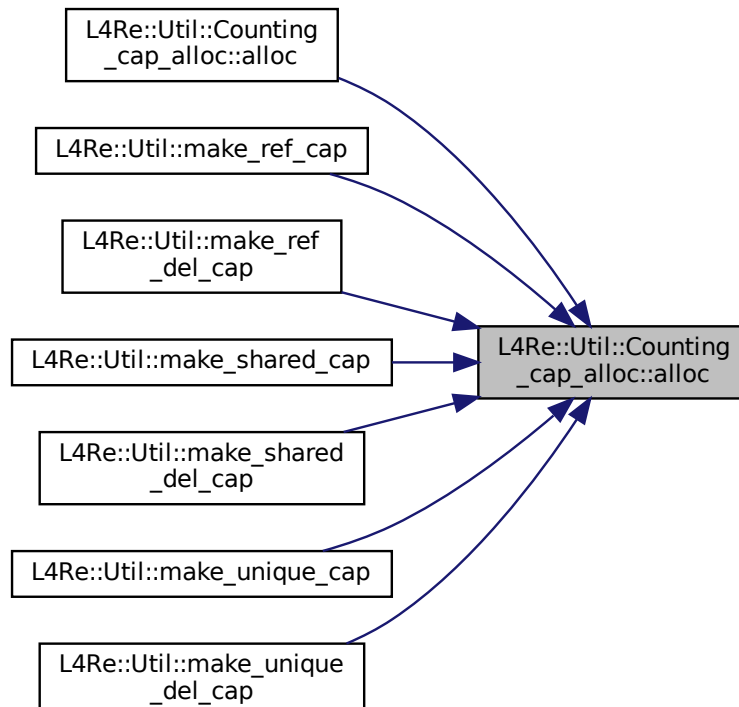
[examples/libs/l4re/c++/mem\\_alloc/ma+rm.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#), and [examples/libs/l4re/c++/shared\\_](#)

Definition at line 135 of file [counting\\_cap\\_alloc](#).

References [L4\\_CAP\\_SHIFT](#).

Referenced by [L4Re::Util::Counting\\_cap\\_alloc< COUNTERTYPE >::alloc\(\)](#), [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\(\)](#) and [L4Re::Util::make\\_unique\\_del\\_cap\(\)](#).

Here is the caller graph for this function:



**15.264.3.2 alloc()** [2/2]

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
template<typename T >
L4::Cap<T> L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc () [inline], [noexcept]
```

Allocate a new capability slot.

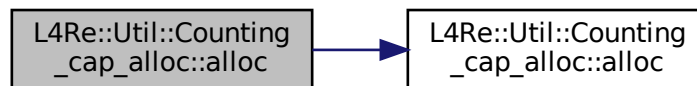
**Returns**

The newly allocated capability slot, invalid if the allocator was exhausted.

Definition at line 153 of file [counting\\_cap\\_alloc](#).

References [L4Re::Util::Counting\\_cap\\_alloc< COUNTERTYPE >::alloc\(\)](#).

Here is the call graph for this function:

**15.264.3.3 free()**

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
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.

**Examples**

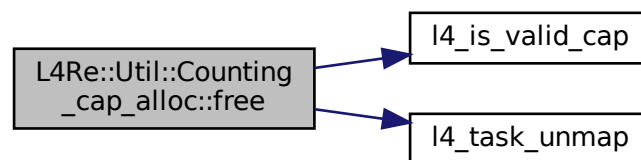
[examples/libs/l4re/c++/mem\\_alloc/ma+rm.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#), [examples/libs/l4re/c++/shared\\_ds/d](#)  
and [examples/libs/l4re/streammap/client.cc](#).

Definition at line 195 of file [counting\\_cap\\_alloc](#).

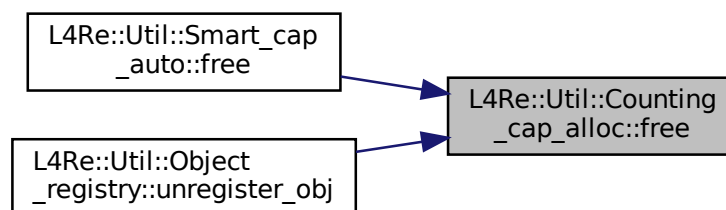
References [l4\\_assert](#), [L4\\_CAP\\_SHIFT](#), [l4\\_is\\_valid\\_cap\(\)](#), and [l4\\_task\\_unmap\(\)](#).

Referenced by [L4Re::Util::Smart\\_cap\\_auto< Unmap\\_flags >::free\(\)](#), and [L4Re::Util::Object\\_registry::unregister\\_obj\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.264.3.4 release()**

```

template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
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 <code>I4_unmap_flags_t</code> . |

## Precondition

The capability has been allocated. Calling release on a free capability results in undefined behaviour.

## Returns

True, if the capability was freed as a result of this operation. If false is returned the capability is either still in use or is not managed by this allocator.

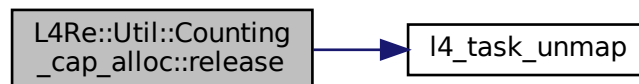
Does nothing apart from returning false if the capability is not managed by this allocator.

Definition at line 238 of file `counting_cap_alloc`.

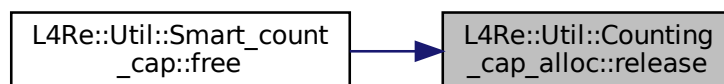
References [L4::dec](#), [I4\\_assert](#), [L4\\_CAP\\_SHIFT](#), [L4\\_INVALID\\_CAP](#), and [I4\\_task\\_unmap\(\)](#).

Referenced by [L4Re::Util::Smart\\_count\\_cap< Unmap\\_flags >::free\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.264.3.5 setup()

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
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 121 of file [counting\\_cap\\_alloc](#).

**15.264.3.6 take()**

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
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. |
|------------|----------------------------------------------------------|

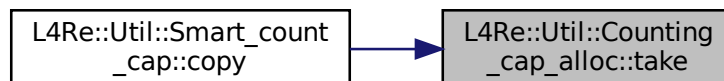
If the capability was still free, it will be automatically allocated. Silently does nothing if the capability is not managed by this allocator.

Definition at line 168 of file [counting\\_cap\\_alloc](#).

References [L4\\_CAP\\_SHIFT](#).

Referenced by [L4Re::Util::Smart\\_count\\_cap< Unmap\\_flags >::copy\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- [l4/re/util/counting\\_cap\\_alloc](#)

## 15.265 L4Re::Util::Dataspace\_svr Class Reference

[Dataspace](#) server class.

Collaboration diagram for L4Re::Util::Dataspace\_svr:

| L4Re::Util::Dataspace_svr                                                                                                                                                                                                                                       |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre># _ds_start # _ds_size # _map_flags # _cache_flags # _rw_flags</pre>                                                                                                                                                                                       |
| <pre>+ L4_RPC_LEGACY_DISPATCH() + Dataspace_svr() + ~Dataspace_svr() + map() + map_hook() + take() + release() + copy() + clear() + allocate() + and 7 more... # size() # map_flags() # page_size() # round_size() # check_limit() # map_flags() # size()</pre> |

### 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.*

### 15.265.1 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](#).

### 15.265.2 Member Function Documentation

#### 15.265.2.1 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](#).

### 15.265.2.2 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

|              |         |
|--------------|---------|
| <i>0</i>     | Success |
| <i>&lt;0</i> | Error   |

### 15.265.2.3 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

|               |                                                                     |
|---------------|---------------------------------------------------------------------|
| <i>&gt;=0</i> | Number of bytes copied                                              |
| <i>&lt;0</i>  | An error occurred. The error code may depend on the implementation. |

Definition at line 128 of file [dataspace\\_svr](#).

References [L4\\_ENODEV](#).

#### 15.265.2.4 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](#).

#### 15.265.2.5 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>      | <a href="#">Dataspace</a> flags, see <a href="#">L4Re::Dataspace::F::Flags</a> . |
|     | <i>min_addr</i>   | Defines start of receive window.                                                 |
|     | <i>max_addr</i>   | Defines end of receive window.                                                   |
| out | <i>memory</i>     | Send fpage to map                                                                |

##### Return values

|    |         |
|----|---------|
| 0  | Success |
| <0 | Error   |

#### 15.265.2.6 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**

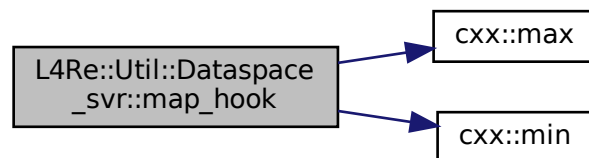
|               |                                                            |
|---------------|------------------------------------------------------------|
| <i>&lt;0</i>  | Error and the map request will be aborted with that error. |
| <i>&gt;=0</i> | Success                                                    |

**See also**[map](#)

Definition at line 89 of file [dataspace\\_svr](#).

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

Here is the call graph for this function:

**15.265.2.7 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](#).



### 15.265.2.8 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](#).

### 15.265.2.9 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](#).

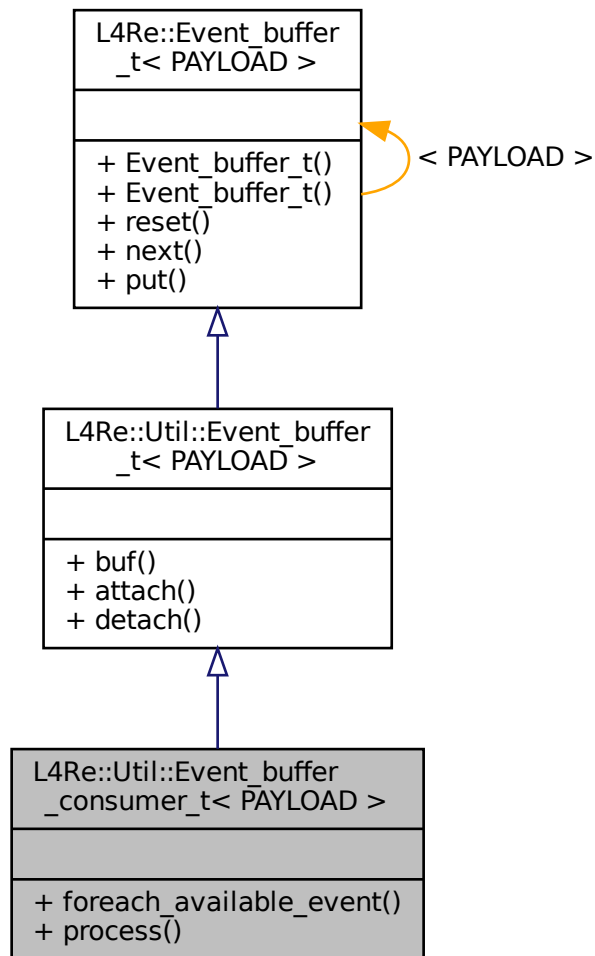
The documentation for this class was generated from the following file:

- I4/re/util/dataspace\_svr

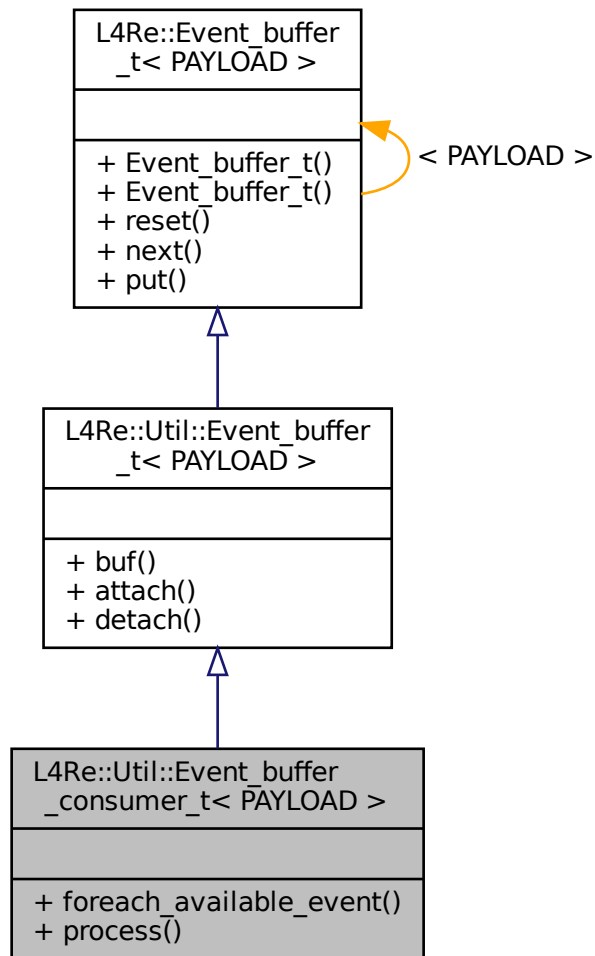
## 15.266 L4Re::Util::Event\_buffer\_consumer\_t< PAYLOAD > Class Template Reference

An event buffer consumer.

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 (L4::Cap< L4::Irq > irq, L4::Cap< L4::Thread > thread, CB const &cb, D data=D())`  
*Continuously wait for events and process them.*

### 15.266.1 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](#).

## 15.266.2 Member Function Documentation

### 15.266.2.1 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.

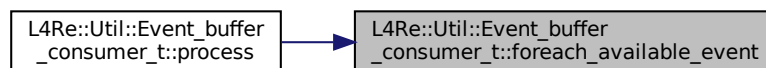
#### Parameters

|             |                                              |
|-------------|----------------------------------------------|
| <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](#).

Referenced by [L4Re::Util::Event\\_buffer\\_consumer\\_t< PAYLOAD >::process\(\)](#).

Here is the caller graph for this function:



### 15.266.2.2 process()

```
template<typename PAYLOAD >
template<typename CB , typename D >
void L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::process (
 L4::Cap< L4::Irq > irq,
 L4::Cap< L4::Thread > thread,
 CB const & cb,
 D data = D()) [inline]
```

Continuously wait for events and process them.

#### Parameters

|               |                                                           |
|---------------|-----------------------------------------------------------|
| <i>irq</i>    | <a href="#">Event</a> signal to wait for.                 |
| <i>thread</i> | Thread capability of the thread calling this function.    |
| <i>cb</i>     | Callback function that is called for each received event. |
| <i>data</i>   | Data to pass as an argument to the processing callback.   |

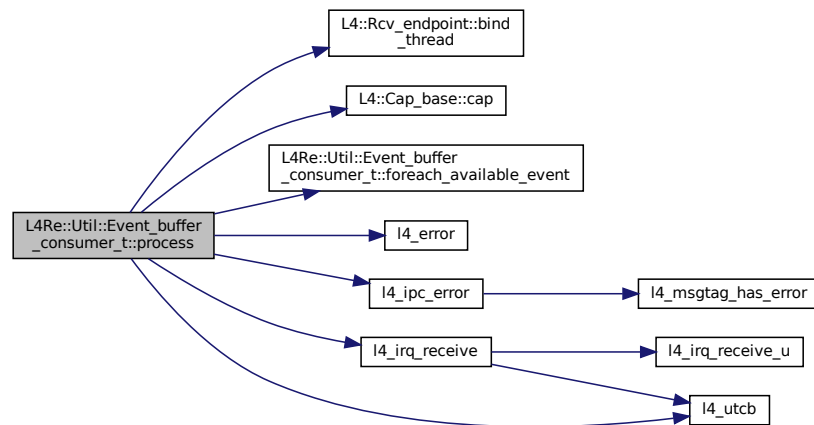
**Note**

This function never returns.

Definition at line 125 of file [event\\_buffer](#).

References [L4::Rcv\\_endpoint::bind\\_thread\(\)](#), [L4::Cap\\_base::cap\(\)](#), [L4Re::Util::Event\\_buffer\\_consumer\\_t< PAYLOAD >::foreach\\_available\\_event](#), [l4\\_error\(\)](#), [l4\\_ipc\\_error\(\)](#), [l4\\_irq\\_receive\(\)](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



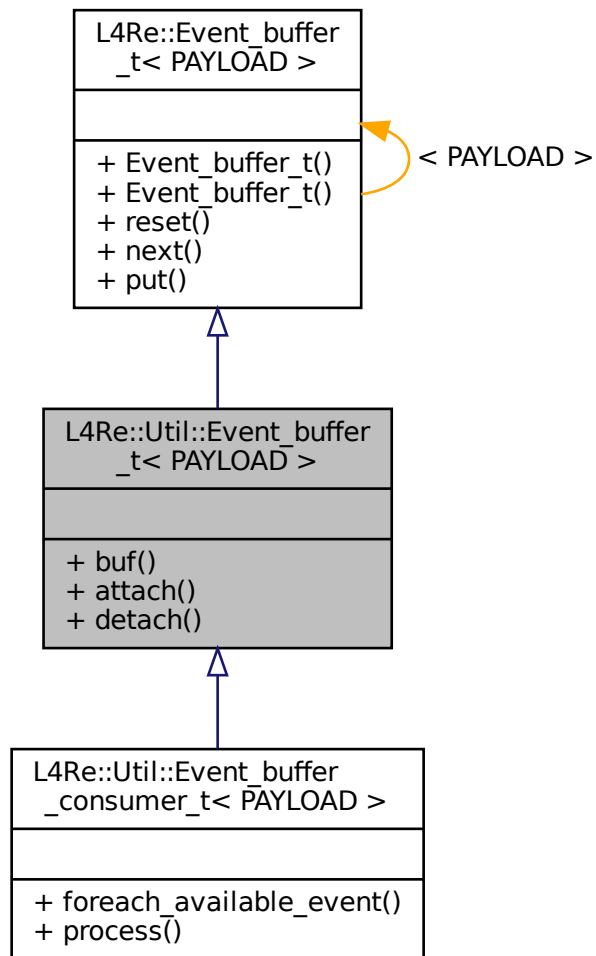
The documentation for this class was generated from the following file:

- `l4/re/util/event_buffer`

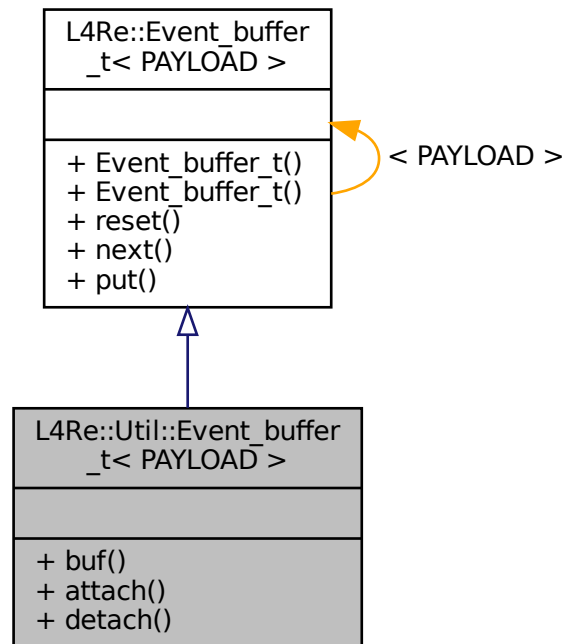
## 15.267 L4Re::Util::Event\_buffer\_t< PAYLOAD > Class Template Reference

Event\_buffer utility class.

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.*

### 15.267.1 Detailed Description

```
template<typename PAYLOAD>
class L4Re::Util::Event_buffer_t< PAYLOAD >
```

Event\_buffer utility class.

Definition at line 36 of file [event\\_buffer](#).

### 15.267.2 Member Function Documentation

### 15.267.2.1 attach()

```
template<typename PAYLOAD >
long L4Re::Util::Event_buffer_t< PAYLOAD >::attach (
 L4::Cap< L4Re::Dataspace > ds,
 L4::Cap< L4Re::Rm > rm) [inline], [noexcept]
```

Attach event buffer from address space.

#### Parameters

|           |                                                |
|-----------|------------------------------------------------|
| <i>ds</i> | <a href="#">Dataspace</a> of the event buffer. |
| <i>rm</i> | Region manager to attach buffer to.            |

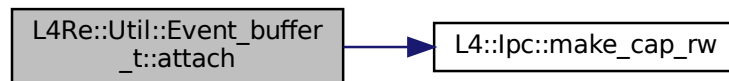
#### Returns

0 on success, negative error code otherwise.

Definition at line 56 of file [event\\_buffer](#).

References [L4::lpc::make\\_cap\\_rw\(\)](#), and [L4Re::Rm::F::Search\\_addr](#).

Here is the call graph for this function:



### 15.267.2.2 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](#).

### 15.267.2.3 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:

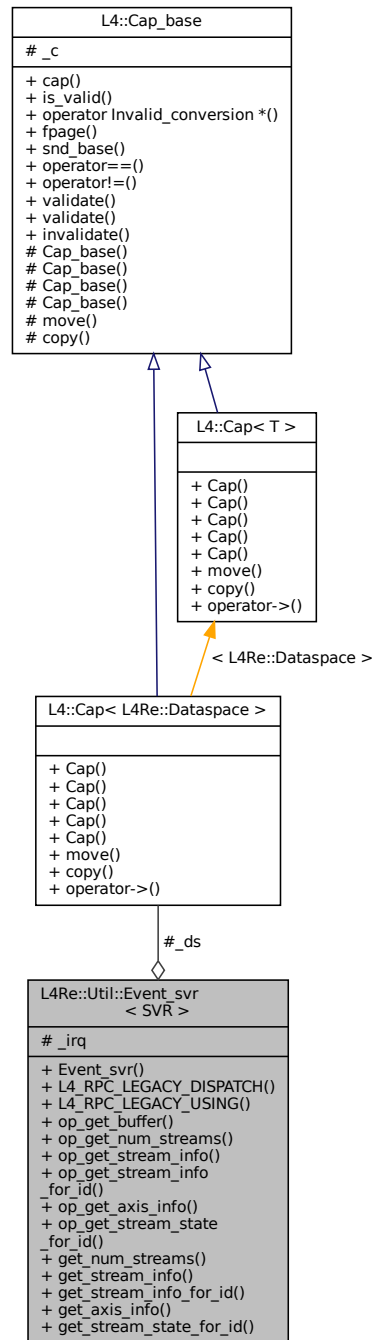
- l4/re/util/event\_buffer

## 15.268 L4Re::Util::Event\_svr< SVR > Class Template Reference

Convenience wrapper for implementing an event server.

Inherits L4Re::Util::Icu\_cap\_array\_svr< ICU >.

Collaboration diagram for L4Re::Util::Event\_svr< SVR >:



## Public Member Functions

- long `op_get_buffer` (L4Re::Event::Rights, [L4::lpc::Cap< L4Re::Dataspace >](#) &ds)  
Handle [L4Re::Event](#) protocol.

### 15.268.1 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](#).

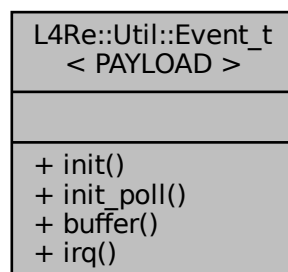
The documentation for this class was generated from the following file:

- [l4/re/util/event\\_svr](#)

## 15.269 L4Re::Util::Event\_t< PAYLOAD > Class Template Reference

Convenience wrapper for getting access to an event object.

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.*

### 15.269.1 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`.

### 15.269.2 Member Enumeration Documentation

#### 15.269.2.1 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`.

### 15.269.3 Member Function Documentation

### 15.269.3.1 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](#).

### 15.269.3.2 init()

```
template<typename PAYLOAD >
template<typename IRQ_TYPE >
int L4Re::Util::Event_t< PAYLOAD >::init (
 L4::Cap< L4Re::Event > event,
 L4Re::Env const * env = L4Re::Env::env(),
 L4Re::Cap_alloc * ca = L4Re::Cap_alloc::get_cap_alloc(L4Re::Util::cap_alloc))
[inline]
```

Initialise an event object.

#### Template Parameters

|                 |                                                                                                                           |
|-----------------|---------------------------------------------------------------------------------------------------------------------------|
| <i>IRQ_TYPE</i> | Type used for handling notifications from the event provider. This must be derived from <a href="#">L4::Triggerable</a> . |
|-----------------|---------------------------------------------------------------------------------------------------------------------------|

#### Parameters

|              |                                  |
|--------------|----------------------------------|
| <i>event</i> | Capability to event.             |
| <i>env</i>   | Pointer to L4Re-Environment      |
| <i>ca</i>    | Pointer to capability allocator. |

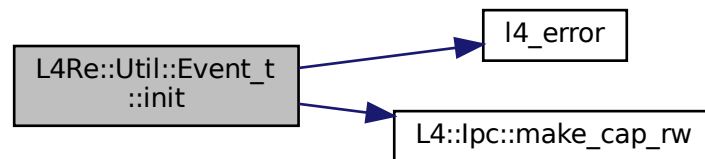
#### Return values

|            |                                              |
|------------|----------------------------------------------|
| 0          | Success                                      |
| -L4_ENOMEM | No memory to allocate required capabilities. |
| <0         | Other IPC errors.                            |

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

References [L4\\_ENOMEM](#), [l4\\_error\(\)](#), [L4::ipc::make\\_cap\\_rw\(\)](#), and [L4Re::Rm::F::Search\\_addr](#).

Here is the call graph for this function:



### 15.269.3.3 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. |

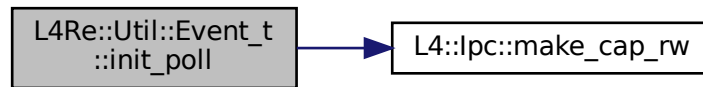
#### Return values

|            |                                              |
|------------|----------------------------------------------|
| 0          | Success                                      |
| -L4_ENOMEM | No memory to allocate required capabilities. |
| <0         | Other IPC errors.                            |

Definition at line 123 of file [event](#).

References [L4\\_ENOMEM](#), [L4::lpc::make\\_cap\\_rw\(\)](#), and [L4Re::Rm::F::Search\\_addr](#).

Here is the call graph for this function:



### 15.269.3.4 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:

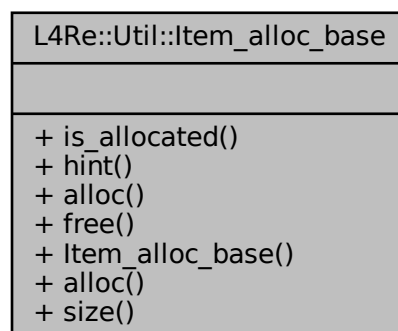
- [l4/re/util/event](#)

## 15.270 L4Re::Util::Item\_alloc\_base Class Reference

Item allocator.

Inherited by `L4Re::Util::Item_alloc< Bits >`.

Collaboration diagram for `L4Re::Util::Item_alloc_base`:



### 15.270.1 Detailed Description

Item allocator.

Definition at line 38 of file [item\\_alloc](#).

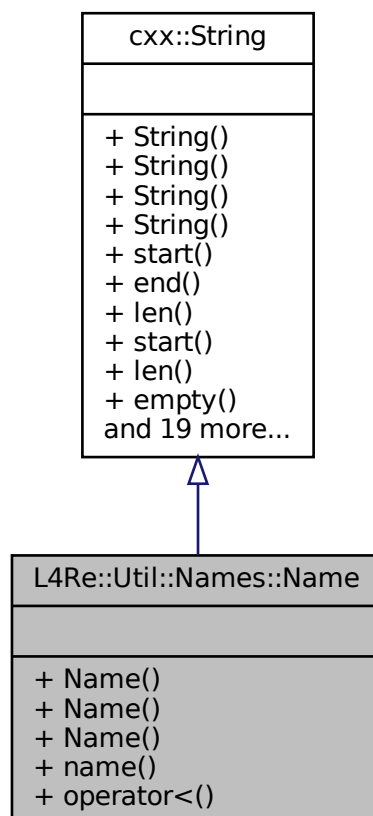
The documentation for this class was generated from the following file:

- [l4/re/util/item\\_alloc](#)

### 15.271 L4Re::Util::Names::Name Class Reference

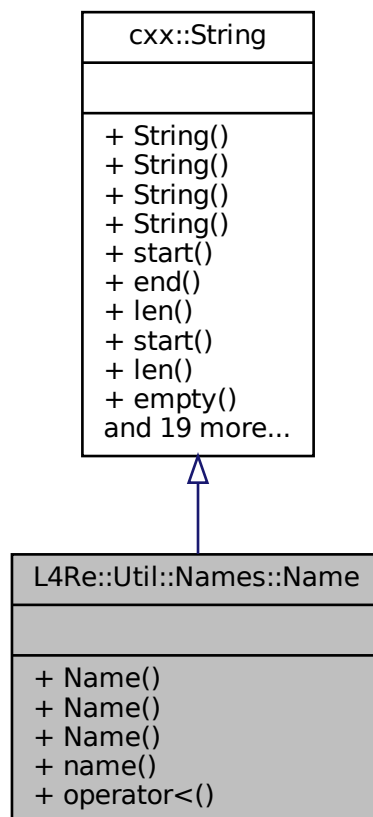
[Name](#) class.

Inheritance diagram for L4Re::Util::Names::Name:





Collaboration diagram for L4Re::Util::Names::Name:



## Additional Inherited Members

### 15.271.1 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`

## 15.272 L4Re::Util::Names::Name\_space Class Reference

Abstract server-side implementation of the `L4::Namespace` interface.

Collaboration diagram for L4Re::Util::Names::Name\_space:

| L4Re::Util::Names::<br>Name_space                                                                                                                                                                                                                                                                                                                     |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| # _dbg<br># _err                                                                                                                                                                                                                                                                                                                                      |
| + begin()<br>+ end()<br>+ Name_space()<br>+ ~Name_space()<br>+ find()<br>+ remove()<br>+ find_iter()<br>+ insert()<br>+ dump()<br>+ op_query()<br>+ op_register_obj()<br>+ op_unlink()<br># alloc_dynamic_entry()<br># free_dynamic_entry()<br># get_epiface()<br># copy_receive_cap()<br># free_capability()<br># free_epiface()<br># insert_entry() |

## 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](#)().*

### 15.272.1 Detailed Description

Abstract server-side implementation of the L4::Namespace interface.

Definition at line 184 of file [name\\_space\\_svr](#).

## 15.272.2 Member Function Documentation

### 15.272.2.1 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.

### 15.272.2.2 copy\_receive\_cap()

```
virtual int L4Re::Util::Names::Name_space::copy_receive_cap (
 L4::Cap< void > * cap) [protected], [pure virtual]
```

Return the receive capability for permanent use.

#### Parameters

|     |            |                                                                                                                                |
|-----|------------|--------------------------------------------------------------------------------------------------------------------------------|
| out | <i>cap</i> | Capability slot with the received capability. Must be permanently available until <a href="#">free_capability()</a> is called. |
|-----|------------|--------------------------------------------------------------------------------------------------------------------------------|

This method is called when a new entry is registered together with a capability mapping. It must decide whether and where to store the capability and return the final capability slot. Typical implementations return the capability slot in the receive window and allocate a new receive window.

### 15.272.2.3 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.

**15.272.2.4 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. |
|----------|----------------|

**15.272.2.5 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.

**15.272.2.6 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.

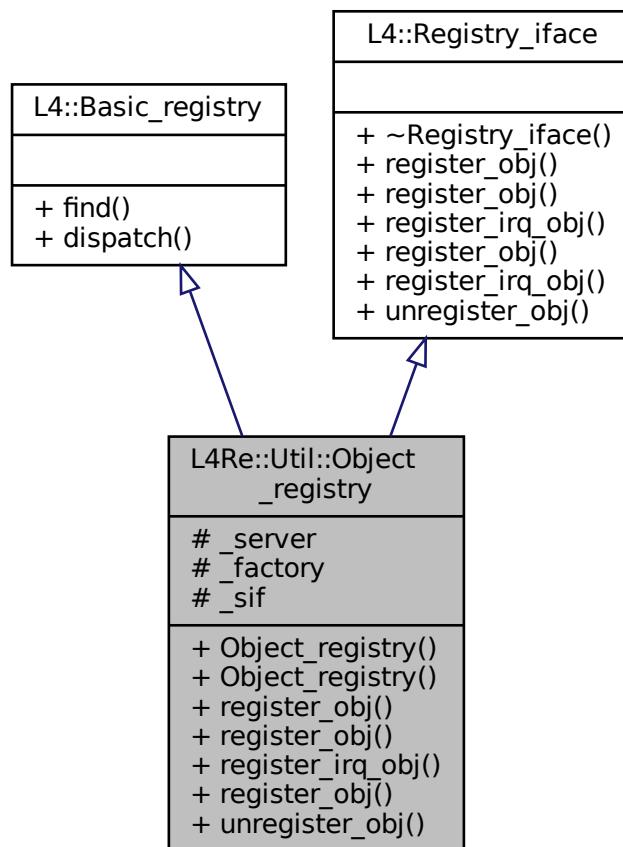
The documentation for this class was generated from the following file:

- `l4/re/util/name_space_svr`

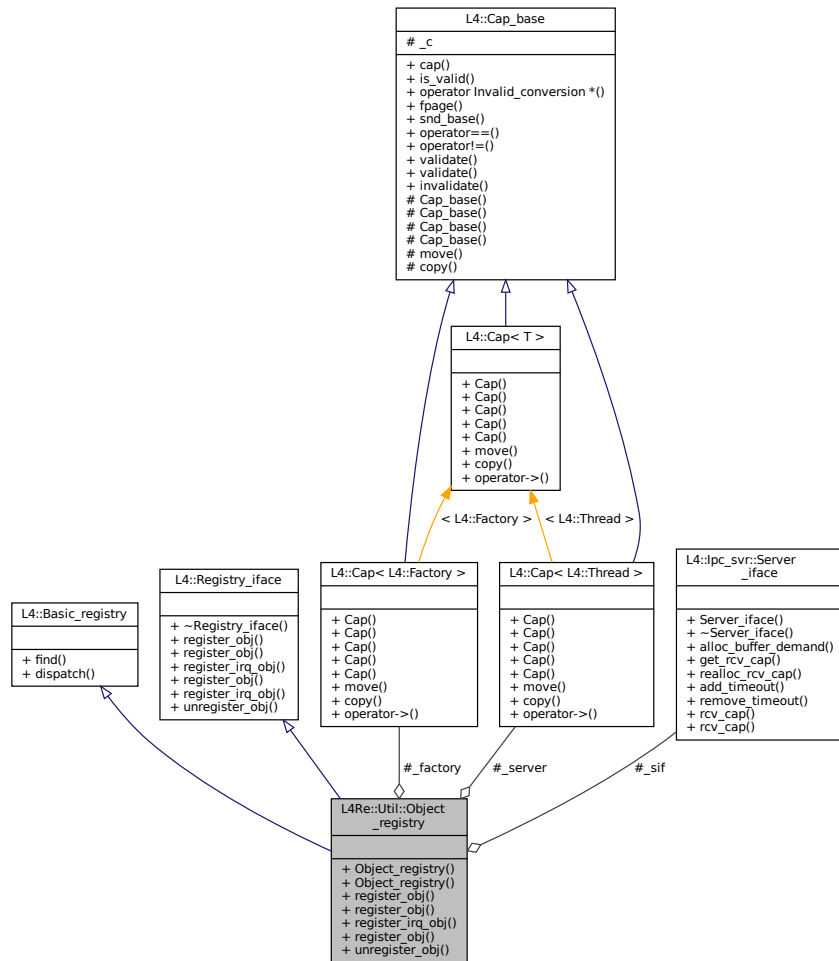
## 15.273 L4Re::Util::Object\_registry Class Reference

A registry that manages server objects and their attached IPC gates for a single server loop for a specific thread.

Inheritance diagram for L4Re::Util::Object\_registry:



Collaboration diagram for L4Re::Util::Object\_registry:



## Public Member Functions

- [Object\\_registry](#) ([L4::ipc\\_srvr::Server\\_iface](#) \*sif)  
*Create a registry for the main thread of the task using the default factory.*
- [Object\\_registry](#) ([L4::ipc\\_srvr::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

### 15.273.1 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](#).

### 15.273.2 Constructor & Destructor Documentation

#### 15.273.2.1 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](#).

#### 15.273.2.2 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](#).

### 15.273.3 Member Function Documentation

#### 15.273.3.1 register\_irq\_obj()

```
L4::Cap<L4::Irq> L4Re::Util::Object_registry::register_irq_obj (
 L4::Epiface * o) [inline], [override], [virtual]
```

Register a handler for an interrupt.

##### Parameters

|          |                                  |
|----------|----------------------------------|
| <i>o</i> | Server object that handles IRQs. |
|----------|----------------------------------|

##### Return values

|                                        |                                            |
|----------------------------------------|--------------------------------------------|
| <i>L4::Cap&lt;L4::Irq&gt;</i>          | Capability to a new IRQ object on success. |
| <i>L4::Cap&lt;L4::Irq&gt;::Invalid</i> | The allocation of the IRQ has failed.      |

The IRQ will be newly allocated using the registry's factory object. The caller must call [unregister\\_obj\(\)](#) to free all resources.

Implements [L4::Registry\\_iface](#).

##### Examples

[examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#).

Definition at line 238 of file [object\\_registry](#).

#### 15.273.3.2 register\_obj() [1/3]

```
L4::Cap<void> L4Re::Util::Object_registry::register_obj (
 L4::Epiface * o) [inline], [override], [virtual]
```

Register a new server object on a newly allocated capability.

##### Parameters

|          |                                          |
|----------|------------------------------------------|
| <i>o</i> | Server object that handles IPC requests. |
|----------|------------------------------------------|



## Return values

|                                     |                                            |
|-------------------------------------|--------------------------------------------|
| <i>L4::Cap&lt;void&gt;</i>          | A valid capability to a new IPC gate.      |
| <i>L4::Cap&lt;void&gt;::Invalid</i> | The allocation of the IPC gate has failed. |

The IPC gate will be allocated using the registry's factory. The caller must call [unregister\\_obj\(\)](#) to free all resources.

Implements [L4::Registry\\_iface](#).

Definition at line 222 of file [object\\_registry](#).

**15.273.3.3 register\_obj()** [2/3]

```
L4::Cap<void> L4Re::Util::Object_registry::register_obj (
 L4::Epiface * o,
 char const * service) [inline], [override], [virtual]
```

Register a new server object to a pre-allocated receive endpoint.

## Parameters

|                |                                           |
|----------------|-------------------------------------------|
| <i>o</i>       | Server object that handles IPC requests.  |
| <i>service</i> | Name of a pre-allocated receive endpoint. |

## Return values

|                                     |                                                    |
|-------------------------------------|----------------------------------------------------|
| <i>L4::Cap&lt;void&gt;</i>          | The capability known as <i>service</i> on success. |
| <i>L4::Cap&lt;void&gt;::Invalid</i> | No capability with the given name found.           |

The interface must be freed with [unregister\\_obj\(\)](#) by the caller to unbind the thread from the capability.

Implements [L4::Registry\\_iface](#).

## Examples

[examples/clntsrv/server.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#), and [examples/libs/l4re/streammap/server.cc](#).

Definition at line 205 of file [object\\_registry](#).

**15.273.3.4 register\_obj()** [3/3]

```
L4::Cap<L4::Rcv_endpoint> L4Re::Util::Object_registry::register_obj (
 L4::Epiface * o,
 L4::Cap< L4::Rcv_endpoint > ep) [inline], [override], [virtual]
```

Register a handler for an already existing interrupt.

## Parameters

|           |                                                                                           |
|-----------|-------------------------------------------------------------------------------------------|
| <i>o</i>  | Server object that handles the IPC.                                                       |
| <i>ep</i> | Capability to a receive endpoint, may be a hardware or software interrupt or an IPC gate. |

## Return values

|                                                 |                                      |
|-------------------------------------------------|--------------------------------------|
| <i>L4::Cap&lt;L4::Rcv_endpoint&gt;</i>          | Capability <i>ep</i> on success.     |
| <i>L4::Cap&lt;L4::Rcv_endpoint&gt;::Invalid</i> | The IRQ attach operation has failed. |

The interface must be freed with [unregister\\_obj\(\)](#) by the caller to unbind the thread from the capability.

Implements [L4::Registry\\_iface](#).

Definition at line 260 of file [object\\_registry](#).

### 15.273.3.5 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. |

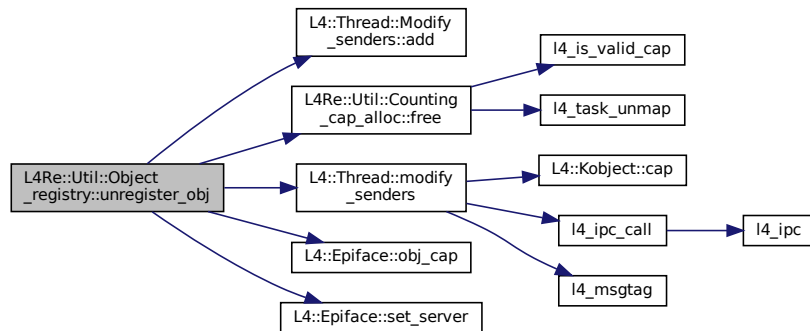
The capability used by the server object will be unmapped if *unmap* is true.

Implements [L4::Registry\\_iface](#).

Definition at line 276 of file [object\\_registry](#).

References [L4::Thread::Modify\\_senders::add\(\)](#), [L4Re::Util::cap\\_alloc](#), [L4Re::Util::Counting\\_cap\\_alloc< COUNTERTYPE >::free\(\)](#), [L4\\_FP\\_ALL\\_SPACES](#), [L4::Thread::modify\\_senders\(\)](#), [L4::Epiface::obj\\_cap\(\)](#), and [L4::Epiface::set\\_server\(\)](#).

Here is the call graph for this function:



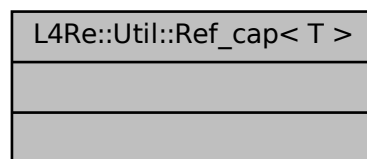
The documentation for this class was generated from the following file:

- l4/re/util/object\_registry

## 15.274 L4Re::Util::Ref\_cap< T > Struct Template Reference

Automatic capability that implements automatic free and unmap of the capability selector.

Collaboration diagram for L4Re::Util::Ref\_cap< T >:



### 15.274.1 Detailed Description

```
template<typename T>
struct L4Re::Util::Ref_cap< T >
```

Automatic capability that implements automatic free and unmap of the capability selector.

#### Template Parameters

|          |                                                        |
|----------|--------------------------------------------------------|
| <i>T</i> | Type of the object that is referred by the capability. |
|----------|--------------------------------------------------------|

This kind of automatic capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero.

#### Usage:

```
L4Re::Util::Ref_cap<L4Re::Dataspace>::Cap global_ds_cap;
{
 L4Re::Util::Ref_cap<L4Re::Dataspace>::Cap
 ds_cap(L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>());
 // reference count for the allocated cap selector is now 1
 // use the dataspace cap
 L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));
 global_ds_cap = ds_cap;
 // reference count is now 2
 ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
```

Definition at line 165 of file [cap\\_alloc](#).

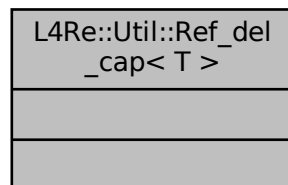
The documentation for this struct was generated from the following file:

- [l4/re/util/cap\\_alloc](#)

## 15.275 L4Re::Util::Ref\_del\_cap< T > Struct Template Reference

Automatic capability that implements automatic free and unmap+delete of the capability selector.

Collaboration diagram for L4Re::Util::Ref\_del\_cap< T >:



### 15.275.1 Detailed Description

```
template<typename T>
struct L4Re::Util::Ref_del_cap< T >
```

Automatic capability that implements automatic free and unmap+delete of the capability selector.

#### Template Parameters

|          |                                                        |
|----------|--------------------------------------------------------|
| <i>T</i> | Type of the object that is referred by the capability. |
|----------|--------------------------------------------------------|

This kind of automatic capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero. The main difference to [Ref\\_cap](#) is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

#### Usage:

```
L4Re::Util::Ref_del_cap<L4Re::Dataspace>::Cap global_ds_cap;
{
 L4Re::Util::Ref_del_cap<L4Re::Dataspace>::Cap
 ds_cap(L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>());
 // reference count for the allocated cap selector is now 1
 // use the dataspace cap
 L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));
 global_ds_cap = ds_cap;
 // reference count is now 2
 ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
...
global_ds_cap = L4_INVALID_CAP;
// reference count dropped to 0 (data space shall be deleted).
```

Definition at line 206 of file [cap\\_alloc](#).

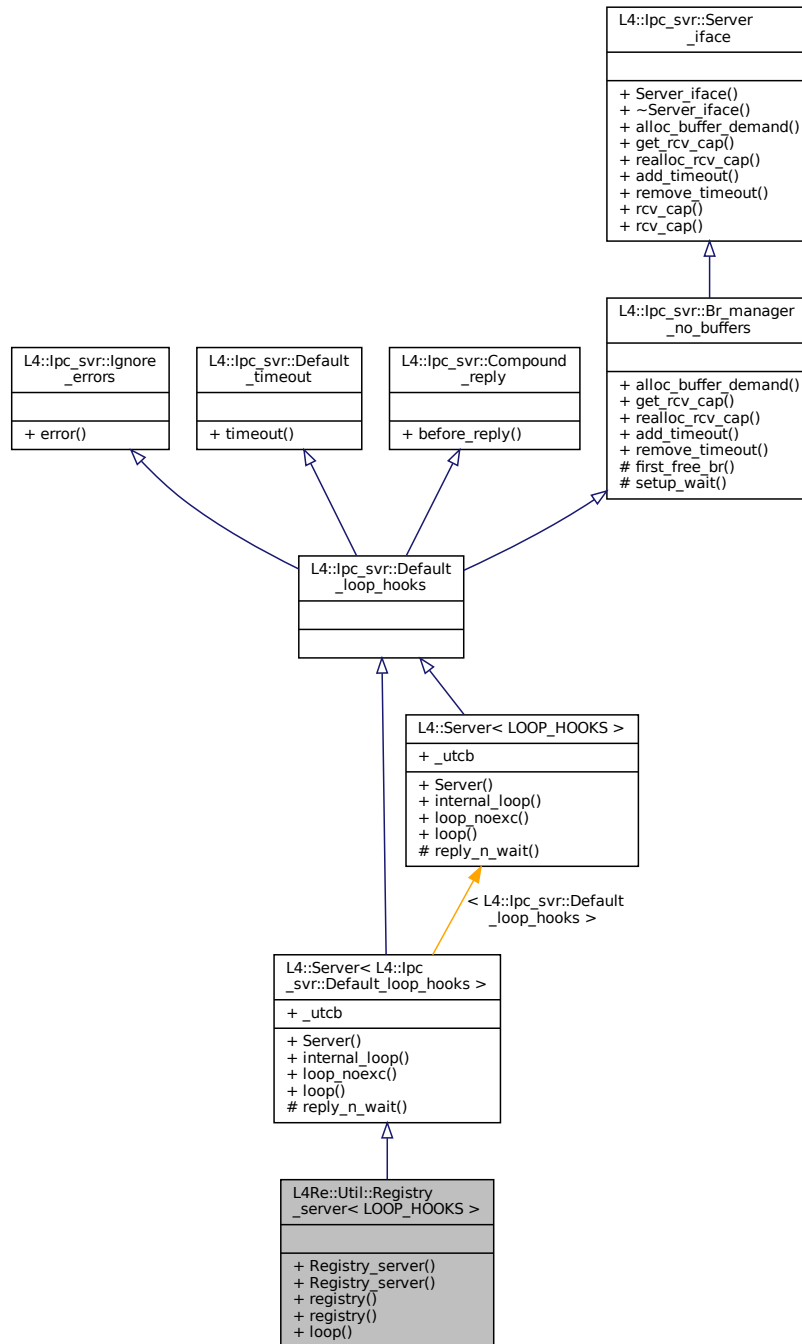
The documentation for this struct was generated from the following file:

- [l4/re/util/cap\\_alloc](#)

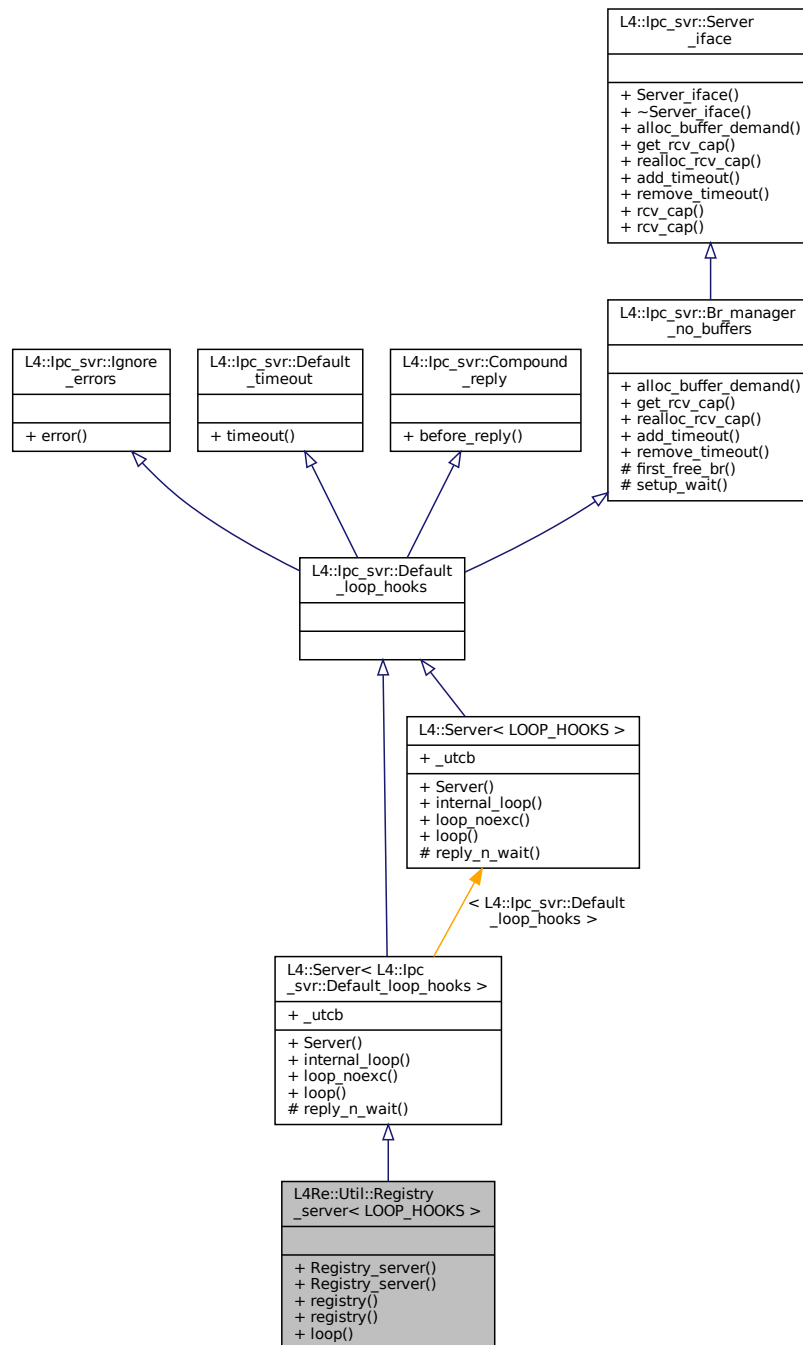
## 15.276 L4Re::Util::Registry\_server< LOOP\_HOOKS > Class Template Reference

A server loop object which has a [Object\\_registry](#) included.

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 \(l4\\_utcb\\_t \\*utcb, L4::Cap< L4::Thread > server, L4::Cap< L4::Factory > factory\)](#)  
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 ()`  
*Start the server loop.*

## Additional Inherited Members

### 15.276.1 Detailed Description

```
template<typename LOOP_HOOKS = L4::ipc_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`.

### 15.276.2 Constructor & Destructor Documentation

#### 15.276.2.1 Registry\_server() [1/2]

```
template<typename LOOP_HOOKS = L4::ipc_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`.

#### 15.276.2.2 Registry\_server() [2/2]

```
template<typename LOOP_HOOKS = L4::ipc_svr::Default_loop_hooks>
L4Re::Util::Registry_server< LOOP_HOOKS >::Registry_server (
 l4_utcb_t * utcb,
 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>utcb</i>    | The UTCB of the thread running the server loop.            |
| <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 328 of file [object\\_registry](#).

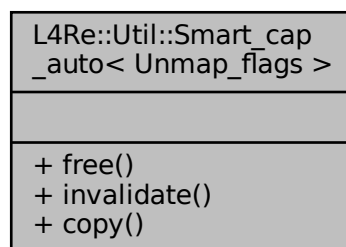
The documentation for this class was generated from the following file:

- [l4/re/util/object\\_registry](#)

## 15.277 L4Re::Util::Smart\_cap\_auto< Unmap\_flags > Class Template Reference

Helper for Unique\_cap and Unique\_del\_cap.

Collaboration diagram for L4Re::Util::Smart\_cap\_auto< Unmap\_flags >:



### 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](#).*

### 15.277.1 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](#).

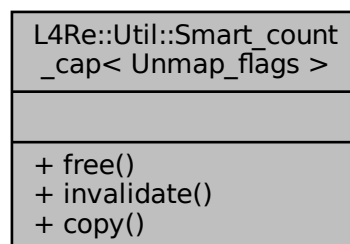
The documentation for this class was generated from the following file:

- [l4/re/util/cap\\_alloc](#)

## 15.278 L4Re::Util::Smart\_count\_cap< Unmap\_flags > Class Template Reference

Helper for [Ref\\_cap](#) and [Ref\\_del\\_cap](#).

Collaboration diagram for L4Re::Util::Smart\_count\_cap< Unmap\_flags >:



### 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).*

### 15.278.1 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](#).

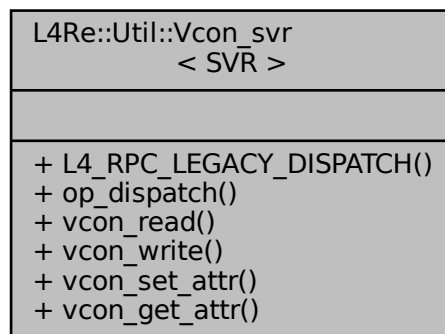
The documentation for this class was generated from the following file:

- [l4/re/util/cap\\_alloc](#)

## 15.279 L4Re::Util::Vcon\_svr< SVR > Class Template Reference

[Console](#) server template class.

Collaboration diagram for L4Re::Util::Vcon\_svr< SVR >:



### 15.279.1 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](#).

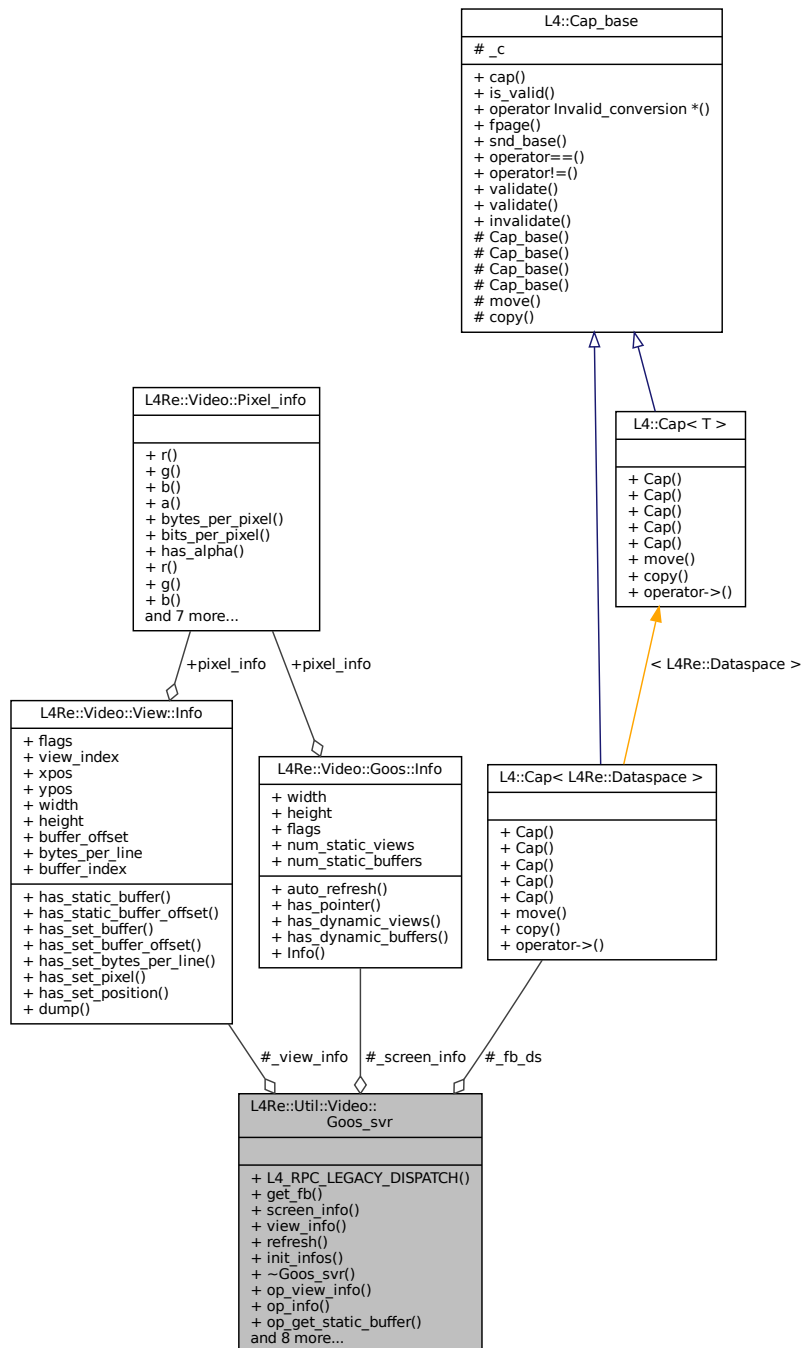
The documentation for this class was generated from the following file:

- [l4/re/util/vcon\\_svr](#)

## 15.280 L4Re::Util::Video::Goos\_svr Class Reference

Goos server class.

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.*

### 15.280.1 Detailed Description

Goos server class.

Definition at line 36 of file [goos\\_svr](#).

### 15.280.2 Member Function Documentation

#### 15.280.2.1 [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](#).

### 15.280.2.2 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](#).

### 15.280.2.3 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](#).

### 15.280.2.4 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](#).

#### 15.280.2.5 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](#).

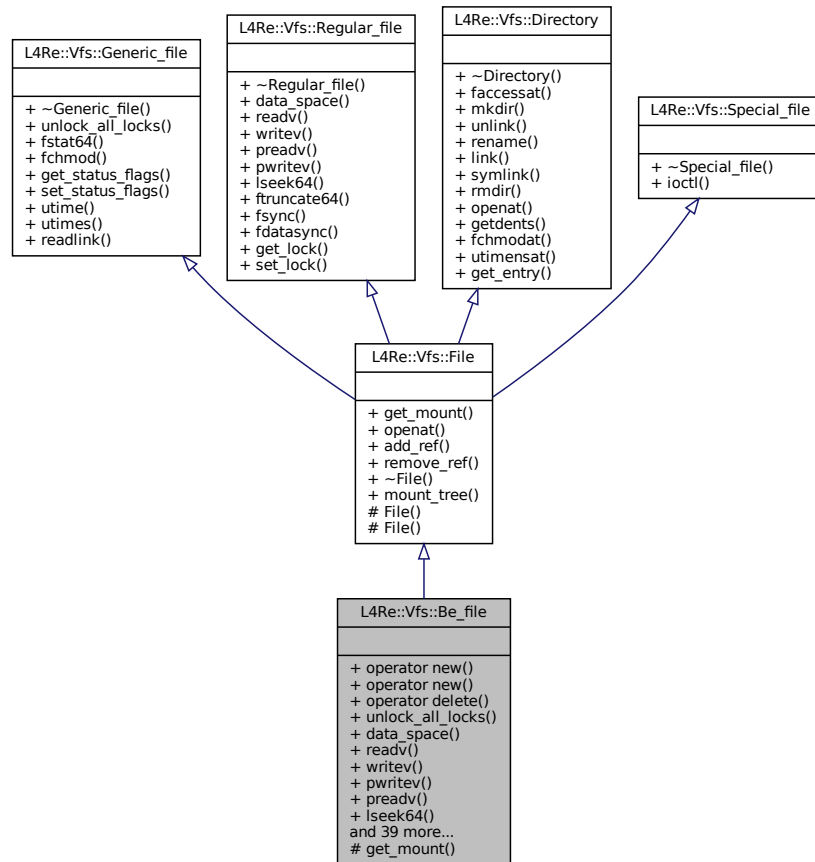
The documentation for this class was generated from the following file:

- [l4/re/util/video/goos\\_svr](#)

## 15.281 L4Re::Vfs::Be\_file Class Reference

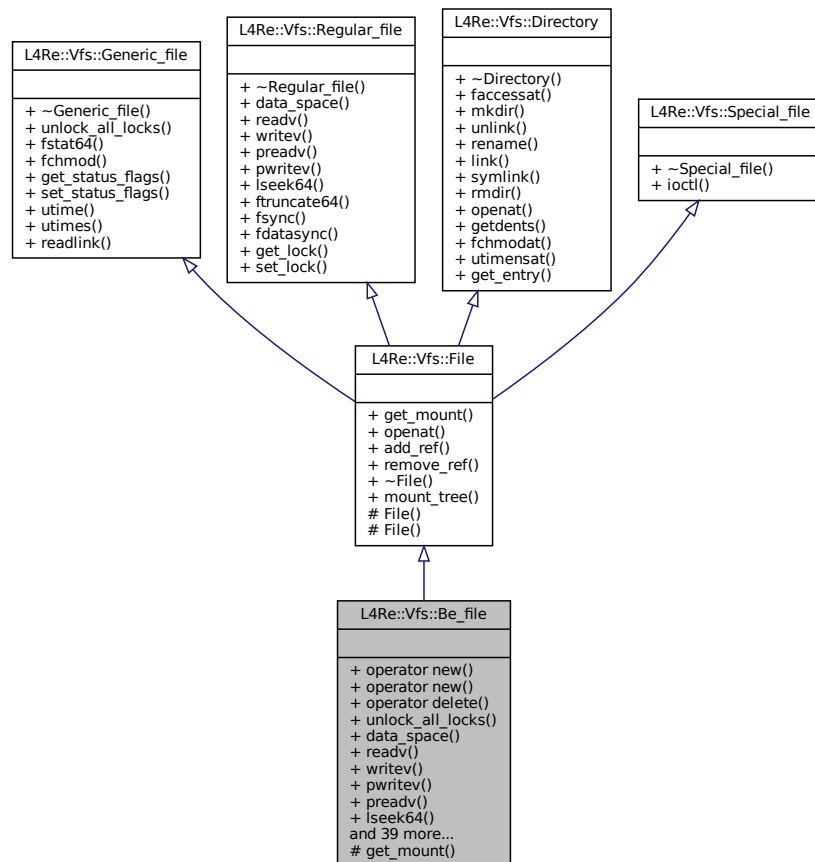
Boiler plate class for implementing an open file for [L4Re::Vfs](#).

Inheritance diagram for L4Re::Vfs::Be\_file:





Collaboration diagram for L4Re::Vfs::Be\_file:



## Public Member Functions

- `int unlock_all_locks () throw ()`  
*Unlock all locks on the file.*
- `L4::Cap< L4Re::Dataspace > data_space () const throw ()`  
*Get an [L4Re::Dataspace](#) object for the file.*
- `ssize_t readv (const struct iovec *, int) throw ()`  
*Default backend for POSIX read and readv functions.*
- `ssize_t writev (const struct iovec *, int) throw ()`  
*Default backend for POSIX write and writev functions.*
- `ssize_t pwritev (const struct iovec *, int, off64_t) throw ()`  
*Default backend for POSIX pwrite and pwritev functions.*
- `ssize_t preadv (const struct iovec *, int, off64_t) throw ()`  
*Default backend for POSIX pread and preadv functions.*
- `off64_t lseek64 (off64_t, int) throw ()`  
*Default backend for POSIX seek and lseek functions.*
- `int ftruncate64 (off64_t) throw ()`  
*Default backend for the POSIX truncate, ftruncate and similar functions.*
- `int fsync () const throw ()`  
*Default backend for POSIX fsync.*

- int [fdatasync](#) () const throw ()  
*Default backend for POSIX fdatasync.*
- int [ioctl](#) (unsigned long, va\_list) throw ()  
*Default backend for POSIX ioctl.*
- int [fstat64](#) (struct stat64 \*) const throw ()  
*Get status information for the file.*
- int [fchmod](#) (mode\_t) throw ()  
*Default backend for POSIX chmod and fchmod.*
- int [get\\_status\\_flags](#) () const throw ()  
*Default backend for POSIX fcntl subfunctions.*
- int [set\\_status\\_flags](#) (long) throw ()  
*Default backend for POSIX fcntl subfunctions.*
- int [get\\_lock](#) (struct flock64 \*) throw ()  
*Default backend for POSIX fcntl subfunctions.*
- int [set\\_lock](#) (struct flock64 \*, bool) throw ()  
*Default backend for POSIX fcntl subfunctions.*
- int [faccessat](#) (const char \*, int, int) throw ()  
*Default backend for POSIX access and faccessat functions.*
- int [fchmodat](#) (const char \*, mode\_t, int) throw ()  
*Default backend for POSIX fchmodat function.*
- int [utime](#) (const struct utimbuf \*) throw ()  
*Default backend for POSIX utime.*
- int [utimes](#) (const struct timeval[2]) throw ()  
*Default backend for POSIX utimes.*
- int [utimensat](#) (const char \*, const struct timespec[2], int) throw ()  
*Default backend for POSIX utimensat.*
- int [mkdir](#) (const char \*, mode\_t) throw ()  
*Default backend for POSIX mkdir and mkdirat.*
- int [unlink](#) (const char \*) throw ()  
*Default backend for POSIX unlink, unlinkat.*
- int [rename](#) (const char \*, const char \*) throw ()  
*Default backend for POSIX rename, renameat.*
- int [link](#) (const char \*, const char \*) throw ()  
*Default backend for POSIX link, linkat.*
- int [symlink](#) (const char \*, const char \*) throw ()  
*Default backend for POSIX symlink, symlinkat.*
- int [rmdir](#) (const char \*) throw ()  
*Default backend for POSIX rmdir, rmdirat.*
- ssize\_t [readlink](#) (char \*, size\_t)  
*Default backend for POSIX readlink, readlinkat.*

### 15.281.1 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.

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

## 15.281.2 Member Function Documentation

### 15.281.2.1 data\_space()

```
L4::Cap<L4Re::Dataspace> L4Re::Vfs::Be_file::data_space () const throw () [inline], [virtual]
```

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 57 of file [backend](#).

### 15.281.2.2 fstat64()

```
int L4Re::Vfs::Be_file::fstat64 (
 struct stat64 * buf) const throw () [inline], [virtual]
```

Get status information for the file.

This is the backend for POSIX fstat, stat, fstat64 and friends.

#### Parameters

|            |            |                                                    |
|------------|------------|----------------------------------------------------|
| <i>out</i> | <i>buf</i> | This buffer is filled with the status information. |
|------------|------------|----------------------------------------------------|

#### Returns

0 on success, or <0 on error.

Implements [L4Re::Vfs::Generic\\_file](#).

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

### 15.281.2.3 unlock\_all\_locks()

```
int L4Re::Vfs::Be_file::unlock_all_locks () throw () [inline], [virtual]
```

Unlock all locks on the file.

#### Note

All locks means all locks independent by which file the locks were taken.

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 53 of file [backend](#).

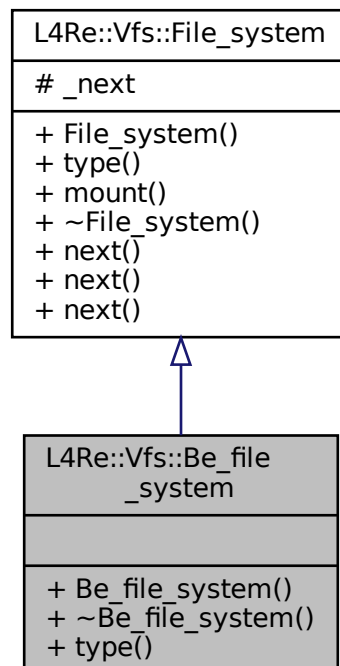
The documentation for this class was generated from the following file:

- l4/l4re\_vfs/backend

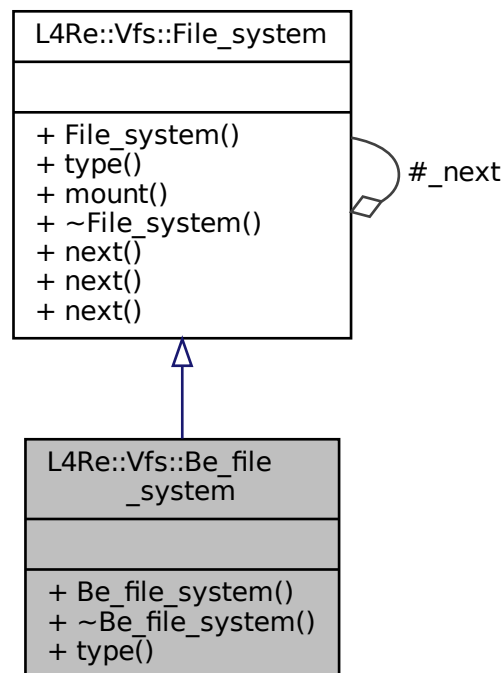
## 15.282 L4Re::Vfs::Be\_file\_system Class Reference

Boilerplate class for implementing a [L4Re::Vfs::File\\_system](#).

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) throw ()  
*Create a file-system object for the given fstype.*
- [~Be\\_file\\_system](#) () throw ()  
*Destroy a file-system object.*
- char const \* [type](#) () const throw ()  
*Return the file-system type.*

### 15.282.1 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.

Definition at line 308 of file [backend](#).

### 15.282.2 Constructor & Destructor Documentation

### 15.282.2.1 Be\_file\_system()

```
L4Re::Vfs::Be_file_system::Be_file_system (
 char const * fstype) throw () [inline], [explicit]
```

Create a file-system object for the given *fstype*.

#### Parameters

|               |                                                    |
|---------------|----------------------------------------------------|
| <i>fstype</i> | The type that <a href="#">type()</a> shall return. |
|---------------|----------------------------------------------------|

This constructor takes care of registering the file system in the registry of L4Re::Vfs::vfs\_ops.

Definition at line [322](#) of file [backend](#).

### 15.282.2.2 ~Be\_file\_system()

```
L4Re::Vfs::Be_file_system::~~Be_file_system () throw () [inline]
```

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](#).

## 15.282.3 Member Function Documentation

### 15.282.3.1 type()

```
char const* L4Re::Vfs::Be_file_system::type () const throw () [inline], [virtual]
```

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:

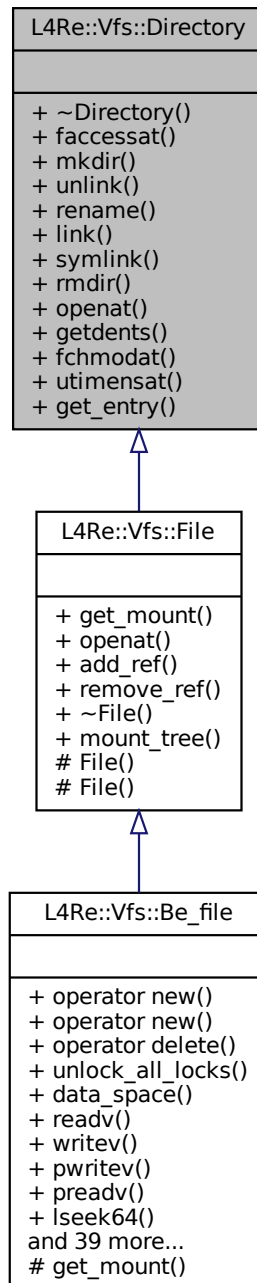
- [l4/l4re\\_vfs/backend](#)

## 15.283 L4Re::Vfs::Directory Class Reference

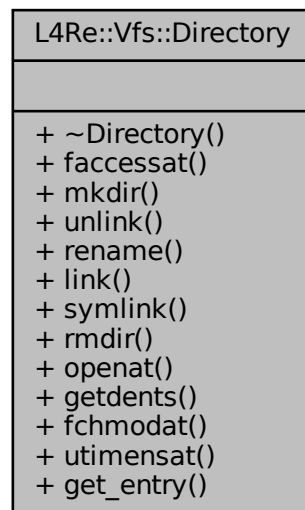
Interface for a POSIX file that is a directory.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Directory:



Collaboration diagram for L4Re::Vfs::Directory:



## Public Member Functions

- virtual int [faccessat](#) (const char \*path, int mode, int flags)=0 throw ()  
*Check access permissions on the given file.*
- virtual int [mkdir](#) (const char \*path, mode\_t mode)=0 throw ()  
*Create a new subdirectory.*
- virtual int [unlink](#) (const char \*path)=0 throw ()  
*Unlink the given file from that directory.*
- virtual int [rename](#) (const char \*src\_path, const char \*dst\_path)=0 throw ()  
*Rename the given file.*
- virtual int [link](#) (const char \*src\_path, const char \*dst\_path)=0 throw ()  
*Create a hard link (second name) for the given file.*
- virtual int [symlink](#) (const char \*src\_path, const char \*dst\_path)=0 throw ()  
*Create a symbolic link for the given file.*
- virtual int [rmdir](#) (const char \*path)=0 throw ()  
*Delete an empty directory.*

### 15.283.1 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 use always the combined [L4Re::Vfs::File](#) interface.

Definition at line 140 of file [vfs.h](#).



## 15.283.2 Member Function Documentation

### 15.283.2.1 faccessat()

```
virtual int L4Re::Vfs::Directory::faccessat (
 const char * path,
 int mode,
 int flags) throw () [pure virtual]
```

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](#).

### 15.283.2.2 link()

```
virtual int L4Re::Vfs::Directory::link (
 const char * src_path,
 const char * dst_path) throw () [pure virtual]
```

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 to 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](#).

### 15.283.2.3 mkdir()

```
virtual int L4Re::Vfs::Directory::mkdir (
 const char * path,
 mode_t mode) throw () [pure virtual]
```

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 is not a directory.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 15.283.2.4 rename()

```
virtual int L4Re::Vfs::Directory::rename (
 const char * src_path,
 const char * dst_path) throw () [pure virtual]
```

Rename the given file.

Backend for the POSIX rename, renameat functions.

#### Parameters

|                 |                                                                                                                         |
|-----------------|-------------------------------------------------------------------------------------------------------------------------|
| <i>src_path</i> | The old name to 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](#).

### 15.283.2.5 rmdir()

```
virtual int L4Re::Vfs::Directory::rmdir (
 const char * path) throw () [pure virtual]
```

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](#).

### 15.283.2.6 symlink()

```
virtual int L4Re::Vfs::Directory::symlink (
 const char * src_path,
 const char * dst_path) throw () [pure virtual]
```

Create a symbolic link for the given file.

Backend for the POSIX symlink and symlinkat functions.

#### Parameters

|                 |                                                                                                           |
|-----------------|-----------------------------------------------------------------------------------------------------------|
| <i>src_path</i> | The old name to 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](#).

### 15.283.2.7 unlink()

```
virtual int L4Re::Vfs::Directory::unlink (
 const char * path) throw () [pure virtual]
```

Unlink the given file from that directory.

Backend for the POSIX unlink and unlinkat functions.

## Parameters

|             |                                                                                                                 |
|-------------|-----------------------------------------------------------------------------------------------------------------|
| <i>path</i> | The name to 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:

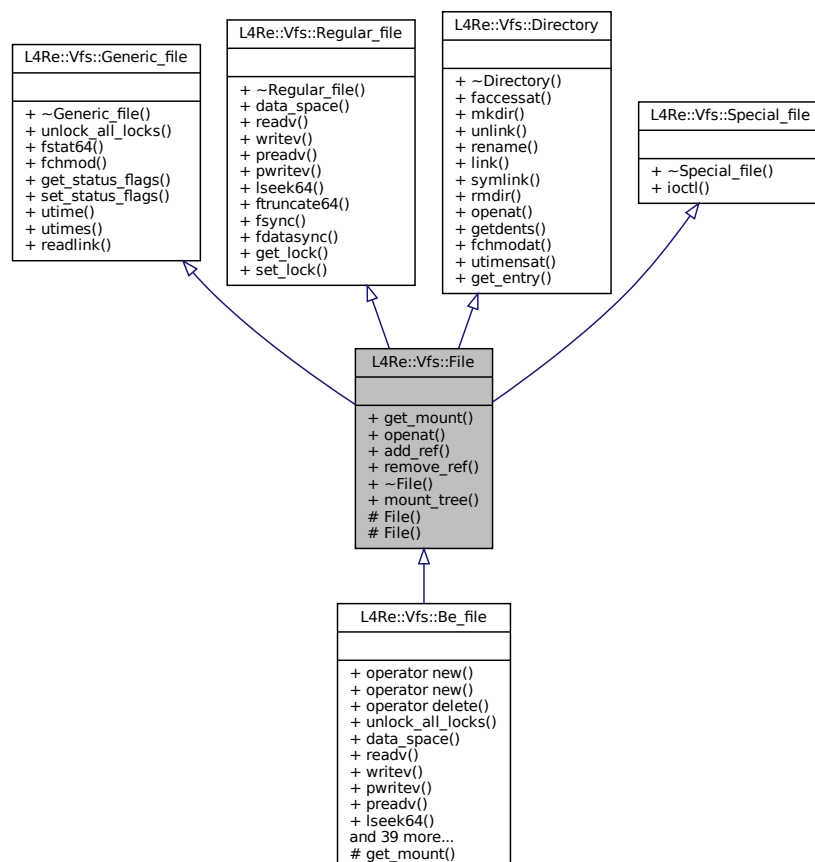
- l4/l4re\_vfs/vfs.h

## 15.284 L4Re::Vfs::File Class Reference

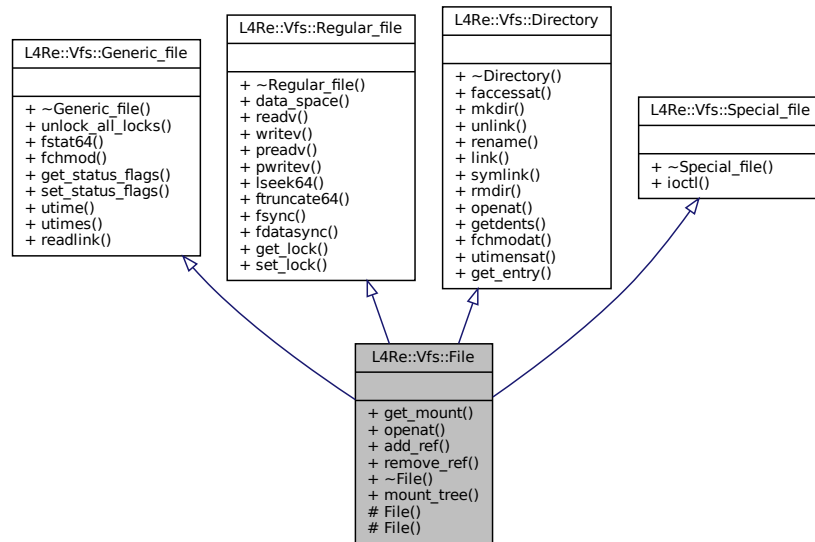
The basic interface for an open POSIX file.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::File:



Collaboration diagram for L4Re::Vfs::File:



## Additional Inherited Members

### 15.284.1 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 a 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](#) you may use [L4Re::Vfs::Be\\_file](#) as a base class.

Definition at line 435 of file [vfs.h](#).

The documentation for this class was generated from the following file:

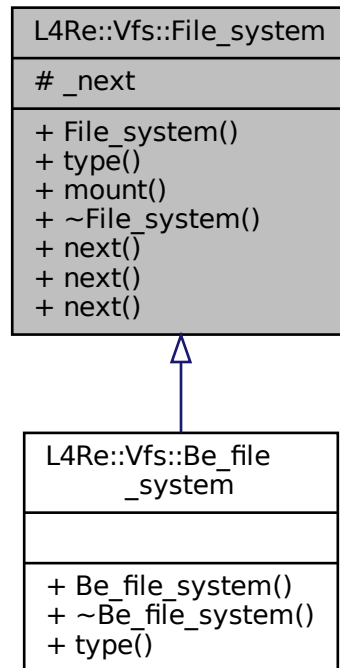
- [l4/l4re\\_vfs/vfs.h](#)

## 15.285 L4Re::Vfs::File\_system Class Reference

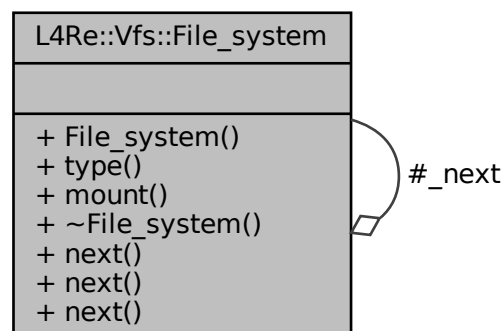
Basic interface for an [L4Re::Vfs](#) file system.

```
#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 =0 throw ()  
*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)=0 throw ()  
*Create a directory object dir representing source mounted with this file system.*

### 15.285.1 Detailed Description

Basic interface for an [L4Re::Vfs](#) file system.

#### Note

For implementing a special file system you may use [L4Re::Vfs::Be\\_file\\_system](#) as a base class.

The may purpose of this interface is that there is a single object for each supported file-system type (e.g., ext2, vfat) exists in your application and is registered at the [L4Re::Vfs::Fs](#) singleton available in via [L4Re::Vfs::vfs\\_ops](#). At the end the POSIX mount function call the [File\\_system::mount](#) method for the given file-system type given in mount.

Definition at line [829](#) of file [vfs.h](#).

### 15.285.2 Member Function Documentation

#### 15.285.2.1 mount()

```
virtual int L4Re::Vfs::File_system::mount (
 char const * source,
 unsigned long mountflags,
 void const * data,
 cxx::Ref_ptr< File > * dir) throw () [pure virtual]
```

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.               |

#### Return values

|            |                                                                     |
|------------|---------------------------------------------------------------------|
| <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:

**15.285.2.2 type()**

```
virtual char const* L4Re::Vfs::File_system::type () const throw () [pure virtual]
```

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

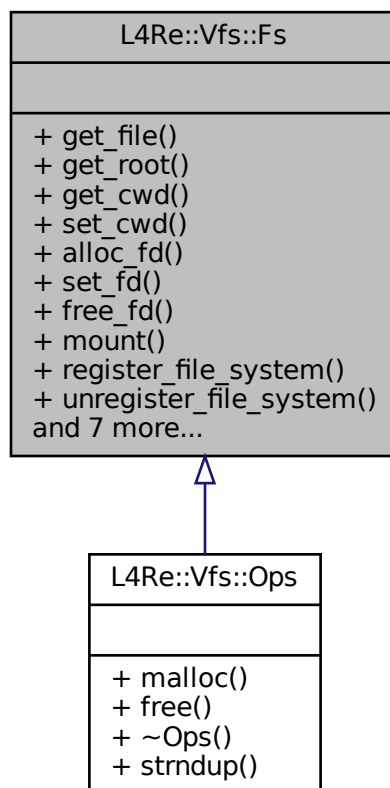
**15.286 L4Re::Vfs::Fs Class Reference**

POSIX File-system related functionality.

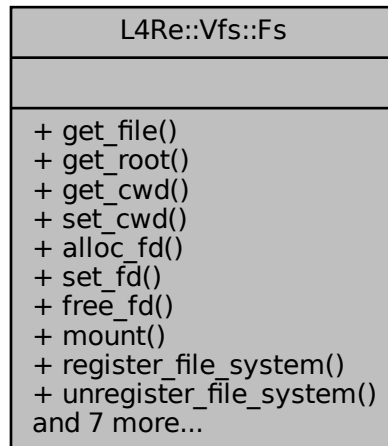
```
#include <vfs.h>
```



Inheritance diagram for L4Re::Vfs::Fs:



Collaboration diagram for L4Re::Vfs::Fs:



## Public Member Functions

- virtual [cxx::Ref\\_ptr< File >](#) [get\\_file](#) (int fd)=0 throw ()  
*Get the [L4Re::Vfs::File](#) for the file descriptor fd.*
- virtual [cxx::Ref\\_ptr< File >](#) [get\\_root](#) ()=0 throw ()  
*Get the directory object for the applications root directory.*
- virtual [cxx::Ref\\_ptr< File >](#) [get\\_cwd](#) () throw ()  
*Get the directory object for the applications current working directory.*
- virtual void [set\\_cwd](#) ([cxx::Ref\\_ptr< File >](#) const &) throw ()  
*Set the current working directory for the application.*
- virtual int [alloc\\_fd](#) ([cxx::Ref\\_ptr< File >](#) const &f=[cxx::Ref\\_ptr<>::Nil](#))=0 throw ()  
*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](#))=0 throw ()  
*Set the file object referenced by the file descriptor fd.*
- virtual [cxx::Ref\\_ptr< File >](#) [free\\_fd](#) (int fd)=0 throw ()  
*Free the file descriptor fd.*
- virtual int [mount](#) (char const \*path, [cxx::Ref\\_ptr< File >](#) const &dir)=0 throw ()  
*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) throw ()  
*Backend for the POSIX mount call.*

### 15.286.1 Detailed Description

POSIX File-system related functionality.

**Note**

This class usually exists as a singleton as a superclass of [L4Re::Vfs::Ops](#) (

**See also**

[L4Re::Vfs::vfs\\_ops](#)).

Definition at line [881](#) of file [vfs.h](#).

## 15.286.2 Member Function Documentation

### 15.286.2.1 alloc\_fd()

```
virtual int L4Re::Vfs::Fs::alloc_fd (
 cxx::Ref_ptr< File > const & f = cxx::Ref_ptr<>::Nil) throw () [pure virtual]
```

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.

### 15.286.2.2 free\_fd()

```
virtual cxx::Ref_ptr<File> L4Re::Vfs::Fs::free_fd (
 int fd) throw () [pure virtual]
```

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.

**15.286.2.3 get\_file()**

```
virtual cxx::Ref_ptr<File> L4Re::Vfs::Fs::get_file (
 int fd) throw () [pure virtual]
```

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.

**15.286.2.4 mount()**

```
virtual int L4Re::Vfs::Fs::mount (
 char const * path,
 cxx::Ref_ptr< File > const & dir) throw () [pure virtual]
```

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.

**15.286.2.5 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) throw () [pure virtual]
```

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:

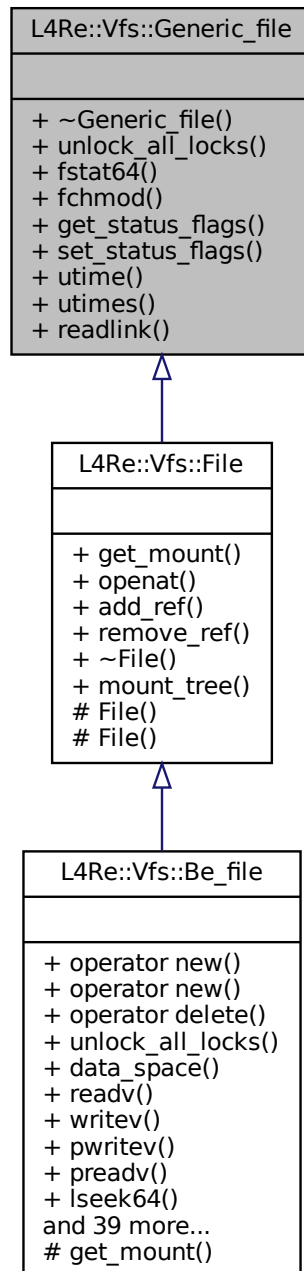
- `l4/l4re_vfs/vfs.h`

## 15.287 L4Re::Vfs::Generic\_file Class Reference

The common interface for an open POSIX file.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Generic\_file:



Collaboration diagram for L4Re::Vfs::Generic\_file:

| L4Re::Vfs::Generic_file                                                                                                                                                                                                                                             |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>+ ~Generic_file()</li> <li>+ unlock_all_locks()</li> <li>+ fstat64()</li> <li>+ fchmod()</li> <li>+ get_status_flags()</li> <li>+ set_status_flags()</li> <li>+ utime()</li> <li>+ utimes()</li> <li>+ readlink()</li> </ul> |

## Public Member Functions

- virtual int [unlock\\_all\\_locks](#) ()=0 throw ()  
*Unlock all locks on the file.*
- virtual int [fstat64](#) (struct stat64 \*buf) const =0 throw ()  
*Get status information for the file.*
- virtual int [fchmod](#) (mode\_t)=0 throw ()  
*Change POSIX access rights on that file.*
- virtual int [get\\_status\\_flags](#) () const =0 throw ()  
*Get file status flags (fcntl F\_GETFL).*
- virtual int [set\\_status\\_flags](#) (long flags)=0 throw ()  
*Set file status flags (fcntl F\_SETFL).*

### 15.287.1 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 mor information.

Definition at line 62 of file [vfs.h](#).

### 15.287.2 Member Function Documentation

**15.287.2.1 fchmod()**

```
virtual int L4Re::Vfs::Generic_file::fchmod (
 mode_t) throw () [pure virtual]
```

Change POSIX access rights on that file.

Backend for POSIX chmod and fchmod.

Implemented in [L4Re::Vfs::Be\\_file](#).

**15.287.2.2 fstat64()**

```
virtual int L4Re::Vfs::Generic_file::fstat64 (
 struct stat64 * buf) const throw () [pure virtual]
```

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](#).

**15.287.2.3 get\_status\_flags()**

```
virtual int L4Re::Vfs::Generic_file::get_status_flags () const throw () [pure virtual]
```

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](#).

**15.287.2.4 set\_status\_flags()**

```
virtual int L4Re::Vfs::Generic_file::set_status_flags (
 long flags) throw () [pure virtual]
```

Set file status flags (fcntl F\_SETFL).

This function is used by the fcntl implementation for the F\_SETFL command.



#### Parameters

|              |                                                                                                                                             |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| <i>flags</i> | 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](#).

### 15.287.2.5 unlock\_all\_locks()

```
virtual int L4Re::Vfs::Generic_file::unlock_all_locks () throw () [pure virtual]
```

Unlock all locks on the file.

#### Note

All locks means all locks independent by which file the locks were taken.

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:

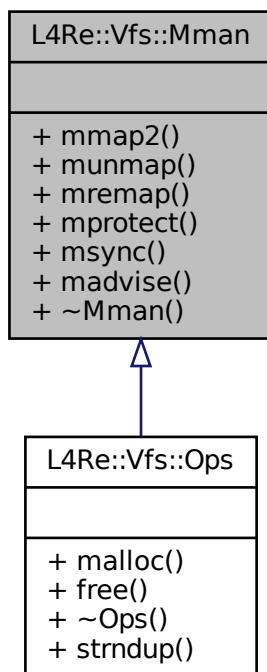
- l4/l4re\_vfs/vfs.h

## 15.288 L4Re::Vfs::Mman Class Reference

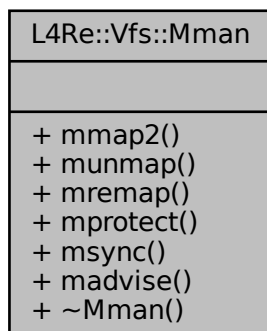
Interface for the POSIX memory management.

```
#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)=0 throw ()  
*Backend for the mmap2 system call.*
- virtual int [munmap](#) (void \*start, size\_t len)=0 throw ()  
*Backend for the munmap system call.*
- virtual int [mremap](#) (void \*old, size\_t old\_sz, size\_t new\_sz, int flags, void \*\*new\_addr)=0 throw ()  
*Backend for the mremap system call.*
- virtual int [mprotect](#) (const void \*a, size\_t sz, int prot)=0 throw ()  
*Backend for the mprotect system call.*
- virtual int [msync](#) (void \*addr, size\_t len, int flags)=0 throw ()  
*Backend for the msync system call.*
- virtual int [madvise](#) (void \*addr, size\_t len, int advice)=0 throw ()  
*Backend for the madvise system call.*

### 15.288.1 Detailed Description

Interface for the POSIX memory management.

#### Note

This interface exists usually as a singleton as 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 [745](#) of file [vfs.h](#).

The documentation for this class was generated from the following file:

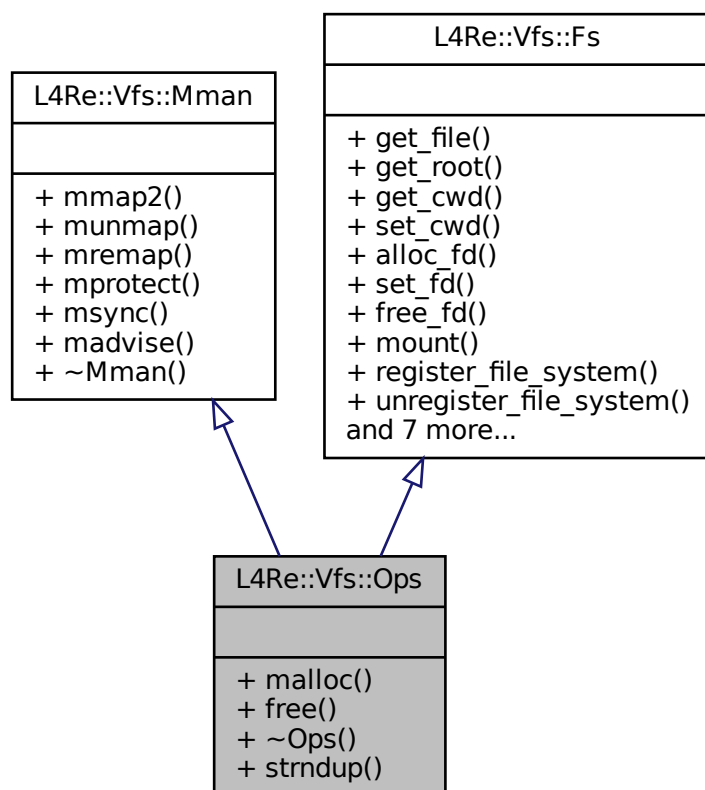
- l4/l4re\_vfs/vfs.h

## 15.289 L4Re::Vfs::Ops Class Reference

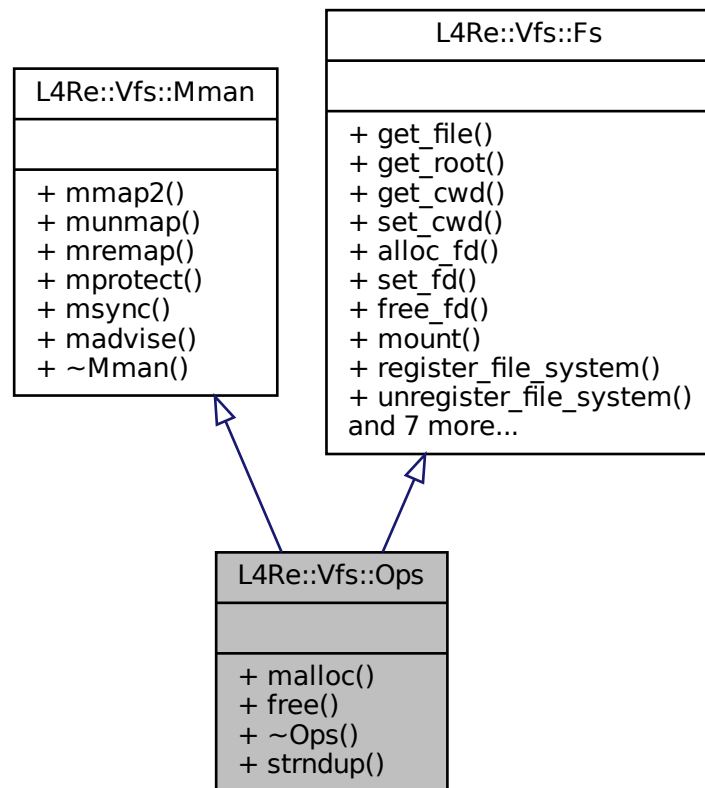
Interface for the POSIX backends for an application.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Ops:



Collaboration diagram for L4Re::Vfs::Ops:



## Additional Inherited Members

### 15.289.1 Detailed Description

Interface for the POSIX backends for 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:

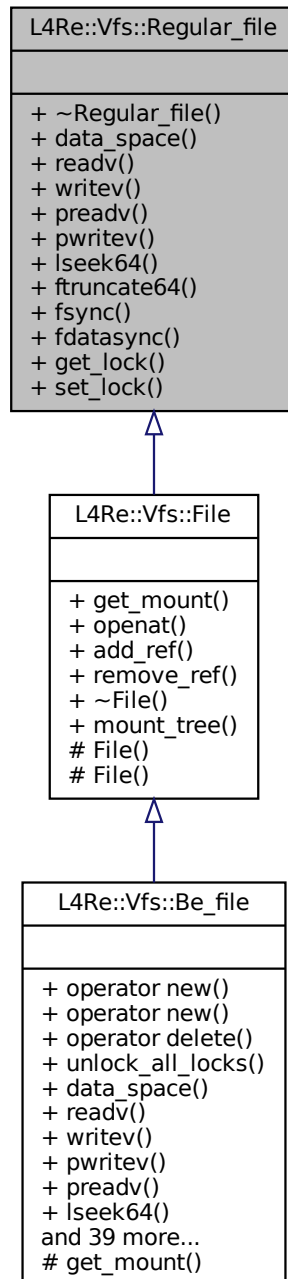
- `l4/l4re_vfs/vfs.h`

## 15.290 L4Re::Vfs::Regular\_file Class Reference

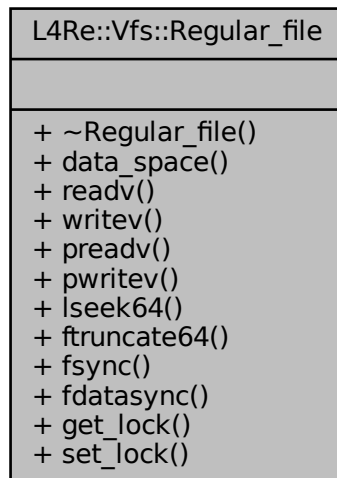
Interface for a POSIX file that provides regular file semantics.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Regular\_file:



Collaboration diagram for L4Re::Vfs::Regular\_file:



## Public Member Functions

- virtual [L4::Cap](#)< [L4Re::Dataspace](#) > [data\\_space](#) () const =0 throw ()  
*Get an [L4Re::Dataspace](#) object for the file.*
- virtual ssize\_t [readv](#) (const struct iovec \*, int iovcnt)=0 throw ()  
*Read one or more blocks of data from the file.*
- virtual ssize\_t [writev](#) (const struct iovec \*, int iovcnt)=0 throw ()  
*Write one or more blocks of data to the file.*
- virtual off64\_t [lseek64](#) (off64\_t, int)=0 throw ()  
*Change the file pointer.*
- virtual int [ftruncate64](#) (off64\_t pos)=0 throw ()  
*Truncate the file at the given position.*
- virtual int [fsync](#) () const =0 throw ()  
*Sync the data and meta data to persistent storage.*
- virtual int [fdatasync](#) () const =0 throw ()  
*Sync the data to persistent storage.*
- virtual int [get\\_lock](#) (struct flock64 \*lock)=0 throw ()  
*Test if the given lock can be placed in the file.*
- virtual int [set\\_lock](#) (struct flock64 \*lock, bool wait)=0 throw ()  
*Acquire or release the given lock on the file.*

### 15.290.1 Detailed Description

Interface for a POSIX file that provides regular file semantics.

Real objects use always the combined [L4Re::Vfs::File](#) interface.

Definition at line 267 of file [vfs.h](#).

## 15.290.2 Member Function Documentation

### 15.290.2.1 data\_space()

```
virtual L4::Cap<L4Re::Dataspace> L4Re::Vfs::Regular_file::data_space () const throw ()
[pure virtual]
```

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](#).

### 15.290.2.2 fdatsync()

```
virtual int L4Re::Vfs::Regular_file::fdatsync () const throw () [pure virtual]
```

Sync the data to persistent storage.

This is the backend for POSIX fdatsync.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 15.290.2.3 fsync()

```
virtual int L4Re::Vfs::Regular_file::fsync () const throw () [pure virtual]
```

Sync the data and meta data to persistent storage.

This is the backend for POSIX fsync.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 15.290.2.4 ftruncate64()

```
virtual int L4Re::Vfs::Regular_file::ftruncate64 (
 off64_t pos) throw () [pure virtual]
```

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](#).

**15.290.2.5 get\_lock()**

```
virtual int L4Re::Vfs::Regular_file::get_lock (
 struct flock64 * lock) throw () [pure virtual]
```

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](#).

**15.290.2.6 lseek64()**

```
virtual off64_t L4Re::Vfs::Regular_file::lseek64 (
 off64_t ,
 int) throw () [pure virtual]
```

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](#).

### 15.290.2.7 readv()

```
virtual ssize_t L4Re::Vfs::Regular_file::readv (
 const struct iovec * ,
 int iovcnt) throw () [pure virtual]
```

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 for the `f_pos` pointer of that open file. The file pointer is advanced according to the number of read bytes.

#### Returns

The number of bytes read from the file. or <0 on error-

Implemented in [L4Re::Vfs::Be\\_file](#).

### 15.290.2.8 set\_lock()

```
virtual int L4Re::Vfs::Regular_file::set_lock (
 struct flock64 * lock,
 bool wait) throw () [pure virtual]
```

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](#).

### 15.290.2.9 writev()

```
virtual ssize_t L4Re::Vfs::Regular_file::writev (
 const struct iovec * ,
 int iovcnt) throw () [pure virtual]
```

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:

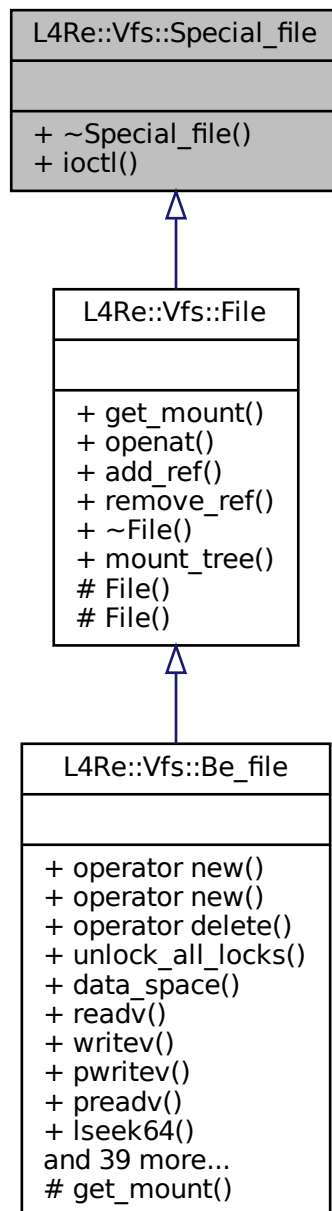
- l4/l4re\_vfs/vfs.h

## 15.291 L4Re::Vfs::Special\_file Class Reference

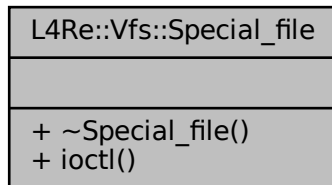
Interface for a POSIX file that provides special file semantics.

```
#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)=0 throw ()  
*The famous IO control.*

### 15.291.1 Detailed Description

Interface for a POSIX file that provides special file semantics.

Real objects use always the combined [L4Re::Vfs::File](#) interface.

Definition at line [400](#) of file [vfs.h](#).

### 15.291.2 Member Function Documentation

#### 15.291.2.1 ioctl()

```
virtual int L4Re::Vfs::Special_file::ioctl (
 unsigned long cmd,
 va_list args) throw () [pure virtual]
```

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**

$\geq 0$  on success, or  $< 0$  on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

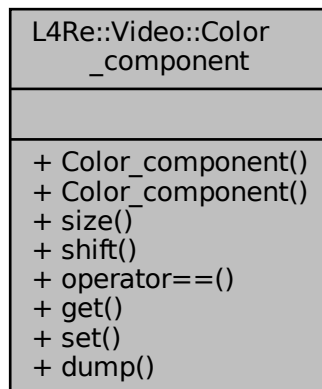
The documentation for this class was generated from the following file:

- [l4/l4re\\_vfs/vfs.h](#)

## 15.292 L4Re::Video::Color\_component Class Reference

A color component.

Collaboration diagram for L4Re::Video::Color\_component:



### 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.*

## 15.292.1 Detailed Description

A color component.

Definition at line 31 of file [colors](#).

## 15.292.2 Constructor & Destructor Documentation

### 15.292.2.1 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 46 of file [colors](#).

## 15.292.3 Member Function Documentation

### 15.292.3.1 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 93 of file [colors](#).

### 15.292.3.2 get()

```
int L4Re::Video::Color_component::get (
 unsigned long v) const [inline]
```

Get component from value (normalized to 16bits).

#### Parameters

|   |       |
|---|-------|
| v | Value |
|---|-------|

#### Returns

Converted value

Definition at line 73 of file [colors](#).

### 15.292.3.3 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 65 of file [colors](#).

### 15.292.3.4 set()

```
long unsigned L4Re::Video::Color_component::set (
 int v) const [inline]
```

Transform 16bit normalized value to the component in the color space.

#### Parameters

|   |                               |
|---|-------------------------------|
| v | Value return Converted value. |
|---|-------------------------------|

Definition at line 84 of file [colors](#).



### 15.292.3.5 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 59 of file [colors](#).

### 15.292.3.6 size()

```
unsigned char L4Re::Video::Color_component::size () const [inline]
```

Return the number of bits used by the component.

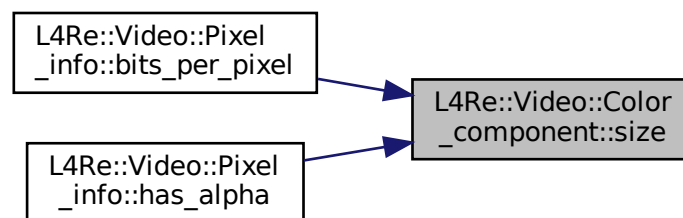
#### Returns

Number of bits used by the component

Definition at line 53 of file [colors](#).

Referenced by [L4Re::Video::Pixel\\_info::bits\\_per\\_pixel\(\)](#), and [L4Re::Video::Pixel\\_info::has\\_alpha\(\)](#).

Here is the caller graph for this function:



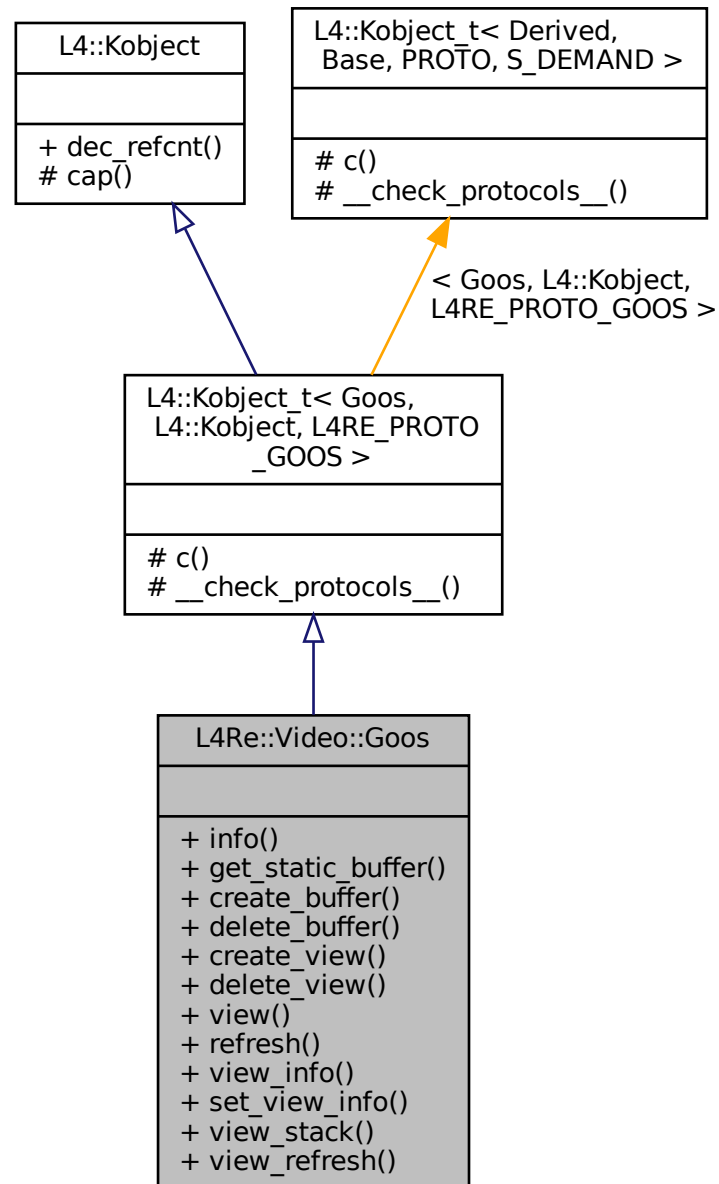
The documentation for this class was generated from the following file:

- `I4/re/video/colors`

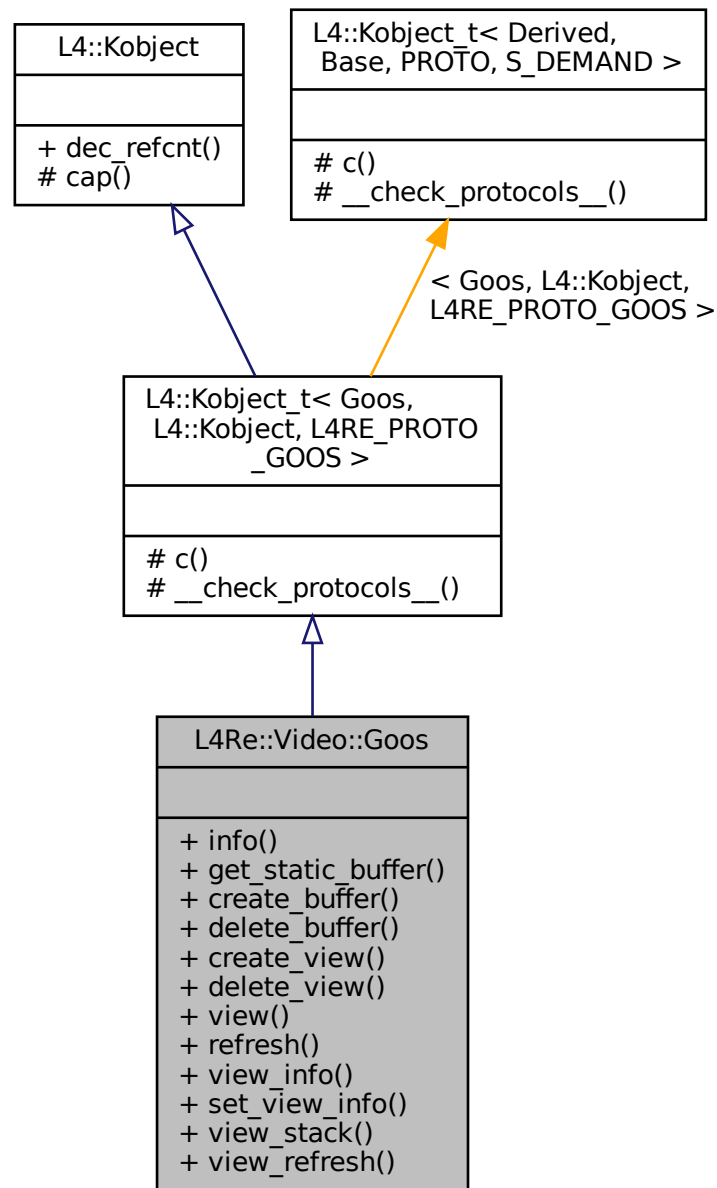
## 15.293 L4Re::Video::Goos Class Reference

A 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.*

## Additional Inherited Members

### 15.293.1 Detailed Description

A goos.

Definition at line 207 of file [goos](#).

### 15.293.2 Member Enumeration Documentation

#### 15.293.2.1 Flags

```
enum L4Re::Video::Goos::Flags
```

Flags for a goos.

Enumerator

|                                   |                                                  |
|-----------------------------------|--------------------------------------------------|
| <a href="#">F_auto_refresh</a>    | The graphics display is automatically refreshed. |
| <a href="#">F_pointer</a>         | We have a mouse pointer.                         |
| <a href="#">F_dynamic_views</a>   | Supports dynamically allocated views.            |
| <a href="#">F_dynamic_buffers</a> | Supports dynamically allocated buffers.          |

Definition at line 212 of file [goos](#).

### 15.293.3 Member Function Documentation

#### 15.293.3.1 create\_buffer()

```
long L4Re::Video::Goos::create_buffer (
 unsigned long size,
 L4::Ipc::Out< L4::Cap< L4Re::Dataspace > > rbuf)
```

Create a buffer.

##### Parameters

|             |                                                   |
|-------------|---------------------------------------------------|
| <i>size</i> | Size of buffer in bytes.                          |
| <i>rbuf</i> | Capability slot to point the buffer dataspace to. |

##### Return values

|          |                                                  |
|----------|--------------------------------------------------|
| $\geq 0$ | Success, the value returned is the buffer index. |
| $< 0$    | Error                                            |

#### 15.293.3.2 create\_view()

```
int L4Re::Video::Goos::create_view (
 View * view,
 l4_utcb_t * utcb = l4_utcb()) const [inline], [noexcept]
```

Create a view.

##### Parameters

|            |             |                                                  |
|------------|-------------|--------------------------------------------------|
| <i>out</i> | <i>view</i> | A view object.                                   |
|            | <i>utcb</i> | UTCB of the caller. This is a default parameter. |

##### Return values

|          |                                                |
|----------|------------------------------------------------|
| $\geq 0$ | Success, the value returned is the view index. |
| $< 0$    | Error                                          |

Definition at line 296 of file [goos](#).

**15.293.3.3 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   |

**15.293.3.4 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 316 of file [goos](#).

**15.293.3.5 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   |

**15.293.3.6 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   |

**15.293.3.7 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 347 of file [goos](#).

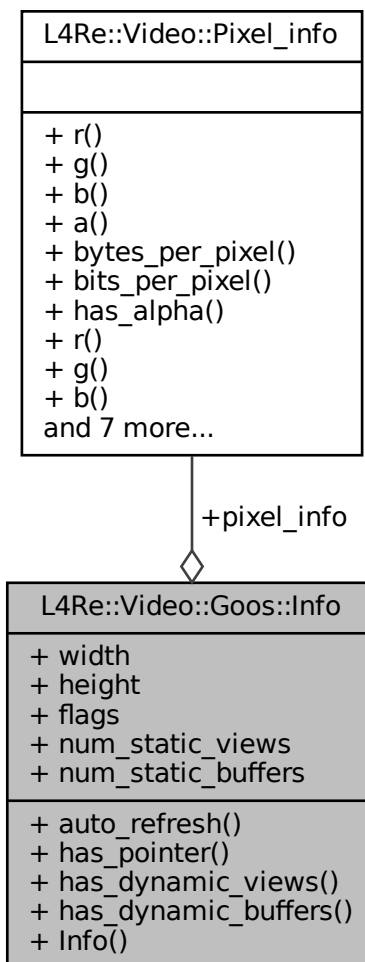
The documentation for this class was generated from the following file:

- [l4/re/video/goos](#)

## 15.294 L4Re::Video::Goos::Info Struct Reference

Information structure of a 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.*

### 15.294.1 Detailed Description

Information structure of a goos.

Definition at line [221](#) of file [goos](#).

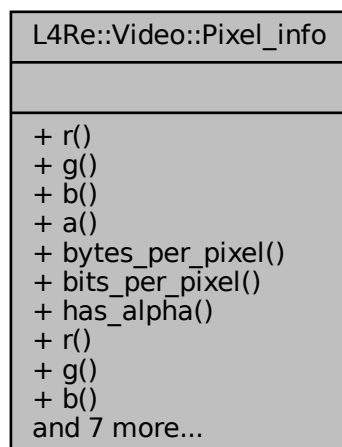
The documentation for this struct was generated from the following file:

- [l4/re/video/goos](#)

## 15.295 L4Re::Video::Pixel\_info Class Reference

Pixel information.

Collaboration diagram for L4Re::Video::Pixel\_info:



## 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.*
- 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.*

### 15.295.1 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 106 of file [colors](#).

### 15.295.2 Constructor & Destructor Documentation

**15.295.2.1 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 203 of file [colors](#).

**15.295.2.2 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 215 of file [colors](#).

**15.295.3 Member Function Documentation**

**15.295.3.1 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 135 of file [colors](#).

**15.295.3.2 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 178 of file [colors](#).

**15.295.3.3 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 129 of file [colors](#).

**15.295.3.4 b()** [2/2]

```
void L4Re::Video::Pixel_info::b (
 Color_component const & c) [inline]
```

Set the blue color component of the pixel.

#### Parameters

|          |                       |
|----------|-----------------------|
| <b>c</b> | Blue color component. |
|----------|-----------------------|

Definition at line 172 of file [colors](#).

#### 15.295.3.5 bits\_per\_pixel()

```
unsigned char L4Re::Video::Pixel_info::bits_per_pixel () const [inline]
```

Number of bits of the pixel.

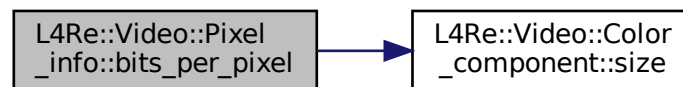
#### Returns

Number of bits used by the pixel.

Definition at line 147 of file [colors](#).

References [L4Re::Video::Color\\_component::size\(\)](#).

Here is the call graph for this function:



#### 15.295.3.6 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 141 of file [colors](#).

#### 15.295.3.7 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 184 of file [colors](#).

**15.295.3.8 dump()**

```
template<typename OUT >
void L4Re::Video::Pixel_info::dump (
 OUT & s) const [inline]
```

Dump information on the pixel to a stream.

**Parameters**

|          |        |
|----------|--------|
| <i>s</i> | Stream |
|----------|--------|

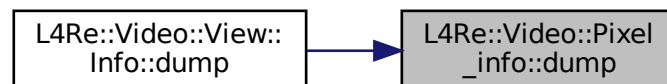
**Returns**

The stream

Definition at line 238 of file [colors](#).

Referenced by [L4Re::Video::View::Info::dump\(\)](#).

Here is the caller graph for this function:

**15.295.3.9 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 123 of file [colors](#).

### 15.295.3.10 g() [2/2]

```
void L4Re::Video::Pixel_info::g (
 Color_component const & c) [inline]
```

Set the green color component of the pixel.

#### Parameters

|          |                        |
|----------|------------------------|
| <i>c</i> | Green color component. |
|----------|------------------------|

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

### 15.295.3.11 has\_alpha()

```
bool L4Re::Video::Pixel_info::has_alpha () const [inline]
```

Return whether the pixel has an alpha channel.

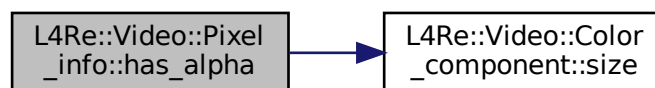
#### Returns

True if the pixel has an alpha channel, false if not.

Definition at line 154 of file [colors](#).

References [L4Re::Video::Color\\_component::size\(\)](#).

Here is the call graph for this function:



### 15.295.3.12 operator==( )

```
bool L4Re::Video::Pixel_info::operator== (
 Pixel_info const & o) const [inline]
```

Compare for complete equality of the color space.

## Parameters

|          |                                             |
|----------|---------------------------------------------|
| <i>o</i> | A <a href="#">Pixel_info</a> to compare to. |
|----------|---------------------------------------------|

## Returns

true if the both [Pixel\\_info](#)'s are equal, false if not.

Definition at line [227](#) of file [colors](#).

**15.295.3.13** [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 [117](#) of file [colors](#).

**15.295.3.14** [r\(\)](#) [2/2]

```
void L4Re::Video::Pixel_info::r (
 Color_component const & c) [inline]
```

Set the red color component of the pixel.

## Parameters

|          |                      |
|----------|----------------------|
| <i>c</i> | Red color component. |
|----------|----------------------|

Definition at line [160](#) of file [colors](#).

The documentation for this class was generated from the following file:

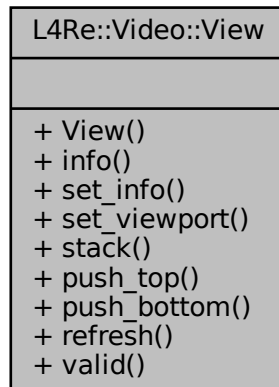
- [l4/re/video/colors](#)

**15.296** [L4Re::Video::View](#) Class Reference

[View](#).



Collaboration diagram for L4Re::Video::View:



## 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.*

## 15.296.1 Detailed Description

[View](#).

Definition at line 34 of file [goos](#).

## 15.296.2 Member Enumeration Documentation

### 15.296.2.1 Flags

```
enum L4Re::Video::View::Flags
```

Flags on a view.

Enumerator

|                      |                                                                    |
|----------------------|--------------------------------------------------------------------|
| F_none               | everything for this view is static (the VESA-FB case)              |
| F_set_buffer         | buffer object for this view can be changed                         |
| F_set_buffer_offset  | buffer offset can be set                                           |
| F_set_bytes_per_line | bytes per line can be set                                          |
| F_set_pixel          | pixel type can be set                                              |
| F_set_position       | position on screen can be set                                      |
| F_dyn_allocated      | <a href="#">View</a> is dynamically allocated.                     |
| F_set_background     | Set view as background for session.                                |
| F_set_flags          | Set view flags (.<br><br>See also<br><br><a href="#">V_flags</a> ) |
| F_fully_dynamic      | Flags for a fully dynamic view.                                    |

Definition at line 54 of file [goos](#).

### 15.296.2.2 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 77 of file [goos](#).

## 15.296.3 Member Function Documentation

### 15.296.3.1 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 351 of file [goos](#).

### 15.296.3.2 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 363 of file [goos](#).

### 15.296.3.3 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 355 of file [goos](#).

### 15.296.3.4 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 367 of file [goos](#).

References [L4Re::Video::View::Info::buffer\\_offset](#), [L4Re::Video::View::Info::flags](#), [L4Re::Video::View::Info::height](#), [L4Re::Video::View::Info::width](#), [L4Re::Video::View::Info::xpos](#), and [L4Re::Video::View::Info::ypos](#).

### 15.296.3.5 stack()

```
int L4Re::Video::View::stack (
 View const & pivot,
 bool behind = true) const [inline], [noexcept]
```

Move this view in the view stack.

#### Parameters

|               |                                                                                              |
|---------------|----------------------------------------------------------------------------------------------|
| <i>pivot</i>  | <a href="#">View</a> to move relative to                                                     |
| <i>behind</i> | When true move the view behind the pivot view, if false move the view before the pivot view. |

#### Return values

|    |         |
|----|---------|
| 0  | Success |
| <0 | Error   |

Definition at line 359 of file [goos](#).

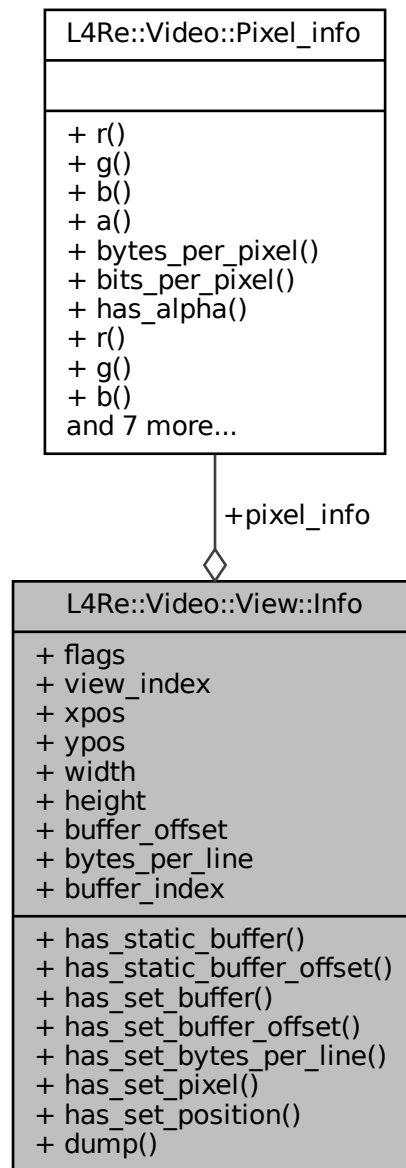
The documentation for this class was generated from the following file:

- [l4/re/video/goos](#)

## 15.297 L4Re::Video::View::Info Struct Reference

Information structure of a view.

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.*

### 15.297.1 Detailed Description

Information structure of a view.

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

The documentation for this struct was generated from the following file:

- l4/re/video/goos

## 15.298 l4re\_aux\_t Struct Reference

Auxiliary descriptor.

```
#include <l4aux.h>
```

Collaboration diagram for l4re\_aux\_t:



### 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.*

### 15.298.1 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](#)

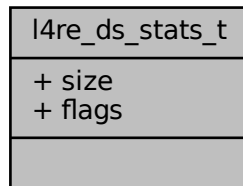


## 15.299 l4re\_ds\_stats\_t Struct Reference

Information about the data space.

```
#include <dataspace.h>
```

Collaboration diagram for l4re\_ds\_stats\_t:



### Data Fields

- l4re\_ds\_size\_t [size](#)  
*size*
- l4re\_ds\_flags\_t [flags](#)  
*flags*

### 15.299.1 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:

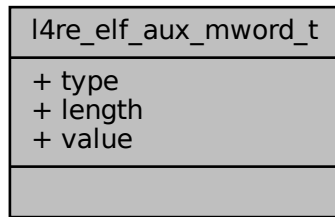
- l4/re/c/[dataspace.h](#)

## 15.300 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`:



### 15.300.1 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:

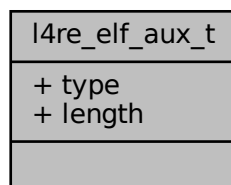
- [l4/re/elf\\_aux.h](#)

## 15.301 l4re\_elf\_aux\_t Struct Reference

Generic header for each auxiliary vector element.

```
#include <elf_aux.h>
```

Collaboration diagram for `l4re_elf_aux_t`:



### 15.301.1 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:

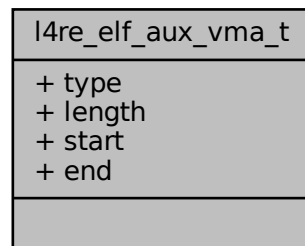
- [l4/re/elf\\_aux.h](#)

## 15.302 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:



### 15.302.1 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](#)

## 15.303 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<br>+ flags<br>+ name                                                |
| + l4re_env_cap_entry_t()<br>+ l4re_env_cap_entry_t()<br>+ is_valid_name() |

### 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.*

### 15.303.1 Detailed Description

Entry in the [L4Re](#) environment array for the named initial objects.

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

### 15.303.2 Constructor & Destructor Documentation

### 15.303.2.1 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](#).

## 15.303.3 Field Documentation

### 15.303.3.1 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:

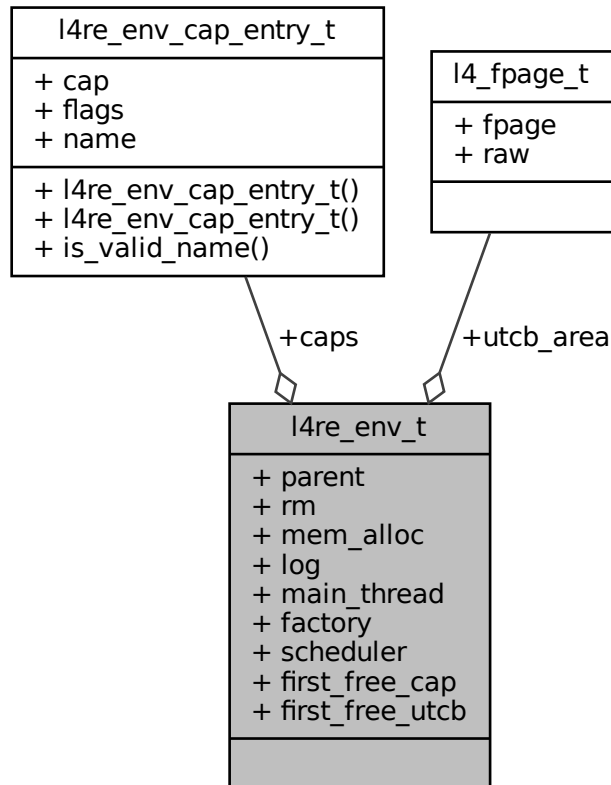
- [l4/re/env.h](#)

## 15.304 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.*

### 15.304.1 Detailed Description

Initial environment data structure.

See also

[Initial environment](#)

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

The documentation for this struct was generated from the following file:

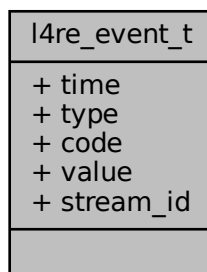
- [l4re/env.h](#)

## 15.305 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.*

### 15.305.1 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](#)

## 15.306 l4re\_video\_color\_component\_t Struct Reference

Color component structure.

```
#include <colors.h>
```

Collaboration diagram for `l4re_video_color_component_t`:

| <code>l4re_video_color_component_t</code> |
|-------------------------------------------|
| + size<br>+ shift                         |
|                                           |

## Data Fields

- unsigned char [size](#)  
*Size in bits.*
- unsigned char [shift](#)  
*offset in pixel*



### 15.306.1 Detailed Description

Color component structure.

Definition at line 31 of file [colors.h](#).

The documentation for this struct was generated from the following file:

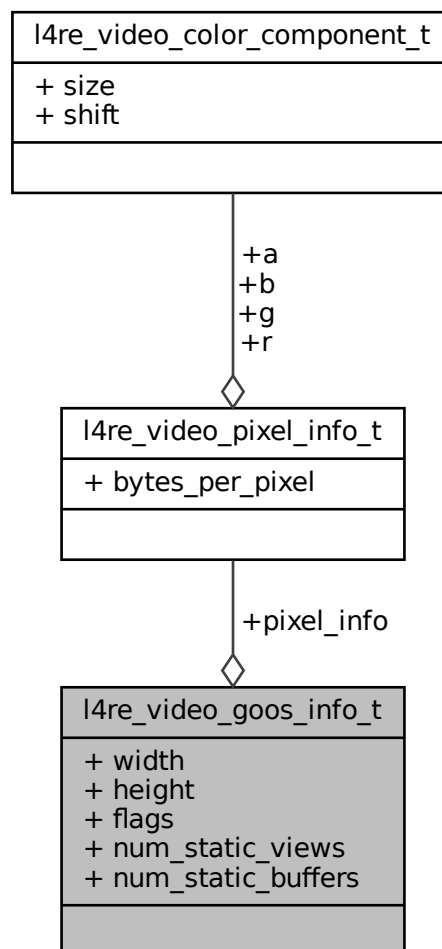
- [l4re/c/video/colors.h](#)

## 15.307 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.*

### 15.307.1 Detailed Description

Goos information structure.

Definition at line 51 of file [goos.h](#).

The documentation for this struct was generated from the following file:

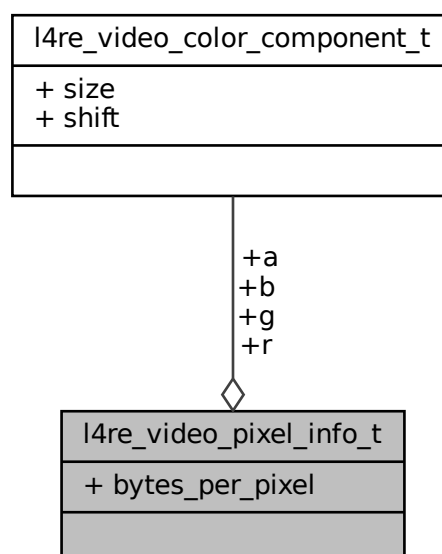
- [l4re/c/video/goos.h](#)

### 15.308 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.*

### 15.308.1 Detailed Description

Pixel\_info structure.

Definition at line 41 of file [colors.h](#).

The documentation for this struct was generated from the following file:

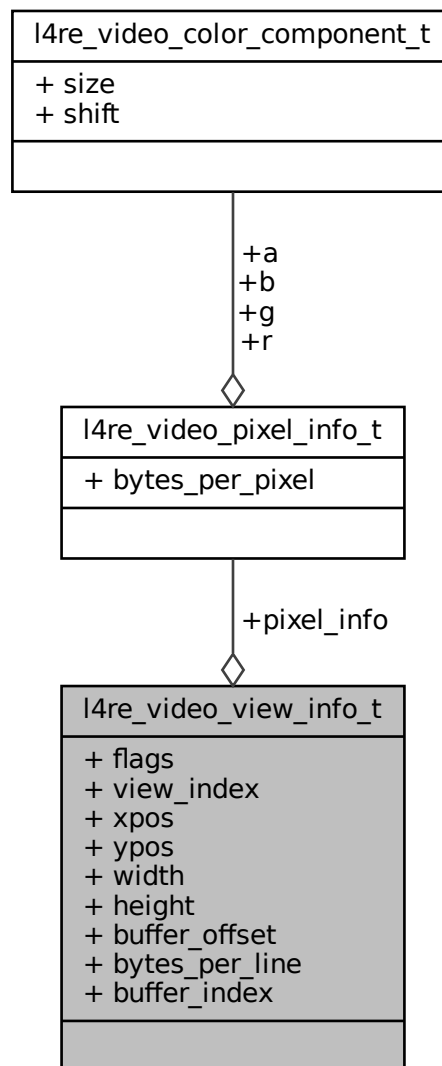
- [l4re/c/video/colors.h](#)

## 15.309 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.*

### 15.309.1 Detailed Description

View information structure.

Definition at line 59 of file [view.h](#).

The documentation for this struct was generated from the following file:

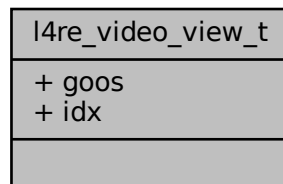
- [l4re/c/video/view.h](#)

## 15.310 l4re\_video\_view\_t Struct Reference

C representation of a goos view.

```
#include <view.h>
```

Collaboration diagram for l4re\_video\_view\_t:



### 15.310.1 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:

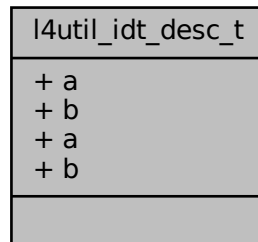
- [l4re/c/video/view.h](#)

## 15.311 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*

### 15.311.1 Detailed Description

IDT entry.

Definition at line 33 of file [idt.h](#).

The documentation for this struct was generated from the following file:

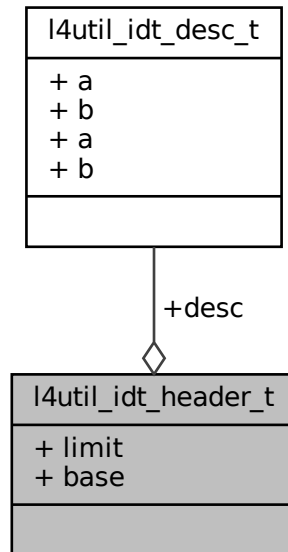
- amd64/l4/util/[idt.h](#)

## 15.312 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)*

### 15.312.1 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 file:

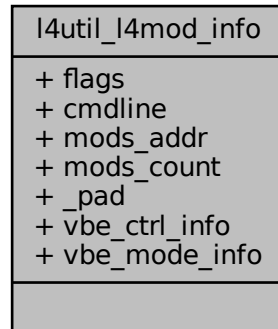
- `amd64/l4/util/idt.h`

## 15.313 l4util\_l4mod\_info Struct Reference

Base module structure.

```
#include <l4mod.h>
```

Collaboration diagram for l4util\_l4mod\_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.*

### 15.313.1 Detailed Description

Base module structure.

Definition at line 30 of file [l4mod.h](#).

### 15.313.2 Field Documentation



### 15.313.2.1 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 42 of file [l4mod.h](#).

The documentation for this struct was generated from the following file:

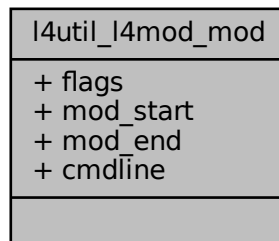
- `l4/util/l4mod.h`

## 15.314 l4util\_l4mod\_mod Struct Reference

A single module.

```
#include <l4mod.h>
```

Collaboration diagram for `l4util_l4mod_mod`:



### 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.*

### 15.314.1 Detailed Description

A single module.

Definition at line 21 of file [l4mod.h](#).

The documentation for this struct was generated from the following file:

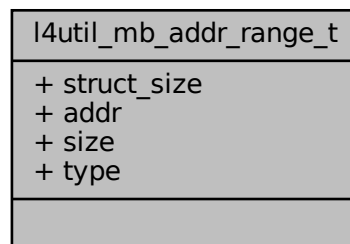
- [l4/util/l4mod.h](#)

## 15.315 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`:



### 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*

### 15.315.1 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 49 of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/mb\\_info.h](#)

## 15.316 l4util\_mb\_apm\_t Struct Reference

APM BIOS info.

```
#include <mb_info.h>
```

Collaboration diagram for l4util\_mb\_apm\_t:

| l4util_mb_apm_t                                                                                                                                                              |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <div>+ version</div> <div>+ cseg</div> <div>+ offset</div> <div>+ cseg_16</div> <div>+ dseg_16</div> <div>+ cseg_len</div> <div>+ cseg_16_len</div> <div>+ dseg_16_len</div> |
|                                                                                                                                                                              |

### 15.316.1 Detailed Description

APM BIOS info.

Definition at line 91 of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

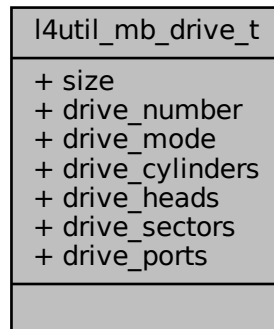
- [l4/util/mb\\_info.h](#)

## 15.317 l4util\_mb\_drive\_t Struct Reference

Drive Info structure.

```
#include <mb_info.h>
```

Collaboration diagram for l4util\_mb\_drive\_t:



### 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

### 15.317.1 Detailed Description

Drive Info structure.

Definition at line 74 of file [mb\\_info.h](#).

## 15.317.2 Field Documentation

### 15.317.2.1 drive\_cylinders

`l4_uint16_t l4util_mb_drive_t::drive_cylinders`

<The access mode (see below).

Definition at line 79 of file [mb\\_info.h](#).

### 15.317.2.2 drive\_mode

`l4_uint8_t l4util_mb_drive_t::drive_mode`

<The BIOS drive number.

Definition at line 78 of file [mb\\_info.h](#).

### 15.317.2.3 drive\_number

`l4_uint8_t l4util_mb_drive_t::drive_number`

<The size of this structure.

Definition at line 77 of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

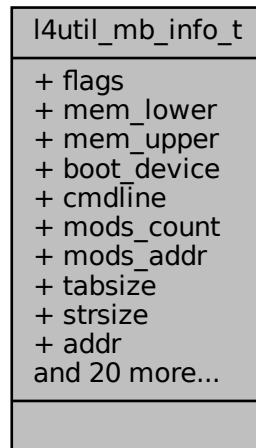
- [l4/util/mb\\_info.h](#)

## 15.318 l4util\_mb\_info\_t Struct Reference

MultiBoot Info description.

```
#include <mb_info.h>
```

Collaboration diagram for l4util\_mb\_info\_t:



### 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.*
- [l4\\_uint32\\_t tabsize](#)  
*(a.out) Kernel symbol table info*
- [l4\\_uint32\\_t num](#)  
*(ELF) Kernel section header table*

### 15.318.1 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 202 of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

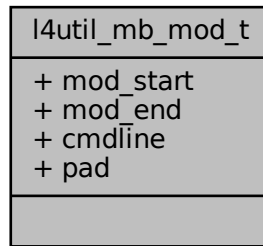
- [l4/util/mb\\_info.h](#)

## 15.319 l4util\_mb\_mod\_t Struct Reference

The structure type "mod\_list" is used by the [multiboot\\_info](#) structure.

```
#include <mb_info.h>
```

Collaboration diagram for `l4util_mb_mod_t`:



## Data Fields

- [l4\\_uint32\\_t mod\\_start](#)  
*Starting address of module in memory.*
- [l4\\_uint32\\_t mod\\_end](#)  
*End address of module in memory.*
- [l4\\_uint32\\_t cmdline](#)  
*Module command line.*
- [l4\\_uint32\\_t pad](#)  
*padding to take it to 16 bytes*

### 15.319.1 Detailed Description

The structure type "mod\_list" is used by the [multiboot\\_info](#) structure.

Definition at line 34 of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/mb\\_info.h](#)

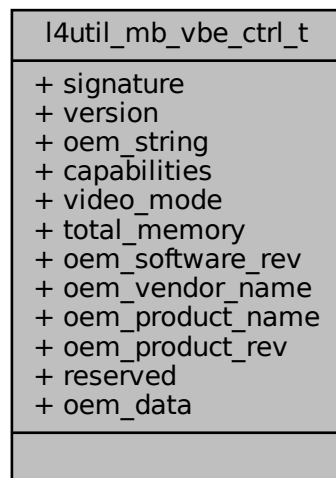
## 15.320 l4util\_mb\_vbe\_ctrl\_t Struct Reference

VBE controller information.

```
#include <mb_info.h>
```



Collaboration diagram for l4util\_mb\_vbe\_ctrl\_t:



### 15.320.1 Detailed Description

VBE controller information.

Definition at line 105 of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/mb\\_info.h](#)

## 15.321 l4util\_mb\_vbe\_mode\_t Struct Reference

VBE mode information.

```
#include <mb_info.h>
```

Collaboration diagram for `l4util_mb_vbe_mode_t`:

| <code>l4util_mb_vbe_mode_t</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>+ <code>mode_attributes</code></li> <li>+ <code>win_a_attributes</code></li> <li>+ <code>win_b_attributes</code></li> <li>+ <code>win_granularity</code></li> <li>+ <code>win_size</code></li> <li>+ <code>win_a_segment</code></li> <li>+ <code>win_b_segment</code></li> <li>+ <code>win_func</code></li> <li>+ <code>bytes_per_scanline</code></li> <li>+ <code>x_resolution</code></li> <li>+ <code>y_resolution</code></li> <li>+ <code>x_char_size</code></li> <li>+ <code>y_char_size</code></li> <li>+ <code>number_of_planes</code></li> <li>+ <code>bits_per_pixel</code></li> <li>+ <code>number_of_banks</code></li> <li>+ <code>memory_model</code></li> <li>+ <code>bank_size</code></li> <li>+ <code>number_of_image_pages</code></li> <li>+ <code>reserved0</code></li> <li>+ <code>red_mask_size</code></li> <li>+ <code>red_field_position</code></li> <li>+ <code>green_mask_size</code></li> <li>+ <code>green_field_position</code></li> <li>+ <code>blue_mask_size</code></li> <li>+ <code>blue_field_position</code></li> <li>+ <code>reserved_mask_size</code></li> <li>+ <code>reserved_field_position</code></li> <li>+ <code>direct_color_mode_info</code></li> <li>+ <code>phys_base</code></li> <li>+ <code>reserved1</code></li> <li>+ <code>reserved2</code></li> <li>+ <code>linear_bytes_per_scanline</code></li> <li>+ <code>banked_number_of_image_pages</code></li> <li>+ <code>linear_number_of_image_pages</code></li> <li>+ <code>linear_red_mask_size</code></li> <li>+ <code>linear_red_field_position</code></li> <li>+ <code>linear_green_mask_size</code></li> <li>+ <code>linear_green_field_position</code></li> <li>+ <code>linear_blue_mask_size</code></li> <li>+ <code>linear_blue_field_position</code></li> <li>+ <code>linear_reserved_mask_size</code></li> <li>+ <code>linear_reserved_field_position</code></li> <li>+ <code>max_pixel_clock</code></li> <li>+ <code>reserved3</code></li> </ul> |
| <ul style="list-style-type: none"> <li>* <code>mode_attributes</code></li> <li>* <code>win_a_attributes</code></li> <li>* <code>win_b_attributes</code></li> <li>* <code>win_granularity</code></li> <li>* <code>win_size</code></li> <li>* <code>win_a_segment</code></li> <li>* <code>win_b_segment</code></li> <li>* <code>win_func</code></li> <li>* <code>bytes_per_scanline</code></li> <li>* <code>x_resolution</code></li> <li>* <code>y_resolution</code></li> <li>* <code>x_char_size</code></li> <li>* <code>y_char_size</code></li> <li>* <code>number_of_planes</code></li> <li>* <code>bits_per_pixel</code></li> <li>* <code>number_of_banks</code></li> <li>* <code>memory_model</code></li> <li>* <code>bank_size</code></li> <li>* <code>number_of_image_pages</code></li> <li>* <code>reserved0</code></li> <li>* <code>red_mask_size</code></li> <li>* <code>red_field_position</code></li> <li>* <code>green_mask_size</code></li> <li>* <code>green_field_position</code></li> <li>* <code>blue_mask_size</code></li> <li>* <code>blue_field_position</code></li> <li>* <code>reserved_mask_size</code></li> <li>* <code>reserved_field_position</code></li> <li>* <code>direct_color_mode_info</code></li> <li>* <code>phys_base</code></li> <li>* <code>reserved1</code></li> <li>* <code>reserved2</code></li> <li>* <code>linear_bytes_per_scanline</code></li> <li>* <code>banked_number_of_image_pages</code></li> <li>* <code>linear_number_of_image_pages</code></li> <li>* <code>linear_red_mask_size</code></li> <li>* <code>linear_red_field_position</code></li> <li>* <code>linear_green_mask_size</code></li> <li>* <code>linear_green_field_position</code></li> <li>* <code>linear_blue_mask_size</code></li> <li>* <code>linear_blue_field_position</code></li> <li>* <code>linear_reserved_mask_size</code></li> <li>* <code>linear_reserved_field_position</code></li> <li>* <code>max_pixel_clock</code></li> <li>* <code>reserved3</code></li> </ul> |

## Data Fields

all VESA versions

- [l4\\_uint16\\_t mode\\_attributes](#)
- [l4\\_uint8\\_t win\\_a\\_attributes](#)
- [l4\\_uint8\\_t win\\_b\\_attributes](#)
- [l4\\_uint16\\_t win\\_granularity](#)

- [l4\\_uint16\\_t win\\_size](#)
- [l4\\_uint16\\_t win\\_a\\_segment](#)
- [l4\\_uint16\\_t win\\_b\\_segment](#)
- [l4\\_uint32\\_t win\\_func](#)
- [l4\\_uint16\\_t bytes\\_per\\_scanline](#)

#### >= VESA version 1.2

- [l4\\_uint16\\_t x\\_resolution](#)
- [l4\\_uint16\\_t y\\_resolution](#)
- [l4\\_uint8\\_t x\\_char\\_size](#)
- [l4\\_uint8\\_t y\\_char\\_size](#)
- [l4\\_uint8\\_t number\\_of\\_planes](#)
- [l4\\_uint8\\_t bits\\_per\\_pixel](#)
- [l4\\_uint8\\_t number\\_of\\_banks](#)
- [l4\\_uint8\\_t memory\\_model](#)
- [l4\\_uint8\\_t bank\\_size](#)
- [l4\\_uint8\\_t number\\_of\\_image\\_pages](#)
- [l4\\_uint8\\_t reserved0](#)

#### direct color

- [l4\\_uint8\\_t red\\_mask\\_size](#)
- [l4\\_uint8\\_t red\\_field\\_position](#)
- [l4\\_uint8\\_t green\\_mask\\_size](#)
- [l4\\_uint8\\_t green\\_field\\_position](#)
- [l4\\_uint8\\_t blue\\_mask\\_size](#)
- [l4\\_uint8\\_t blue\\_field\\_position](#)
- [l4\\_uint8\\_t reserved\\_mask\\_size](#)
- [l4\\_uint8\\_t reserved\\_field\\_position](#)
- [l4\\_uint8\\_t direct\\_color\\_mode\\_info](#)

#### >= VESA version 2.0

- [l4\\_uint32\\_t phys\\_base](#)
- [l4\\_uint32\\_t reserved1](#)
- [l4\\_uint16\\_t reversed2](#)

#### >= VESA version 3.0

- [l4\\_uint16\\_t linear\\_bytes\\_per\\_scanline](#)
- [l4\\_uint8\\_t banked\\_number\\_of\\_image\\_pages](#)
- [l4\\_uint8\\_t linear\\_number\\_of\\_image\\_pages](#)
- [l4\\_uint8\\_t linear\\_red\\_mask\\_size](#)
- [l4\\_uint8\\_t linear\\_red\\_field\\_position](#)
- [l4\\_uint8\\_t linear\\_green\\_mask\\_size](#)
- [l4\\_uint8\\_t linear\\_green\\_field\\_position](#)
- [l4\\_uint8\\_t linear\\_blue\\_mask\\_size](#)
- [l4\\_uint8\\_t linear\\_blue\\_field\\_position](#)
- [l4\\_uint8\\_t linear\\_reserved\\_mask\\_size](#)
- [l4\\_uint8\\_t linear\\_reserved\\_field\\_position](#)
- [l4\\_uint32\\_t max\\_pixel\\_clock](#)
- [l4\\_uint8\\_t reserved3 \[189+1\]](#)

### 15.321.1 Detailed Description

VBE mode information.

Definition at line 123 of file [mb\\_info.h](#).

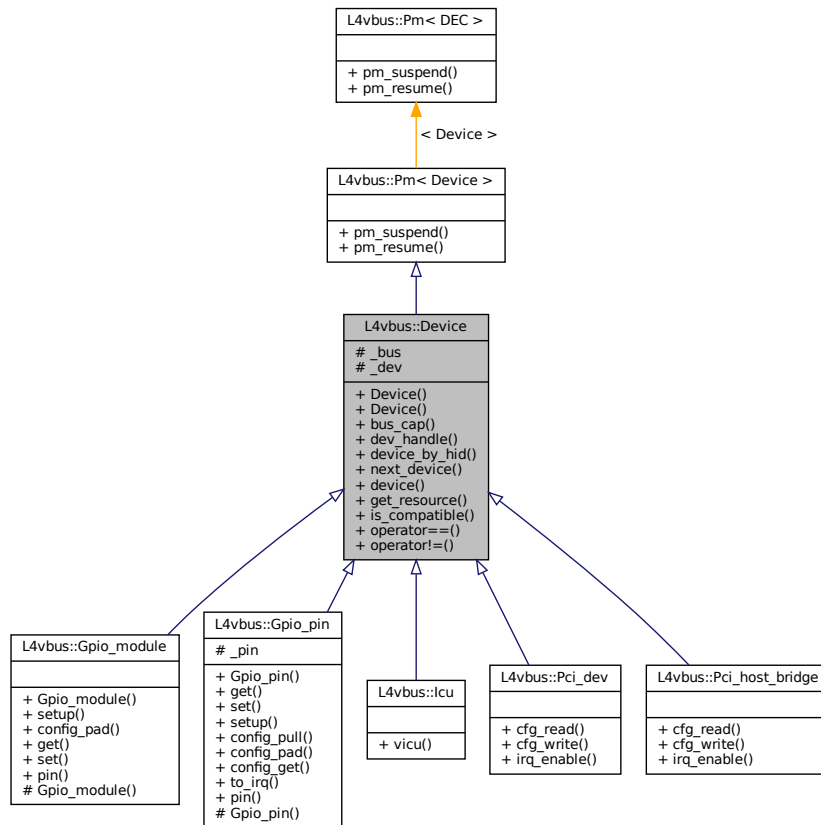
The documentation for this struct was generated from the following file:

- [l4/util/mb\\_info.h](#)

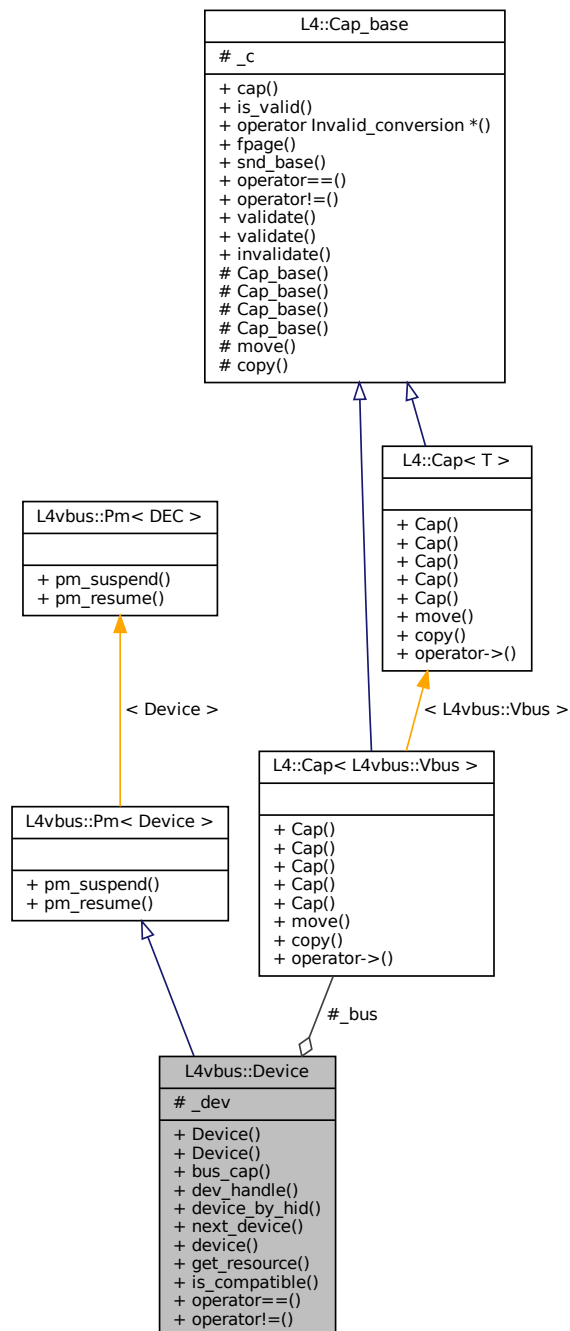
## 15.322 L4vbus::Device Class Reference

[Device](#) on a [L4vbus::Vbus](#).

Inheritance diagram for L4vbus::Device:



Collaboration diagram for L4vbus::Device:



## Public Member Functions

- `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 devices are not the same.

## Protected Attributes

- `L4::Cap< Vbus > _bus`
- `l4vbus_device_handle_t _dev`  
The device handle for this device.

### 15.322.1 Detailed Description

*Device* on a `L4vbus::Vbus`.

Definition at line 75 of file `vbus`.

### 15.322.2 Member Function Documentation

#### 15.322.2.1 bus\_cap()

```
L4::Cap<Vbus> L4vbus::Device::bus_cap () const [inline]
```

Access the *Vbus* capability of the underlying virtual bus.

**Returns**

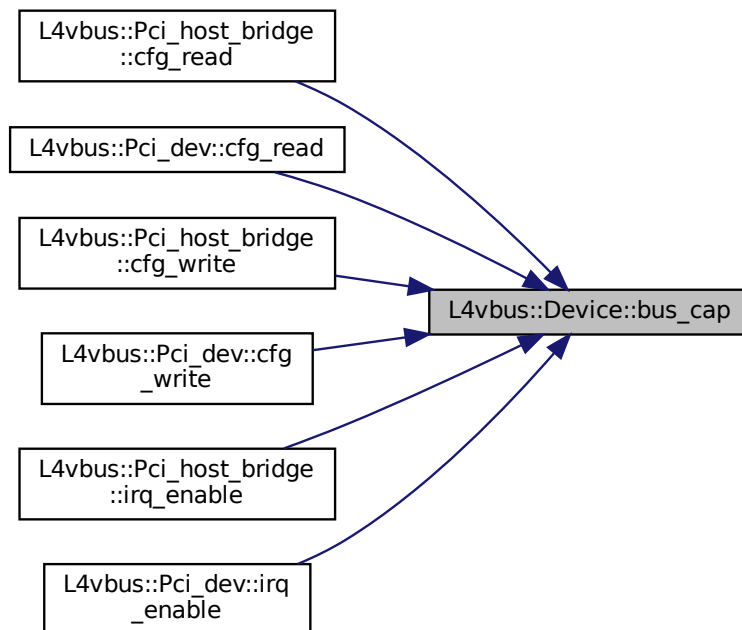
the capability to the underlying [Vbus](#).

Definition at line 87 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:

**15.322.2.2 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 96 of file [vbus](#).

References [\\_dev](#).

### 15.322.2.3 device()

```
int L4vbus::Device::device (
 l4vbus_device_t * devinfo) const [inline]
```

Obtain detailed information about a [Vbus](#) device.

#### Parameters

|     |         |                                                                                           |
|-----|---------|-------------------------------------------------------------------------------------------|
| out | devinfo | 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 dev could be found. |

Definition at line 168 of file [vbus](#).

References [\\_bus](#), [\\_dev](#), and [l4vbus\\_get\\_device\(\)](#).

Here is the call graph for this function:



### 15.322.2.4 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

|                |                |                                                                                                                                                                                                                                                                                    |
|----------------|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>in, out</i> | <i>child</i>   | Handle of the device from where in the device tree the search should start. To start searching from the beginning <i>child</i> must be initialized using the default ( <a href="#">L4VBUS_NULL</a> ). If a matching device is found its handle is returned through this parameter. |
|                | <i>hid</i>     | HID of the device                                                                                                                                                                                                                                                                  |
|                | <i>depth</i>   | Maximum depth for the recursive lookup                                                                                                                                                                                                                                             |
| <i>out</i>     | <i>devinfo</i> | <a href="#">Device</a> information structure (might be NULL)                                                                                                                                                                                                                       |

## Return values

|                            |                                                          |
|----------------------------|----------------------------------------------------------|
| $\geq 0$                   | A device with the given HID was found.                   |
| <a href="#">-L4_ENOENT</a> | No device with the given HID could be found on the vbus. |
| <a href="#">-L4_EINVAL</a> | Invalid or no HID provided.                              |
| <a href="#">-L4_ENODEV</a> | Function called on a non-existing device.                |

Definition at line [128](#) of file [vbus](#).

15.322.2.5 `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 <a href="#">l4vbus_device_t</a> structure that is returned by <a href="#">L4vbus::Device::device_by_hid()</a> and <a href="#">L4vbus::Device::next_device()</a> . |
| <i>out</i> | <i>res</i>     | Descriptor of the resource.                                                                                                                                                                                                                                                                                   |

This function returns the resource descriptor of an individual device resource selected by the *res\_idx* parameter.

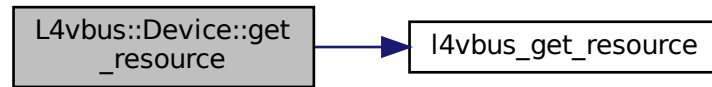
## Return values

|                            |                                         |
|----------------------------|-----------------------------------------|
| <i>0</i>                   | Success.                                |
| <a href="#">-L4_ENOENT</a> | Invalid resource index <i>res_idx</i> . |

Definition at line [188](#) of file [vbus](#).

References [\\_bus](#), [\\_dev](#), and [l4vbus\\_get\\_resource\(\)](#).

Here is the call graph for this function:



#### 15.322.2.6 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 |
|------------|------------------------------|

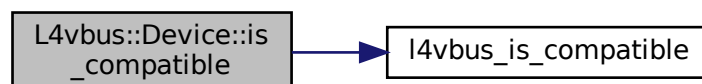
##### Returns

1 when the given ID (*cid*) matches this device, 0 when the given ID does not match, <0 on error.

Definition at line 202 of file [vbus](#).

References [\\_bus](#), [\\_dev](#), and [l4vbus\\_is\\_compatible\(\)](#).

Here is the call graph for this function:



### 15.322.2.7 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

|                      |                      |                                                                                                                                                                                                                                                |
|----------------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>in, out</code> | <code>child</code>   | Handle of the device that precedes the device that shall be returned. To start from the beginning, <code>child</code> must be initialized with <code>L4VBUS_NULL</code> . If a device is found, its handle is returned through this parameter. |
|                      | <code>depth</code>   | Depth to look for                                                                                                                                                                                                                              |
| <code>out</code>     | <code>devinfo</code> | <code>Device</code> information (might be NULL)                                                                                                                                                                                                |

#### Returns

0 on success, else failure

Definition at line 150 of file `vbus`.

### 15.322.2.8 operator"!="()

```
bool L4vbus::Device::operator!= (
 Device const & o) const [inline]
```

Test if two devices are not the same.

#### Returns

true if the two devices are different, false else.

Definition at line 218 of file `vbus`.

References `_bus`, and `_dev`.

### 15.322.2.9 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 209 of file `vbus`.

References `_bus`, and `_dev`.

### 15.322.3 Field Documentation

#### 15.322.3.1 `_bus`

`L4::Cap<Vbus> L4vbus::Device::_bus` [protected]

The `Vbus` capability (where this device is located on).

Definition at line 224 of file `vbus`.

Referenced by `bus_cap()`, `L4vbus::Gpio_pin::config_get()`, `L4vbus::Gpio_module::config_pad()`, `L4vbus::Gpio_pin::config_pad()`, `L4vbus::Gpio_pin::config_pull()`, `device()`, `L4vbus::Gpio_pin::get()`, `L4vbus::Gpio_module::get()`, `get_resource()`, `is_compatible()`, `operator!=()`, `operator==()`, `L4vbus::Gpio_pin::set()`, `L4vbus::Gpio_module::set()`, `L4vbus::Gpio_module::setup()`, `L4vbus::Gpio_pin::setup()`, `L4vbus::Gpio_pin::to_irq()`, and `L4vbus::Icu::vicu()`.

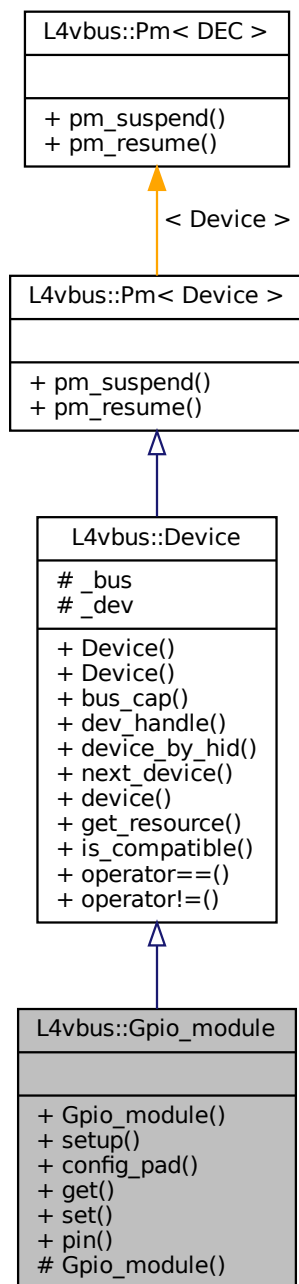
The documentation for this class was generated from the following file:

- `I4/vbus/vbus`

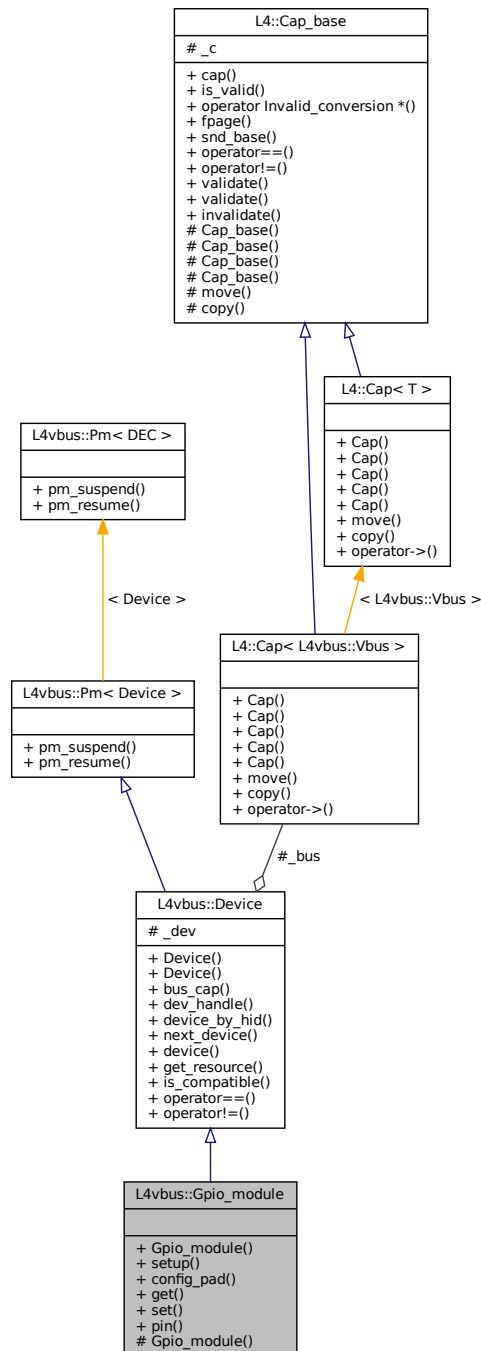
### 15.323 L4vbus::Gpio\_module Class Reference

A `Gpio_module` groups multiple GPIO pins together.

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](#).*

## Additional Inherited Members

### 15.323.1 Detailed Description

A [Gpio\\_module](#) groups multiple GPIO pins together.

Definition at line 135 of file [vbus\\_gpio](#).

### 15.323.2 Member Function Documentation

#### 15.323.2.1 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                                                                                            |

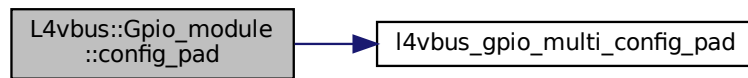
#### Returns

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:



### 15.323.2.2 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.                                       |
| 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

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:



### 15.323.2.3 pin()

```
Gpio_pin L4vbus::Gpio_module::pin (
 unsigned pin) const [inline]
```

Get [Gpio\\_pin](#) for a specific pin of this [Gpio\\_module](#).



## Parameters

|            |                                                      |
|------------|------------------------------------------------------|
| <i>pin</i> | GPIO pin number to get <a href="#">Gpio_pin</a> for. |
|------------|------------------------------------------------------|

## Returns

[Gpio\\_pin](#)

Definition at line 231 of file [vbus\\_gpio](#).

### 15.323.2.4 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:



### 15.323.2.5 setup()

```
int L4vbus::Gpio_module::setup (
 Pin_slice const & mask,
 unsigned mode,
 unsigned value) const [inline]
```

Configure function of multiple GPIO pins at once.

#### Parameters

|              |                                                                                                                                                                                            |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>mask</i>  | Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. The real number depends on the hardware and the driver implementation. |
| <i>mode</i>  | GPIO function, see <a href="#">L4vbus_gpio_generic_func</a> for generic functions. Hardware specific functions must be provided in the lower 8 bits.                                       |
| <i>value</i> | Optional value to set the GPIO pins to if they are configured as output pins                                                                                                               |

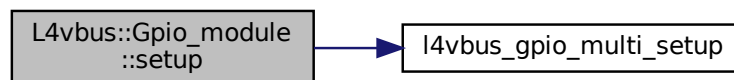
#### Returns

0 if OK, error code otherwise

Definition at line 168 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_multi\\_setup\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- [l4/vbus/vbus\\_gpio](#)

## 15.324 L4vbus::Gpio\_module::Pin\_slice Struct Reference

A slice of the pins provided by this module.

Collaboration diagram for L4vbus::Gpio\_module::Pin\_slice:

| L4vbus::Gpio_module<br>::Pin_slice |
|------------------------------------|
| + offset<br>+ mask                 |
| + Pin_slice()                      |

### 15.324.1 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](#).

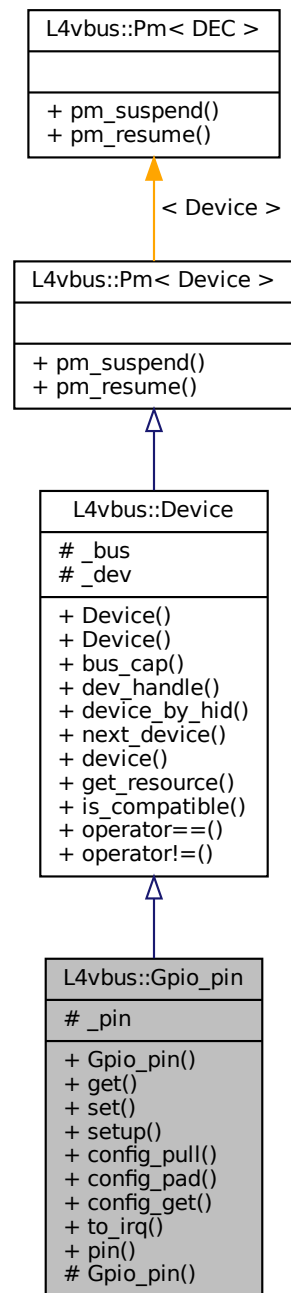
The documentation for this struct was generated from the following file:

- [l4/vbus/vbus\\_gpio](#)

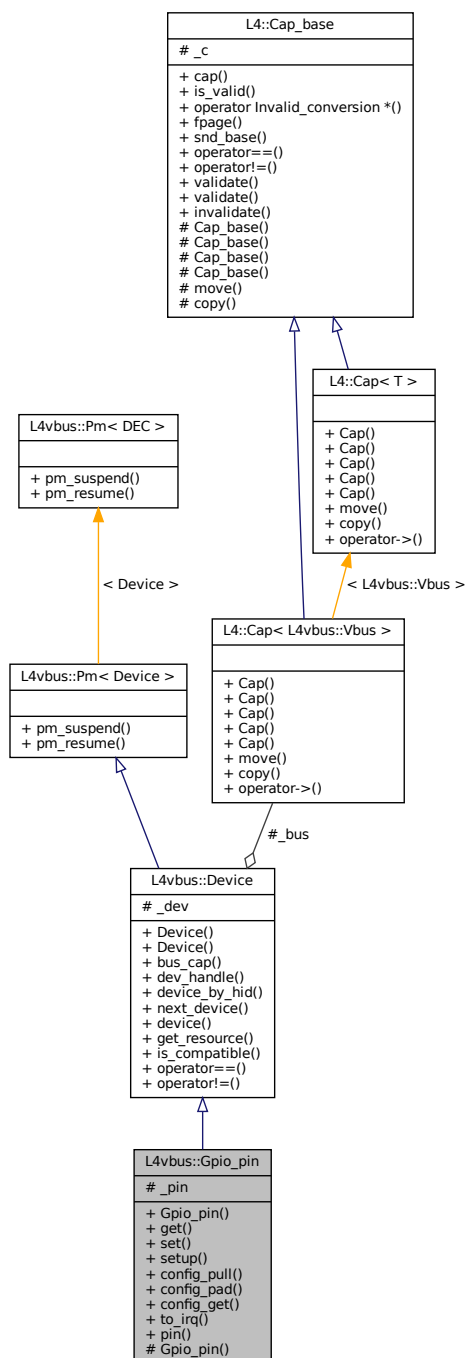
## 15.325 L4vbus::Gpio\_pin Class Reference

A GPIO pin.

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.*

## Additional Inherited Members

### 15.325.1 Detailed Description

A GPIO pin.

Definition at line 28 of file [vbus\\_gpio](#).

### 15.325.2 Member Function Documentation

#### 15.325.2.1 [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:



### 15.325.2.2 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:



### 15.325.2.3 config\_pull()

```
int L4vbus::Gpio_pin::config_pull (
 unsigned mode) const [inline]
```

Generic function to set pull up/down mode.

**Parameters**

|             |                                                                             |
|-------------|-----------------------------------------------------------------------------|
| <i>mode</i> | mode for pull up/down resistors, see <a href="#">L4vbus_gpio_pull_modes</a> |
|-------------|-----------------------------------------------------------------------------|

**Returns**

0 if OK, error code otherwise

Definition at line 77 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_config\\_pull\(\)](#).

Here is the call graph for this function:

**15.325.2.4 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:





### 15.325.2.5 pin()

```
unsigned L4vbus::Gpio_pin::pin () const [inline]
```

Get pin number.

#### Returns

GPIO pin number

Definition at line 124 of file [vbus\\_gpio](#).

### 15.325.2.6 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:



### 15.325.2.7 setup()

```
int L4vbus::Gpio_pin::setup (
 unsigned mode,
 unsigned value) const [inline]
```

Configure the function of a GPIO pin.

## Parameters

|              |                                                                                                                                                      |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>mode</i>  | GPIO function, see <a href="#">L4vbus_gpio_generic_func</a> for generic functions. Hardware specific functions must be provided in the lower 8 bits. |
| <i>value</i> | Optional value to set the GPIO pin to if it is configured as an output pin                                                                           |

## Returns

0 if OK, error code otherwise

Definition at line 66 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_setup\(\)](#).

Here is the call graph for this function:

**15.325.2.8 to\_irq()**

```
int L4vbus::Gpio_pin::to_irq () const [inline]
```

Create IRQ for GPIO pin.

## Returns

IRQ number if OK, negative error code otherwise

Definition at line 114 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_to\\_irq\(\)](#).

Here is the call graph for this function:



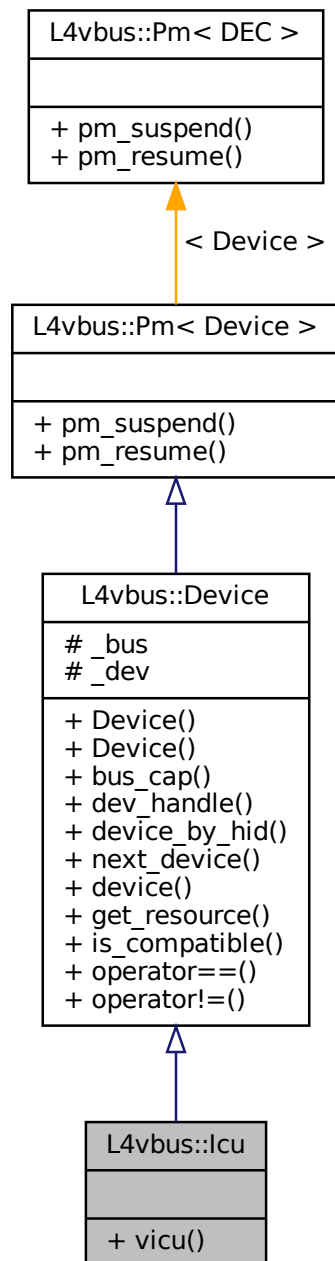
The documentation for this class was generated from the following file:

- `I4/vbus/vbus_gpio`

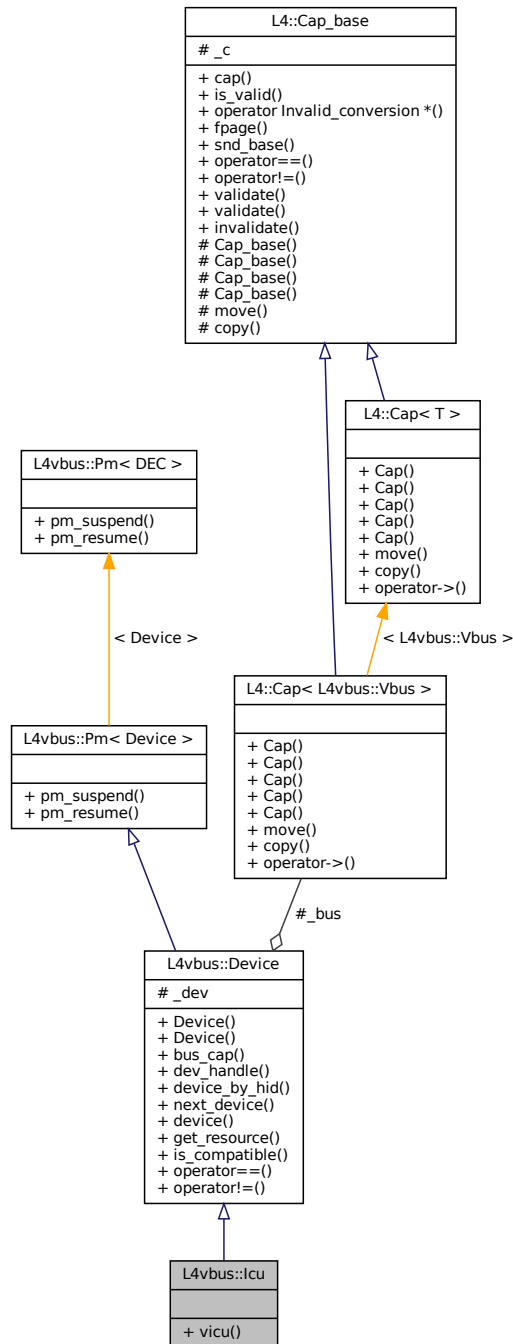
## 15.326 L4vbus::lcu Class Reference

[Vbus](#) Interrupt controller API.

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 the [L4::Icu](#) capability for this [Vbus](#) ICU.

## Additional Inherited Members

### 15.326.1 Detailed Description

[Vbus](#) Interrupt controller API.

Allows to access the underlying [L4::Icu](#) capability managing IRQs for the [L4vbus::Vbus](#).

See also

[L4::Icu](#)

Definition at line 238 of file [vbus](#).

### 15.326.2 Member Enumeration Documentation

#### 15.326.2.1 Src\_types

```
enum L4vbus::Icu::Src_types
```

Flags that can be used with the ICU on a vbus device.

Enumerator

|                             |                                                                                                                                                                                                                                                     |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>Src_dev_handle</code> | Flag to denote that the value should be interpreted as a device handle. This flag may be used in the <code>source</code> parameter in <a href="#">L4::Icu::msi_info()</a> to denote that the ICU should interpret the source ID as a device handle. |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

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

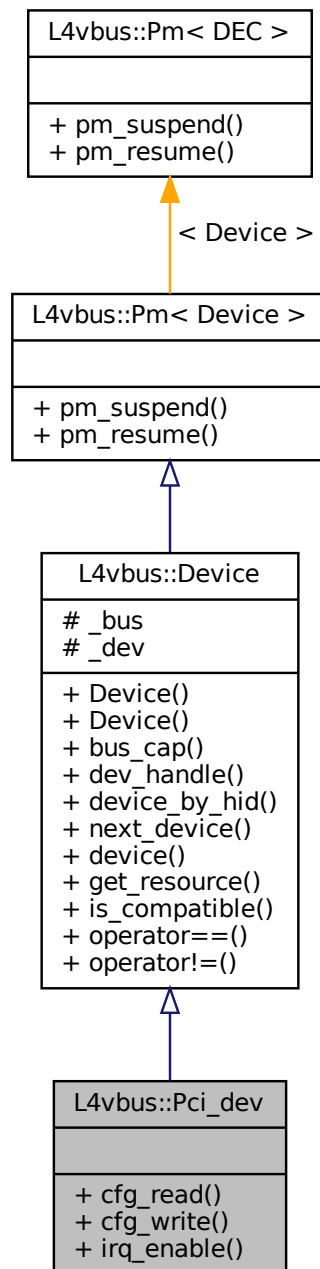
The documentation for this class was generated from the following file:

- `I4/vbus/vbus`

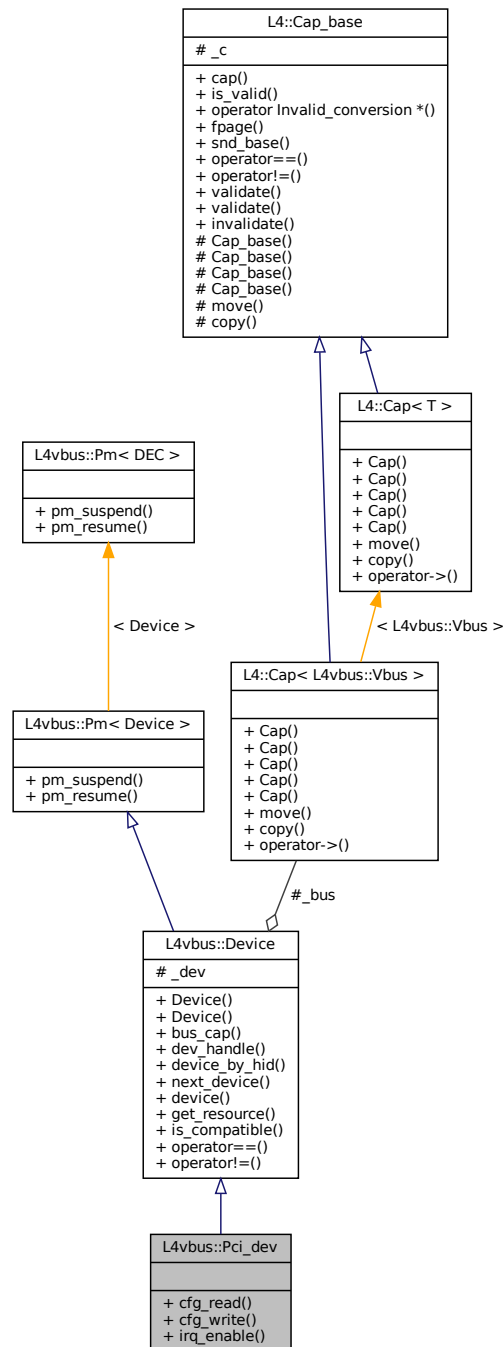
## 15.327 L4vbus::Pci\_dev Class Reference

A PCI device.

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.

## Additional Inherited Members

### 15.327.1 Detailed Description

A PCI device.

Definition at line 95 of file [vbus\\_pci](#).

### 15.327.2 Member Function Documentation

#### 15.327.2.1 `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)  |

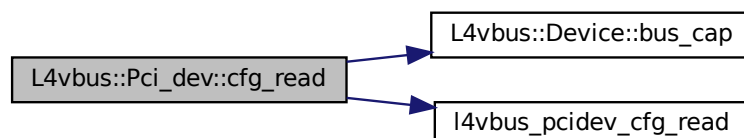
#### Returns

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:





### 15.327.2.2 cfg\_write()

```
int L4vbus::Pci_dev::cfg_write (
 l4_uint32_t reg,
 l4_uint32_t value,
 l4_uint32_t width) const [inline]
```

Write to the device's vPCI configuration space.

#### Parameters

|              |                                          |
|--------------|------------------------------------------|
| <i>reg</i>   | Register in configuration space to write |
| <i>value</i> | Value to write                           |
| <i>width</i> | Width to write in bits (e.g. 8, 16, 32)  |

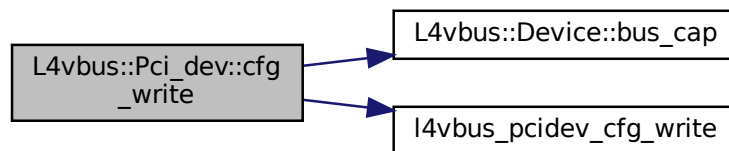
#### Returns

0 on success, else failure

Definition at line 123 of file [vbus\\_pci](#).

References [L4vbus::Device::\\_dev](#), [L4vbus::Device::bus\\_cap\(\)](#), and [l4vbus\\_pcidev\\_cfg\\_write\(\)](#).

Here is the call graph for this function:



### 15.327.2.3 irq\_enable()

```
int L4vbus::Pci_dev::irq_enable (
 unsigned char * trigger,
 unsigned char * polarity) const [inline]
```

Enable the device's PCI interrupt.

#### Parameters

|     |                 |                                       |
|-----|-----------------|---------------------------------------|
| out | <i>trigger</i>  | False if interrupt is level-triggered |
| out | <i>polarity</i> | True if interrupt is of low polarity  |

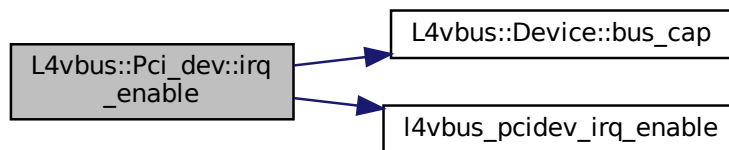
**Returns**

On success: Interrupt line to be used, else failure

Definition at line 139 of file [vbus\\_pci](#).

References [L4vbus::Device::\\_dev](#), [L4vbus::Device::bus\\_cap\(\)](#), and [l4vbus\\_pcidev\\_irq\\_enable\(\)](#).

Here is the call graph for this function:



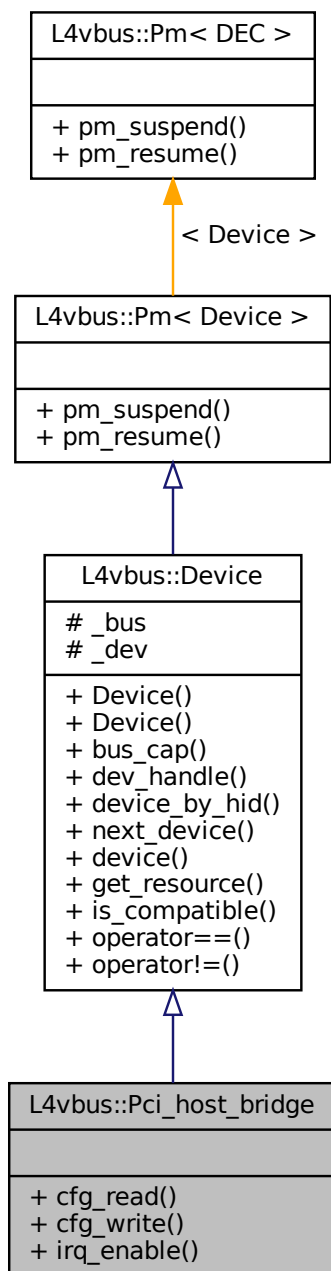
The documentation for this class was generated from the following file:

- `l4/vbus/vbus_pci`

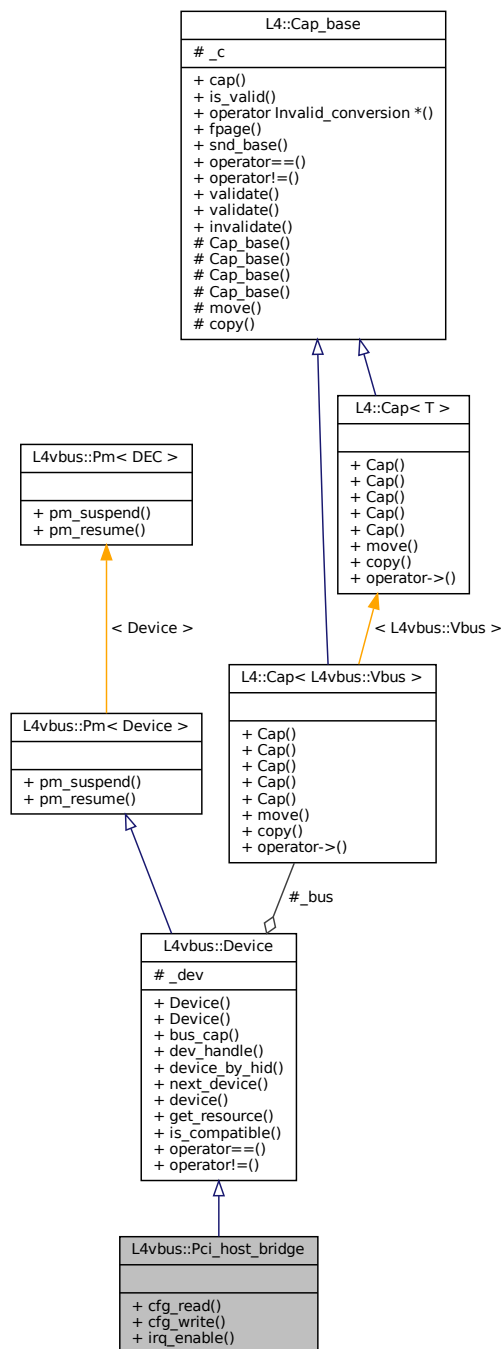
## 15.328 L4vbus::Pci\_host\_bridge Class Reference

A Pci host bridge.

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.*

## Additional Inherited Members

### 15.328.1 Detailed Description

A Pci host bridge.

Definition at line 27 of file [vbus\\_pci](#).

### 15.328.2 Member Function Documentation

#### 15.328.2.1 [cfg\\_read\(\)](#)

```
int L4vbus::Pci_host_bridge::cfg_read (
 l4_uint32_t bus,
 l4_uint32_t devfn,
 l4_uint32_t reg,
 l4_uint32_t * value,
 l4_uint32_t width) const [inline]
```

Read from the vPCI configuration space using the PCI root bridge.

#### Parameters

|     |              |                                                                    |
|-----|--------------|--------------------------------------------------------------------|
|     | <i>bus</i>   | Bus number                                                         |
|     | <i>devfn</i> | <a href="#">Device</a> id (upper 16bit) and function (lower 16bit) |
|     | <i>reg</i>   | Register in configuration space to read                            |
| out | <i>value</i> | Value that has been read                                           |
|     | <i>width</i> | Width to read in bits (e.g. 8, 16, 32)                             |

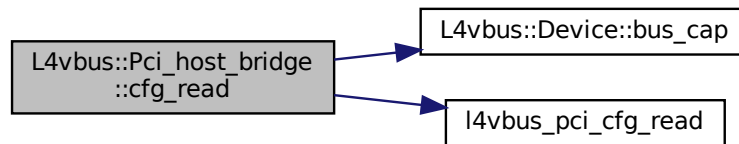
#### Returns

0 on success, else failure

Definition at line 41 of file [vbus\\_pci](#).

References [L4vbus::Device::\\_dev](#), [L4vbus::Device::bus\\_cap\(\)](#), and [l4vbus\\_pci\\_cfg\\_read\(\)](#).

Here is the call graph for this function:



### 15.328.2.2 `cfg_write()`

```
int L4vbus::Pci_host_bridge::cfg_write (
 14_uint32_t bus,
 14_uint32_t devfn,
 14_uint32_t reg,
 14_uint32_t value,
 14_uint32_t width) const [inline]
```

Write to the vPCI configuration space using the PCI root bridge.

#### Parameters

|              |                                                                    |
|--------------|--------------------------------------------------------------------|
| <i>bus</i>   | Bus number                                                         |
| <i>devfn</i> | <a href="#">Device</a> id (upper 16bit) and function (lower 16bit) |
| <i>reg</i>   | Register in configuration space to write                           |
| <i>value</i> | Value to write                                                     |
| <i>width</i> | Width to write in bits (e.g. 8, 16, 32)                            |

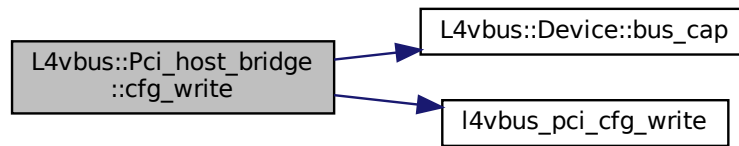
#### Returns

0 on success, else failure

Definition at line 60 of file [vbus\\_pci](#).

References [L4vbus::Device::\\_dev](#), [L4vbus::Device::bus\\_cap\(\)](#), and [l4vbus\\_pci\\_cfg\\_write\(\)](#).

Here is the call graph for this function:



### 15.328.2.3 irq\_enable()

```
int L4vbus::Pci_host_bridge::irq_enable (
 l4_uint32_t bus,
 l4_uint32_t devfn,
 int pin,
 unsigned char * trigger,
 unsigned char * polarity) const [inline]
```

Enable PCI interrupt for a specific device using the PCI root bridge.

#### Parameters

|     |                 |                                                                     |
|-----|-----------------|---------------------------------------------------------------------|
|     | <i>bus</i>      | Bus number                                                          |
|     | <i>devfn</i>    | <a href="#">Device</a> id (upper 16bit) and function (lower 16bit)  |
|     | <i>pin</i>      | Interrupt pin (normally as reported in configuration register INTR) |
| out | <i>trigger</i>  | False if interrupt is level-triggered                               |
| out | <i>polarity</i> | True if interrupt is of low polarity                                |

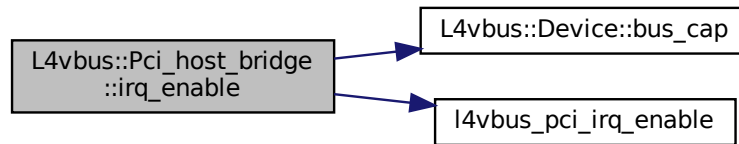
#### Returns

On success: Interrupt line to be used, else failure

Definition at line 81 of file [vbus\\_pci](#).

References [L4vbus::Device::\\_dev](#), [L4vbus::Device::bus\\_cap\(\)](#), and [l4vbus\\_pci\\_irq\\_enable\(\)](#).

Here is the call graph for this function:



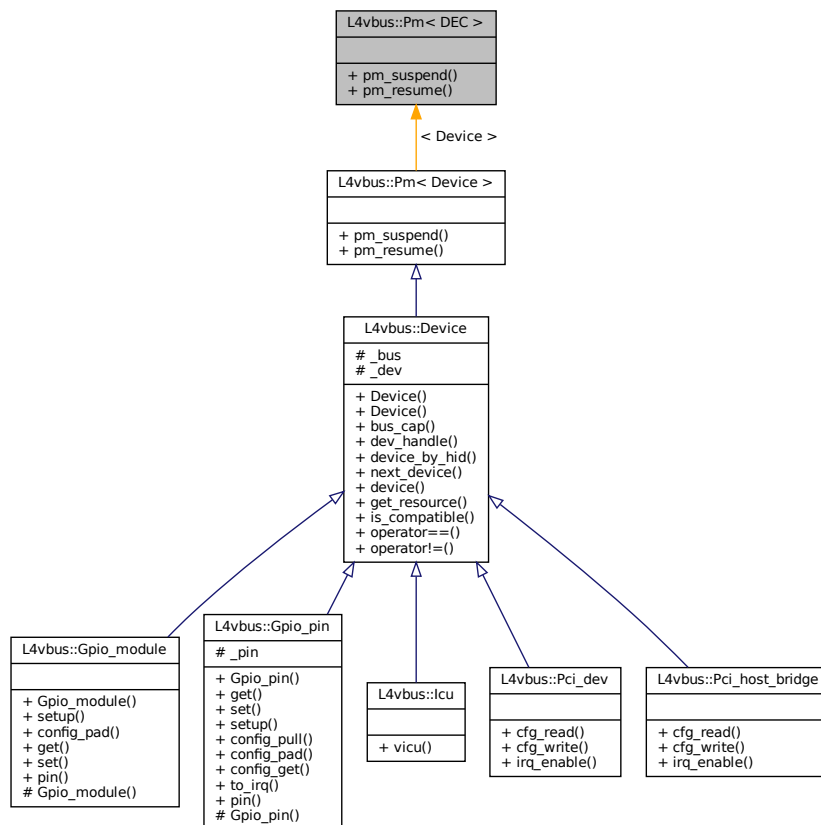
The documentation for this class was generated from the following file:

- l4/vbus/vbus\_pci

## 15.329 L4vbus::Pm< DEC > Class Template Reference

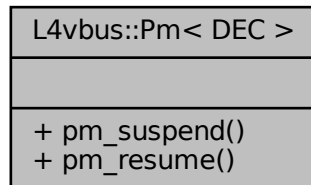
Power-management API mixin.

Inheritance diagram for L4vbus::Pm< DEC >:





Collaboration diagram for L4vbus::Pm< DEC >:



## Public Member Functions

- int [pm\\_suspend](#) () const  
*Suspend the module.*
- int [pm\\_resume](#) () const  
*Resume the module.*

### 15.329.1 Detailed Description

```
template<typename DEC>
class L4vbus::Pm< DEC >
```

Power-management API mixin.

Definition at line 51 of file [vbus](#).

The documentation for this class was generated from the following file:

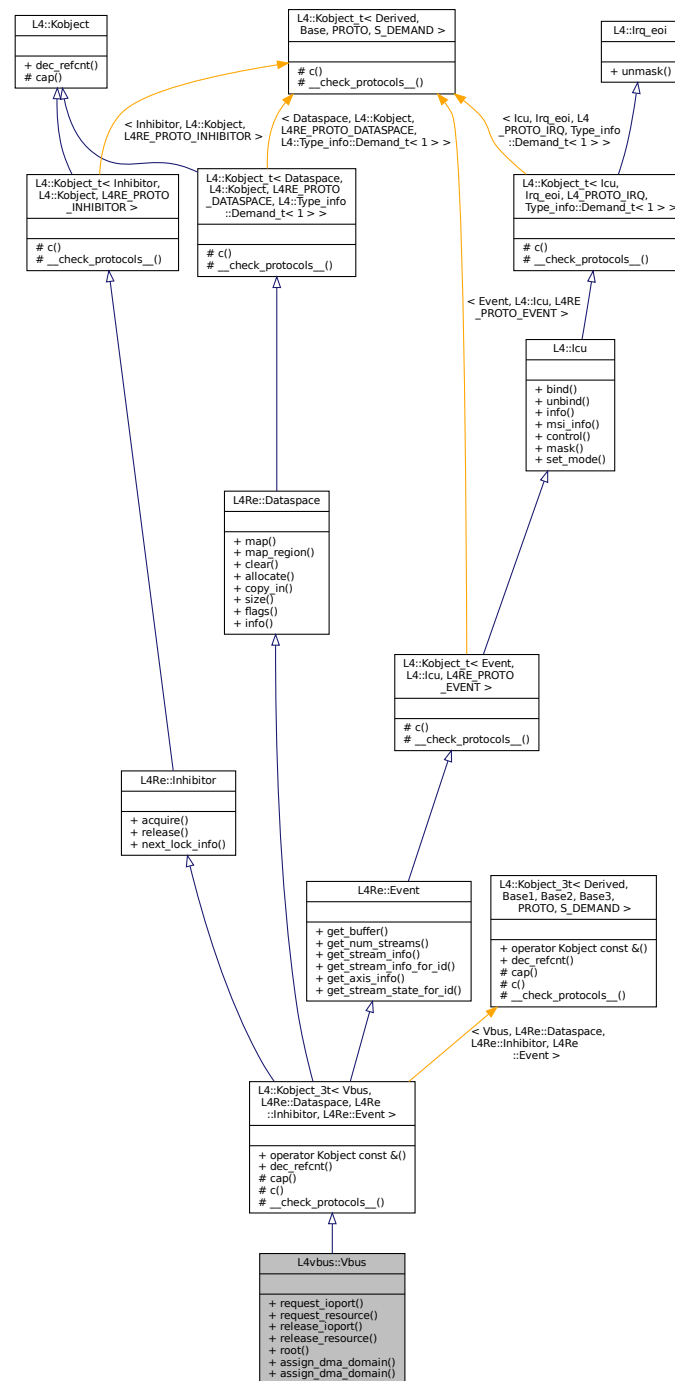
- [l4/vbus/vbus](#)

## 15.330 L4vbus::Vbus Class Reference

The virtual bus ([Vbus](#)) interface.



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 request_resource (l4vbus_resource_t *res, int=0) const`  
Request the given resource from the bus.
- `int release_ioport (l4vbus_resource_t *res) const`

- *Release the given IO port resource from the bus.*  
int [release\\_resource](#) ([l4vbus\\_resource\\_t](#) \*res) const
- *Release the given 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
- *Assign an [L4Re::Dma\\_space](#) to a DMA domain.*  
int [assign\\_dma\\_domain](#) (unsigned domain\_id, unsigned flags, [L4::Cap](#)< [L4::Task](#) > dma\_space) const
- *Assign a kernel DMA space to a DMA domain.*

## Additional Inherited Members

### 15.330.1 Detailed Description

The virtual bus ([Vbus](#)) interface.

See also

[L4Re Vbus API](#)

Definition at line 270 of file [vbus](#).

### 15.330.2 Member Function Documentation

#### 15.330.2.1 [assign\\_dma\\_domain\(\)](#) [1/2]

```
int L4vbus::Vbus::assign_dma_domain (
 unsigned domain_id,
 unsigned flags,
 L4::Cap< L4::Task > dma_space) const [inline]
```

Assign a kernel DMA space to a DMA domain.

#### Parameters

|                  |                                                                                                                                         |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| <i>domain_id</i> | DMA domain ID (resource address of DMA domain found on the vBUS). If the value is ~0U the DMA space of the whole vBUS is used.          |
| <i>flags</i>     | A combination of <a href="#">L4vbus_dma_domain_assign_flags</a> .                                                                       |
| <i>dma_space</i> | The DMA space capability to bind or unbind, this must be a kernel DMA space ( <a href="#">L4::Task</a> created with L4_PROTO_DMA_SPACE) |

#### Return values

|   |                                   |
|---|-----------------------------------|
| 0 | Operation completed successfully. |
|---|-----------------------------------|

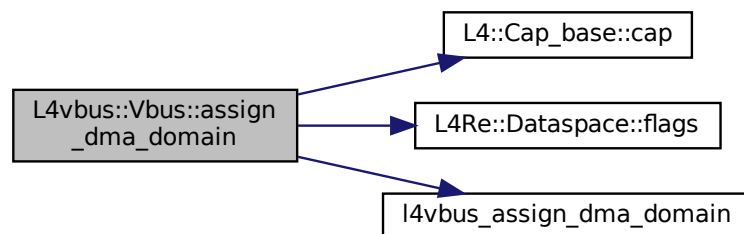
## Return values

|            |                                                                                 |
|------------|---------------------------------------------------------------------------------|
| -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 385 of file [vbus](#).

References [L4::Cap\\_base::cap\(\)](#), [L4Re::Dataspace::flags\(\)](#), [l4vbus\\_assign\\_dma\\_domain\(\)](#), [L4VBUS\\_DMAD\\_KERNEL\\_DMA\\_SPACE](#) and [L4VBUS\\_DMAD\\_L4RE\\_DMA\\_SPACE](#).

Here is the call graph for this function:



## 15.330.2.2 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]

```

Assign 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 <code>~0U</code> the DMA space of the whole vBUS is used. |
| <i>flags</i>     | A combination of <a href="#">L4vbus_dma_domain_assign_flags</a> .                                                                           |
| <i>dma_space</i> | The DMA space capability to bind or unbind, this must be an <a href="#">L4Re::Dma_space</a>                                                 |

## Return values

|            |                                                                                |
|------------|--------------------------------------------------------------------------------|
| 0          | Operation completed successfully.                                              |
| -L4_ENOENT | The vbus does not support a global DMA domain or no DMA domain could be found. |
| -L4_EINVAL | Invalid argument used.                                                         |

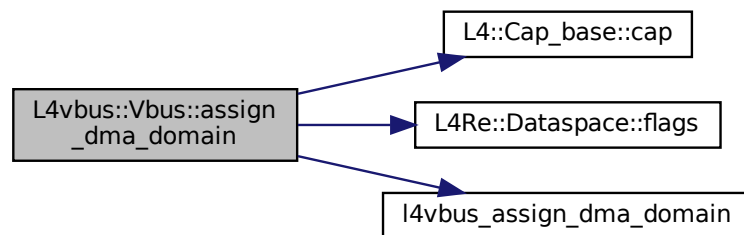
## Return values

|                        |                                                                                 |
|------------------------|---------------------------------------------------------------------------------|
| <code>-L4_EBUSY</code> | DMA domain is already active, this means another DMA space is already assigned. |
|------------------------|---------------------------------------------------------------------------------|

Definition at line 360 of file [vbus](#).

References [L4::Cap\\_base::cap\(\)](#), [L4Re::Dataspace::flags\(\)](#), [l4vbus\\_assign\\_dma\\_domain\(\)](#), [L4VBUS\\_DMAD\\_KERNEL\\_DMA\\_SPACE](#) and [L4VBUS\\_DMAD\\_L4RE\\_DMA\\_SPACE](#).

Here is the call graph for this function:



### 15.330.2.3 release\_ioport()

```
int L4vbus::Vbus::release_ioport (
 l4vbus_resource_t * res) const [inline]
```

Release the given IO port resource from the bus.

## Parameters

|                 |                  |                                                   |
|-----------------|------------------|---------------------------------------------------|
| <code>in</code> | <code>res</code> | The IO port resource to be released from the bus. |
|-----------------|------------------|---------------------------------------------------|

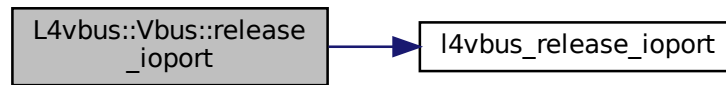
## Returns

$\geq 0$  on success,  $< 0$  on error.

Definition at line 313 of file [vbus](#).

References [l4vbus\\_release\\_ioport\(\)](#).

Here is the call graph for this function:



#### 15.330.2.4 release\_resource()

```
int L4vbus::Vbus::release_resource (
 l4vbus_resource_t * res) const [inline]
```

Release the given resource from the bus.

##### Parameters

|    |            |                                           |
|----|------------|-------------------------------------------|
| in | <i>res</i> | The resource to be released from the bus. |
|----|------------|-------------------------------------------|

##### Returns

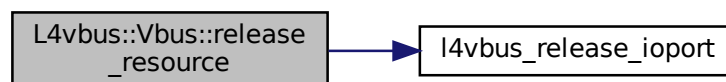
$\geq 0$  on success,  $< 0$  on error.

**Deprecated** This function is deprecated since Q3 2019. Use [release\\_ioport\(\)](#) instead.

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

References [l4vbus\\_release\\_ioport\(\)](#).

Here is the call graph for this function:



#### 15.330.2.5 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 283 of file [vbus](#).

References [l4vbus\\_request\\_ioport\(\)](#).

Here is the call graph for this function:



### 15.330.2.6 request\_resource()

```
int L4vbus::Vbus::request_resource (
 l4vbus_resource_t * res,
 int = 0) const [inline]
```

Request the given resource from the bus.

## Parameters

|           |              |                                            |
|-----------|--------------|--------------------------------------------|
| <i>in</i> | <i>res</i>   | The resource to be requested from the bus. |
|           | <i>flags</i> | Ignored.                                   |

## Returns

0 on success, else failure.

**Deprecated** This function is deprecated since Q3 2019. Use [request\\_ioport\(\)](#) instead.

Definition at line 301 of file [vbus](#).

References [l4vbus\\_request\\_ioport\(\)](#).

Here is the call graph for this function:



### 15.330.2.7 root()

`Device L4vbus::Vbus::root ( ) const [inline]`

Get the root device of the device tree of this bus.

Returns

A [Vbus](#) device representing the root of the device tree.

Definition at line [339](#) of file [vbus](#).

References [L4VBUS\\_ROOT\\_BUS](#).

The documentation for this class was generated from the following file:

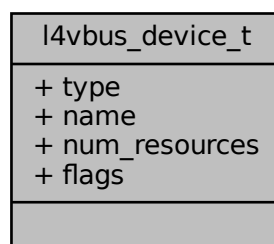
- `I4/vbus/vbus`

## 15.331 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](#).*

### 15.331.1 Detailed Description

Detailed information about a vbus device.

Definition at line 69 of file [vbus\\_types.h](#).

The documentation for this struct was generated from the following file:

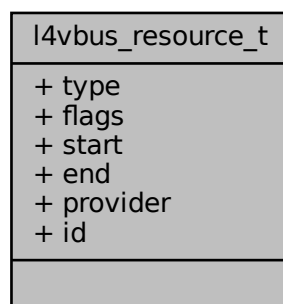
- [l4/vbus/vbus\\_types.h](#)

## 15.332 l4vbus\_resource\_t Struct Reference

Description of a single vbus resource.

```
#include <vbus_types.h>
```

Collaboration diagram for l4vbus\_resource\_t:



## 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.*

### 15.332.1 Detailed Description

Description of a single vbus resource.

Definition at line 23 of file [vbus\\_types.h](#).

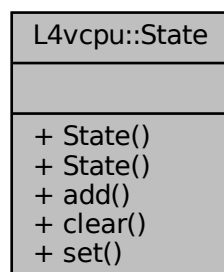
The documentation for this struct was generated from the following file:

- [l4/vbus/vbus\\_types.h](#)

## 15.333 L4vcpu::State Class Reference

C++ implementation of state word in the vCPU area.

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.*

### 15.333.1 Detailed Description

C++ implementation of state word in the vCPU area.

Definition at line 31 of file [vcpu](#).

### 15.333.2 Constructor & Destructor Documentation

#### 15.333.2.1 State()

```
L4vcpu::State::State (
 unsigned v) [inline], [explicit]
```

Initialize state.

##### Parameters

|                   |                |
|-------------------|----------------|
| <a href="#">v</a> | Initial state. |
|-------------------|----------------|

Definition at line 41 of file [vcpu](#).

### 15.333.3 Member Function Documentation

#### 15.333.3.1 add()

```
void L4vcpu::State::add (
 unsigned bits) throw () [inline]
```

Add flags.

## Parameters

|             |                          |
|-------------|--------------------------|
| <i>bits</i> | Bits to add to the word. |
|-------------|--------------------------|

Definition at line 48 of file [vcpu](#).

**15.333.3.2 clear()**

```
void L4vcpu::State::clear (
 unsigned bits) throw () [inline]
```

Clear flags.

## Parameters

|             |                            |
|-------------|----------------------------|
| <i>bits</i> | Bits to clear in the word. |
|-------------|----------------------------|

Definition at line 55 of file [vcpu](#).

**15.333.3.3 set()**

```
void L4vcpu::State::set (
 unsigned v) throw () [inline]
```

Set flags.

## Parameters

|          |                                 |
|----------|---------------------------------|
| <i>v</i> | Set the word to the value of v. |
|----------|---------------------------------|

Definition at line 62 of file [vcpu](#).

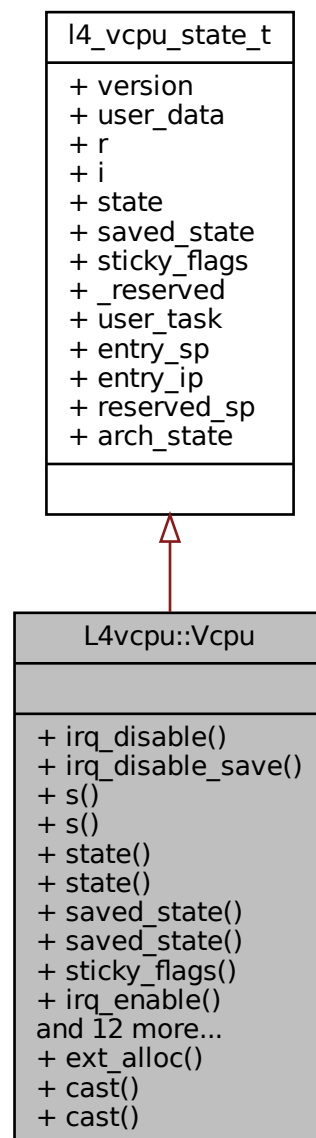
The documentation for this class was generated from the following file:

- [l4/vcpu/vcpu](#)

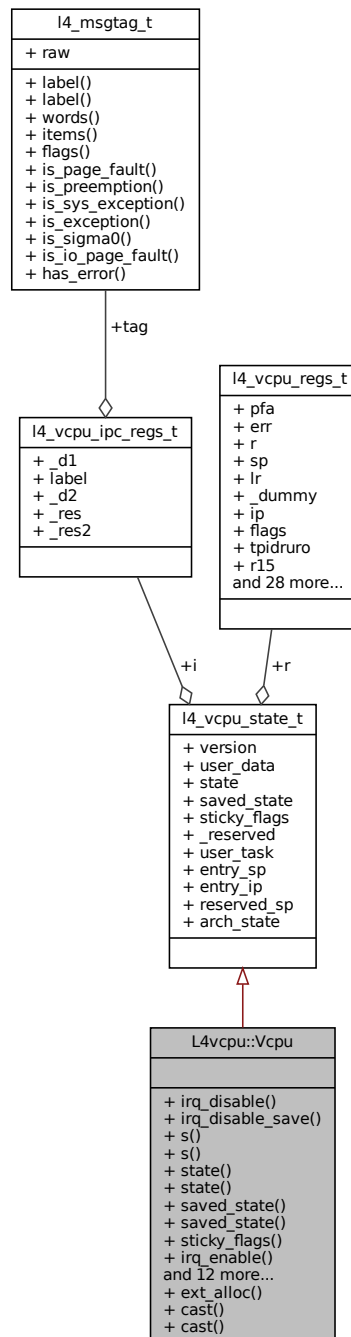
**15.334 L4vcpu::Vcpu Class Reference**

C++ implementation of the vCPU save state area.

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.*

### 15.334.1 Detailed Description

C++ implementation of the vCPU save state area.

Definition at line 72 of file [vcpu](#).

### 15.334.2 Member Function Documentation

#### 15.334.2.1 `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 275 of file [vcpu](#).

#### 15.334.2.2 `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 265 of file [vcpu](#).

**15.334.2.3 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 239 of file `vcpu`.

References `l4_vcpu_state_t::entry_ip`.

**15.334.2.4 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 232 of file `vcpu`.

References `l4_vcpu_state_t::entry_sp`.

**15.334.2.5 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.

## Return values

|                  |                                     |
|------------------|-------------------------------------|
| <i>vcpu</i>      | Allocated vcpu-state area.          |
| <i>ext_state</i> | Allocated extended vcpu-state area. |

## Parameters

|             |                                                                           |
|-------------|---------------------------------------------------------------------------|
| <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

**15.334.2.6 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 216 of file [vcpu](#).

References [l4\\_vcpu\\_state\\_t::i](#).

**15.334.2.7 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 223 of file [vcpu](#).

References [l4\\_vcpu\\_state\\_t::i](#).

### 15.334.2.8 irq\_disable\_save()

```
unsigned L4vcpu::Vcpu::irq_disable_save () throw () [inline]
```

Disable the vCPU for event delivery and return previous state.

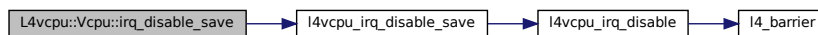
#### Returns

IRQ state before disabling IRQs.

Definition at line 85 of file [vcpu](#).

References [l4vcpu\\_irq\\_disable\\_save\(\)](#).

Here is the call graph for this function:



### 15.334.2.9 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.

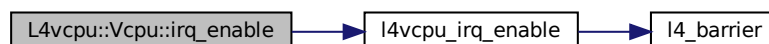
#### Parameters

|                         |                                                                                                            |
|-------------------------|------------------------------------------------------------------------------------------------------------|
| <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 142 of file [vcpu](#).

References [l4vcpu\\_irq\\_enable\(\)](#).

Here is the call graph for this function:



### 15.334.2.10 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.

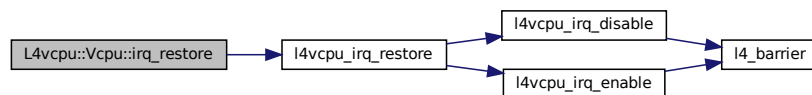
#### Parameters

|                         |                                                                                                            |
|-------------------------|------------------------------------------------------------------------------------------------------------|
| <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 157 of file [vcpu](#).

References [l4vcpu\\_irq\\_restore\(\)](#).

Here is the call graph for this function:



### 15.334.2.11 is\_irq\_entry()

```
int L4vcpu::Vcpu::is_irq_entry () const [inline]
```

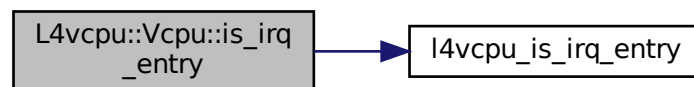
Return whether the entry reason was an IRQ/IPC message.

return 0 if not, !=0 otherwise.

Definition at line 195 of file [vcpu](#).

References [l4vcpu\\_is\\_irq\\_entry\(\)](#).

Here is the call graph for this function:



#### 15.334.2.12 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 188 of file `vcpu`.

References `l4vcpu_is_page_fault_entry()`.

Here is the call graph for this function:



#### 15.334.2.13 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 202 of file `vcpu`.

References `l4_vcpu_state_t::r`.

**15.334.2.14** `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 209 of file `vcpu`.

References `l4_vcpu_state_t::r`.

**15.334.2.15** `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 113 of file `vcpu`.

**15.334.2.16** `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 123 of file `vcpu`.

References `l4_vcpu_state_t::saved_state`.



**15.334.2.17 state()** [1/2]

```
State* L4vcpu::Vcpu::state () throw () [inline]
```

Get state word.

**Returns**

Pointer to state word in the vCPU

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

**15.334.2.18 state()** [2/2]

```
State L4vcpu::Vcpu::state () const throw () [inline]
```

Get state word.

**Returns**

Pointer to state word in the vCPU

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

References [l4\\_vcpu\\_state\\_t::state](#).

**15.334.2.19 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 181 of file [vcpu](#).

References [l4\\_vcpu\\_state\\_t::user\\_task](#).

**15.334.2.20 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

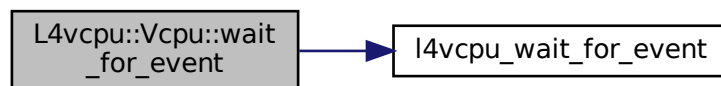
|                         |                                                                                       |
|-------------------------|---------------------------------------------------------------------------------------|
| <i>utcb</i>             | The UTCB to use.                                                                      |
| <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 173 of file [vcpu](#).

References [l4vcpu\\_wait\\_for\\_event\(\)](#).

Here is the call graph for this function:



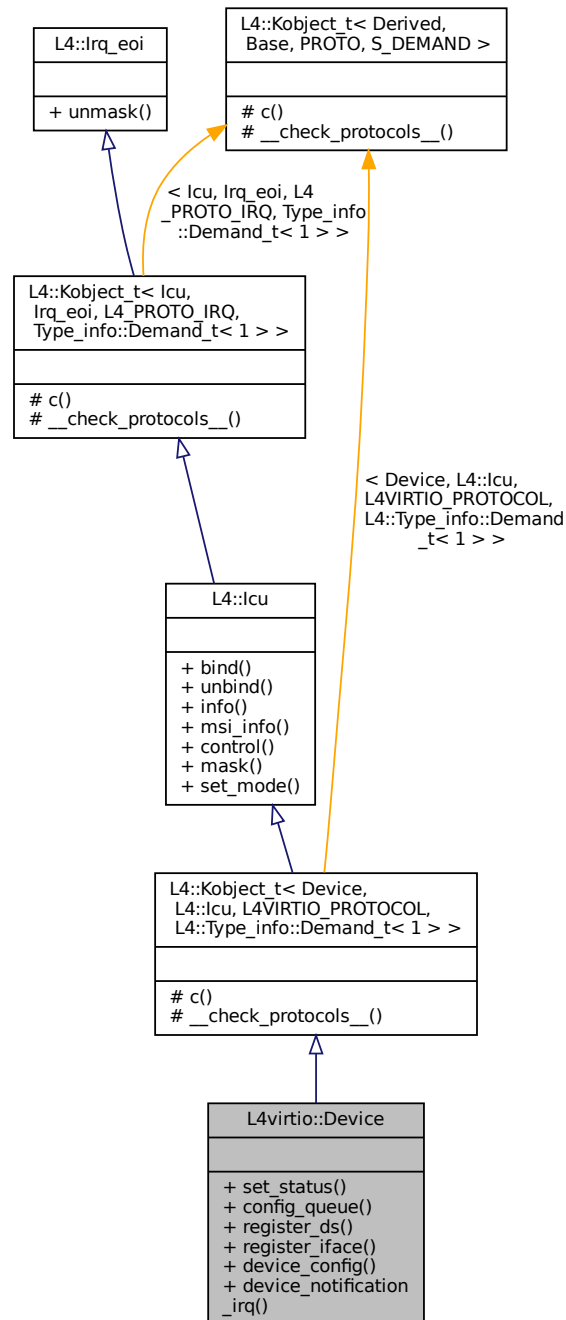
The documentation for this class was generated from the following file:

- [l4/vcpu/vcpu](#)

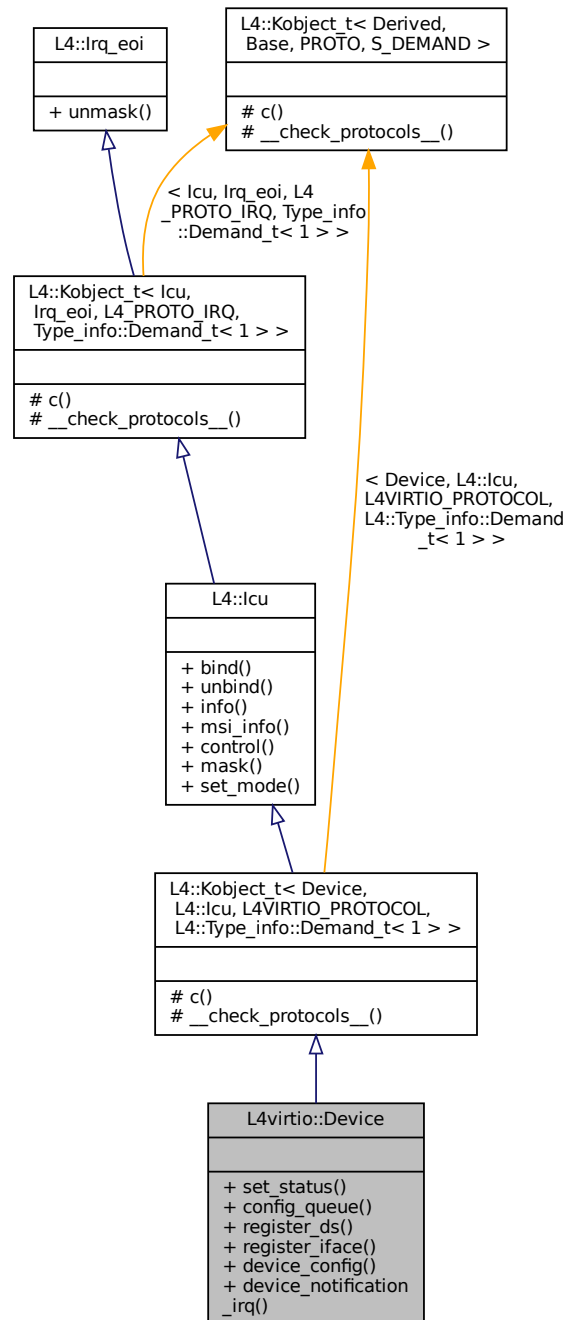
## 15.335 L4virtio::Device Class Reference

IPC interface for virtio over [L4](#) IPC.

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 `register_iface` (`L4::lpc::Cap< L4::Triggerable >` guest\_irq, `L4::lpc::Out< L4::Cap< L4::Triggerable > >` host\_irq, `L4::lpc::Out< L4::Cap< L4Re::Dataspace > >` config\_ds)  
*Register client to the L4-VIRTIO device.*
- 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.*

## Additional Inherited Members

### 15.335.1 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`.

### 15.335.2 Member Function Documentation

#### 15.335.2.1 `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

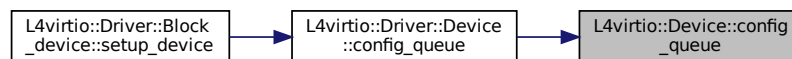
|                    |                                             |
|--------------------|---------------------------------------------|
| <code>queue</code> | Queue index for the queue to be configured. |
|--------------------|---------------------------------------------|

## Return values

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>0</code>          | on success.                                   |
| <code>-L4_EIO</code>    | The queue's status is invalid.                |
| <code>-L4_ERANGE</code> | The queue index exceeds the number of queues. |
| <code>-L4_EINVAL</code> | Otherwise.                                    |

Referenced by [L4virtio::Driver::Device::config\\_queue\(\)](#).

Here is the caller graph for this function:



### 15.335.2.2 device\_config()

```

long L4virtio::Device::device_config (
 L4::Ipc::Out< L4::Cap< L4Re::Dataspace > > config_ds,
 l4_addr_t * ds_offset)

```

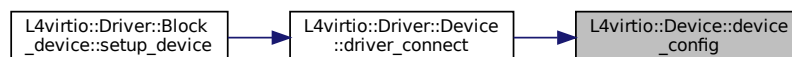
Get the dataspace with the [L4virtio](#) configuration page.

## Parameters

|                        |                                                                                               |
|------------------------|-----------------------------------------------------------------------------------------------|
| <code>config_ds</code> | Capability for receiving the dataspace capability for the shared L4-VIRTIO config data space. |
| <code>ds_offset</code> | 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:



### 15.335.2.3 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.          |
| <i>out</i> | <i>irq</i>   | Triggerable for the given index. |

#### Return values

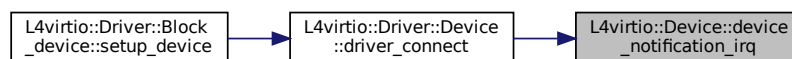
|                  |                                           |
|------------------|-------------------------------------------|
| <i>L4_EOK</i>    | Success.                                  |
| <i>L4_ENOSYS</i> | IRQ notification not supported by device. |
| <i>&lt;0</i>     | Other error.                              |

An index is only guaranteed to return an IRQ object when the index is set in one of the device notify index fields. The device must return the same interrupt for a given index as long as the index is in use. If an index disappears as a result of a configuration change and then is reused later, the interrupt is not guaranteed to be the same.

Interrupts must always be rerequested after a device reset.

Referenced by [L4virtio::Driver::Device::driver\\_connect\(\)](#).

Here is the caller graph for this function:



### 15.335.2.4 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

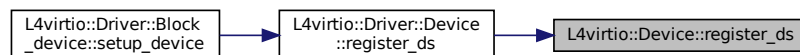
|               |                                                                                                                                                                                     |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>ds_cap</i> | Data-space 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                                                                                                                                              |

## Returns

0 on success, < 0 on error

Referenced by [L4virtio::Driver::Device::register\\_ds\(\)](#).

Here is the caller graph for this function:



## 15.335.2.5 register\_iface()

```

long L4virtio::Device::register_iface (
 L4::Ipc::Cap< L4::Triggerable > guest_irq,
 L4::Ipc::Out< L4::Cap< L4::Triggerable > > host_irq,
 L4::Ipc::Out< L4::Cap< L4Re::Dataspace > > config_ds)

```

Register client to the L4-VIRTIO device.

## Parameters

|                  |                                                                                               |
|------------------|-----------------------------------------------------------------------------------------------|
| <i>guest_irq</i> | IRQ capability for valid IRQ object for device-to-driver notifications.                       |
| <i>host_irq</i>  | Capability selector for receiving the driver-to-device notifications IRQ capability.          |
| <i>config_ds</i> | Capability for receiving the dataspace capability for the shared L4-VIRTIO config data space. |

## Return values

|                   |                                                                                                         |
|-------------------|---------------------------------------------------------------------------------------------------------|
| <i>L4_EOK</i>     | Success.                                                                                                |
| <i>-L4_ENOSYS</i> | This interface is no longer supported by the server. Use <code>get_device_config()</code> etc. instead. |

**Deprecated** Use [device\\_config\(\)](#), [device\\_notification\\_irq\(\)](#) and [lcu::bind\(\)](#) instead.



### 15.335.2.6 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

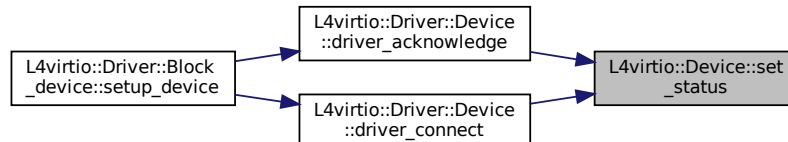
|   |             |
|---|-------------|
| 0 | on success. |
|---|-------------|

#### 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:



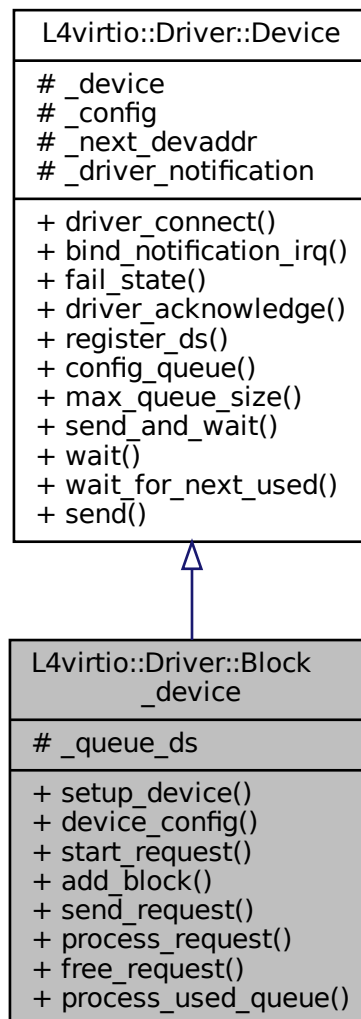
The documentation for this class was generated from the following file:

- `l4/l4virtio/l4virtio`

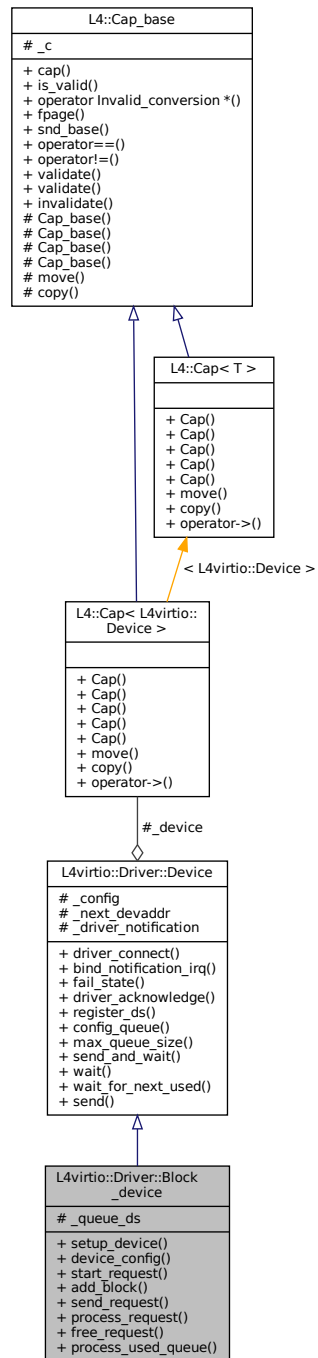
## 15.336 L4virtio::Driver::Block\_device Class Reference

Simple class for accessing a virtio block device synchronously.

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_uint32_t` `fmask0=-1U`, `l4_uint32_t` `fmask1=-1U`)  
*Setup a connection to a 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.*

### 15.336.1 Detailed Description

Simple class for accessing a virtio block device synchronously.

Definition at line 346 of file [virtio-block](#).

### 15.336.2 Member Function Documentation

#### 15.336.2.1 `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> | <code>Handle</code> to request previously set up with <code>start_request()</code> . |
| <i>addr</i>   | Address of data block in device address space.                                       |
| <i>size</i>   | Size of data block.                                                                  |

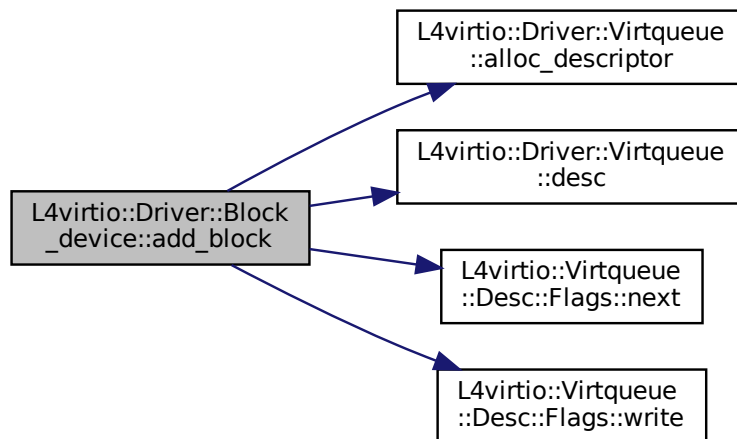
##### Return values

|                         |                                            |
|-------------------------|--------------------------------------------|
| <code>L4_OK</code>      | Block was successfully added.              |
| <code>-L4_EAGAIN</code> | No descriptors available. Try again later. |

Definition at line 524 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::Flags::write](#).

Here is the call graph for this function:



### 15.336.2.2 process\_request()

```
int L4virtio::Driver::Block_device::process_request (
 Handle handle) [inline]
```

Process request synchronously.

#### Parameters

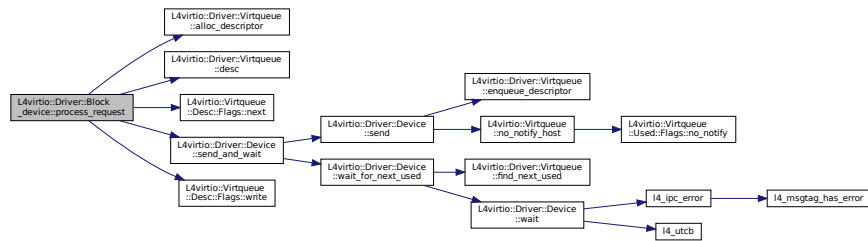
|            |                               |
|------------|-------------------------------|
| <i>req</i> | Request to send to the device |
|------------|-------------------------------|

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 595 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:



### 15.336.2.3 process\_used\_queue()

```
void L4virtio::Driver::Block_device::process_used_queue () [inline]
```

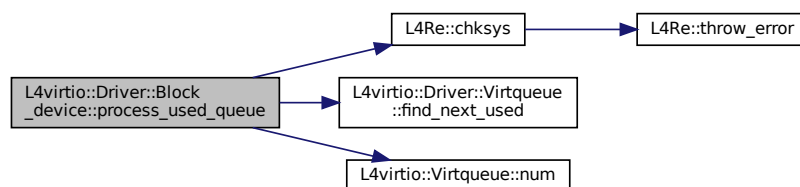
Process and free all items in the used queue.

If the request has a callback registered it is called after the item has been removed from the queue.

Definition at line 646 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:



### 15.336.2.4 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

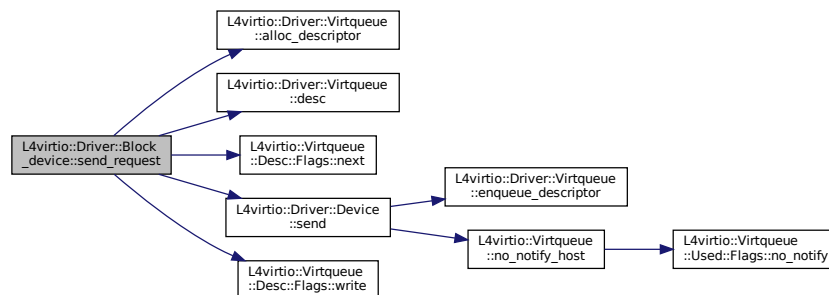
|                   |                                            |
|-------------------|--------------------------------------------|
| <i>L4_OK</i>      | Request was successfully scheduled.        |
| <i>-L4_EAGAIN</i> | No descriptors available. Try again later. |

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 560 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:



## 15.336.2.5 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_uint32_t fmask0 = -1U,
 l4_uint32_t fmask1 = -1U) [inline]

```

Setup a connection to a device and set up shared memory.

## Parameters

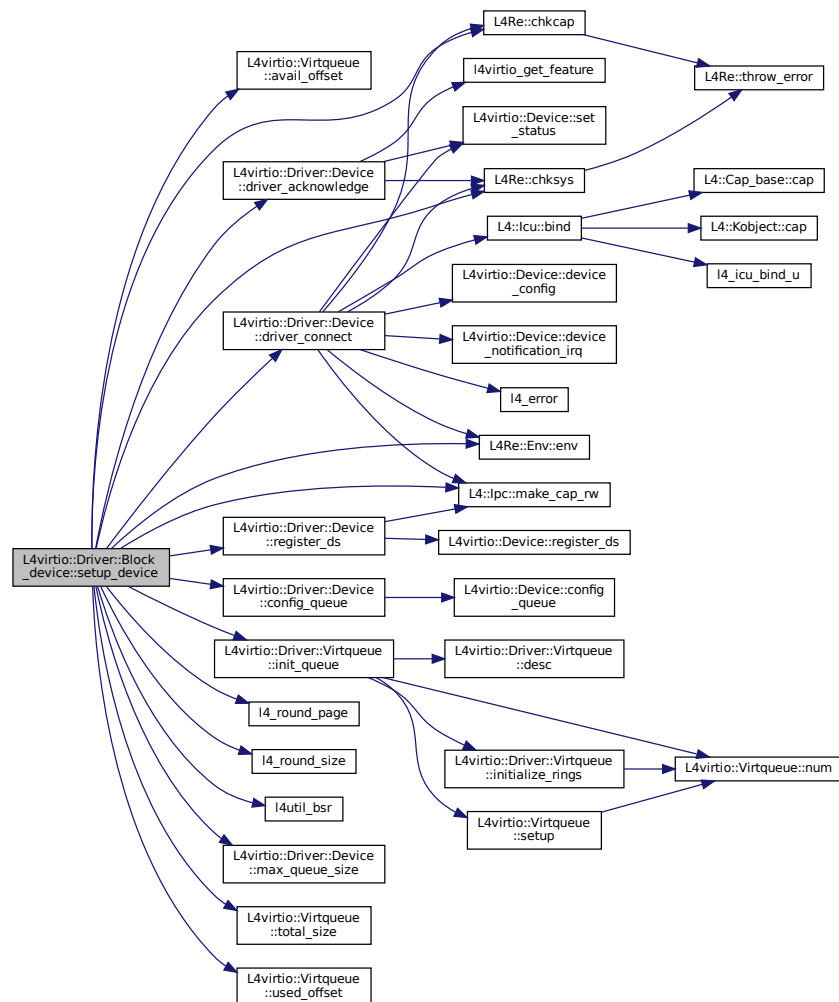
|                     |                                                        |
|---------------------|--------------------------------------------------------|
| <i>srvcap</i>       | IPC capability of the channel to the server.           |
| <i>usermem</i>      | Size of additional memory to share with device.        |
| <i>userdata</i>     | Pointer to the region of user-usable memory.           |
| <i>user_devaddr</i> | Address of user-usable memory in device address space. |
| <i>fmask0</i>       | Feature bits 0..31 that the driver supports.           |
| <i>fmask1</i>       | Feature bits 32..63 that the driver supports.          |

This function starts a hand shake with the device and sets up the virtqueues for communication and the additional data structures for the block device. It will also allocate and share additional memory that the caller then can use freely, i.e. normally this memory would be used as a reception buffer. The caller may also decide to not make use of this convenience function and request 0 bytes in usermem. Then it has to allocate the block buffers for sending/receiving payload manually and share them using `register_ds()`.

Definition at line 397 of file `virtio-block`.

References `L4virtio::Virtqueue::avail_offset()`, `L4Re::chkcap()`, `L4Re::chksys()`, `L4virtio::Driver::Device::config_queue()`, `L4Re::Mem_alloc::Continuous`, `L4virtio::Driver::Device::driver_acknowledge()`, `L4virtio::Driver::Device::driver_connect()`, `L4Re::Env::env()`, `L4virtio::Driver::Virtqueue::init_queue()`, `L4_EINVAL`, `L4_ENODEV`, `L4_PAGESHIFT`, `I4_round_page()`, `I4_round_size()`, `I4util_bsr()`, `L4VIRTIO_ID_BLOCK`, `L4::lpc::make_cap_rw()`, `L4virtio::Driver::Device::max_queue_size()`, `L4Re::Mem_alloc::Pinned`, `L4virtio::Driver::Device::register_ds()`, `L4Re::Rm::F::Search_addr`, `L4virtio::Virtqueue::total_size()`, and `L4virtio::Virtqueue::used_offset()`.

Here is the call graph for this function:





### 15.336.2.6 start\_request()

```
Handle L4virtio::Driver::Block_device::start_request (
 l4_uint64_t sector,
 l4_uint32_t type,
 Callback callback) [inline]
```

Start the setup of a new request.

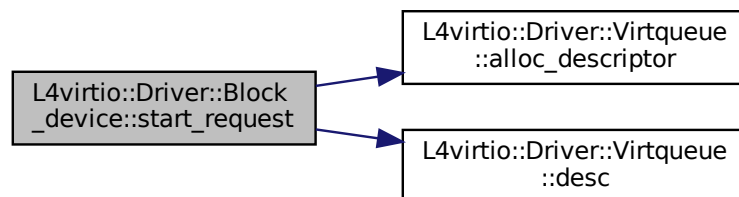
#### Parameters

|                 |                                                                                    |
|-----------------|------------------------------------------------------------------------------------|
| <i>sector</i>   | First sector to write to/read from.                                                |
| <i>type</i>     | Request type.                                                                      |
| <i>callback</i> | Function to call, when the request is finished. May be 0 for synchronous requests. |

Definition at line 486 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:



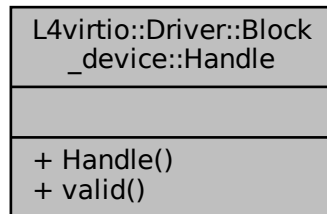
The documentation for this class was generated from the following file:

- `l4/l4virtio/client/virtio-block`

## 15.337 L4virtio::Driver::Block\_device::Handle Class Reference

[Handle](#) to an ongoing request.

Collaboration diagram for L4virtio::Driver::Block\_device::Handle:



### 15.337.1 Detailed Description

[Handle](#) to an ongoing request.

Definition at line [366](#) of file [virtio-block](#).

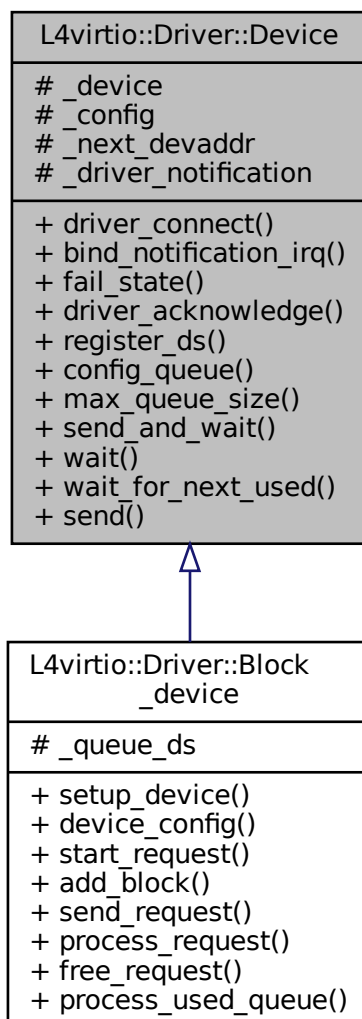
The documentation for this class was generated from the following file:

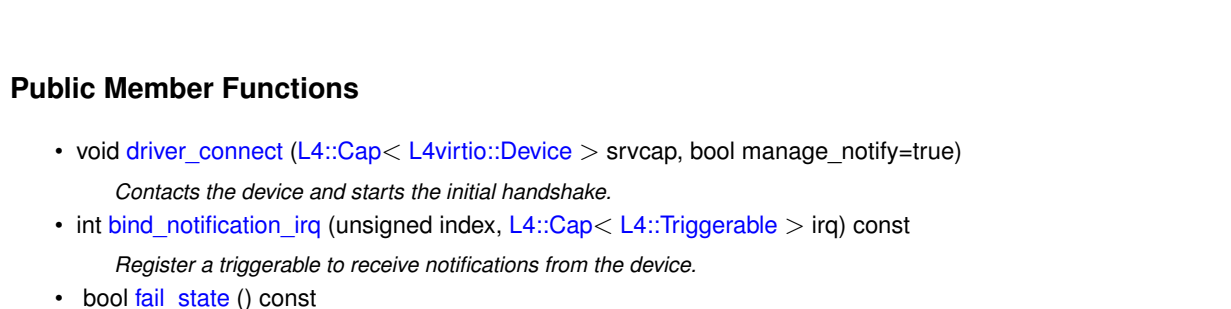
- `l4/l4virtio/client/virtio-block`

## 15.338 L4virtio::Driver::Device Class Reference

Client-side implementation for a general virtio device.

Inheritance diagram for L4virtio::Driver::Device:





- Return true if the device is in a fail state.*

  - 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) 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.*

### 15.338.1 Detailed Description

Client-side implementation for a general virtio device.

Definition at line 35 of file [virtio-block](#).

### 15.338.2 Member Function Documentation

#### 15.338.2.1 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.

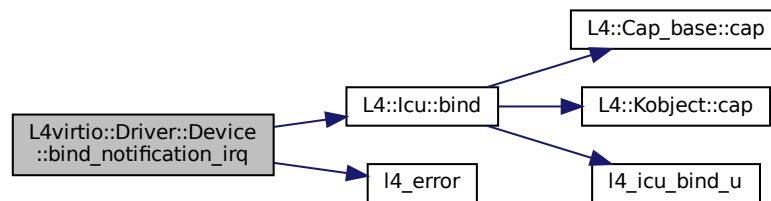
##### Parameters

|     |              |                                            |
|-----|--------------|--------------------------------------------|
|     | <i>index</i> | Index of the interrupt.                    |
| out | <i>irq</i>   | Triggerable to register for notifications. |

Definition at line 133 of file [virtio-block](#).

References [L4::lcu::bind\(\)](#), and [l4\\_error\(\)](#).

Here is the call graph for this function:



### 15.338.2.2 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

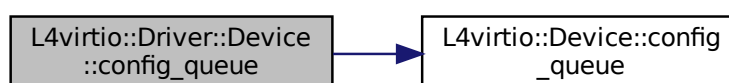
|                   |                                                   |
|-------------------|---------------------------------------------------|
| <i>num</i>        | Number of queue to configure.                     |
| <i>size</i>       | Size of rings in the queue, must be a power of 2) |
| <i>desc_addr</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 203 of file [virtio-block](#).

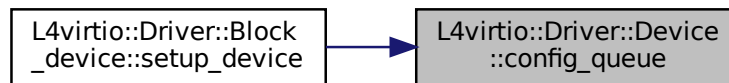
References [L4virtio::Device::config\\_queue\(\)](#).

Referenced by [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.338.2.3 driver\_acknowledge()

```
int L4virtio::Driver::Device::driver_acknowledge () [inline]
```

Finalize handshake with the device.

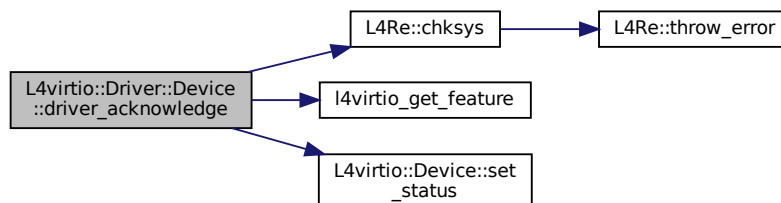
Must be called after all queues have been set up and before the first request is sent. It is still possible to add more shared dataspace after the handshake has been finished.

Definition at line 147 of file [virtio-block](#).

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::Block\\_device::setup\\_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.338.2.4 driver\_connect()

```
void L4virtio::Driver::Device::driver_connect (
 L4::Cap< L4virtio::Device > srvcap,
 bool manage_notify = true) [inline]
```

Contacts the device and starts the initial handshake.

#### Parameters

|                      |                                                                  |
|----------------------|------------------------------------------------------------------|
| <i>srvcap</i>        | Capability for device communication.                             |
| <i>manage_notify</i> | Set up a semaphore for notifications from the device. See below. |

#### Exceptions

|                                   |                             |
|-----------------------------------|-----------------------------|
| <a href="#">L4::Runtime_error</a> | if the initialisation fails |
|-----------------------------------|-----------------------------|

This function contacts the server, sets up the notification channels and the configuration dataspace. After this is done, the caller can set up any dataspace it needs. The initialisation then needs to be finished by calling [driver\\_acknowledge\(\)](#).

Per default this function creates and registers a semaphore for receiving notification from the device. This semaphore is used in the blocking functions [send\\_and\\_wait\(\)](#), [wait\(\)](#) and [next\\_used\(\)](#).

When `manage_notify` is false, then the caller may manually register and handle notification interrupts from the device. This is for example useful, when the client runs in an application with a server loop.

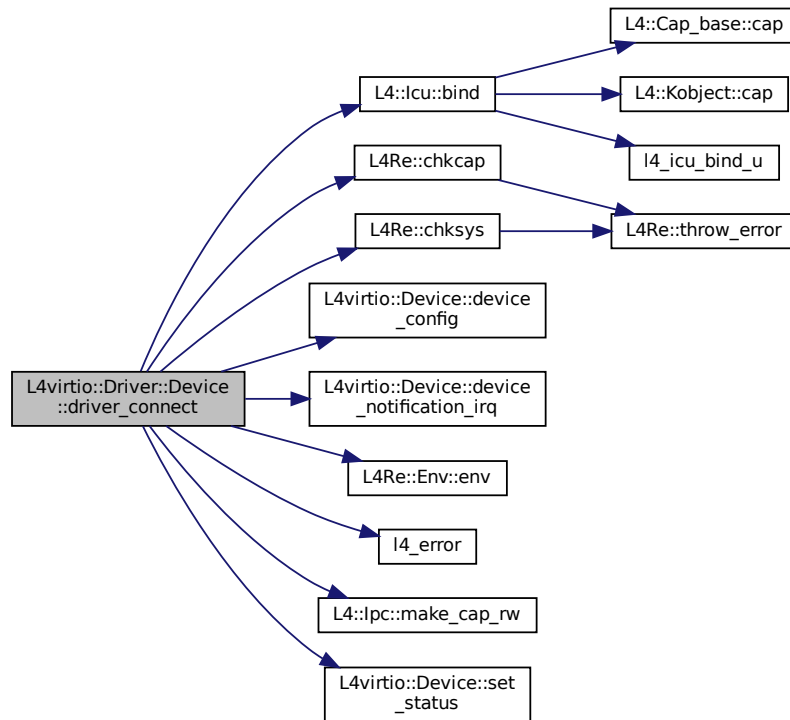
Definition at line 60 of file [virtio-block](#).

References [L4::lcu::bind\(\)](#), [L4Re::chkcap\(\)](#), [L4Re::chksys\(\)](#), [L4virtio::Device::device\\_config\(\)](#), [L4virtio::Device::device\\_notification\\_irq\(\)](#), [L4Re::Env::env\(\)](#), [L4\\_EINVAL](#), [L4\\_EIO](#), [L4\\_ENODEV](#), [l4\\_error\(\)](#), [L4\\_PAGEMASK](#), [L4\\_PAGESHIFT](#), [L4\\_PAGESIZE](#), [L4\\_SUPERPAGESIZE](#), [L4VIRTIO\\_STATUS\\_ACKNOWLEDGE](#), [L4VIRTIO\\_STATUS\\_DRIVER](#), [L4::lpc::make\\_cap\\_rw\(\)](#), [L4Re::Rm::F::Search\\_addr](#), and [L4virtio::Device::set\\_status\(\)](#).

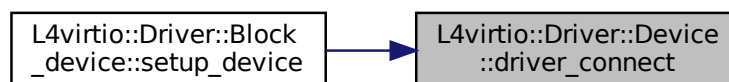
Referenced by [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#).



Here is the call graph for this function:



Here is the caller graph for this function:



### 15.338.2.5 max\_queue\_size()

```
int L4virtio::Driver::Device::max_queue_size (
 int num) const [inline]
```

Maximum queue size allowed by the device.

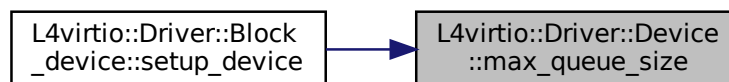
## Parameters

|            |                                                          |
|------------|----------------------------------------------------------|
| <i>num</i> | Number of queue for which to determine the maximum size. |
|------------|----------------------------------------------------------|

Definition at line 221 of file [virtio-block](#).

Referenced by [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#).

Here is the caller graph for this function:



### 15.338.2.6 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.

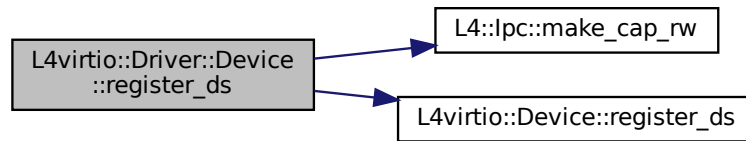
When communicating with the device, addresses must be given with respect to the device address space. This is not the same as the virtual address space of the client in order to not leak information about the address space layout.

Definition at line 187 of file [virtio-block](#).

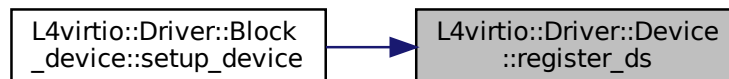
References [L4::lpc::make\\_cap\\_rw\(\)](#), and [L4virtio::Device::register\\_ds\(\)](#).

Referenced by [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.338.2.7 send()

```
void L4virtio::Driver::Device::send (
 Virtqueue & queue,
 l4_uint16_t descno) [inline]
```

Send a request to the device.

#### Parameters

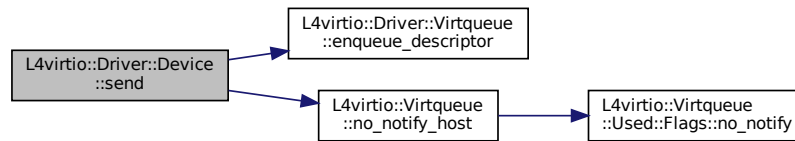
|               |                                                         |
|---------------|---------------------------------------------------------|
| <i>queue</i>  | Queue that contains the request in its descriptor table |
| <i>descno</i> | Index of first entry in descriptor table where          |

Definition at line 298 of file [virtio-block](#).

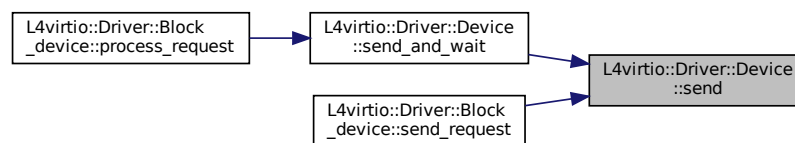
References [L4virtio::Driver::Virtqueue::enqueue\\_descriptor\(\)](#), and [L4virtio::Virtqueue::no\\_notify\\_host\(\)](#).

Referenced by [send\\_and\\_wait\(\)](#), and [L4virtio::Driver::Block\\_device::send\\_request\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.338.2.8 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

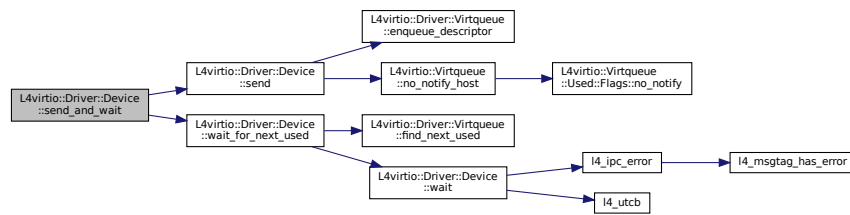
`driver_connect()` was called with `manage_notify`.

Definition at line 238 of file `virtio-block`.

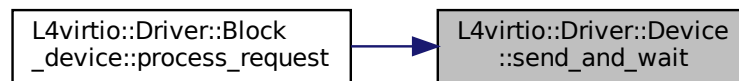
References `L4_EINVAL`, `L4_EOK`, `send()`, and `wait_for_next_used()`.

Referenced by `L4virtio::Driver::Block_device::process_request()`.

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.338.2.9 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

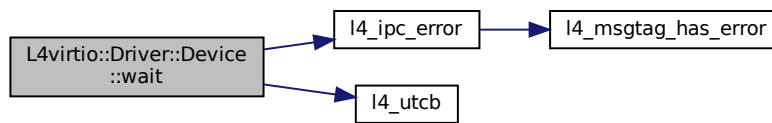
[driver\\_connect\(\)](#) was called with `manage_notify`.

Definition at line 259 of file [virtio-block](#).

References [L4\\_EEXIST](#), [I4\\_ipc\\_error\(\)](#), and [I4\\_utcb\(\)](#).

Referenced by [wait\\_for\\_next\\_used\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 15.338.2.10 wait\_for\_next\_used()

```
int L4virtio::Driver::Device::wait_for_next_used (
 Virtqueue & queue) const [inline]
```

Wait for the next item to arrive in the used queue and return it.

##### Return values

|          |                                                    |
|----------|----------------------------------------------------|
| $\geq 0$ | Descriptor number of item removed from used queue. |
| $< 0$    | IPC error while waiting for notification.          |

The call blocks until the next item is available in the used queue.

##### Precondition

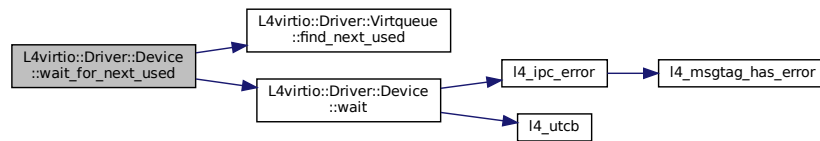
`driver_connect()` was called with `manage_notify`.

Definition at line 277 of file `virtio-block`.

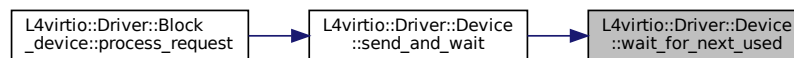
References `L4virtio::Driver::Virtqueue::find_next_used()`, and `wait()`.

Referenced by `send_and_wait()`.

Here is the call graph for this function:



Here is the caller graph for this function:



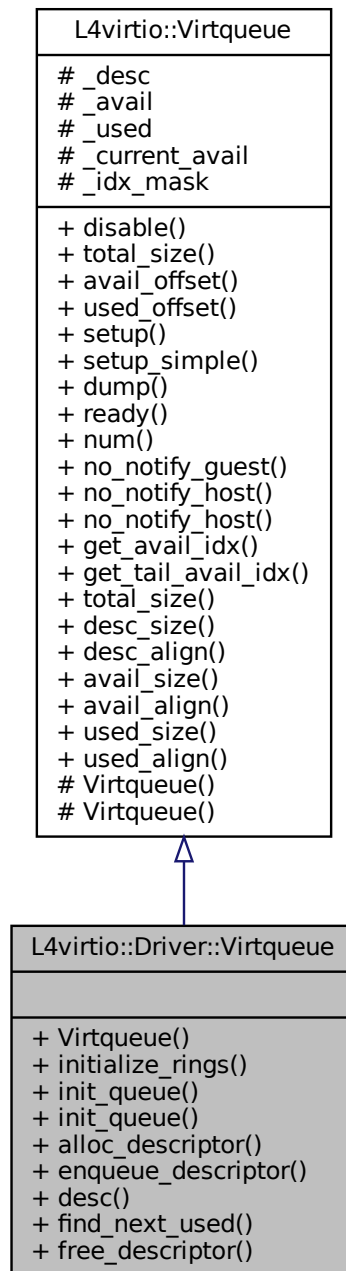
The documentation for this class was generated from the following file:

- `I4/I4virtio/client/virtio-block`

## 15.339 L4virtio::Driver::Virtqueue Class Reference

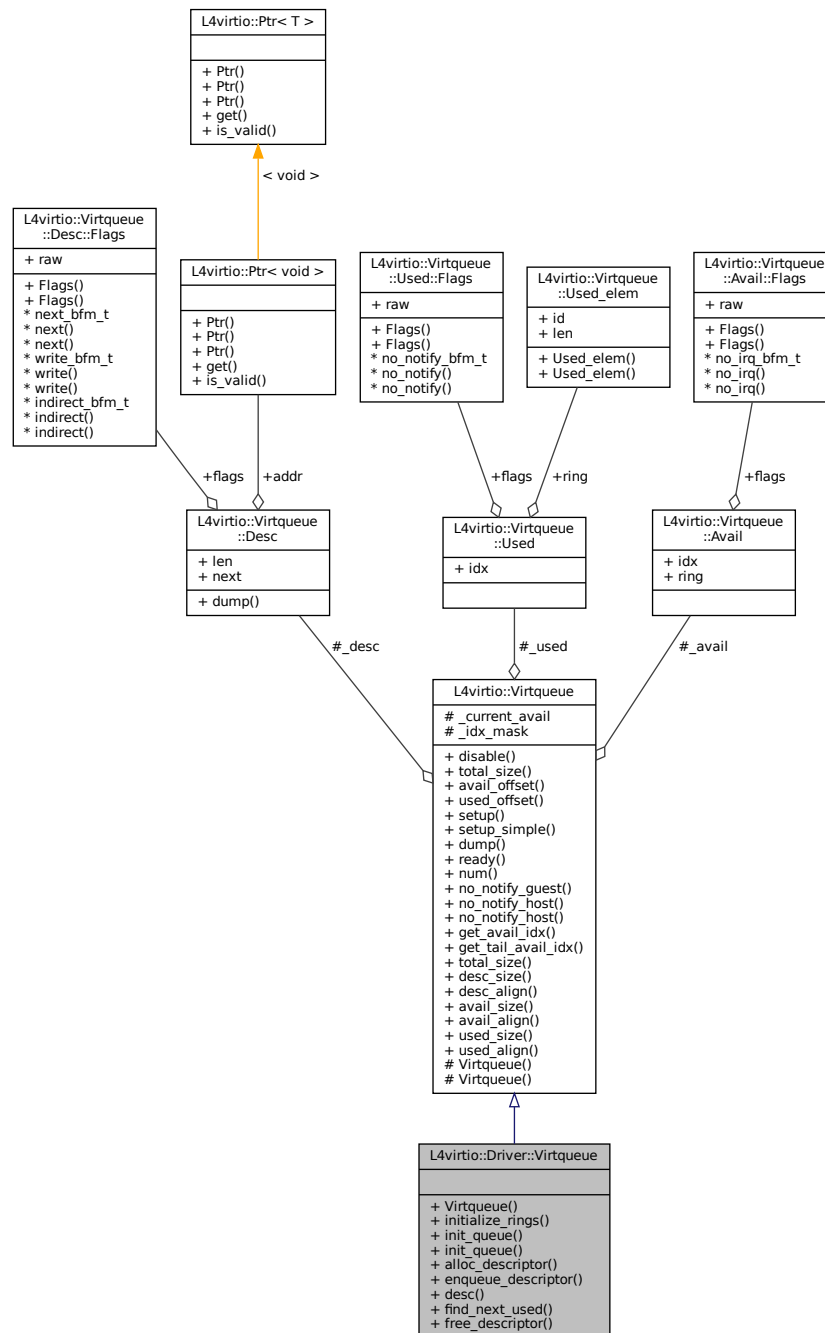
Driver-side implementation of a [Virtqueue](#).

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.*

## Additional Inherited Members

### 15.339.1 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 [475](#) of file [virtqueue](#).

### 15.339.2 Member Function Documentation

#### 15.339.2.1 [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.

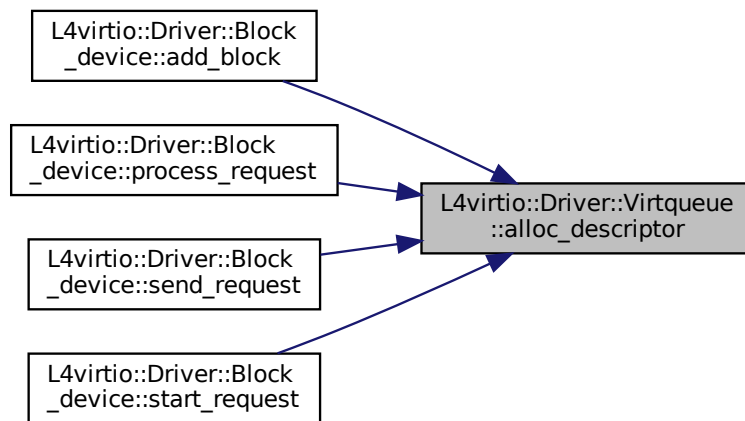
Note: the implementation uses  $(2^{16} - 1)$  as the end of queue marker. That means that the final entry in the queue can not be allocated iff the queue size is  $2^{16}$ .

Definition at line 563 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::Block\\_device::send\\_request\(\)](#), and [L4virtio::Driver::Block\\_device::start\\_request\(\)](#).

Here is the caller graph for this function:



### 15.339.2.2 desc()

```
Desc& L4virtio::Driver::Virtqueue::desc (
 14_uint16_t descno) [inline]
```

Return a reference to a descriptor in the descriptor table.

#### Parameters

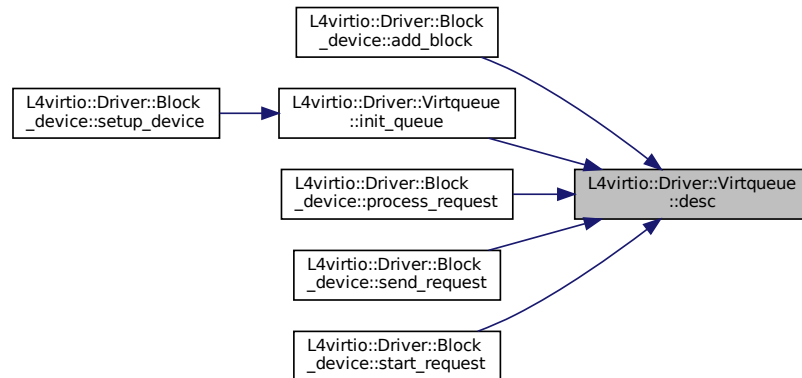
|               |                                                           |
|---------------|-----------------------------------------------------------|
| <i>descno</i> | Index of the descriptor, expected to be in correct range. |
|---------------|-----------------------------------------------------------|

Definition at line 595 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\(\)](#), and [L4virtio::Driver::Block\\_device::start\\_request\(\)](#).

Here is the caller graph for this function:



### 15.339.2.3 enqueue\_descriptor()

```
void L4virtio::Driver::Virtqueue::enqueue_descriptor (
 l4_uint16_t descno) [inline]
```

Enqueue a descriptor in the available ring.

#### Parameters

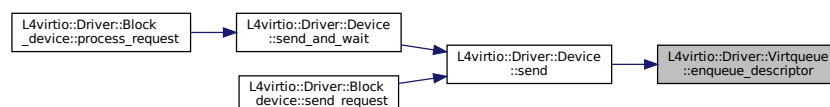
|               |                                          |
|---------------|------------------------------------------|
| <i>descno</i> | Index of the head descriptor to enqueue. |
|---------------|------------------------------------------|

Definition at line 579 of file [virtqueue](#).

References [L4virtio::Virtqueue::\\_avail](#), [L4virtio::Virtqueue::\\_idx\\_mask](#), [L4virtio::Virtqueue::Avail::idx](#), and [L4virtio::Virtqueue::Avail::ring](#).

Referenced by [L4virtio::Driver::Device::send\(\)](#).

Here is the caller graph for this function:



### 15.339.2.4 find\_next\_used()

```
l4_uint16_t L4virtio::Driver::Virtqueue::find_next_used (
 l4_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

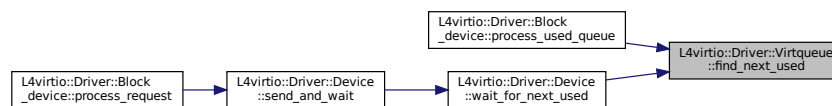
Index of the head or Virtqueue::Eoq if no used element is currently available.

Definition at line 614 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:



### 15.339.2.5 free\_descriptor()

```
void L4virtio::Driver::Virtqueue::free_descriptor (
 l4_uint16_t head,
 l4_uint16_t tail) [inline]
```

Free a chained list of descriptors in the descriptor queue.

#### Parameters

|      |                                                     |
|------|-----------------------------------------------------|
| head | Index of the first element in the descriptor chain. |
| tail | 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 636 of file [virtqueue](#).

References [L4virtio::Virtqueue::\\_desc](#), [L4virtio::Virtqueue::\\_idx\\_mask](#), and [L4virtio::Virtqueue::Desc::next](#).

### 15.339.2.6 `init_queue()` [1/2]

```
void L4virtio::Driver::Virtqueue::init_queue (
 unsigned num,
 void * base) [inline]
```

Initialize this virtqueue.

#### Parameters

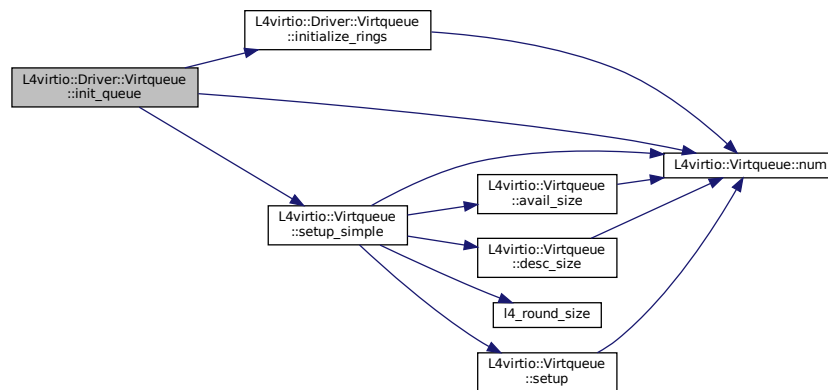
|             |                                                                                                              |
|-------------|--------------------------------------------------------------------------------------------------------------|
| <i>num</i>  | The number of entries in the descriptor 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 542 of file [virtqueue](#).

References [initialize\\_rings\(\)](#), [L4virtio::Virtqueue::num\(\)](#), and [L4virtio::Virtqueue::setup\\_simple\(\)](#).

Here is the call graph for this function:



### 15.339.2.7 `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 <code>Desc_align</code> aligned and at least <code>desc_size(num)</code> bytes in size.) |
| <i>avail</i> | The address of the available ring. (Must be <code>Avail_align</code> aligned and at least <code>avail_size(num)</code> bytes in size.) |
| <i>used</i>  | The address of the used ring. (Must be <code>Used_align</code> aligned and at least <code>used_size(num)</code> bytes in size.)        |

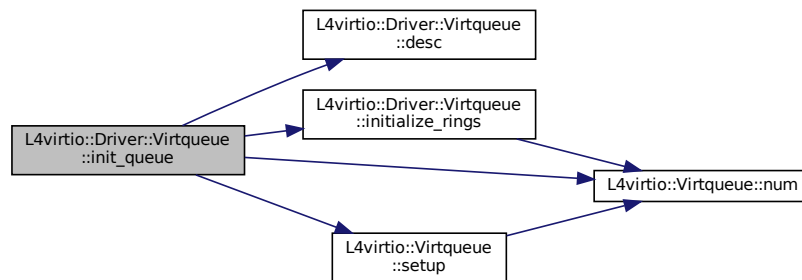
This function sets up the memory and initializes the freelist.

Definition at line 527 of file [virtqueue](#).

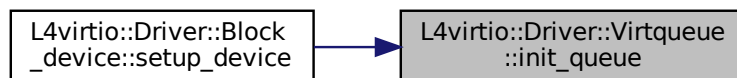
References [desc\(\)](#), [initialize\\_rings\(\)](#), [L4virtio::Virtqueue::num\(\)](#), and [L4virtio::Virtqueue::setup\(\)](#).

Referenced by [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.339.2.8 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

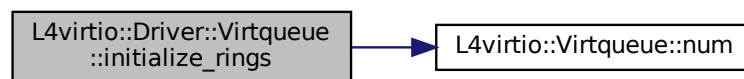
The queue must be set up correctly with [setup\(\)](#) or [setup\\_simple\(\)](#).

Definition at line 499 of file [virtqueue](#).

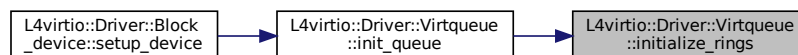
References [L4virtio::Virtqueue::\\_avail](#), [L4virtio::Virtqueue::\\_desc](#), [L4virtio::Virtqueue::\\_used](#), [L4virtio::Virtqueue::Avail::idx](#), [L4virtio::Virtqueue::Used::idx](#), [L4virtio::Virtqueue::Desc::next](#), and [L4virtio::Virtqueue::num\(\)](#).

Referenced by [init\\_queue\(\)](#).

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:

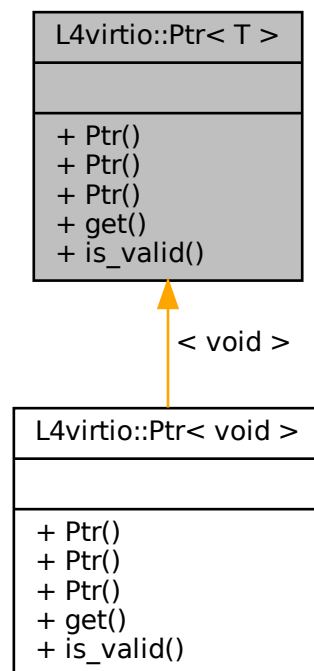
- `l4/l4virtio/virtqueue`

## 15.340 L4virtio::Ptr< T > Class Template Reference

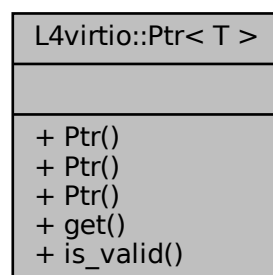
Pointer used in virtio descriptors.



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

### 15.340.1 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 [56](#) of file [virtqueue](#).

### 15.340.2 Member Enumeration Documentation

#### 15.340.2.1 Invalid\_type

```
template<typename T >
enum L4virtio::Ptr::Invalid_type
```

Type for making an invalid (NULL) [Ptr](#).

Enumerator

|         |                                                    |
|---------|----------------------------------------------------|
| Invalid | Use to set a <a href="#">Ptr</a> to invalid (NULL) |
|---------|----------------------------------------------------|

Definition at line [60](#) of file [virtqueue](#).

### 15.340.3 Member Function Documentation

#### 15.340.3.1 get()

```
template<typename T >
l4_uint64_t L4virtio::Ptr< T >::get () const [inline]
```

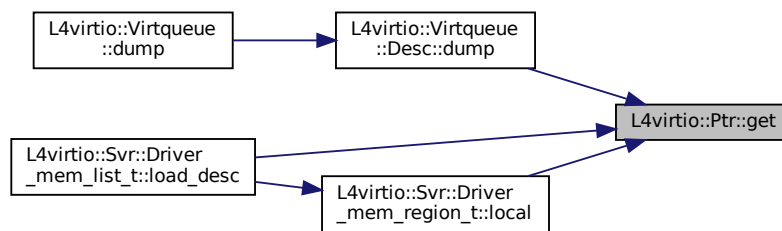
**Returns**

The raw 64bit address of the pointer.

Definition at line 71 of file [virtqueue](#).

Referenced by [L4virtio::Virtqueue::Desc::dump\(\)](#), [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:

**15.340.3.2 is\_valid()**

```

template<typename T >
bool L4virtio::Ptr< T >::is_valid () const [inline]

```

**Returns**

true if the pointer is invalid (NULL).

Definition at line 74 of file [virtqueue](#).

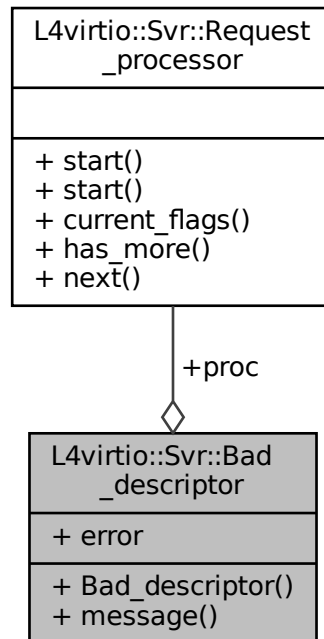
The documentation for this class was generated from the following file:

- `I4/I4virtio/virtqueue`

## 15.341 L4virtio::Svr::Bad\_descriptor Struct Reference

Exception used by Queue to indicate descriptor errors.

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.*

## 15.341.1 Detailed Description

Exception used by Queue to indicate descriptor errors.

Definition at line 325 of file [virtio](#).

## 15.341.2 Member Enumeration Documentation

### 15.341.2.1 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](#).

## 15.341.3 Constructor & Destructor Documentation

### 15.341.3.1 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](#).

## 15.341.4 Member Function Documentation

### 15.341.4.1 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](#).

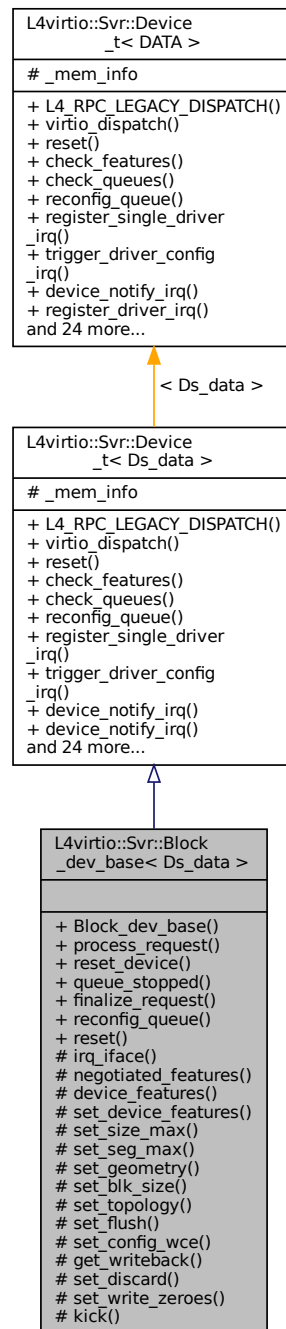
The documentation for this struct was generated from the following file:

- [l4/l4virtio/server/virtio](#)

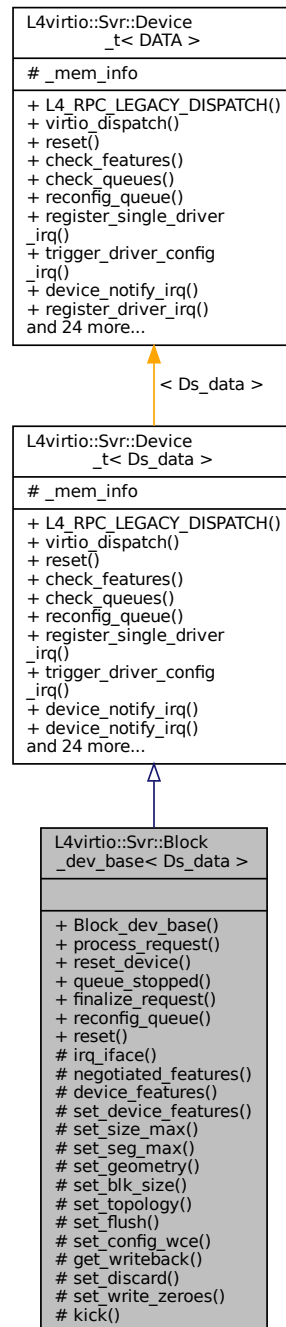
## 15.342 L4virtio::Svr::Block\_dev\_base< Ds\_data > Class Template Reference

Base class for virtio block devices.

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)  
*callback for client queue-config request*
- void `reset` ()  
*reset callback, called for doing a device reset*

## 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 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

### 15.342.1 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`.

## 15.342.2 Constructor & Destructor Documentation

### 15.342.2.1 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.

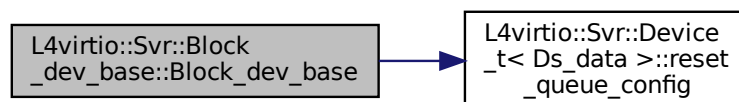
#### Parameters

|                   |                                                       |
|-------------------|-------------------------------------------------------|
| <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:



## 15.342.3 Member Function Documentation

### 15.342.3.1 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

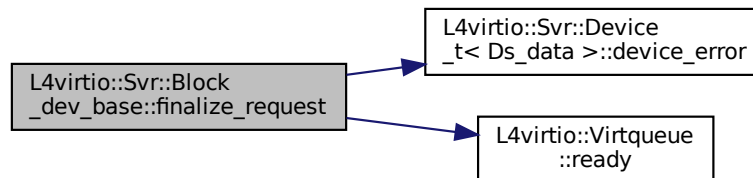
|               |                                                |
|---------------|------------------------------------------------|
| <i>req</i>    | Pointer to request that has finished.          |
| <i>sz</i>     | Number of bytes consumed.                      |
| <i>status</i> | Status of request (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:



### 15.342.3.2 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](#).

### 15.342.3.3 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.

**15.342.3.4 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](#).

**15.342.3.5 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 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](#).

**15.342.3.6 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](#).

### 15.342.3.7 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](#).

### 15.342.3.8 set\_topology()

```
template<typename Ds_data >
void L4virtio::Svr::Block_dev_base< Ds_data >::set_topology (
 14_uint8_t physical_block_exp,
 14_uint8_t alignment_offset,
 14_uint32_t min_io_size,
 14_uint32_t opt_io_size) [inline], [protected]
```

Sets the I/O alignment information reported back to the client.

## Parameters

|                           |                                                   |
|---------------------------|---------------------------------------------------|
| <i>physical_block_exp</i> | Number of logical blocks per physical block(log2) |
| <i>alignment_offset</i>   | Offset of the first aligned logical block         |
| <i>min_io_size</i>        | Suggested minimum I/O size in blocks              |
| <i>opt_io_size</i>        | Optimal I/O size in blocks                        |

Definition at line 366 of file [virtio-block](#).

### 15.342.3.9 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](#).

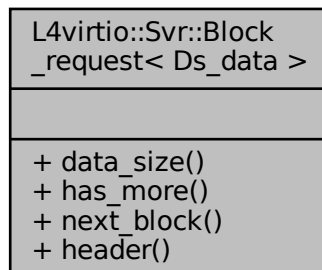
The documentation for this class was generated from the following file:

- l4/l4virtio/server/virtio-block

## 15.343 L4virtio::Svr::Block\_request< Ds\_data > Class Template Reference

A request to read or write data.

Collaboration diagram for L4virtio::Svr::Block\_request< Ds\_data >:



### 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.*

### 15.343.1 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](#).

### 15.343.2 Member Function Documentation

#### 15.343.2.1 data\_size()

```
template<typename Ds_data >
unsigned L4virtio::Svr::Block_request< Ds_data >::data_size () const [inline]
```

Compute the total size of the data in the request.

Return values

|             |                                      |
|-------------|--------------------------------------|
| <i>Size</i> | in bytes or 0 if there was an error. |
|-------------|--------------------------------------|

Exceptions

|                                            |                           |
|--------------------------------------------|---------------------------|
| <a href="#">L4::Runtime_error(-L4_EIO)</a> | Request has a bad format. |
|--------------------------------------------|---------------------------|

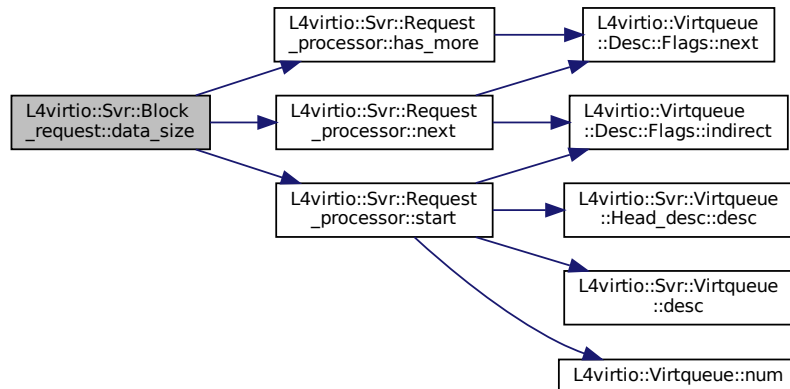
Note that this operation is relatively expensive as it has to iterate over the complete list of blocks.

Definition at line 63 of file [virtio-block](#).

References [L4virtio::Svr::Request\\_processor::has\\_more\(\)](#), [L4\\_EIO](#), [L4virtio::Svr::Request\\_processor::next\(\)](#), and [L4virtio::Svr::Request\\_processor::start\(\)](#).



Here is the call graph for this function:



### 15.343.2.2 next\_block()

```

template<typename Ds_data >
Data_block L4virtio::Svr::Block_request< Ds_data >::next_block () [inline]

```

Return next block in scatter-gather list.

#### Returns

Information about the next data block.

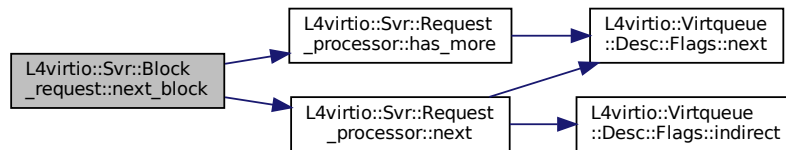
#### Exceptions

|                                          |                                  |
|------------------------------------------|----------------------------------|
| <a href="#"><i>L4::Runtime_error</i></a> | No more data block is available. |
| <a href="#"><i>Bad_descriptor</i></a>    | Virtio request is corrupted.     |

Definition at line 113 of file [virtio-block](#).

References [L4virtio::Svr::Bad\\_descriptor::Bad\\_size](#), [L4virtio::Svr::Request\\_processor::has\\_more\(\)](#), [L4\\_EEXIST](#), and [L4virtio::Svr::Request\\_processor::next\(\)](#).

Here is the call graph for this function:



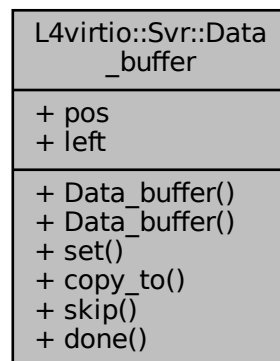
The documentation for this class was generated from the following file:

- l4/l4virtio/server/virtio-block

## 15.344 L4virtio::Svr::Data\_buffer Struct Reference

Abstract data buffer.

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.*

### 15.344.1 Detailed Description

Abstract data buffer.

Definition at line 239 of file [virtio](#).

### 15.344.2 Constructor & Destructor Documentation

#### 15.344.2.1 Data\_buffer()

```
template<typename T >
L4virtio::Svr::Data_buffer::Data_buffer (
 T * p) [inline], [explicit]
```

Create buffer for object *p*.

##### Template Parameters

|          |                           |
|----------|---------------------------|
| <i>T</i> | type of object (implicit) |
|----------|---------------------------|

##### Parameters

|          |                    |
|----------|--------------------|
| <i>p</i> | pointer to object. |
|----------|--------------------|

The buffer shall point to the start of the object *p* and the size left is `sizeof(T)`.

Definition at line 255 of file [virtio](#).

### 15.344.3 Member Function Documentation

#### 15.344.3.1 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

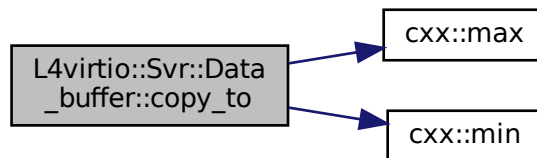
the number of bytes copied.

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

Here is the call graph for this function:

15.344.3.2 `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](#).

15.344.3.3 `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](#).

**15.344.3.4 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

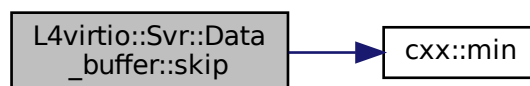
The number of bytes skipped.

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:

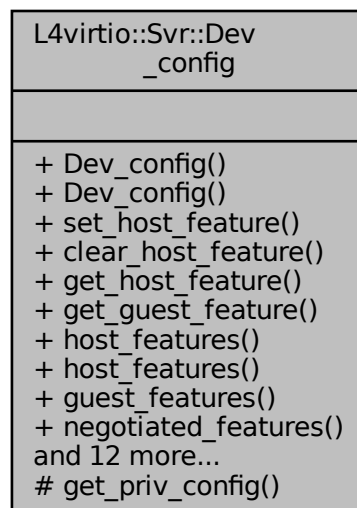
- `I4/I4virtio/server/virtio`

## 15.345 L4virtio::Svr::Dev\_config Class Reference

Abstraction for L4-Virtio device config memory.

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.*

### 15.345.1 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 48 of file [l4virtio](#).

### 15.345.2 Constructor & Destructor Documentation

#### 15.345.2.1 Dev\_config() [1/2]

```
L4virtio::Svr::Dev_config::Dev_config (
 l4_uint32_t vendor,
 l4_uint32_t device,
 unsigned cfg_size,
 l4_uint32_t num_queues = 0) [inline]
```

Create a L4-Virtio config data space.

#### Parameters

|                   |                                                       |
|-------------------|-------------------------------------------------------|
| <i>vendor</i>     | The vendor ID to store in config header.              |
| <i>device</i>     | The device ID to store in config header.              |
| <i>cfg_size</i>   | The size of the device-specific config data in bytes. |
| <i>num_queues</i> | The number of queues provided by the device.          |

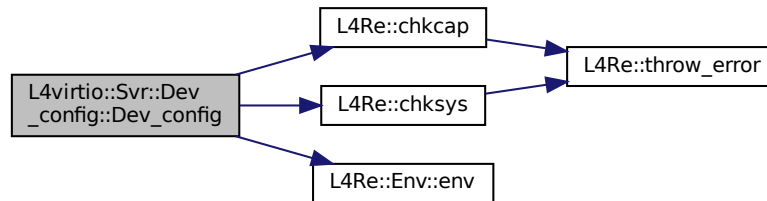
This constructor allocates a data space used for L4-virtio config attaches the data space to the local address space

and writes the initial contents to the config header.

Definition at line 108 of file [l4virtio](#).

References [L4Re::chkcap\(\)](#), [L4Re::chksys\(\)](#), [L4Re::Env::env\(\)](#), and [L4\\_PAGESIZE](#).

Here is the call graph for this function:



### 15.345.2.2 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 142 of file [l4virtio](#).

References [L4\\_PAGESIZE](#).

## 15.345.3 Member Function Documentation



### 15.345.3.1 change\_queue\_config()

```
bool L4virtio::Svr::Dev_config::change_queue_config (
 l4_uint32_t num_queues) [inline]
```

Setup new queue configuration.

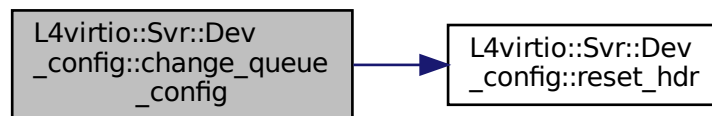
#### Parameters

|                   |                                              |
|-------------------|----------------------------------------------|
| <i>num_queues</i> | The number of queues provided by the device. |
|-------------------|----------------------------------------------|

Definition at line 265 of file [l4virtio](#).

References [L4\\_PAGESIZE](#), and [reset\\_hdr\(\)](#).

Here is the call graph for this function:



### 15.345.3.2 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 354 of file [l4virtio](#).

### 15.345.3.3 `get_cmd()`

```
l4_uint32_t L4virtio::Svr::Dev_config::get_cmd () const [inline]
```

Get the value from the `cmd` register.

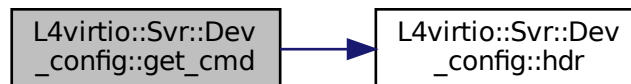
Note, the most significant eight bits are the command (0 is nothing to do). The upper eight bit are reset to zero after the command was handled.

Definition at line 220 of file `l4virtio`.

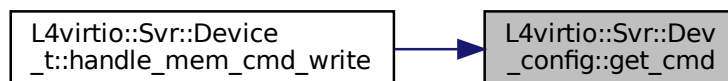
References `l4virtio_config_hdr_t::cmd`, and `hdr()`.

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:



### 15.345.3.4 `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 188 of file [l4virtio](#).

15.345.3.5 `hdr()`

```
l4virtio_config_hdr_t const volatile* L4virtio::Svr::Dev_config::hdr () const [inline]
```

Get a read-only pointer to the config header.

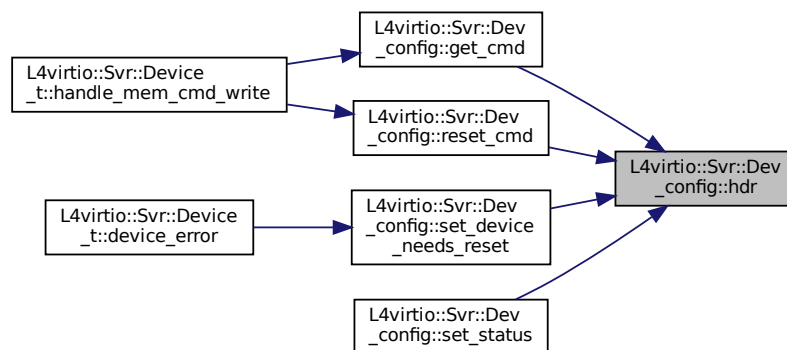
## Returns

Read-only pointer to the shared config header.

Definition at line 347 of file [l4virtio](#).

Referenced by [get\\_cmd\(\)](#), [reset\\_cmd\(\)](#), [set\\_device\\_needs\\_reset\(\)](#), and [set\\_status\(\)](#).

Here is the caller graph for this function:

15.345.3.6 `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 202 of file [l4virtio](#).

**15.345.3.7 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

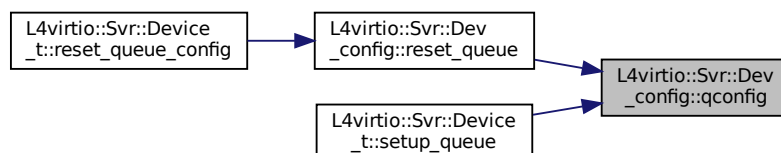
Read-only pointer to the config of the queue with the given *index*, or NULL if *index* is out of range.

Definition at line 282 of file [l4virtio](#).

References [L4\\_UNLIKELY](#).

Referenced by [reset\\_queue\(\)](#), and [L4virtio::Svr::Device\\_t< DATA >::setup\\_queue\(\)](#).

Here is the caller graph for this function:



**15.345.3.8 reset\_cmd()**

```
void L4virtio::Svr::Dev_config::reset_cmd () [inline]
```

Reset the `cmd` register after execution of a command.

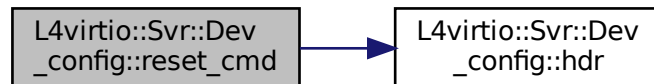
This function resets the `cmd` register in order for the client to detect that the command was executed by the device.

Definition at line 231 of file [l4virtio](#).

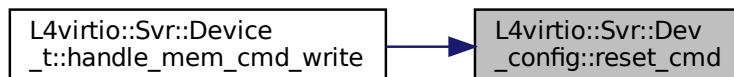
References [l4virtio\\_config\\_hdr\\_t::cmd](#), and [hdr\(\)](#).

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:

**15.345.3.9 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**

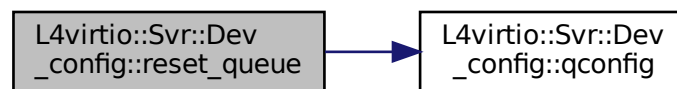
true on success, or false when *index* is out of range.

Definition at line 324 of file [l4virtio](#).

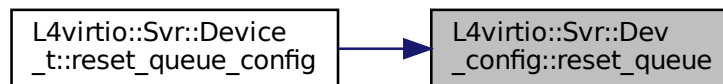
References [L4\\_UNLIKELY](#), [l4virtio\\_config\\_queue\\_t::num](#), [l4virtio\\_config\\_queue\\_t::num\\_max](#), [qconfig\(\)](#), and [l4virtio\\_config\\_queue\\_t::ready](#).

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:

**15.345.3.10 set\_device\_needs\_reset()**

```
void L4virtio::Svr::Dev_config::set_device_needs_reset () [inline]
```

Set DEVICE\_NEEDS\_RESET bit in device status register.

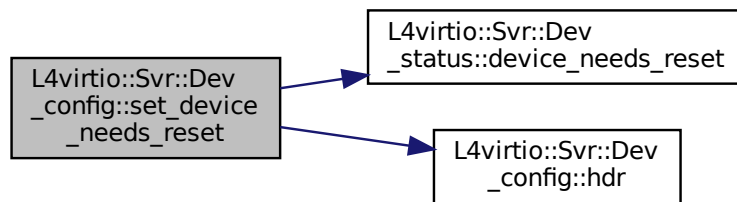
This function sets the internal status register and also the status register in the shared memory to DEVICE\_↔NEEDS\_RESET.

Definition at line 255 of file [l4virtio](#).

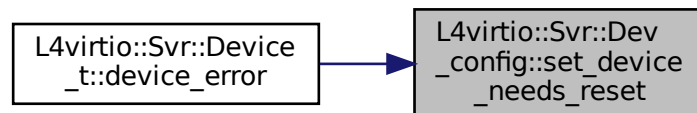
References [L4virtio::Svr::Dev\\_status::device\\_needs\\_reset\(\)](#), [hdr\(\)](#), [L4virtio::Svr::Dev\\_status::raw](#), and [l4virtio\\_config\\_hdr\\_t::status](#).

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:



### 15.345.3.11 set\_status()

```
void L4virtio::Svr::Dev_config::set_status (
 Status status) [inline]
```

Set device status register.

#### Parameters

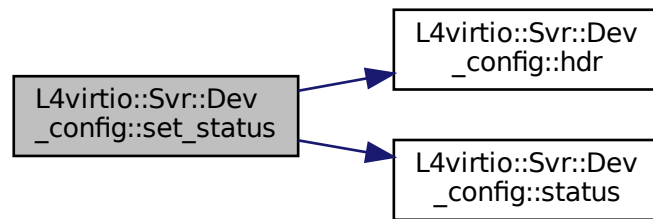
|               |                                               |
|---------------|-----------------------------------------------|
| <i>status</i> | The new value for the device status register. |
|---------------|-----------------------------------------------|

This function sets the internal status register and also the status register in the shared memory to *status*.

Definition at line 243 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:



### 15.345.3.12 status()

```
Status L4virtio::Svr::Dev_config::status () const [inline]
```

Get current device status (trusted).

#### Returns

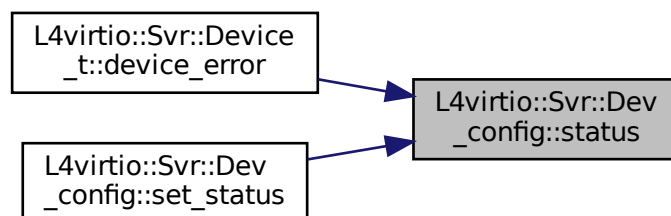
Current device status register (trusted).

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 212 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`



## 15.346 L4virtio::Svr::Dev\_features Struct Reference

Type for device feature bitmap.

Inherited by L4virtio::Svr::Block\_features.

Collaboration diagram for L4virtio::Svr::Dev\_features:

| L4virtio::Svr::Dev_features                                                                                                                                                                  |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| + raw                                                                                                                                                                                        |
| + Dev_features()<br>+ Dev_features()<br>* ring_indirect_desc_bfm_t<br>* ring_indirect_desc()<br>* ring_indirect_desc()<br>* ring_event_idx_bfm_t<br>* ring_event_idx()<br>* 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.*
- typedef [cxx::Bitfield](#)< decltype(raw), 28, 28 > [ring\\_indirect\\_desc\\_bfm\\_t](#)  
*Type to access the ring\_indirect\_desc bits ( 28 to 28 ) of raw.*
- [ring\\_indirect\\_desc\\_bfm\\_t::Val](#) ring\_indirect\_desc () const  
*Get the ring\_indirect\_desc bits ( 28 to 28 ) of raw.*
- [ring\\_indirect\\_desc\\_bfm\\_t::Ref](#) ring\_indirect\_desc ()  
*Get a reference to the ring\_indirect\_desc bits ( 28 to 28 ) of raw.*
- typedef [cxx::Bitfield](#)< decltype(raw), 29, 29 > [ring\\_event\\_idx\\_bfm\\_t](#)  
*Type to access the ring\_event\_idx bits ( 29 to 29 ) of raw.*
- [ring\\_event\\_idx\\_bfm\\_t::Val](#) ring\_event\_idx () const  
*Get the ring\_event\_idx bits ( 29 to 29 ) of raw.*
- [ring\\_event\\_idx\\_bfm\\_t::Ref](#) ring\_event\_idx ()  
*Get a reference to the ring\_event\_idx bits ( 29 to 29 ) of raw.*

### 15.346.1 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

## 15.347 L4virtio::Svr::Dev\_status Struct Reference

Type of the device status register.

Collaboration diagram for L4virtio::Svr::Dev\_status:

| L4virtio::Svr::Dev_status                                                                                                                                                                                                                                                                                                                                                                                                                              |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| + raw                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| + Dev_status()<br>+ Dev_status()<br>+ running()<br>* acked_bfm_t<br>* acked()<br>* acked()<br>* driver_bfm_t<br>* driver()<br>* driver()<br>* driver_ok_bfm_t<br>* driver_ok()<br>* driver_ok()<br>* features_ok_bfm_t<br>* features_ok()<br>* features_ok()<br>* fail_state_bfm_t<br>* fail_state()<br>* fail_state()<br>* device_needs_reset_bfm_t<br>* device_needs_reset()<br>* device_needs_reset()<br>* failed_bfm_t<br>* failed()<br>* 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.*
- [typedef cxx::Bitfield< decltype\(raw\), 0, 0 > \[acked\\\_bfm\\\_t\]\(#\)](#)  
*Type to access the `acked` bits ( 0 to 0 ) of `raw`.*
- [acked\\_bfm\\_t::Val \[acked\]\(#\)](#) () const  
*Get the `acked` bits ( 0 to 0 ) of `raw`.*
- [acked\\_bfm\\_t::Ref \[acked\]\(#\)](#) ()  
*Get a reference to the `acked` bits ( 0 to 0 ) of `raw`.*
- [typedef cxx::Bitfield< decltype\(raw\), 1, 1 > \[driver\\\_bfm\\\_t\]\(#\)](#)  
*Type to access the `driver` bits ( 1 to 1 ) of `raw`.*
- [driver\\_bfm\\_t::Val \[driver\]\(#\)](#) () const  
*Get the `driver` bits ( 1 to 1 ) of `raw`.*
- [driver\\_bfm\\_t::Ref \[driver\]\(#\)](#) ()  
*Get a reference to the `driver` bits ( 1 to 1 ) of `raw`.*
- [typedef cxx::Bitfield< decltype\(raw\), 2, 2 > \[driver\\\_ok\\\_bfm\\\_t\]\(#\)](#)  
*Type to access the `driver_ok` bits ( 2 to 2 ) of `raw`.*
- [driver\\_ok\\_bfm\\_t::Val \[driver\\\_ok\]\(#\)](#) () const  
*Get the `driver_ok` bits ( 2 to 2 ) of `raw`.*
- [driver\\_ok\\_bfm\\_t::Ref \[driver\\\_ok\]\(#\)](#) ()  
*Get a reference to the `driver_ok` bits ( 2 to 2 ) of `raw`.*
- [typedef cxx::Bitfield< decltype\(raw\), 3, 3 > \[features\\\_ok\\\_bfm\\\_t\]\(#\)](#)  
*Type to access the `features_ok` bits ( 3 to 3 ) of `raw`.*
- [features\\_ok\\_bfm\\_t::Val \[features\\\_ok\]\(#\)](#) () const  
*Get the `features_ok` bits ( 3 to 3 ) of `raw`.*
- [features\\_ok\\_bfm\\_t::Ref \[features\\\_ok\]\(#\)](#) ()  
*Get a reference to the `features_ok` bits ( 3 to 3 ) of `raw`.*
- [typedef cxx::Bitfield< decltype\(raw\), 6, 7 > \[fail\\\_state\\\_bfm\\\_t\]\(#\)](#)  
*Type to access the `fail_state` bits ( 6 to 7 ) of `raw`.*
- [fail\\_state\\_bfm\\_t::Val \[fail\\\_state\]\(#\)](#) () const  
*Get the `fail_state` bits ( 6 to 7 ) of `raw`.*
- [fail\\_state\\_bfm\\_t::Ref \[fail\\\_state\]\(#\)](#) ()  
*Get a reference to the `fail_state` bits ( 6 to 7 ) of `raw`.*
- [typedef cxx::Bitfield< decltype\(raw\), 6, 6 > \[device\\\_needs\\\_reset\\\_bfm\\\_t\]\(#\)](#)

- Type to access the `device_needs_reset` bits ( 6 to 6 ) of `raw`.*

  - `device_needs_reset_bfm_t::Val device_needs_reset ()` const

*Get the `device_needs_reset` bits ( 6 to 6 ) of `raw`.*
- `device_needs_reset_bfm_t::Ref device_needs_reset ()`

*Get a reference to the `device_needs_reset` bits ( 6 to 6 ) of `raw`.*
- typedef `cx::Bitfield`< `decltype(raw)`, 7, 7 > `failed_bfm_t`

*Type to access the `failed` bits ( 7 to 7 ) of `raw`.*

  - `failed_bfm_t::Val failed ()` const

*Get the `failed` bits ( 7 to 7 ) of `raw`.*
- `failed_bfm_t::Ref failed ()`

*Get a reference to the `failed` bits ( 7 to 7 ) of `raw`.*

### 15.347.1 Detailed Description

Type of the device status register.

Definition at line 32 of file [virtio](#).

### 15.347.2 Member Function Documentation

#### 15.347.2.1 `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](#).

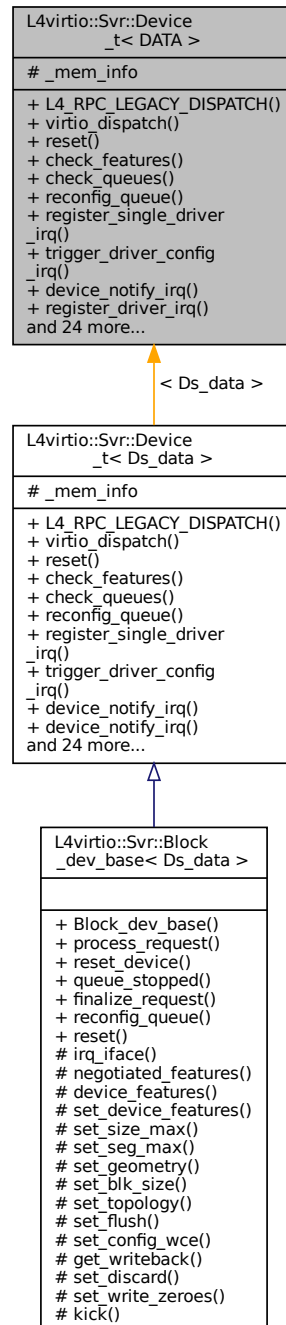
The documentation for this struct was generated from the following file:

- `l4/l4virtio/server/virtio`

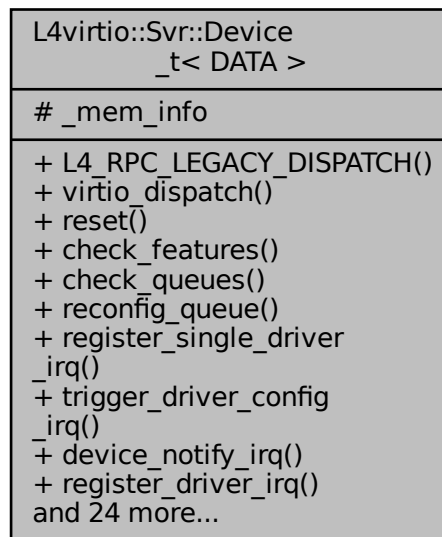
## 15.348 L4virtio::Svr::Device\_t< DATA > Class Template Reference

Server-side L4-VIRTIO device stub.

Inheritance diagram for L4virtio::Svr::Device\_t< DATA >:



Collaboration diagram for L4virtio::Svr::Device\_t< DATA >:



## 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.*

## Protected Attributes

- [Mem\\_list\\_mem\\_info](#)  
*Memory region list.*

### 15.348.1 Detailed Description

```
template<typename DATA>
class L4virtio::Svr::Device_t< DATA >
```

Server-side L4-VIRTIO device stub.

This stub supports old-style device registration with single IRQs (via [register\\_iface\(\)](#)) and new-style multi-event registration (using [get\\_device\\_config\(\)](#), [bind\(\)](#) and [get\\_device\\_notification\\_irq\(\)](#)).

In their default implementation the callbacks provide a wrapper from old-style to new-style functions, so that legacy devices provide both interfaces without any changes to their implementation.

New devices should always implement the new-style interface. If required, they can also provide a backward-compatibility mode by implementing the old-style interface as well.

The old-style interface is considered deprecated and will be removed at some point.

Definition at line [783](#) of file [l4virtio](#).

### 15.348.2 Member Function Documentation

### 15.348.2.1 device\_error()

```
template<typename DATA >
void L4virtio::Svr::Device_t< DATA >::device_error () [inline]
```

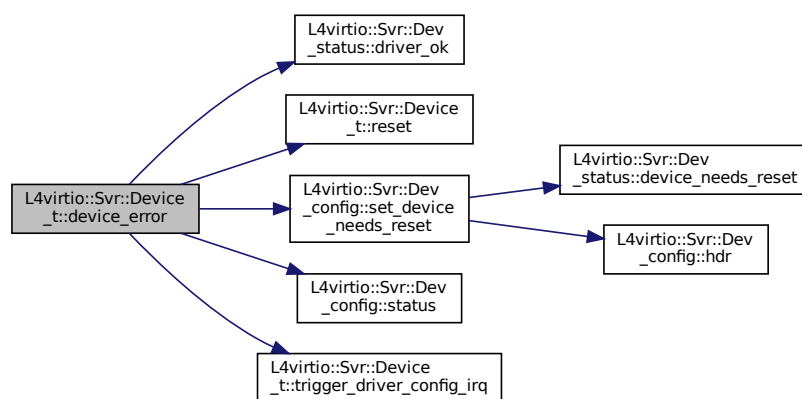
Transition device into DEVICE\_NEEDS\_RESET state.

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 1020 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:



### 15.348.2.2 device\_notify\_irq()

```
template<typename DATA >
virtual L4::Cap<L4::Irq> L4virtio::Svr::Device_t< DATA >::device_notify_irq (
 unsigned idx) [inline], [virtual]
```

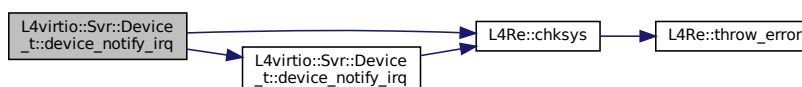
Callback to gather the device notification IRQ (multi IRQ).

The default implementation maps to the implementation for single IRQ notification points.

Definition at line 846 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:





### 15.348.2.3 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.

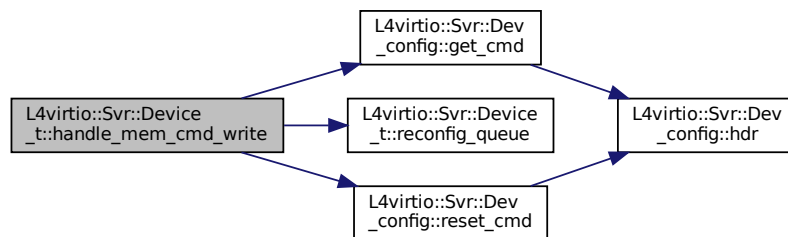
This function checks for a value in the `cmd` register and executes the command if there is any, or returns false if there was no command.

Execution of the command is signaled by a zero in the `cmd` register.

Definition at line 1154 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:



### 15.348.2.4 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.

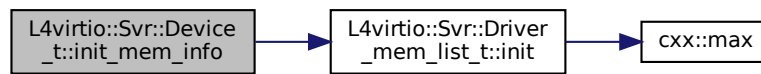
#### Parameters

|            |                                                       |
|------------|-------------------------------------------------------|
| <i>num</i> | Maximum number of memory regions that can be managed. |
|------------|-------------------------------------------------------|

Definition at line 1008 of file `l4virtio`.

References `L4virtio::Svr::Device_t< DATA >::_mem_info`, and `L4virtio::Svr::Driver_mem_list_t< DATA >::init()`.

Here is the call graph for this function:



### 15.348.2.5 register\_driver\_irq()

```

template<typename DATA >
virtual void L4virtio::Svr::Device_t< DATA >::register_driver_irq (
 unsigned idx) [inline], [virtual]

```

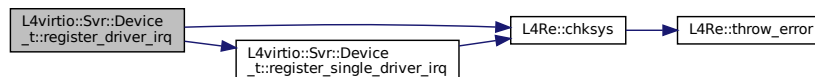
Callback for registering an notification IRQ (multi IRQ).

The default implementation maps to the implementation for single IRQ notification points.

Definition at line 832 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:



### 15.348.2.6 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

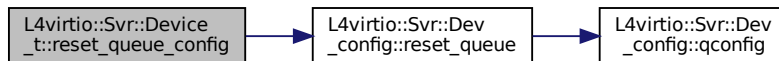
|                       |                                                              |
|-----------------------|--------------------------------------------------------------|
| <i>idx</i>            | The queue index to reset.                                    |
| <i>num_max</i>        | Maximum 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 998 of file [l4virtio](#).

References [L4virtio::Svr::Dev\\_config::reset\\_queue\(\)](#).

Here is the call graph for this function:



### 15.348.2.7 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

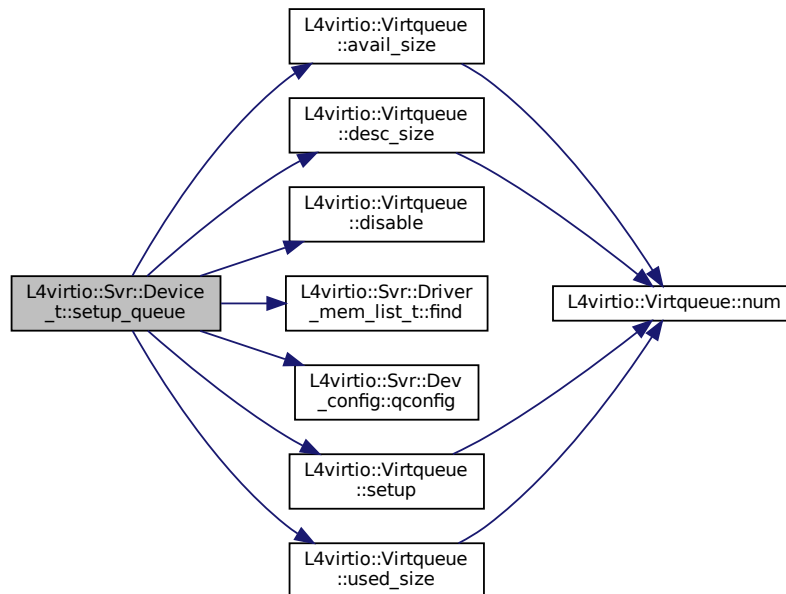
true for success.

- 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 1047 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\(\)](#), [L4virtio::Svr::Driver\\_mem\\_list\\_t< 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:



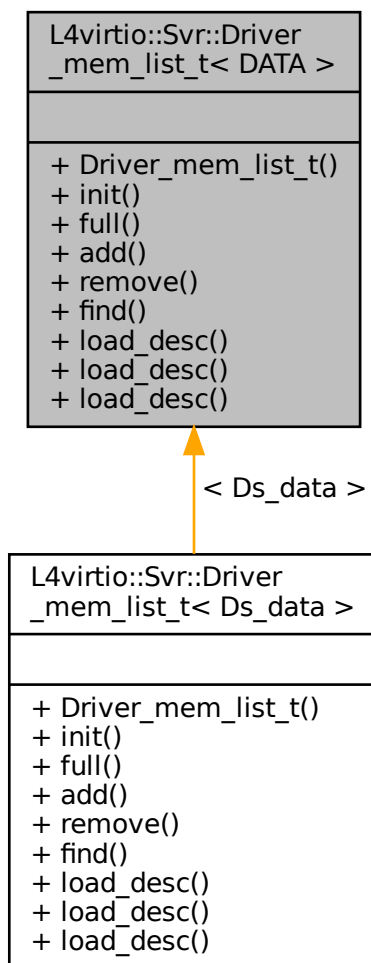
The documentation for this class was generated from the following file:

- `l4/l4virtio/server/l4virtio`

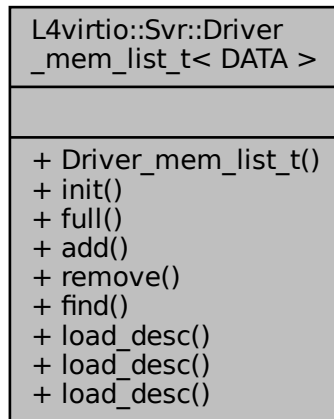
## 15.349 L4virtio::Svr::Driver\_mem\_list\_t< DATA > Class Template Reference

List of driver memory regions assigned to a single L4-VIRTIO transport instance.

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.*

## 15.349.1 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 605 of file [l4virtio](#).

## 15.349.2 Member Function Documentation

### 15.349.2.1 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.                      |

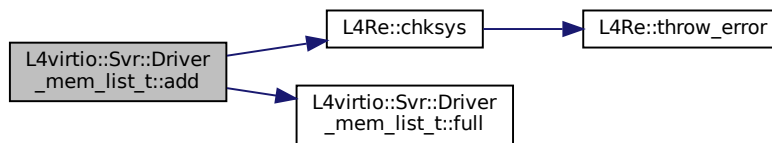
### Returns

A pointer to the new region.

Definition at line 645 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:



### 15.349.2.2 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.  |

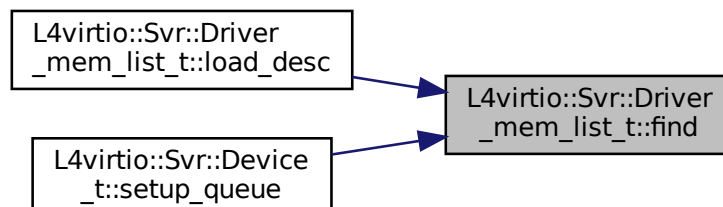
#### Returns

Pointer to the region containing the given region, NULL if none is found.

Definition at line 679 of file `l4virtio`.

Referenced by `L4virtio::Svr::Driver_mem_list_t< DATA >::load_desc()`, and `L4virtio::Svr::Device_t< DATA >::setup_queue()`.

Here is the caller graph for this function:





### 15.349.2.3 full()

```
template<typename DATA >
bool L4virtio::Svr::Driver_mem_list_t< DATA >::full () const [inline]
```

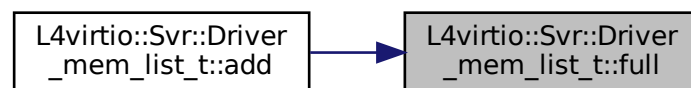
#### Returns

True if the remaining capacity is 0.

Definition at line 634 of file [l4virtio](#).

Referenced by [L4virtio::Svr::Driver\\_mem\\_list\\_t< DATA >::add\(\)](#).

Here is the caller graph for this function:



### 15.349.2.4 init()

```
template<typename DATA >
void L4virtio::Svr::Driver_mem_list_t< DATA >::init (
 unsigned max) [inline]
```

Make a fresh list with capacity *max*.

#### Parameters

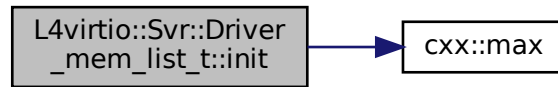
|            |                              |
|------------|------------------------------|
| <i>max</i> | The capacity of this vector. |
|------------|------------------------------|

Definition at line 626 of file [l4virtio](#).

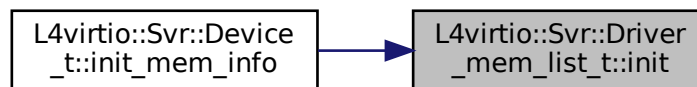
References [cxx::max\(\)](#).

Referenced by [L4virtio::Svr::Device\\_t< DATA >::init\\_mem\\_info\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.349.2.5 load\_desc() [1/3]

```

template<typename DATA >
template<typename ARG >
void L4virtio::Svr::Driver_mem_list_t< DATA >::load_desc (
 Virtqueue::Desc const & desc,
 Request_processor const * p,
 ARG * data) const [inline]

```

Default implementation returning generic information.

#### Template Parameters

|            |                                                                                                                                                                                                                                                                                                                                                                                                                  |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>ARG</i> | Abstract argument type used with <a href="#">Request_processor::start()</a> and <a href="#">Request_processor::next()</a> to deliver the result of loading a descriptor. This type must provide a constructor taking three arguments: (1) pointer to a <code>Driver_mem_region</code> , (2) the <a href="#">Virtqueue::Desc</a> descriptor, and (3) a pointer to the calling <a href="#">Request_processor</a> . |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Parameters

|     |             |                                                     |
|-----|-------------|-----------------------------------------------------|
|     | <i>desc</i> | The descriptor to load                              |
|     | <i>p</i>    | The request processor calling us                    |
| out | <i>data</i> | Shall be assigned to <code>ARG(mem, desc, p)</code> |

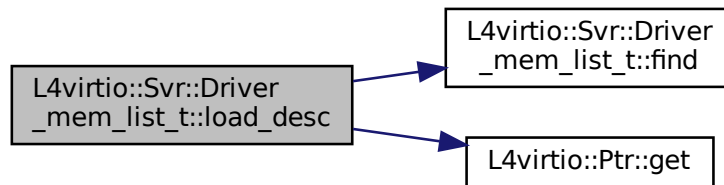
## Exceptions

|                                       |                                                 |
|---------------------------------------|-------------------------------------------------|
| <a href="#"><i>Bad_descriptor</i></a> | The descriptor address could not be translated. |
|---------------------------------------|-------------------------------------------------|

Definition at line 740 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:



## 15.349.2.6 load\_desc() [2/3]

```

template<typename DATA >
void L4virtio::Svr::Driver_mem_list_t< DATA >::load_desc (
 Virtqueue::Desc const & desc,
 Request_processor const * p,
 Mem_region const ** data) const [inline]

```

Default implementation returning the Driver\_mem\_region.

## Parameters

|     |             |                                                                                |
|-----|-------------|--------------------------------------------------------------------------------|
|     | <i>desc</i> | The descriptor to load                                                         |
|     | <i>p</i>    | The request processor calling us                                               |
| out | <i>data</i> | Shall be set to a pointer to the Driver_mem_region that covers the descriptor. |

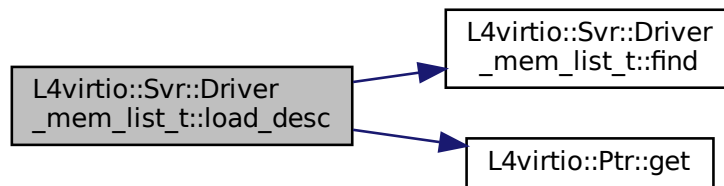
## Exceptions

|                                       |                                                 |
|---------------------------------------|-------------------------------------------------|
| <a href="#"><i>Bad_descriptor</i></a> | The descriptor address could not be translated. |
|---------------------------------------|-------------------------------------------------|

Definition at line 713 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:



### 15.349.2.7 load\_desc() [3/3]

```

template<typename DATA >
void L4virtio::Svr::Driver_mem_list_t< DATA >::load_desc (
 Virtqueue::Desc const & desc,
 Request_processor const * p,
 Virtqueue::Desc const ** table) const [inline]

```

Default implementation for loading an indirect descriptor.

#### Parameters

|     |              |                                             |
|-----|--------------|---------------------------------------------|
|     | <i>desc</i>  | The descriptor to load                      |
|     | <i>p</i>     | The request processor calling us            |
| out | <i>table</i> | Shall be set to the loaded descriptor table |

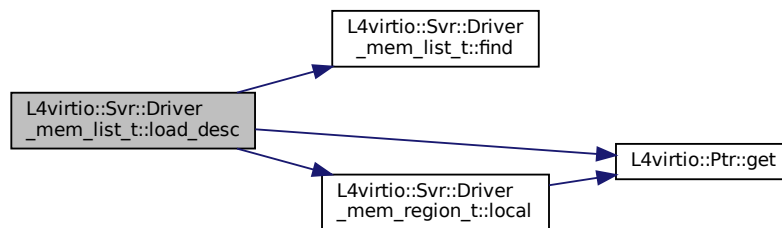
#### Exceptions

|                                       |                                                 |
|---------------------------------------|-------------------------------------------------|
| <a href="#"><i>Bad_descriptor</i></a> | The descriptor address could not be translated. |
|---------------------------------------|-------------------------------------------------|

Definition at line 693 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](#), [L4virtio::Virtqueue::Desc::len](#), and [L4virtio::Svr::Driver\\_mem\\_region\\_t< DATA >::local\(\)](#).

Here is the call graph for this function:



### 15.349.2.8 remove()

```

template<typename DATA >
void L4virtio::Svr::Driver_mem_list_t< DATA >::remove (
 Mem_region const * r) [inline]

```

Remove the given region from the list.

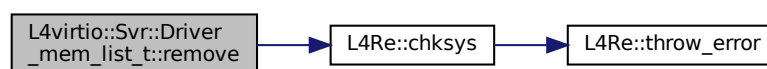
#### Parameters

|          |                                                                                        |
|----------|----------------------------------------------------------------------------------------|
| <i>r</i> | The region to remove (result from <a href="#">add()</a> , or <a href="#">find()</a> ). |
|----------|----------------------------------------------------------------------------------------|

Definition at line 659 of file [l4virtio](#).

References [L4Re::chksys\(\)](#), and [L4\\_ERANGE](#).

Here is the call graph for this function:



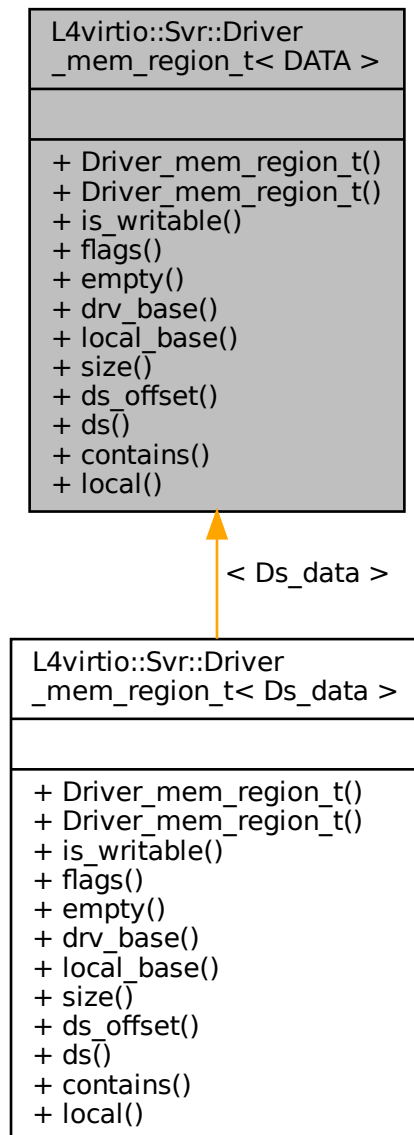
The documentation for this class was generated from the following file:

- `l4/l4virtio/server/l4virtio`

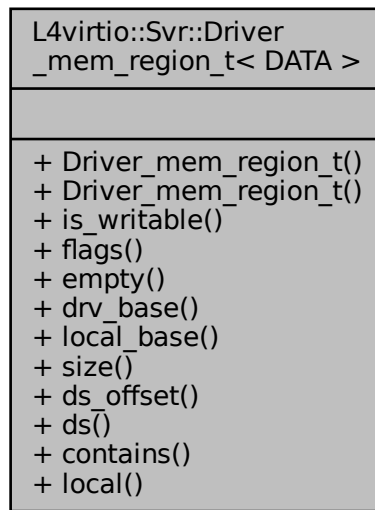
## 15.350 L4virtio::Svr::Driver\_mem\_region\_t< DATA > Class Template Reference

Region of driver memory, that shall be managed locally.

Inheritance diagram for L4virtio::Svr::Driver\_mem\_region\_t< DATA >:



Collaboration diagram for L4virtio::Svr::Driver\_mem\_region\_t< DATA >:



## Public Member Functions

- [Driver\\_mem\\_region\\_t](#) ()  
*Make default empty memroy 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.*

### 15.350.1 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 429 of file [l4virtio](#).

## 15.350.2 Constructor & Destructor Documentation

### 15.350.2.1 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

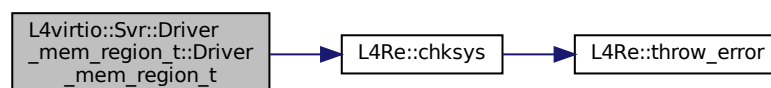
|                 |                                                                                   |
|-----------------|-----------------------------------------------------------------------------------|
| <i>drv_base</i> | Base address of the memory region used by the driver.                             |
| <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 474 of file [l4virtio](#).

References [L4Re::chksys\(\)](#).

Here is the call graph for this function:



### 15.350.3 Member Function Documentation



### 15.350.3.1 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. |

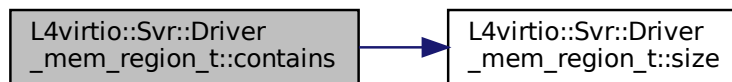
#### Returns

true if the given driver address region is contained in this region, false else.

Definition at line 568 of file [l4virtio](#).

References [L4virtio::Svr::Driver\\_mem\\_region\\_t< DATA >::size\(\)](#).

Here is the call graph for this function:



### 15.350.3.2 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 547 of file [l4virtio](#).

### 15.350.3.3 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 559 of file [l4virtio](#).

### 15.350.3.4 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 556 of file [l4virtio](#).

### 15.350.3.5 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 else.

Definition at line 543 of file [l4virtio](#).

### 15.350.3.6 flags()

```
template<typename DATA >
Flags L4virtio::Svr::Driver_mem_region_t< DATA >::flags () const [inline]
```

#### Returns

The flags for this region.

Definition at line 540 of file [l4virtio](#).

### 15.350.3.7 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 else.

Definition at line 537 of file [l4virtio](#).

### 15.350.3.8 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

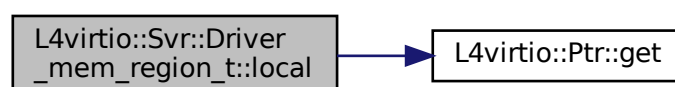
Local address for the given driver address *p*.

Definition at line 592 of file [l4virtio](#).

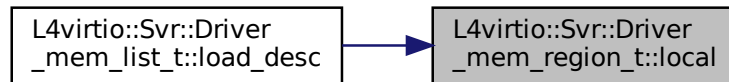
References [L4virtio::Ptr< T >::get\(\)](#).

Referenced by [L4virtio::Svr::Driver\\_mem\\_list\\_t< DATA >::load\\_desc\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.350.3.9 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 550 of file [l4virtio](#).

### 15.350.3.10 size()

```
template<typename DATA >
l4_umword_t L4virtio::Svr::Driver_mem_region_t< DATA >::size () const [inline]
```

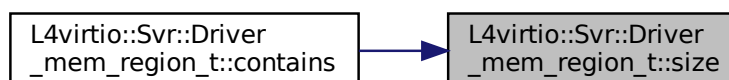
#### Returns

The size of the region in bytes.

Definition at line 553 of file [l4virtio](#).

Referenced by [L4virtio::Svr::Driver\\_mem\\_region\\_t< DATA >::contains\(\)](#).

Here is the caller graph for this function:



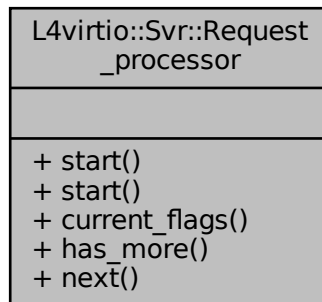
The documentation for this class was generated from the following file:

- [l4/l4virtio/server/l4virtio](#)

## 15.351 L4virtio::Svr::Request\_processor Class Reference

Encapsulate the state for processing a VIRTIO request.

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.*

### 15.351.1 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 [398](#) of file [virtio](#).

## 15.351.2 Member Function Documentation

### 15.351.2.1 `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 [468](#) of file [virtio](#).

References [L4virtio::Virtqueue::Desc::flags](#).

### 15.351.2.2 `has_more()`

```
bool L4virtio::Svr::Request_processor::has_more () const [inline]
```

Are there more chained descriptors?

**Returns**

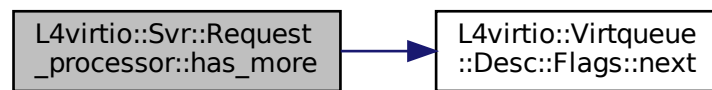
true if there are more chained descriptors in the current request.

Definition at line 475 of file [virtio](#).

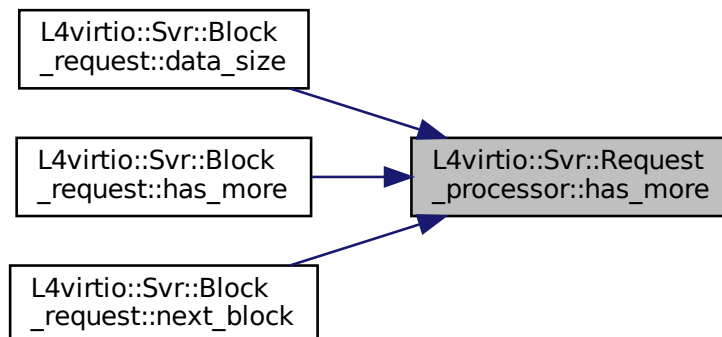
References [L4virtio::Virtqueue::Desc::flags](#), and [L4virtio::Virtqueue::Desc::Flags::next\(\)](#).

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:

**15.351.2.3 next()**

```

template<typename DESC_MAN , typename ... ARGS>
bool L4virtio::Svr::Request_processor::next (
 DESC_MAN * dm,
 ARGS... args) [inline]

```

Switch to the next descriptor in a descriptor chain.

## Template Parameters

|                 |                                        |
|-----------------|----------------------------------------|
| <i>DESC_MAN</i> | Type of descriptor manager (implicit). |
|-----------------|----------------------------------------|

## Parameters

|             |                                                                           |
|-------------|---------------------------------------------------------------------------|
| <i>dm</i>   | Descriptor manager that is used to translate VIRTIO descriptor addresses. |
| <i>args</i> | Extra arguments passed to <code>dm-&gt;load_desc()</code>                 |

## Return values

|              |                                 |
|--------------|---------------------------------|
| <i>true</i>  | A next descriptor is available. |
| <i>false</i> | No descriptor available.        |

## Exceptions

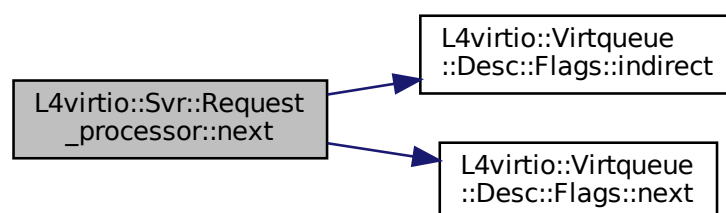
|                                       |                                                            |
|---------------------------------------|------------------------------------------------------------|
| <a href="#"><i>Bad_descriptor</i></a> | The <code>next</code> index of this descriptor is invalid. |
|---------------------------------------|------------------------------------------------------------|

Definition at line 492 of file [virtio](#).

References [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](#).

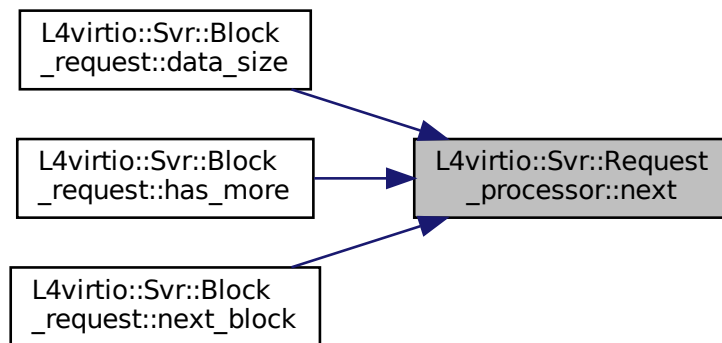
Referenced by [L4virtio::Svr::Block\\_request<Ds\\_data>::data\\_size\(\)](#), [L4virtio::Svr::Block\\_request<Ds\\_data>::has\\_more\(\)](#), and [L4virtio::Svr::Block\\_request<Ds\\_data>::next\\_block\(\)](#).

Here is the call graph for this function:





Here is the caller graph for this function:



#### 15.351.2.4 start() [1/2]

```

template<typename DESC_MAN , typename ... ARGS>
void L4virtio::Svr::Request_processor::start (
 DESC_MAN * dm,
 Virtqueue * ring,
 Virtqueue::Head_desc const & request,
 ARGS... args) [inline]

```

Start processing a new request.

##### Template Parameters

|                 |                                        |
|-----------------|----------------------------------------|
| <i>DESC_MAN</i> | Type of descriptor manager (implicit). |
|-----------------|----------------------------------------|

##### Parameters

|                |                                                                           |
|----------------|---------------------------------------------------------------------------|
| <i>dm</i>      | Descriptor manager that is used to translate VIRTIO descriptor addresses. |
| <i>ring</i>    | VIRTIO ring of the request.                                               |
| <i>request</i> | VIRTIO request from <a href="#">Virtqueue::next_avail()</a>               |
| <i>args</i>    | Extra arguments passed to <code>dm-&gt;load_desc()</code>                 |

##### Precondition

The given request must be valid.

## Exceptions

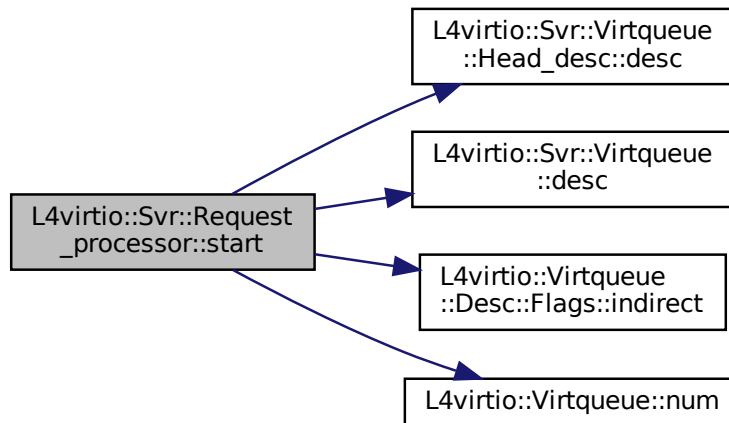
|                                       |                                                                                                   |
|---------------------------------------|---------------------------------------------------------------------------------------------------|
| <a href="#"><i>Bad_descriptor</i></a> | The descriptor has an invalid size or <code>load_desc()</code> has thrown an exception by itself. |
|---------------------------------------|---------------------------------------------------------------------------------------------------|

Definition at line 427 of file [virtio](#).

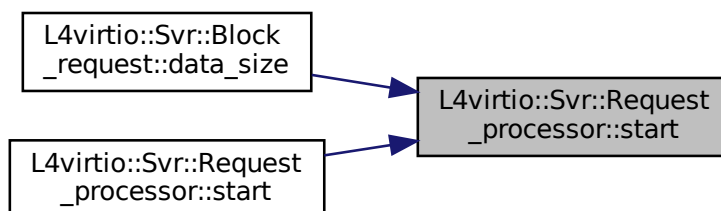
References [L4virtio::Svr::Bad\\_descriptor::Bad\\_size](#), [L4virtio::Svr::Virtqueue::Head\\_desc::desc\(\)](#), [L4virtio::Svr::Virtqueue::desc\(\)](#), [L4virtio::Virtqueue::Desc::flags](#), [L4virtio::Virtqueue::Desc::Flags::indirect\(\)](#), [L4\\_UNLIKELY](#), [L4virtio::Virtqueue::Desc::len](#), 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:



## 15.351.2.5 start() [2/2]

```
template<typename DESC_MAN , typename ... ARGS>
Virtqueue::Request const& L4virtio::Svr::Request_processor::start (
 DESC_MAN * dm,
 Virtqueue::Request const & request,
 ARGS... args) [inline]
```

Start processing a new request.

## Template Parameters

|                 |                                        |
|-----------------|----------------------------------------|
| <i>DESC_MAN</i> | Type of descriptor manager (implicit). |
|-----------------|----------------------------------------|

## Parameters

|                |                                                                           |
|----------------|---------------------------------------------------------------------------|
| <i>dm</i>      | Descriptor manager that is used to translate VIRTIO descriptor addresses. |
| <i>request</i> | VIRTIO request from <a href="#">Virtqueue::next_avail()</a>               |
| <i>args</i>    | Extra arguments passed to dm->load_desc()                                 |

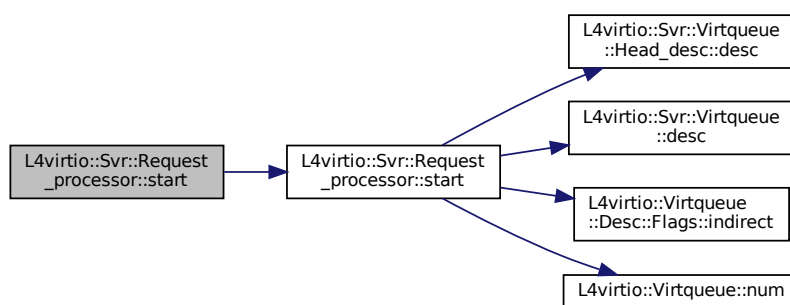
## Precondition

The given request must be valid.

Definition at line [458](#) of file [virtio](#).

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

Here is the call graph for this function:



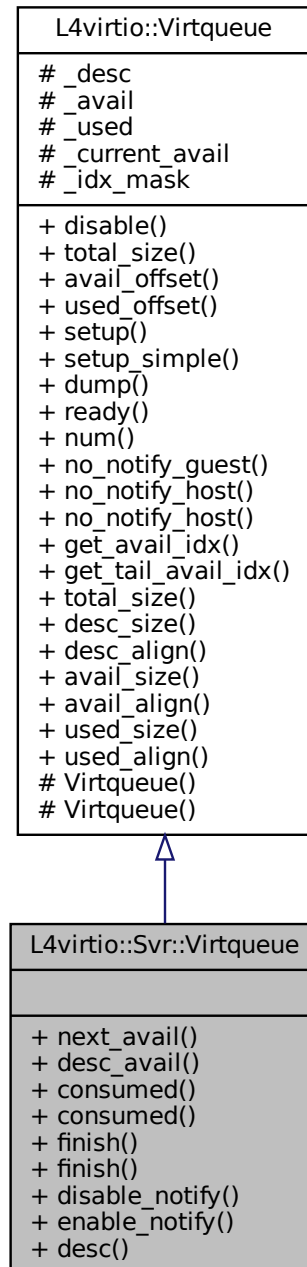
The documentation for this class was generated from the following file:

- `l4/l4virtio/server/virtio`

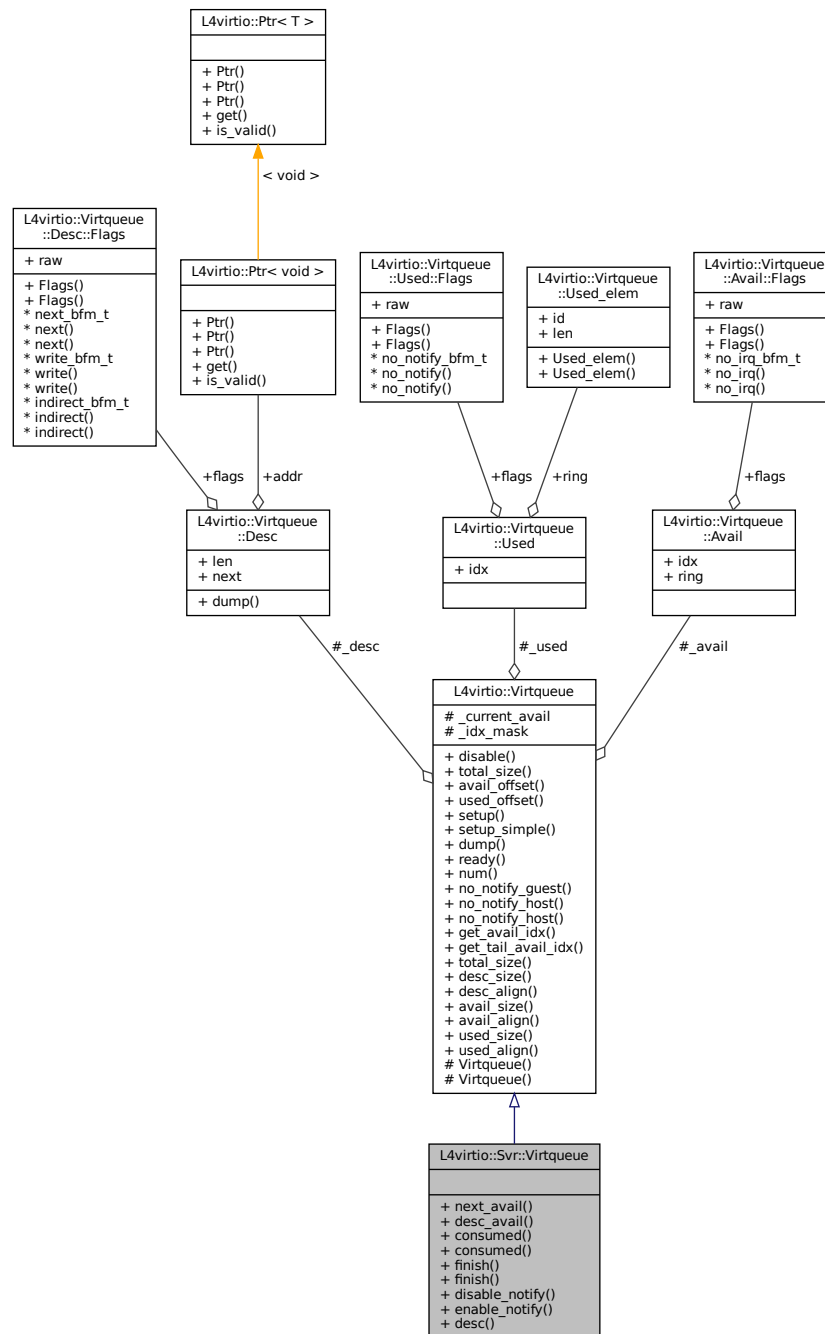
## 15.352 L4virtio::Svr::Virtqueue Class Reference

[Virtqueue](#) implementation for the device.

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.*

## Additional Inherited Members

### 15.352.1 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`.

### 15.352.2 Member Function Documentation

#### 15.352.2.1 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](#).

### 15.352.2.2 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:



### 15.352.2.3 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](#).

**15.352.2.4 disable\_notify()**

```
void L4virtio::Svr::Virtqueue::disable_notify () [inline]
```

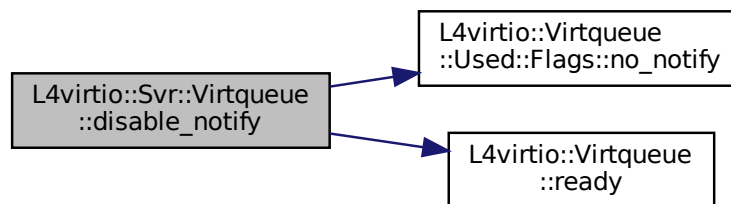
Set the 'no notify' flag for this queue.

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:





### 15.352.2.5 enable\_notify()

```
void L4virtio::Svr::Virtqueue::enable_notify () [inline]
```

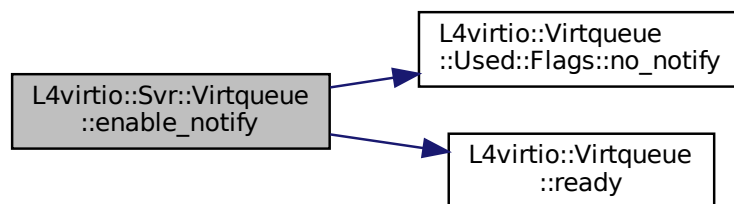
Clear the 'no notify' flag for this queue.

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:



### 15.352.2.6 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:

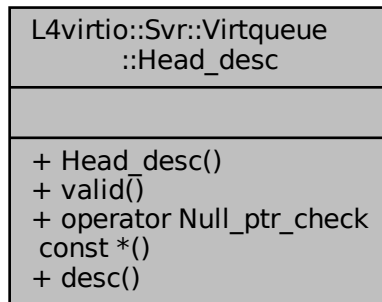
- [I4/I4virtio/server/virtio](#)

## 15.353 L4virtio::Svr::Virtqueue::Head\_desc Class Reference

VIRTIO request, essentially a descriptor from the available ring.

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

### 15.353.1 Detailed Description

VIRTIO request, essentially a descriptor from the available ring.

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

### 15.353.2 Member Function Documentation

### 15.353.2.1 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:



### 15.353.2.2 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](#).

### 15.353.2.3 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](#).

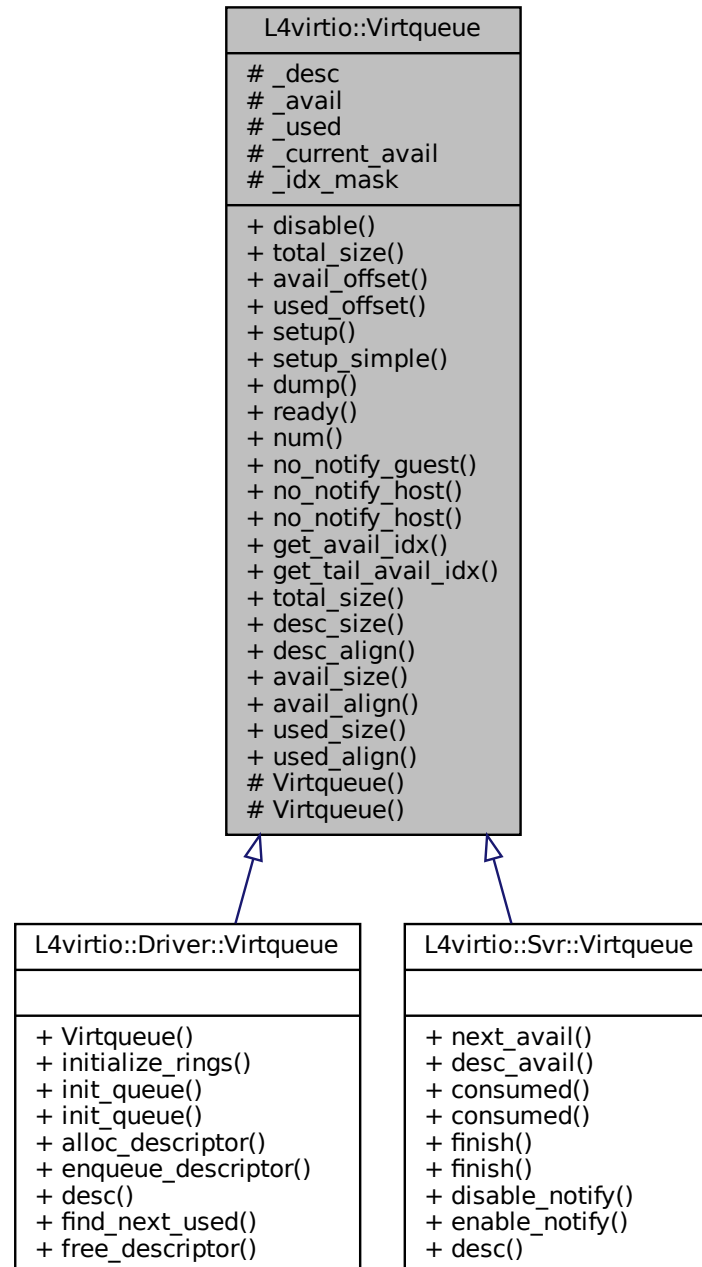
The documentation for this class was generated from the following file:

- `I4/I4virtio/server/virtio`

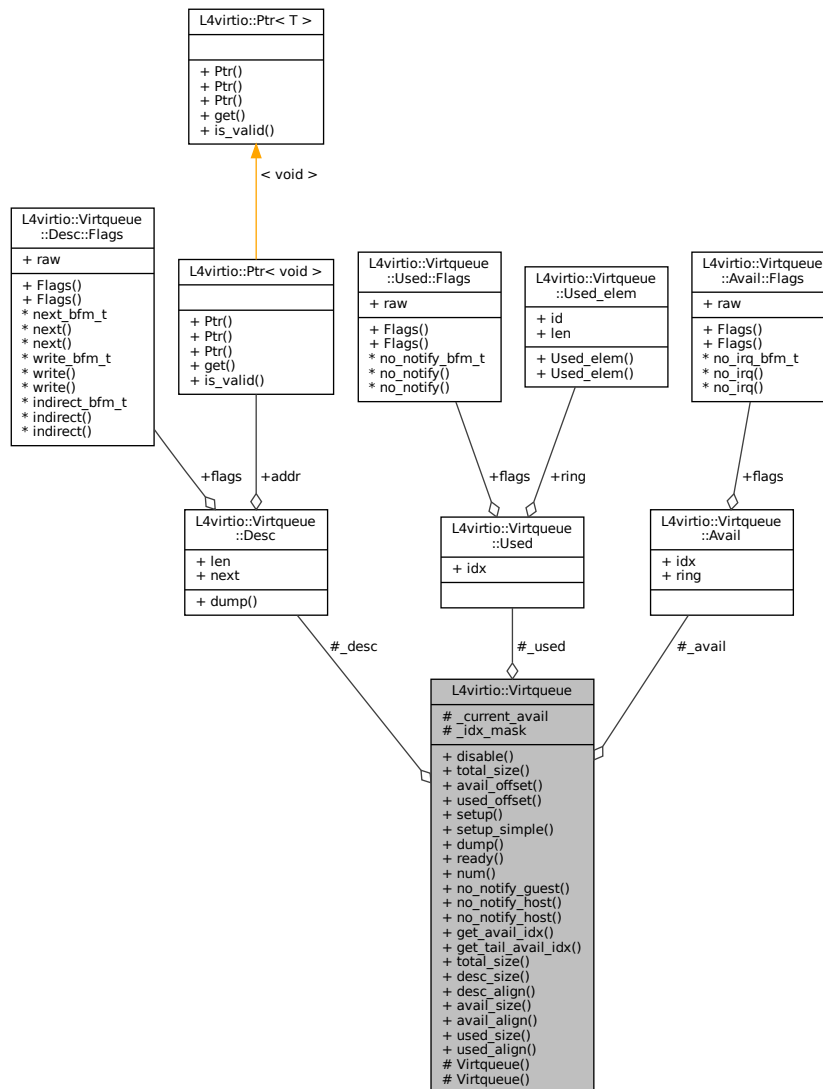
## 15.354 L4virtio::Virtqueue Class Reference

Low-level [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.*

### 15.354.1 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 89 of file [virtqueue](#).

### 15.354.2 Member Function Documentation

#### 15.354.2.1 [avail\\_align\(\)](#)

```
static unsigned long L4virtio::Virtqueue::avail_align () [inline], [static]
```

Get the alignment in zero LSBs needed for the available ring.

#### Returns

The alignment in zero LSBs needed for an available ring.

Definition at line 293 of file [virtqueue](#).

#### 15.354.2.2 [avail\\_size\(\)](#)

```
static unsigned long L4virtio::Virtqueue::avail_size (
 unsigned num) [inline], [static]
```

Calculate the size of the available ring for `num` entries.

## Parameters

|            |                                              |
|------------|----------------------------------------------|
| <i>num</i> | The number of entries in the available ring. |
|------------|----------------------------------------------|

## Returns

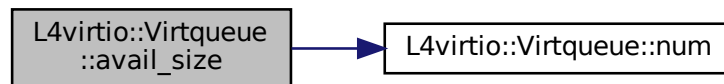
The size in bytes needed for an available ring with *num* entries.

Definition at line 285 of file [virtqueue](#).

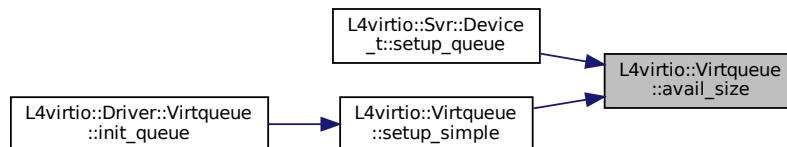
References [num\(\)](#).

Referenced by [L4virtio::Svr::Device\\_t< DATA >::setup\\_queue\(\)](#), and [setup\\_simple\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.354.2.3 desc\_align()

```
static unsigned long L4virtio::Virtqueue::desc_align () [inline], [static]
```

Get the alignment in zero LSBs needed for the descriptor table.

## Returns

The alignment in zero LSBs needed for a descriptor table.

Definition at line 275 of file [virtqueue](#).



#### 15.354.2.4 desc\_size()

```
static unsigned long L4virtio::Virtqueue::desc_size (
 unsigned num) [inline], [static]
```

Calculate the size of the descriptor table for `num` entries.

## Parameters

|            |                                                |
|------------|------------------------------------------------|
| <i>num</i> | The number of entries in the descriptor table. |
|------------|------------------------------------------------|

## Returns

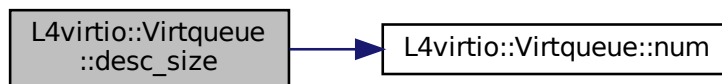
The size in bytes needed for a descriptor table with `num` entries.

Definition at line 267 of file [virtqueue](#).

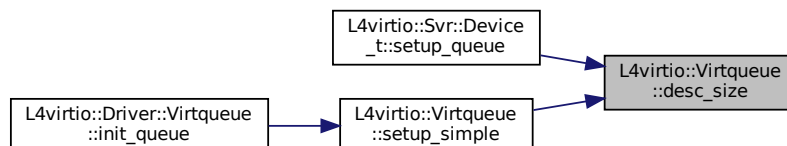
References [num\(\)](#).

Referenced by [L4virtio::Svr::Device\\_t< DATA >::setup\\_queue\(\)](#), and [setup\\_simple\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.354.2.5 disable()

```
void L4virtio::Virtqueue::disable () [inline]
```

Completely disable the queue.

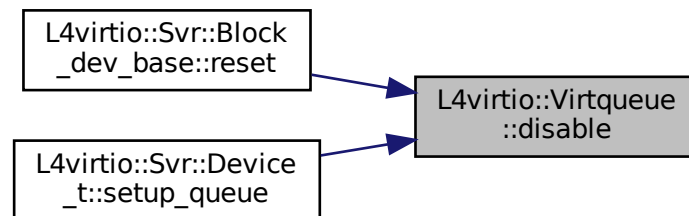
[setup\(\)](#) must be used to enable the queue again.

Definition at line 230 of file [virtqueue](#).

References [\\_desc](#).

Referenced by [L4virtio::Svr::Block\\_dev\\_base< Ds\\_data >::reset\(\)](#), and [L4virtio::Svr::Device\\_t< DATA >::setup\\_queue\(\)](#).

Here is the caller graph for this function:



### 15.354.2.6 dump()

```
void L4virtio::Virtqueue::dump (
 Desc const * d) const [inline]
```

Dump descriptors for this queue.

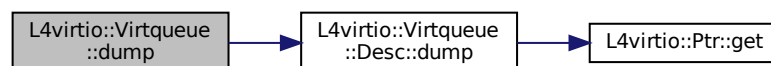
#### Precondition

the queue must be in working state.

Definition at line 397 of file [virtqueue](#).

References [\\_desc](#), and [L4virtio::Virtqueue::Desc::dump\(\)](#).

Here is the call graph for this function:



### 15.354.2.7 `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 454 of file [virtqueue](#).

References [\\_avail](#), and [L4virtio::Virtqueue::Avail::idx](#).

### 15.354.2.8 `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 461 of file [virtqueue](#).

References [\\_current\\_avail](#).

### 15.354.2.9 `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.

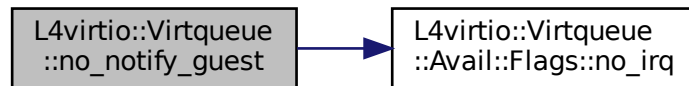
**Returns**

true if the guest does not want to get IRQs (currently).

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

References [\\_avail](#), [L4virtio::Virtqueue::Avail::flags](#), and [L4virtio::Virtqueue::Avail::Flags::no\\_irq\(\)](#).

Here is the call graph for this function:

**15.354.2.10 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**

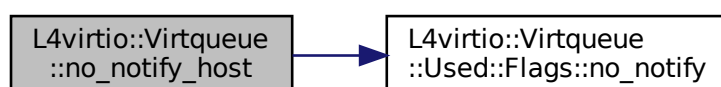
true if the host does not want to get IRQs (currently).

Definition at line 431 of file [virtqueue](#).

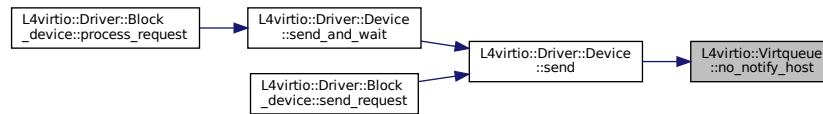
References [\\_used](#), [L4virtio::Virtqueue::Used::flags](#), and [L4virtio::Virtqueue::Used::Flags::no\\_notify\(\)](#).

Referenced by [L4virtio::Driver::Device::send\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.354.2.11 no\_notify\_host() [2/2]

```
void L4virtio::Virtqueue::no_notify_host (
 bool value) [inline]
```

Set the no-notify flag for this queue.

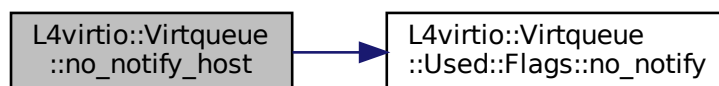
#### Precondition

Queue must be in a working state.

Definition at line 441 of file [virtqueue](#).

References [\\_used](#), [L4virtio::Virtqueue::Used::flags](#), and [L4virtio::Virtqueue::Used::Flags::no\\_notify\(\)](#).

Here is the call graph for this function:



### 15.354.2.12 num()

```
unsigned L4virtio::Virtqueue::num () const [inline]
```

**Returns**

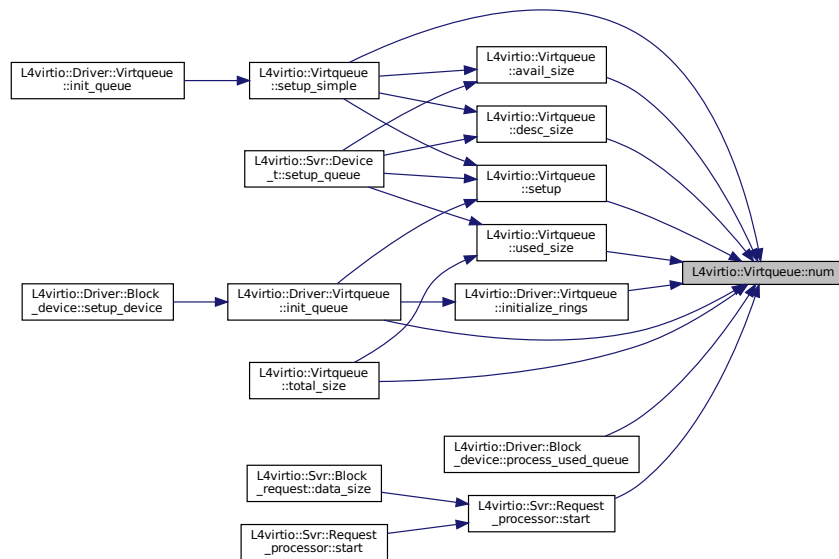
The number of entries in the ring.

Definition at line 409 of file [virtqueue](#).

References [\\_idx\\_mask](#).

Referenced by [avail\\_size\(\)](#), [desc\\_size\(\)](#), [L4virtio::Driver::Virtqueue::init\\_queue\(\)](#), [L4virtio::Driver::Virtqueue::initialize\\_rings\(\)](#), [L4virtio::Driver::Block\\_device::process\\_used\\_queue\(\)](#), [setup\(\)](#), [setup\\_simple\(\)](#), [L4virtio::Svr::Request\\_processor::start\(\)](#), [total\\_size\(\)](#), and [used\\_size\(\)](#).

Here is the caller graph for this function:

**15.354.2.13 ready()**

```
bool L4virtio::Virtqueue::ready () const [inline]
```

Test if this queue is in working state.

**Returns**

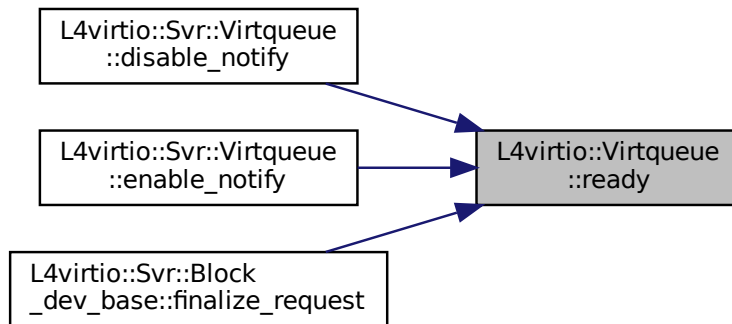
true when the queue is in working state, false else.

Definition at line 405 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:



#### 15.354.2.14 setup()

```

void L4virtio::Virtqueue::setup (
 unsigned num,
 void * desc,
 void * avail,
 void * used) [inline]

```

Enable this queue.

##### Parameters

|              |                                                                                                              |
|--------------|--------------------------------------------------------------------------------------------------------------|
| <i>num</i>   | The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2). |
| <i>desc</i>  | The address of the descriptor table. (Must be Desc_align aligned and at least desc_size(num) bytes in size.) |
| <i>avail</i> | The address of the available ring. (Must be Avail_align aligned and at least avail_size(num) bytes in size.) |
| <i>used</i>  | The address of the used ring. (Must be Used_align aligned and at least used_size(num) bytes in size.)        |

Due to the data type of the descriptors, the queue can have a maximum size of  $2^{16}$ .

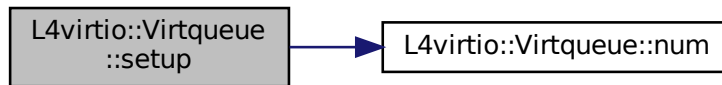
Definition at line 355 of file [virtqueue](#).

References [\\_avail](#), [\\_current\\_avail](#), [\\_desc](#), [\\_idx\\_mask](#), [\\_used](#), [L4\\_EINVAL](#), and [num\(\)](#).

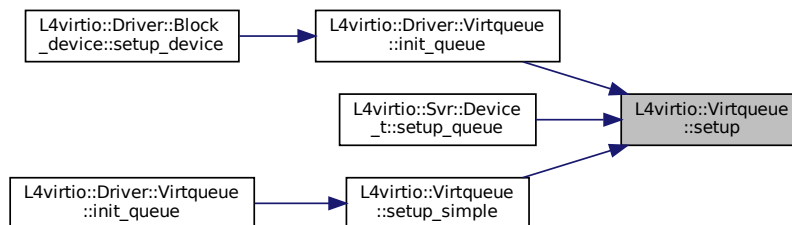
Referenced by [L4virtio::Driver::Virtqueue::init\\_queue\(\)](#), [L4virtio::Svr::Device\\_t< DATA >::setup\\_queue\(\)](#), and [setup\\_simple\(\)](#).



Here is the call graph for this function:



Here is the caller graph for this function:



### 15.354.2.15 setup\_simple()

```
void L4virtio::Virtqueue::setup_simple (
 unsigned num,
 void * ring) [inline]
```

Enable this queue.

#### Parameters

|             |                                                                                                                                                                                                                       |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>num</i>  | The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2).                                                                                                          |
| <i>ring</i> | The base address for the queue data structure. The memory block at <code>ring</code> must be at least <code>total_size(num)</code> bytes in size and have an alignment of <code>Desc_align(desc_align())</code> bits. |

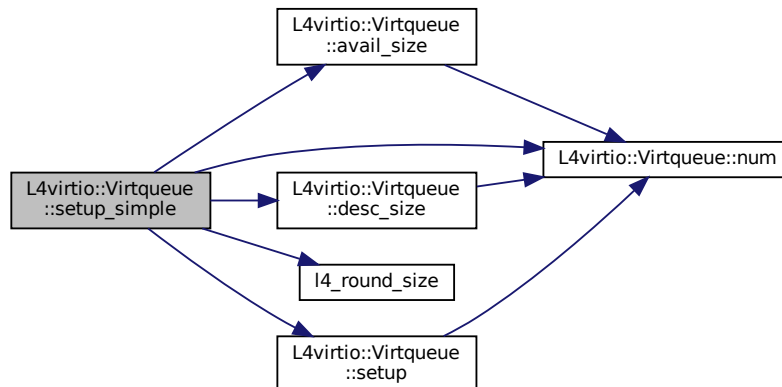
Due to the data type of the descriptors, the queue can have a maximum size of  $2^{16}$ .

Definition at line 384 of file [virtqueue](#).

References [avail\\_size\(\)](#), [desc\\_size\(\)](#), [l4\\_round\\_size\(\)](#), [num\(\)](#), and [setup\(\)](#).

Referenced by [L4virtio::Driver::Virtqueue::init\\_queue\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 15.354.2.16 total\_size() [1/2]

```
unsigned long L4virtio::Virtqueue::total_size () const [inline]
```

Calculate the total size of this virtqueue.

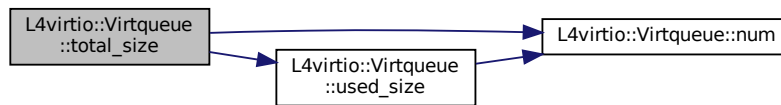
##### Precondition

The queue has been set up.

Definition at line 320 of file [virtqueue](#).

References [\\_desc](#), [\\_used](#), [num\(\)](#), and [used\\_size\(\)](#).

Here is the call graph for this function:



### 15.354.2.17 total\_size() [2/2]

```
static unsigned long L4virtio::Virtqueue::total_size (
 unsigned num) [inline], [static]
```

Calculate the total size for a virtqueue of the given dimensions.

#### Parameters

|            |                                                                                                              |
|------------|--------------------------------------------------------------------------------------------------------------|
| <i>num</i> | The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2). |
|------------|--------------------------------------------------------------------------------------------------------------|

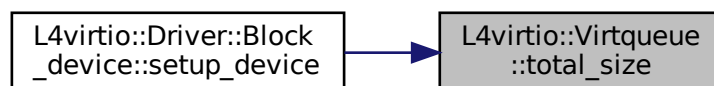
#### Returns

The total size in bytes of the queue data structures.

Definition at line 251 of file [virtqueue](#).

Referenced by [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#).

Here is the caller graph for this function:



#### 15.354.2.18 `used_align()`

```
static unsigned long L4virtio::Virtqueue::used_align () [inline], [static]
```

Get the alignment in zero LSBs needed for the used ring.

##### Returns

The alignment in zero LSBs needed for an used ring.

Definition at line 312 of file [virtqueue](#).

#### 15.354.2.19 `used_size()`

```
static unsigned long L4virtio::Virtqueue::used_size (
 unsigned num) [inline], [static]
```

Calculate the size of the used ring for `num` entries.

##### Parameters

|            |                                         |
|------------|-----------------------------------------|
| <i>num</i> | The number of entries in the used ring. |
|------------|-----------------------------------------|

##### Returns

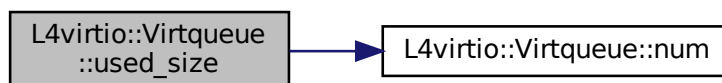
The size in bytes needed for an used ring with `num` entries.

Definition at line 304 of file [virtqueue](#).

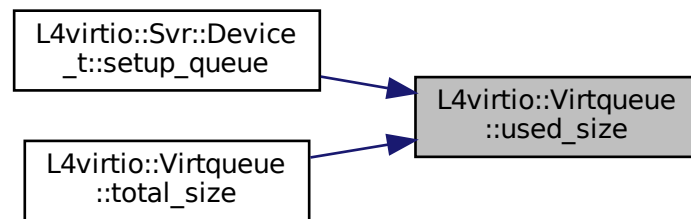
References [num\(\)](#).

Referenced by [L4virtio::Svr::Device\\_t< DATA >::setup\\_queue\(\)](#), and [total\\_size\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



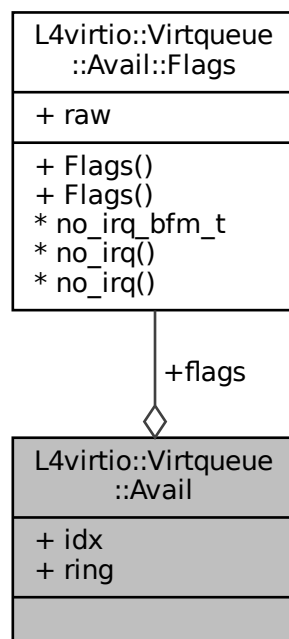
The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

## 15.355 L4virtio::Virtqueue::Avail Class Reference

Type of available ring, this is read-only for the host.

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.*

### 15.355.1 Detailed Description

Type of available ring, this is read-only for the host.

Definition at line [136](#) of file [virtqueue](#).

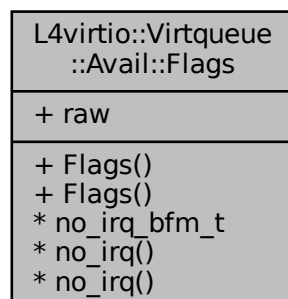
The documentation for this class was generated from the following file:

- [l4/l4virtio/virtqueue](#)

## 15.356 L4virtio::Virtqueue::Avail::Flags Struct Reference

[Flags](#) of the available ring.

Collaboration diagram for L4virtio::Virtqueue::Avail::Flags:



## 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.
- typedef [cxx::Bitfield](#)< decltype([raw](#)), 0, 0 > [no\\_irq\\_bfm\\_t](#)  
Guest does not want to receive interrupts when requests are finished.
- [no\\_irq\\_bfm\\_t::Val](#) [no\\_irq](#) () const  
Get the `no_irq` bits ( 0 to 0 ) of `raw`.
- [no\\_irq\\_bfm\\_t::Ref](#) [no\\_irq](#) ()  
Get a reference to the `no_irq` bits ( 0 to 0 ) of `raw`.

### 15.356.1 Detailed Description

[Flags](#) of the available ring.

Definition at line 142 of file [virtqueue](#).

### 15.356.2 Member Typedef Documentation

#### 15.356.2.1 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 151 of file [virtqueue](#).

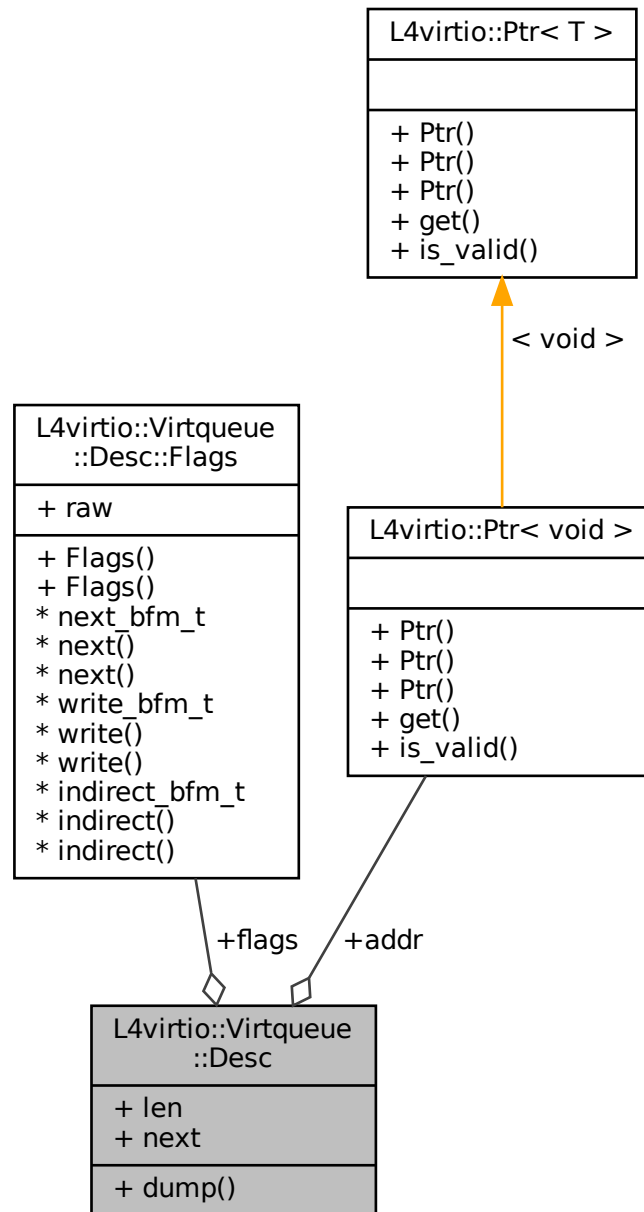
The documentation for this struct was generated from the following file:

- [l4/l4virtio/virtqueue](#)

## 15.357 L4virtio::Virtqueue::Desc Class Reference

Descriptor in the descriptor table.

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.*

### 15.357.1 Detailed Description

Descriptor in the descriptor table.

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

The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

## 15.358 L4virtio::Virtqueue::Desc::Flags Struct Reference

Type for descriptor flags.

Collaboration diagram for L4virtio::Virtqueue::Desc::Flags:

| L4virtio::Virtqueue<br>::Desc::Flags                                                                                                                          |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| + raw                                                                                                                                                         |
| + Flags()<br>+ Flags()<br>* next_bfm_t<br>* next()<br>* next()<br>* write_bfm_t<br>* write()<br>* write()<br>* indirect_bfm_t<br>* indirect()<br>* 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.
- typedef [cxx::Bitfield](#)< decltype(raw), 0, 0 > [next\\_bfm\\_t](#)  
Part of a descriptor chain which is continued with the next field.
- [next\\_bfm\\_t::Val](#) next () const  
Get the *next* bits ( 0 to 0 ) of raw.
- [next\\_bfm\\_t::Ref](#) next ()  
Get a reference to the *next* bits ( 0 to 0 ) of raw.
- typedef [cxx::Bitfield](#)< decltype(raw), 1, 1 > [write\\_bfm\\_t](#)  
Block described by this descriptor is writeable.
- [write\\_bfm\\_t::Val](#) write () const  
Get the *write* bits ( 1 to 1 ) of raw.
- [write\\_bfm\\_t::Ref](#) write ()  
Get a reference to the *write* bits ( 1 to 1 ) of raw.
- typedef [cxx::Bitfield](#)< decltype(raw), 2, 2 > [indirect\\_bfm\\_t](#)  
Indirect descriptor, block contains a list of descriptors.
- [indirect\\_bfm\\_t::Val](#) indirect () const  
Get the *indirect* bits ( 2 to 2 ) of raw.
- [indirect\\_bfm\\_t::Ref](#) indirect ()  
Get a reference to the *indirect* bits ( 2 to 2 ) of raw.

### 15.358.1 Detailed Description

Type for descriptor flags.

Definition at line 101 of file [virtqueue](#).

### 15.358.2 Member Typedef Documentation

#### 15.358.2.1 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 114 of file [virtqueue](#).

### 15.358.2.2 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 110 of file [virtqueue](#).

### 15.358.2.3 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 112 of file [virtqueue](#).

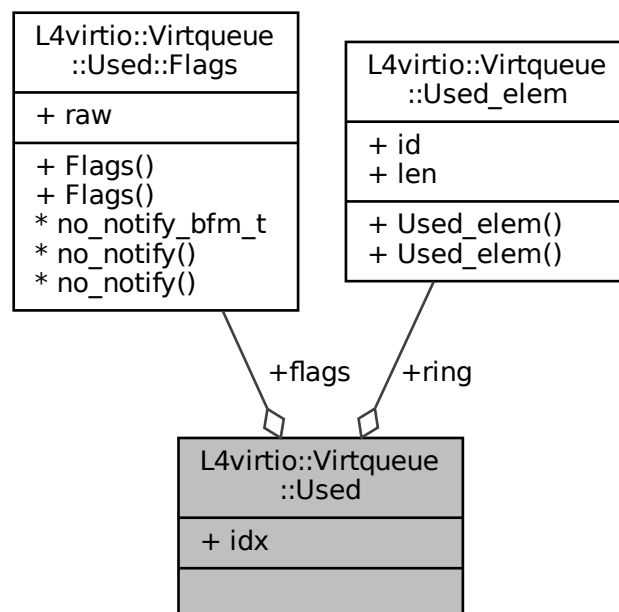
The documentation for this struct was generated from the following file:

- [l4/l4virtio/virtqueue](#)

## 15.359 L4virtio::Virtqueue::Used Class Reference

[Used](#) ring.

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.*

### 15.359.1 Detailed Description

[Used](#) ring.

Definition at line [181](#) of file [virtqueue](#).

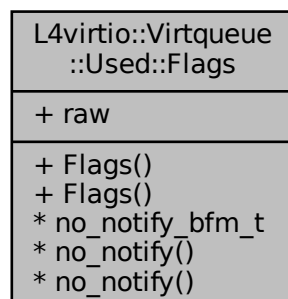
The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

## 15.360 L4virtio::Virtqueue::Used::Flags Struct Reference

flags for the used ring.

Collaboration diagram for L4virtio::Virtqueue::Used::Flags:



## 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.*
- typedef [cxx::Bitfield](#)< decltype([raw](#)), 0, 0 > [no\\_notify\\_bfm\\_t](#)  
*host does not want to be notified when new requests have been queued.*
- [no\\_notify\\_bfm\\_t::Val](#) [no\\_notify](#) () const  
*Get the `no_notify` bits ( 0 to 0 ) of `raw`.*
- [no\\_notify\\_bfm\\_t::Ref](#) [no\\_notify](#) ()  
*Get a reference to the `no_notify` bits ( 0 to 0 ) of `raw`.*

### 15.360.1 Detailed Description

flags for the used ring.

Definition at line 187 of file [virtqueue](#).

### 15.360.2 Member Typedef Documentation

#### 15.360.2.1 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 196 of file [virtqueue](#).

The documentation for this struct was generated from the following file:

- [l4/l4virtio/virtqueue](#)

## 15.361 L4virtio::Virtqueue::Used\_elem Struct Reference

Type of an element of the used ring.

Collaboration diagram for L4virtio::Virtqueue::Used\_elem:

| L4virtio::Virtqueue<br>::Used_elem |
|------------------------------------|
| + id<br>+ len                      |
| + Used_elem()<br>+ 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*

### 15.361.1 Detailed Description

Type of an element of the used ring.

Definition at line [162](#) of file [virtqueue](#).

### 15.361.2 Constructor & Destructor Documentation

#### 15.361.2.1 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 173 of file [virtqueue](#).

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtqueue

## 15.362 l4virtio\_block\_config\_t Struct Reference

Device configuration for block devices.

```
#include <virtio_block.h>
```

Collaboration diagram for l4virtio\_block\_config\_t:



### 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.*

### 15.362.1 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:

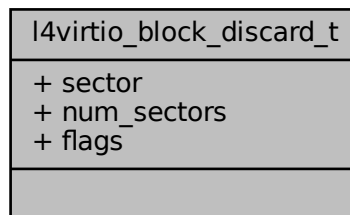
- l4/l4virtio/virtio\_block.h

## 15.363 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:



### 15.363.1 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

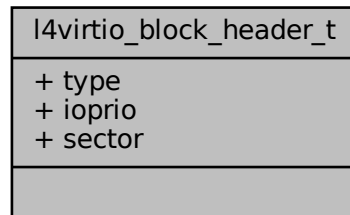


## 15.364 l4virtio\_block\_header\_t Struct Reference

Header structure of a request for a block device.

```
#include <virtio_block.h>
```

Collaboration diagram for l4virtio\_block\_header\_t:



### 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.*

### 15.364.1 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

## 15.365 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:



### 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.*

### 15.365.1 Detailed Description

L4-VIRTIO config header, provided in shared data space.

Definition at line 122 of file [virtio.h](#).

### 15.365.2 Field Documentation

#### 15.365.2.1 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 161 of file [virtio.h](#).

Referenced by [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:

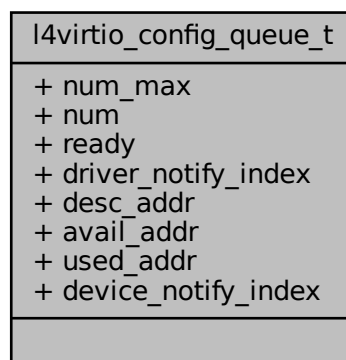
- l4/l4virtio/virtio.h

## 15.366 l4virtio\_config\_queue\_t Struct Reference

Queue configuration entry.

```
#include <virtio.h>
```

Collaboration diagram for l4virtio\_config\_queue\_t:



## 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)*

### 15.366.1 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 203 of file [virtio.h](#).

The documentation for this struct was generated from the following file:

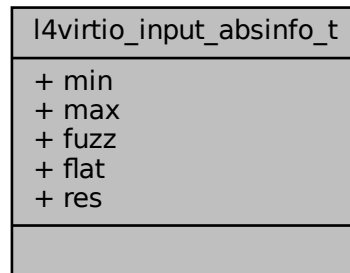
- `l4/l4virtio/virtio.h`

## 15.367 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:



### 15.367.1 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:

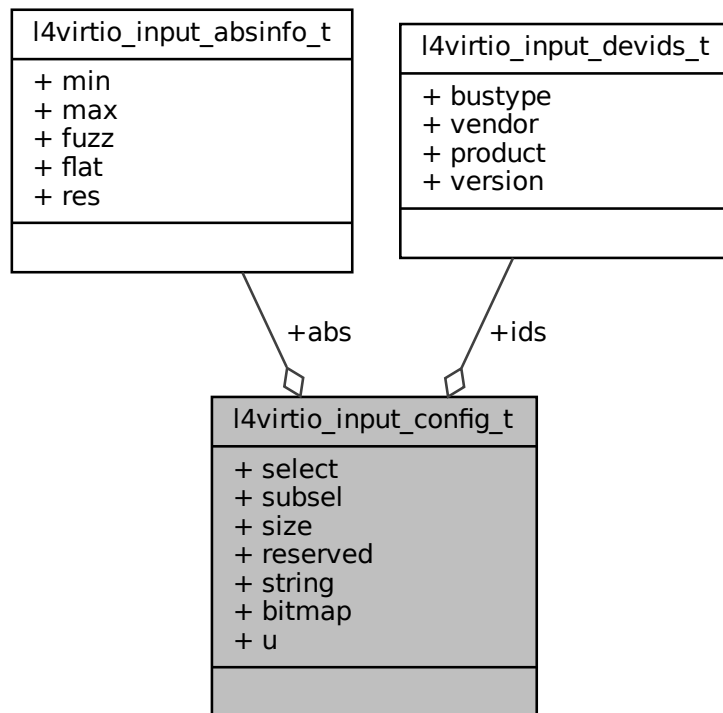
- l4/l4virtio/virtio\_input.h

## 15.368 l4virtio\_input\_config\_t Struct Reference

Device configuration for input devices.

```
#include <virtio_input.h>
```

Collaboration diagram for `l4virtio_input_config_t`:



### 15.368.1 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:

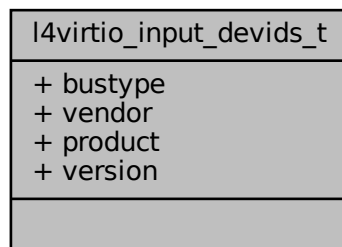
- `l4/l4virtio/virtio_input.h`

## 15.369 l4virtio\_input\_devids\_t Struct Reference

Device ID information for the device.

```
#include <virtio_input.h>
```

Collaboration diagram for l4virtio\_input\_devids\_t:



### 15.369.1 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:

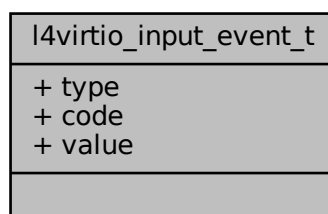
- l4/l4virtio/virtio\_input.h

## 15.370 l4virtio\_input\_event\_t Struct Reference

Single event in event or status queue.

```
#include <virtio_input.h>
```

Collaboration diagram for l4virtio\_input\_event\_t:



### 15.370.1 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



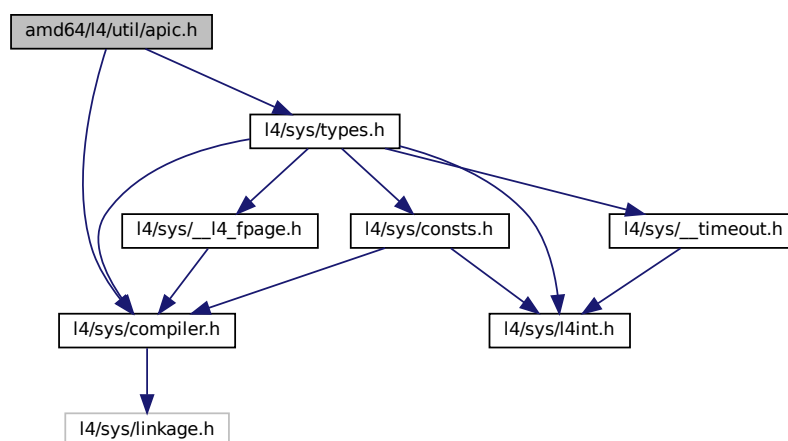
## Chapter 16

# File Documentation

### 16.1 amd64/l4/util/apic.h File Reference

APIC for AMD64.

```
#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
Include dependency graph for apic.h:
```



#### 16.1.1 Detailed Description

APIC for AMD64.

Definition in file [apic.h](#).

## 16.2 apic.h

```

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 __L4_UTIL_APIC_H
00014 #define __L4_UTIL_APIC_H
00015
00016 /*
00017 * Local APIC programming library
00018 *
00019 * For documentation, see
00020 *
00021 * "Intel Architecture Software Developer's Manual", Volume 3, chapter 7.5:
00022 * "Advanced Programmable Interrupt Controller (APIC)"
00023 *
00024 * Local APIC is present since
00025 * - INTEL P6 (PPro)
00026 * - AMD K7 (Athlon), Model 2
00027 *
00028 * In non-SMP-boards, local APIC is disabled, but
00029 * can be activated by writing to a MSR register.
00030 * For using APIC see packets cpufreq and l4rtl.
00031 *
00032 * See linux/include/asm-i386/i82489.h for further details.
00033 */
00034
00035 #define APIC_PHYS_BASE 0xFEE00000
00036 #define APIC_MAP_BASE 0xA0200000
00037 #define APIC_BASE_MSR 0x1b
00038
00039 #define APIC_ID 0x20
00040 #define GET_APIC_ID(x) ((x)>>24)&0x0F)
00041 #define APIC_LVR 0x30
00042 #define GET_APIC_VERSION(x) ((x)&0xFF)
00043 #define APIC_TASKPRI 0x80
00044 #define APIC_TPRI_MASK 0xFF
00045 #define APIC_EOI 0xB0
00046 #define APIC_LDR 0xD0
00047 #define APIC_LDR_MASK (0xFF<<24)
00048 #define APIC_DFR 0xE0
00049 #define SET_APIC_DFR(x) ((x)<<28)
00050 #define APIC_SPIV 0xF0
00051 #define APIC_LVTT 0x320
00052 #define APIC_LVTPC 0x340
00053 #define APIC_LVT0 0x350
00054 #define SET_APIC_TIMER_BASE(x) ((x)<<18)
00055 #define APIC_TIMER_BASE_DIV 0x2
00056 #define APIC_LVT1 0x360
00057 #define APIC_LVTERR 0x370
00058 #define APIC_TMICT 0x380
00059 #define APIC_TMCCT 0x390
00060 #define APIC_TDCR 0x3E0
00061
00062 #define APIC_LVT_MASKED (1<<16)
00063 #define APIC_LVT_TIMER_PERIODIC (1<<17)
00064 #define APIC_TDR_DIV_1 0xB
00065 #define APIC_TDR_DIV_2 0x0
00066 #define APIC_TDR_DIV_4 0x1
00067 #define APIC_TDR_DIV_8 0x2
00068 #define APIC_TDR_DIV_16 0x3
00069 #define APIC_TDR_DIV_32 0x8
00070 #define APIC_TDR_DIV_64 0x9
00071 #define APIC_TDR_DIV_128 0xA
00072
00073 #include <l4/sys/compiler.h>
00074 #include <l4/sys/types.h>
00075
00076 EXTERN_C_BEGIN
00077
00078 /* prototypes */
00079 extern unsigned long apic_map_base;
00080 extern unsigned long apic_timer_divisor;
00081
00082 extern unsigned long l4_scaler_apic_to_ms;
00083
00084 L4_CV void apic_show_registers(void);
00085 L4_CV int apic_check_working(void);
00086 L4_CV void apic_activate_by_io(void);
00087 L4_CV void apic_timer_set_divisor(int divisor);
00088

```

```

00089 L4_CV unsigned long l4_calibrate_apic(void);
00090
00091 EXTERN_C_END
00092
00093 L4_INLINE void apic_write(unsigned long reg, unsigned long v);
00094 L4_INLINE unsigned long apic_read(unsigned long reg);
00095 L4_INLINE void apic_activate_by_msr(void);
00096 L4_INLINE void apic_deactivate_by_msr(void);
00097 L4_INLINE unsigned long apic_read_phys_address(void);
00098 L4_INLINE int apic_test_present(void);
00099 L4_INLINE void apic_soft_enable(void);
00100 L4_INLINE void apic_init(unsigned long map_addr);
00101 L4_INLINE void apic_done(void);
00102 L4_INLINE void apic_irq_ack(void);
00103
00104 L4_INLINE void apic_lvt0_disable_irq(void);
00105 L4_INLINE void apic_lvt0_enable_irq(void);
00106 L4_INLINE void apic_lvt1_disable_irq(void);
00107 L4_INLINE void apic_lvt1_enable_irq(void);
00108
00109 L4_INLINE void apic_timer_write(unsigned long value);
00110 L4_INLINE unsigned long apic_timer_read(void);
00111 L4_INLINE void apic_timer_disable_irq(void);
00112 L4_INLINE void apic_timer_enable_irq(void);
00113 L4_INLINE void apic_timer_assign_irq(unsigned long vector);
00114 L4_INLINE void apic_timer_set_periodic(void);
00115 L4_INLINE void apic_timer_set_one_shot(void);
00116
00117 L4_INLINE void apic_perf_disable_irq(void);
00118 L4_INLINE void apic_perf_enable_irq(void);
00119 L4_INLINE void apic_perf_assign_irq(unsigned long vector);
00120
00121
00122 /* write APIC register */
00123 L4_INLINE void
00124 apic_write(unsigned long reg, unsigned long v)
00125 {
00126 *((volatile unsigned long *) (apic_map_base+reg))=v;
00127 }
00128
00129
00130 /* read APIC register */
00131 L4_INLINE unsigned long
00132 apic_read(unsigned long reg)
00133 {
00134 return *((volatile unsigned long *) (apic_map_base+reg));
00135 }
00136
00137
00138 /* disable LINT0 */
00139 L4_INLINE void
00140 apic_lvt0_disable_irq(void)
00141 {
00142 unsigned long tmp_val;
00143 tmp_val = apic_read(APIC_LVT0);
00144 tmp_val |= APIC_LVT_MASKED;
00145 apic_write(APIC_LVT0, tmp_val);
00146 }
00147
00148
00149 /* enable LINT0 */
00150 L4_INLINE void
00151 apic_lvt0_enable_irq(void)
00152 {
00153 unsigned long tmp_val;
00154 tmp_val = apic_read(APIC_LVT0);
00155 tmp_val &= ~(APIC_LVT_MASKED);
00156 apic_write(APIC_LVT0, tmp_val);
00157 }
00158
00159
00160 /* disable LINT1 */
00161 L4_INLINE void
00162 apic_lvt1_disable_irq(void)
00163 {
00164 unsigned long tmp_val;
00165 tmp_val = apic_read(APIC_LVT1);
00166 tmp_val |= APIC_LVT_MASKED;
00167 apic_write(APIC_LVT1, tmp_val);
00168 }
00169
00170
00171 /* enable LINT1 */
00172 L4_INLINE void
00173 apic_lvt1_enable_irq(void)
00174 {
00175 unsigned long tmp_val;

```

```
00176 tmp_val = apic_read(APIC_LVT1);
00177 tmp_val &= ~(APIC_LVT_MASKED);
00178 apic_write(APIC_LVT1, tmp_val);
00179 }
00180
00181
00182 /* write APIC timer register */
00183 L4_INLINE void
00184 apic_timer_write(unsigned long value)
00185 {
00186 apic_read(APIC_TMICT);
00187 apic_write(APIC_TMICT, value);
00188 }
00189
00190
00191 /* read APIC timer register */
00192 L4_INLINE unsigned long
00193 apic_timer_read(void)
00194 {
00195 return apic_read(APIC_TMCCT);
00196 }
00197
00198
00199 /* disable IRQ when APIC timer passes 0 */
00200 L4_INLINE void
00201 apic_timer_disable_irq(void)
00202 {
00203 unsigned long tmp_val;
00204 tmp_val = apic_read(APIC_LVTT);
00205 tmp_val |= APIC_LVT_MASKED;
00206 apic_write(APIC_LVTT, tmp_val);
00207 }
00208
00209
00210 /* enable IRQ when APIC timer passes 0 */
00211 L4_INLINE void
00212 apic_timer_enable_irq(void)
00213 {
00214 unsigned long tmp_val;
00215 tmp_val = apic_read(APIC_LVTT);
00216 tmp_val &= ~(APIC_LVT_MASKED);
00217 apic_write(APIC_LVTT, tmp_val);
00218 }
00219
00220
00221 L4_INLINE void
00222 apic_timer_set_periodic(void)
00223 {
00224 unsigned long tmp_val;
00225 tmp_val = apic_read(APIC_LVTT);
00226 tmp_val |= APIC_LVT_TIMER_PERIODIC;
00227 tmp_val |= APIC_LVT_MASKED;
00228 apic_write(APIC_LVTT, tmp_val);
00229 }
00230
00231
00232 L4_INLINE void
00233 apic_timer_set_one_shot(void)
00234 {
00235 unsigned long tmp_val;
00236 tmp_val = apic_read(APIC_LVTT);
00237 tmp_val &= ~APIC_LVT_TIMER_PERIODIC;
00238 tmp_val |= APIC_LVT_MASKED;
00239 apic_write(APIC_LVTT, tmp_val);
00240 }
00241
00242
00243 /* set vector of APIC timer irq */
00244 L4_INLINE void
00245 apic_timer_assign_irq(unsigned long vector)
00246 {
00247 unsigned long tmp_val;
00248 tmp_val = apic_read(APIC_LVTT);
00249 tmp_val &= 0xfffff00;
00250 tmp_val |= vector;
00251 tmp_val |= APIC_LVT_MASKED;
00252 apic_write(APIC_LVTT, tmp_val);
00253 }
00254
00255
00256 /* disable IRQ when performance counter passes 0 */
00257 L4_INLINE void
00258 apic_perf_disable_irq(void)
00259 {
00260 unsigned long tmp_val;
00261 tmp_val = apic_read(APIC_LVTPC);
00262 tmp_val |= APIC_LVT_MASKED;
```

```

00263 apic_write(APIC_LVTPC, tmp_val);
00264 }
00265
00266
00267 /* enable IRQ when performance counter passes 0 */
00268 L4_INLINE void
00269 apic_perf_enable_irq(void)
00270 {
00271 unsigned long tmp_val;
00272 tmp_val = apic_read(APIC_LVTPC);
00273 tmp_val &= ~(APIC_LVT_MASKED);
00274 apic_write(APIC_LVTPC, tmp_val);
00275 }
00276
00277
00278 /* set vector of performance counter irq */
00279 L4_INLINE void
00280 apic_perf_assign_irq(unsigned long vector)
00281 {
00282 unsigned long tmp_val;
00283 tmp_val = apic_read(APIC_LVTPC);
00284 tmp_val &= 0xfffff00;
00285 tmp_val |= vector;
00286 tmp_val |= APIC_LVT_MASKED;
00287 apic_write(APIC_LVTPC, tmp_val);
00288 }
00289
00290
00291 /* activate APIC by writing to appropriate MSR */
00292 L4_INLINE void
00293 apic_activate_by_msr(void)
00294 {
00295 unsigned long low;
00296 unsigned long high;
00297
00298 /* rdmsr */
00299 asm volatile(".byte 0xf; .byte 0x32\n"
00300 : "=a" (low),
00301 "=d" (high)
00302 : "c" (APIC_BASE_MSR)
00303);
00304
00305 low |= 0x800; /* activate APIC */
00306 low &= 0x00000fff;
00307 low |= (APIC_PHYS_BASE & 0xfffff000); /* set address */
00308
00309 /* wrmsr */
00310 asm volatile(".byte 0xf; .byte 0x30\n"
00311 :
00312 : "c" (APIC_BASE_MSR),
00313 "a" (low),
00314 "d" (high)
00315);
00316 }
00317
00318
00319 /* deactivate APIC by writing to appropriate MSR */
00320 L4_INLINE void
00321 apic_deactivate_by_msr(void)
00322 {
00323 unsigned long low;
00324 unsigned long high;
00325
00326 /* rdmsr */
00327 asm volatile(".byte 0xf; .byte 0x32\n"
00328 : "=a" (low),
00329 "=d" (high)
00330 : "c" (APIC_BASE_MSR)
00331);
00332
00333 low &= 0xfffff7ff; /* deactivate APIC */
00334
00335 /* wrmsr */
00336 asm volatile(".byte 0xf; .byte 0x30\n"
00337 :
00338 : "c" (APIC_BASE_MSR),
00339 "a" (low),
00340 "d" (high)
00341);
00342 }
00343
00344
00345 /* read memory mapped address of apic */
00346 L4_INLINE unsigned long
00347 apic_read_phys_address(void)
00348 {
00349 unsigned long low;

```

```

00350 unsigned long high;
00351
00352 /* rdmsr */
00353 asm volatile(".byte 0xf; .byte 0x32\n"
00354 : "=a" (low),
00355 "=d" (high)
00356 : "c" (APIC_BASE_MSR)
00357);
00358
00359 return (low &= 0xfffff000);
00360 }
00361
00362
00363 /* test if APIC present */
00364 L4_INLINE int
00365 apic_test_present(void)
00366 {
00367 unsigned int dummy;
00368 unsigned int capability;
00369
00370 asm volatile("cpuid"
00371 : "=a" (dummy),
00372 "=d" (capability)
00373 : "a" (0x00000001)
00374 : "cc", "rbx", "rcx");
00375
00376 return ((capability & 1<9) !=0);
00377 }
00378
00379
00380 L4_INLINE void
00381 apic_soft_enable(void)
00382 {
00383 unsigned long tmp_val;
00384 tmp_val = apic_read(APIC_SPIV);
00385 tmp_val |= (1<8); /* enable APIC */
00386 tmp_val &= ~(1<9); /* enable Focus Processor Checking */
00387 tmp_val |= 0xff; /* Set spurious IRQ vector to 0xff */
00388 apic_write(APIC_SPIV, tmp_val);
00389 }
00390
00391
00392 L4_INLINE void
00393 apic_init(unsigned long base_addr)
00394 {
00395 apic_map_base = base_addr;
00396 }
00397
00398
00399 L4_INLINE void
00400 apic_done(void)
00401 {
00402 apic_map_base = 0;
00403 }
00404
00405
00406 L4_INLINE void
00407 apic_irq_ack(void)
00408 {
00409 apic_read(APIC_SPIV);
00410 apic_write(APIC_EOI, 0);
00411 }
00412
00413
00414 #endif /* __L4_UTIL_APIC_H */

```

## 16.3 x86/I4/util/apic.h File Reference

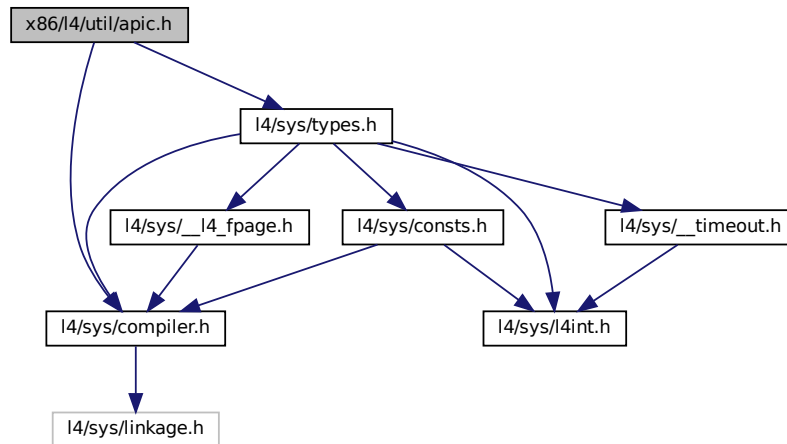
APIC for X86.

```

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

```

Include dependency graph for apic.h:



### 16.3.1 Detailed Description

APIC for X86.

Definition in file [apic.h](#).

## 16.4 apic.h

```

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 __L4_UTIL_APIC_H
00014 #define __L4_UTIL_APIC_H
00015
00016 /*
00017 * Local APIC programming library
00018 *
00019 * For documentation, see
00020 *
00021 * "Intel Architecture Software Developer's Manual", Volume 3, chapter 7.5:
00022 * "Advanced Programmable Interrupt Controller (APIC)"
00023 *
00024 * Local APIC is present since
00025 * - INTEL P6 (PPro)
00026 * - AMD K7 (Athlon), Model 2
00027 *
00028 * In non-SMP-boards, local APIC is disabled, but
00029 * can be activated by writing to a MSR register.
00030 * For using APIC see packets cpufreq and l4rtl.
00031 *
00032 * See linux/include/asm-i386/i82489.h for further details.
00033 */
00034
00035 #define APIC_PHYS_BASE 0xFEE00000
00036 #define APIC_MAP_BASE 0xA0200000
00037 #define APIC_BASE_MSR 0x1b
00038
00039 #define APIC_ID 0x20
00040 #define GET_APIC_ID(x) (((x) >> 24) & 0x0F)

```

```

00041 #define APIC_LVR 0x30
00042 #define GET_APIC_VERSION(x) ((x)&0xFF)
00043 #define APIC_TASKPRI 0x80
00044 #define APIC_TPRI_MASK 0xFF
00045 #define APIC_EOI 0xB0
00046 #define APIC_LDR 0xD0
00047 #define APIC_LDR_MASK (0xFF<<24)
00048 #define APIC_DFR 0xE0
00049 #define SET_APIC_DFR(x) ((x)<<28)
00050 #define APIC_SPIV 0xF0
00051 #define APIC_LVT 0x320
00052 #define APIC_LVTPC 0x340
00053 #define APIC_LVT0 0x350
00054 #define SET_APIC_TIMER_BASE(x) (((x)<<18))
00055 #define APIC_TIMER_BASE_DIV 0x2
00056 #define APIC_LVT1 0x360
00057 #define APIC_LVTERR 0x370
00058 #define APIC_TMICT 0x380
00059 #define APIC_TMCCT 0x390
00060 #define APIC_TDCR 0x3E0
00061
00062 #define APIC_LVT_MASKED (1<<16)
00063 #define APIC_LVT_TIMER_PERIODIC (1<<17)
00064 #define APIC_TDR_DIV_1 0xB
00065 #define APIC_TDR_DIV_2 0x0
00066 #define APIC_TDR_DIV_4 0x1
00067 #define APIC_TDR_DIV_8 0x2
00068 #define APIC_TDR_DIV_16 0x3
00069 #define APIC_TDR_DIV_32 0x8
00070 #define APIC_TDR_DIV_64 0x9
00071 #define APIC_TDR_DIV_128 0xA
00072
00073 #include <14/sys/compiler.h>
00074 #include <14/sys/types.h>
00075
00076 EXTERN_C_BEGIN
00077
00078 /* prototypes */
00079 extern unsigned long apic_map_base;
00080 extern unsigned long apic_timer_divisor;
00081
00082 extern unsigned long l4_scaler_apic_to_ms;
00083
00084 L4_CV void apic_show_registers(void);
00085 L4_CV int apic_check_working(void);
00086 L4_CV void apic_activate_by_io(void);
00087 L4_CV void apic_timer_set_divisor(int divisor);
00088
00089 L4_CV unsigned long l4_calibrate_apic(void);
00090
00091 EXTERN_C_END
00092
00093 L4_INLINE void apic_write(unsigned long reg, unsigned long v);
00094 L4_INLINE unsigned long apic_read(unsigned long reg);
00095 L4_INLINE void apic_activate_by_msr(void);
00096 L4_INLINE void apic_deactivate_by_msr(void);
00097 L4_INLINE unsigned long apic_read_phys_address(void);
00098 L4_INLINE int apic_test_present(void);
00099 L4_INLINE void apic_soft_enable(void);
00100 L4_INLINE void apic_init(unsigned long map_addr);
00101 L4_INLINE void apic_done(void);
00102 L4_INLINE void apic_irq_ack(void);
00103
00104 L4_INLINE void apic_lvt0_disable_irq(void);
00105 L4_INLINE void apic_lvt0_enable_irq(void);
00106 L4_INLINE void apic_lvt1_disable_irq(void);
00107 L4_INLINE void apic_lvt1_enable_irq(void);
00108
00109 L4_INLINE void apic_timer_write(unsigned long value);
00110 L4_INLINE unsigned long apic_timer_read(void);
00111 L4_INLINE void apic_timer_disable_irq(void);
00112 L4_INLINE void apic_timer_enable_irq(void);
00113 L4_INLINE void apic_timer_assign_irq(unsigned long vector);
00114 L4_INLINE void apic_timer_set_periodic(void);
00115 L4_INLINE void apic_timer_set_one_shot(void);
00116
00117 L4_INLINE void apic_perf_disable_irq(void);
00118 L4_INLINE void apic_perf_enable_irq(void);
00119 L4_INLINE void apic_perf_assign_irq(unsigned long vector);
00120
00121
00122 /* write APIC register */
00123 L4_INLINE void
00124 apic_write(unsigned long reg, unsigned long v)
00125 {
00126 *((volatile unsigned long *) (apic_map_base+reg))=v;
00127 }

```



```

00128
00129
00130 /* read APIC register */
00131 L4_INLINE unsigned long
00132 apic_read(unsigned long reg)
00133 {
00134 return *((volatile unsigned long *) (apic_map_base+reg));
00135 }
00136
00137
00138 /* disable LINT0 */
00139 L4_INLINE void
00140 apic_lvt0_disable_irq(void)
00141 {
00142 unsigned long tmp_val;
00143 tmp_val = apic_read(APIC_LVT0);
00144 tmp_val |= APIC_LVT_MASKED;
00145 apic_write(APIC_LVT0, tmp_val);
00146 }
00147
00148
00149 /* enable LINT0 */
00150 L4_INLINE void
00151 apic_lvt0_enable_irq(void)
00152 {
00153 unsigned long tmp_val;
00154 tmp_val = apic_read(APIC_LVT0);
00155 tmp_val &= ~(APIC_LVT_MASKED);
00156 apic_write(APIC_LVT0, tmp_val);
00157 }
00158
00159
00160 /* disable LINT1 */
00161 L4_INLINE void
00162 apic_lvt1_disable_irq(void)
00163 {
00164 unsigned long tmp_val;
00165 tmp_val = apic_read(APIC_LVT1);
00166 tmp_val |= APIC_LVT_MASKED;
00167 apic_write(APIC_LVT1, tmp_val);
00168 }
00169
00170
00171 /* enable LINT1 */
00172 L4_INLINE void
00173 apic_lvt1_enable_irq(void)
00174 {
00175 unsigned long tmp_val;
00176 tmp_val = apic_read(APIC_LVT1);
00177 tmp_val &= ~(APIC_LVT_MASKED);
00178 apic_write(APIC_LVT1, tmp_val);
00179 }
00180
00181
00182 /* write APIC timer register */
00183 L4_INLINE void
00184 apic_timer_write(unsigned long value)
00185 {
00186 apic_read(APIC_TMICT);
00187 apic_write(APIC_TMICT, value);
00188 }
00189
00190
00191 /* read APIC timer register */
00192 L4_INLINE unsigned long
00193 apic_timer_read(void)
00194 {
00195 return apic_read(APIC_TMCCT);
00196 }
00197
00198
00199 /* disable IRQ when APIC timer passes 0 */
00200 L4_INLINE void
00201 apic_timer_disable_irq(void)
00202 {
00203 unsigned long tmp_val;
00204 tmp_val = apic_read(APIC_LVTT);
00205 tmp_val |= APIC_LVT_MASKED;
00206 apic_write(APIC_LVTT, tmp_val);
00207 }
00208
00209
00210 /* enable IRQ when APIC timer passes 0 */
00211 L4_INLINE void
00212 apic_timer_enable_irq(void)
00213 {
00214 unsigned long tmp_val;

```

```

00215 tmp_val = apic_read(APIC_LVTT);
00216 tmp_val &= ~(APIC_LVT_MASKED);
00217 apic_write(APIC_LVTT, tmp_val);
00218 }
00219
00220
00221 L4_INLINE void
00222 apic_timer_set_periodic(void)
00223 {
00224 unsigned long tmp_val;
00225 tmp_val = apic_read(APIC_LVTT);
00226 tmp_val |= APIC_LVT_TIMER_PERIODIC;
00227 tmp_val |= APIC_LVT_MASKED;
00228 apic_write(APIC_LVTT, tmp_val);
00229 }
00230
00231
00232 L4_INLINE void
00233 apic_timer_set_one_shot(void)
00234 {
00235 unsigned long tmp_val;
00236 tmp_val = apic_read(APIC_LVTT);
00237 tmp_val &= ~APIC_LVT_TIMER_PERIODIC;
00238 tmp_val |= APIC_LVT_MASKED;
00239 apic_write(APIC_LVTT, tmp_val);
00240 }
00241
00242
00243 /* set vector of APIC timer irq */
00244 L4_INLINE void
00245 apic_timer_assign_irq(unsigned long vector)
00246 {
00247 unsigned long tmp_val;
00248 tmp_val = apic_read(APIC_LVTT);
00249 tmp_val &= 0xffffffff00;
00250 tmp_val |= vector;
00251 tmp_val |= APIC_LVT_MASKED;
00252 apic_write(APIC_LVTT, tmp_val);
00253 }
00254
00255
00256 /* disable IRQ when performance counter passes 0 */
00257 L4_INLINE void
00258 apic_perf_disable_irq(void)
00259 {
00260 unsigned long tmp_val;
00261 tmp_val = apic_read(APIC_LVTPC);
00262 tmp_val |= APIC_LVT_MASKED;
00263 apic_write(APIC_LVTPC, tmp_val);
00264 }
00265
00266
00267 /* enable IRQ when performance counter passes 0 */
00268 L4_INLINE void
00269 apic_perf_enable_irq(void)
00270 {
00271 unsigned long tmp_val;
00272 tmp_val = apic_read(APIC_LVTPC);
00273 tmp_val &= ~(APIC_LVT_MASKED);
00274 apic_write(APIC_LVTPC, tmp_val);
00275 }
00276
00277
00278 /* set vector of performance counter irq */
00279 L4_INLINE void
00280 apic_perf_assign_irq(unsigned long vector)
00281 {
00282 unsigned long tmp_val;
00283 tmp_val = apic_read(APIC_LVTPC);
00284 tmp_val &= 0xffffffff00;
00285 tmp_val |= vector;
00286 tmp_val |= APIC_LVT_MASKED;
00287 apic_write(APIC_LVTPC, tmp_val);
00288 }
00289
00290
00291 /* activate APIC by writing to appropriate MSR */
00292 L4_INLINE void
00293 apic_activate_by_msr(void)
00294 {
00295 unsigned long low;
00296 unsigned long high;
00297
00298 /* rdmsr */
00299 asm volatile(".byte 0xf; .byte 0x32\n"
00300 : "=a" (low),
00301 : "=d" (high)

```

```

00302 : "c" (APIC_BASE_MSR)
00303);
00304
00305 low |= 0x800; /* activate APIC */
00306 low &= 0x00000fff;
00307 low |= (APIC_PHYS_BASE & 0xfffff000); /* set address */
00308
00309 /* wrmsr */
00310 asm volatile(".byte 0xf; .byte 0x30\n"
00311 :
00312 : "c" (APIC_BASE_MSR),
00313 "a" (low),
00314 "d" (high)
00315);
00316 }
00317
00318
00319 /* deactivate APIC by writing to appropriate MSR */
00320 L4_INLINE void
00321 apic_deactivate_by_msr(void)
00322 {
00323 unsigned long low;
00324 unsigned long high;
00325
00326 /* rdmsr */
00327 asm volatile(".byte 0xf; .byte 0x32\n"
00328 : "=a" (low),
00329 "=d" (high)
00330 : "c" (APIC_BASE_MSR)
00331);
00332
00333 low &= 0xfffff7ff; /* deactivate APIC */
00334
00335 /* wrmsr */
00336 asm volatile(".byte 0xf; .byte 0x30\n"
00337 :
00338 : "c" (APIC_BASE_MSR),
00339 "a" (low),
00340 "d" (high)
00341);
00342 }
00343
00344
00345 /* read memory mapped address of apic */
00346 L4_INLINE unsigned long
00347 apic_read_phys_address(void)
00348 {
00349 unsigned long low;
00350 unsigned long high;
00351
00352 /* rdmsr */
00353 asm volatile(".byte 0xf; .byte 0x32\n"
00354 : "=a" (low),
00355 "=d" (high)
00356 : "c" (APIC_BASE_MSR)
00357);
00358
00359 return (low &= 0xfffff000);
00360 }
00361
00362
00363 /* test if APIC present */
00364 L4_INLINE int
00365 apic_test_present(void)
00366 {
00367 unsigned int dummy;
00368 unsigned int capability;
00369
00370 asm volatile("pushl %%ebx ; cpuid ; popl %%ebx"
00371 : "=a" (dummy),
00372 "=c" (dummy),
00373 "=d" (capability)
00374 : "a" (0x00000001)
00375 : "cc");
00376
00377 return ((capability & 1<9) !=0);
00378 }
00379
00380
00381 L4_INLINE void
00382 apic_soft_enable(void)
00383 {
00384 unsigned long tmp_val;
00385 tmp_val = apic_read(APIC_SPIV);
00386 tmp_val |= (1<8); /* enable APIC */
00387 tmp_val &= ~(1<9); /* enable Focus Processor Checking */
00388 tmp_val |= 0xff; /* Set spurious IRQ vector to 0xff */

```

```

00389 apic_write(APIC_SPIV, tmp_val);
00390 }
00391
00392
00393 L4_INLINE void
00394 apic_init(unsigned long base_addr)
00395 {
00396 apic_map_base = base_addr;
00397 }
00398
00399
00400 L4_INLINE void
00401 apic_done(void)
00402 {
00403 apic_map_base = 0;
00404 }
00405
00406
00407 L4_INLINE void
00408 apic_irq_ack(void)
00409 {
00410 apic_read(APIC_SPIV);
00411 apic_write(APIC_EOI, 0);
00412 }
00413
00414
00415 #endif /* __L4_UTIL_APIC_H */

```

## 16.5 amd64/l4/util/idt.h File Reference

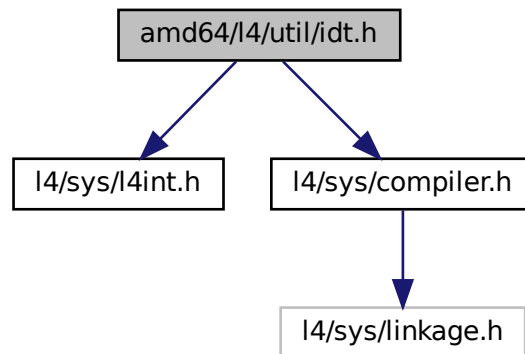
IDT related functions.

```

#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for idt.h:



## Data Structures

- struct [l4util\\_idt\\_desc\\_t](#)  
*IDT entry.*
- struct [l4util\\_idt\\_header\\_t](#)  
*Header of an IDT table.*

## 16.5.1 Detailed Description

IDT related functions.

### Date

2003

### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [idt.h](#).

## 16.6 idt.h

```

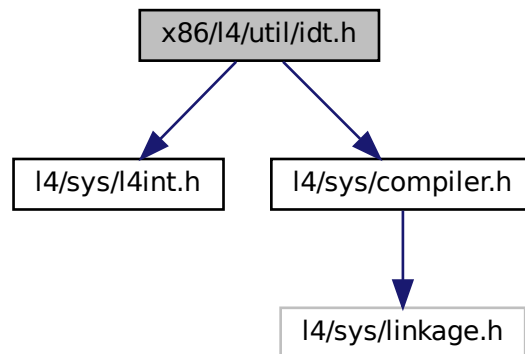
00001
00009 /*
00010 * (c) 2003-2009 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
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
00030
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

```

## 16.7 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 `l4util_idt_desc_t`  
*IDT entry.*
- struct `l4util_idt_header_t`  
*Header of an IDT table.*

### 16.7.1 Detailed Description

IDT related functions.

Date

2003

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [idt.h](#).

## 16.8 idt.h

```

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
00030
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

```

## 16.9 amd64/l4/util/perform.h File Reference

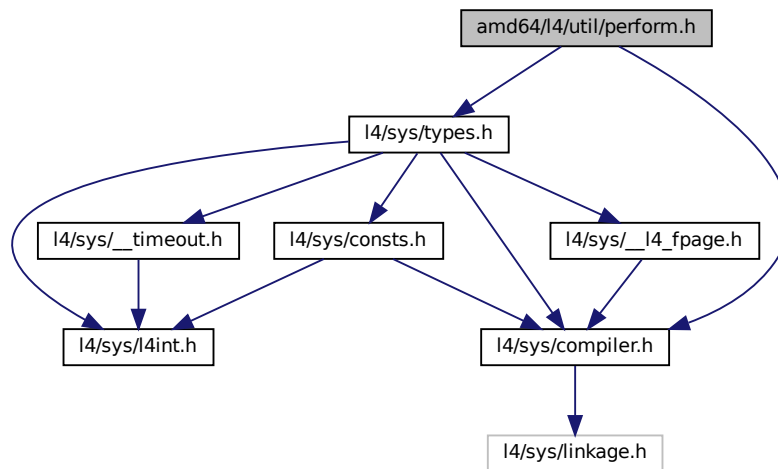
Performance Monitoring using P5/P6 Measurement Counters.

```

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

```

Include dependency graph for `perform.h`:



### 16.9.1 Detailed Description

Performance Monitoring using P5/P6 Measurement Counters.

Define either `CPU_PENTIUM` or `CPU_P6`

Definition in file [perform.h](#).

## 16.10 perform.h

```

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
00068 #ifdef CPU_PENTIUM
00069 /* Pentium section */
00070
00071 /* functions and events defined here are only usable at Pentium
00072 Processors. P6 architecture does NOT support this kind of measuring and
00073 these events. P6 architecture has its own counters and its own events.
00074 See P6-section for details. */
00075
00076 /* from l4linux/arch/l4-i386/include/perform.h */
00077
00078 static inline void
00079 l4_i586_reset_event_counter(void){
00080 asm volatile("xor %%rax, %%rax\n"
00081 "xor %%rdx, %%rdx\n"
00082 "mov $0x12, %%rcx\n"
00083 ".byte 0x0f, 0x30\n"
00084 "movl $0x13, %%rcx\n"
00085 ".byte 0x0f, 0x30\n"
00086 : : : "cx", "ax", "dx"
00087);
00088 };
00089
00090 static inline void
00091 l4_i586_read_event_counter_long(long long *counter0, long long *counter1)
00092 {
00093 asm volatile(
00094 /* "movl $0, %%eax\n"
00095 "movl $0x11, %%ecx\n"
00096 ".byte 0x0f, 0x30\n" *//* stop event counting */
00097 "mov $0x12, %%rcx\n"
00098 ".byte 0x0f, 0x32\n"
00099 "mov %%rax, (%rbx)\n"
00100 "mov %%rdx, 4(%rbx)\n"
00101 "mov $0x13, %%ecx\n"
00102 ".byte 0x0f, 0x32\n"
00103 "mov %%rax, (%rsi)\n"
00104 "mov %%rdx, 4(%rsi)\n"
00105 : /* no output */
00106 : "b" (counter0), "S" (counter1)
00107 : "ax", "cx", "dx"
00108);
00109 }
00110
00111 static inline void
00112 l4_i586_read_event_counter(int *counter0, int *counter1)
00113 {
00114 asm volatile("push %%rdx\n"
00115 ".byte 0x0f, 0x30\n"
00116 "mov $0x12, %%rcx\n"
00117 ".byte 0x0f, 0x32\n"
00118 "mov %%rax, %%rbx\n"
00119 "movl $0x13, %%rcx\n"
00120 ".byte 0x0f, 0x32\n"
00121 "popl %%edx\n"
00122 : "=b" (*counter0), "=a" (*counter1)
00123 : "1" (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_UOPS_RETIRED 0xc2 /* number of micro-ops retired */
00295 #define P6_INST_DECODER 0xd0 /* number of instructions decoded */
00296
00297 /* Interrupts */
00298 #define P6_HW_INT_RX 0xc8 /* number of hardware interrupts */

```

```

00299 #define P6_CYCLES_INT_MASKED 0xc6 /* number of cycles hardints masked */
00300 #define P6_CYCLES_INT_PENDING_AND_MASKED 0xc7 /* #cycles masked but pending */
00301
00302 /* Branches */
00303 #define P6_BR_INST_RETIRED 0xc4 /* number of branch instrs retired */
00304 #define P6_BR_MISS_PRED_RETIRED 0xc5 /* number of mispred'd brs retired */
00305 #define P6_BR_TAKEN_RETIRED 0xc9 /* number of taken branches retired */
00306 #define P6_BR_MISS_PRED_TAKEN_RET 0xca /* #taken mispredictions br's retired*/
00307 #define P6_BR_INST_DECODED 0xe0 /* number of branch instrs decoded */
00308 #define P6_BTБ_MISSES 0xe2 /* # of branches that missed in BTБ */
00309 #define P6_BR_BOGUS 0xe4 /* number of bogus branches */
00310 #define P6_BACLEAR 0xe6 /* # times BACLEAR is asserted */
00311
00312 /* Stalls */
00313 #define P6_RESOURCE_STALLS 0xa2 /* # resource-related stall cycles */
00314 #define P6_PARTIAL_RAT_STALLS 0xd2 /* # cycles/events for partial stalls*/
00315
00316 /* Segment register loads */
00317 #define P6_SEGMENT_REG_LOADS 0x06 /* number of segment register loads */
00318
00319 /* Clocks */
00320 #define P6_CPU_CLK_UNHALTED 0x79 /* #clocks CPU is not halted */
00321
00322 /* Unit field tags */
00323 #define P6_UNIT_M 0x0800
00324 #define P6_UNIT_E 0x0400
00325 #define P6_UNIT_S 0x0200
00326 #define P6_UNIT_I 0x0100
00327 #define P6_UNIT_MESI 0x0f00
00328
00329 #define P6_UNIT_SELF 0x0000
00330 #define P6_UNIT_ANY 0x2000
00331
00332 /*****
00333 ** Flag bit definitions (used for the 'flag' field in select_p6counter()) **
00334 *****/
00335 *
00336 * The driver accepts fully-formed counter specifications from user-level.
00337 * The following flags are mnemonics for the bits that get set in the
00338 * PerfEvtSel0 and PerfEvtSel1 MSR's
00339 *
00340 */
00341 #define P6CNT_U 0x010000 /* Monitor user-level events */
00342 #define P6CNT_K 0x020000 /* Monitor kernel-level events */
00343 #define P6CNT_E 0x040000 /* Edge detect: count state transitions */
00344 #define P6CNT_PC 0x080000 /* Pin control: ?? */
00345 #define P6CNT_IE 0x100000 /* Int enable: enable interrupt on overflow */
00346 #define P6CNT_F 0x200000 /* Freeze counter (handled in software) */
00347 #define P6CNT_EN 0x400000 /* enable counters (in PerfEvtSel0) */
00348 #define P6CNT_IV 0x800000 /* Invert counter mask comparison result */
00349
00350 /*****
00351 ** Miscellaneous constants **
00352 *****/
00353 *
00354 * Number of Pentium Pro programable hardware counters.
00355 */
00356 #define NUM_P6HWC 2
00357
00358 /*****
00359 *
00360 * End of Copyright by Harvard College
00361 *
00362 *****/
00363
00364
00365 #define MSR_P6_EVTSEL0 0x186
00366 #define MSR_P6_EVTSEL1 0x187
00367 #define MSR_P6_PERFCTR0 0xc1
00368 #define MSR_P6_PERFCTR1 0xc2
00369
00370 /* P6-specific Makros to manipulate and read counters */
00371
00372 /* Read the 40 bit performance monitoring counter. This requires
00373 the PCE-flag in CR4 to be set. Otherwise GP0 is raised. Works only
00374 at P6.
00375 */
00376 #define l4_i686_rdpmc(cnt, res_p) \
00377 __asm __volatile(\
00378 "mov %2, %%rcx # put counter number in \n\ \
00379 .byte 0xf; .byte 0x33 # RDPMC instruction \n\ \
00380 mov %%rdx, %1 # High order 32 bits \n\ \
00381 mov %%rax, %0 # Low order 32 bits" \
00382 : "=g" (*(int *) (res_p)), "=g" (((int *) res_p)+1) \
00383 : "g" (cnt) \
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

```

## 16.11 x86/I4/util/perform.h File Reference

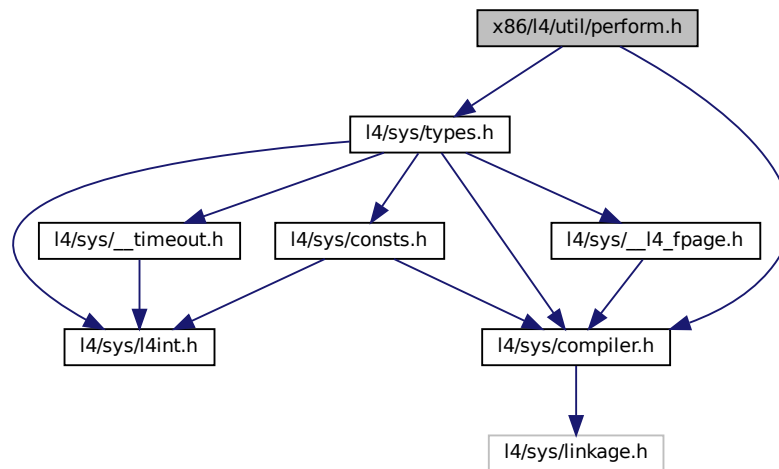
Performance Monitoring using P5/P6 Measurement Counters.

```

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

```

Include dependency graph for `perform.h`:



### 16.11.1 Detailed Description

Performance Monitoring using P5/P6 Measurement Counters.

Define either `CPU_PENTIUM` or `CPU_P6`

Definition in file [perform.h](#).

## 16.12 perform.h

```

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_rdmrs(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 : "l" (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 /* ($) 12 ifetches */
00248 #define P6_L2_LD 0x29 /* ($) 12 data loads */
00249 #define P6_L2_ST 0x2a /* ($) 12 data stores */
00250 #define P6_L2_LINES_IN 0x24 /* lines allocated in L2 */
00251 #define P6_L2_LINES_OUT 0x26 /* lines removed from L2 */
00252 #define P6_L2_M_LINES_INM 0x25 /* modified lines allocated in L2 */
00253 #define P6_L2_M_LINES_OUTM 0x27 /* modified lines removed from L2 */
00254 #define P6_L2_RQSTS 0x2e /* ($) 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_OOPS_RETIRED 0xc2 /* number of micro-ops retired */
00297 #define P6_INST_DECODER 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_BTBMISSES 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_rdpmc(cntnr, res_p) \
00379 __asm __volatile(\
00380 "movl %2, %%ecx # put counter number in \n\
00381 .byte 0xf, .byte 0x33 # RDPMC 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" (cntnr) \

```

```

00386 : "ecx", "eax", "edx")
00387
00388 static inline l4_uint32_t l4_i686_rdpmc_32(int cnt){
00389 l4_uint32_t x;
00390
00391 __asm__ __volatile__(
00392 ".byte 0xf; .byte 0x33 # RDPMC instruction"
00393 : "=a" (x)
00394 : "c" (cnt)
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

```

## 16.13 amd64/l4/util/rdtsc.h File Reference

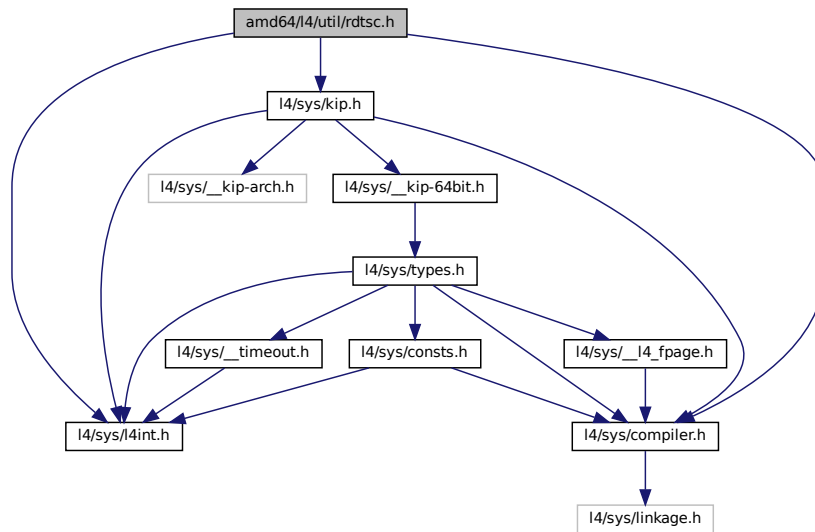
time stamp counter related functions

```

#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
#include <l4/sys/kip.h>

```

Include dependency graph for rdtsc.h:



## Macros

- `#define L4_TSC_INIT_AUTO 0`  
*Automatic init.*
- `#define L4_TSC_INIT_KERNEL 1`  
*Initialized by kernel.*
- `#define L4_TSC_INIT_CALIBRATE 2`  
*Initialized by user-level.*

## Functions

- `l4_cpu_time_t l4_rdtsc (void)`  
*Read current value of CPU-internal time stamp 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 time stamp to ns value.*
- `l4_uint64_t l4_tsc_to_us (l4_cpu_time_t tsc)`  
*Convert time stamp 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 time stamp 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](#) \*kip)
- Calibrate scalers for time stamp calculations.*
- [l4\\_uint32\\_t](#) [l4\\_tsc\\_init](#) (int constraint, [l4\\_kernel\\_info\\_t](#) \*kip)
- Initialize scaler for TSC calibrations.*
- [l4\\_uint32\\_t](#) [l4\\_get\\_hz](#) (void)
- Get CPU frequency in Hz.*

### 16.13.1 Detailed Description

time stamp counter related functions

#### Date

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [rdtsc.h](#).

### 16.13.2 Function Documentation

#### 16.13.2.1 [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 |
|-----------|----------------------|

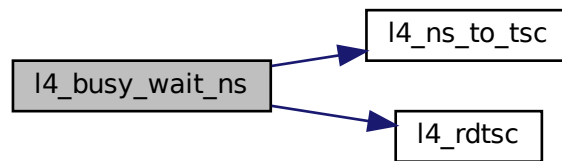
#### Attention

Not intended for any use!

Definition at line 319 of file [rdtsc.h](#).

References [l4\\_ns\\_to\\_tsc\(\)](#), and [l4\\_rdtsc\(\)](#).

Here is the call graph for this function:



#### 16.13.2.2 `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 |
|-----------|-----------------------|

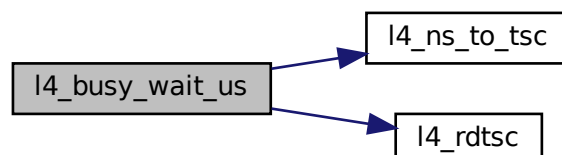
##### Attention

Not intended for any use!

Definition at line 329 of file [rdtsc.h](#).

References [l4\\_ns\\_to\\_tsc\(\)](#), and [l4\\_rdtsc\(\)](#).

Here is the call graph for this function:



### 16.13.2.3 l4\_calibrate\_tsc()

```
l4_uint32_t l4_calibrate_tsc (
 l4_kernel_info_t * kip) [inline]
```

Calibrate scalers for time stamp calculations.

Determine some scalers to be able to convert between real time and CPU ticks. This test uses channel 0 of the PIT (i8254) or the kernel KIP, depending on availability. Just calls `l4_tsc_init(L4_TSC_INIT_AUTO)`.

#### Examples

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

Definition at line 181 of file [rdtsc.h](#).

References [l4\\_tsc\\_init\(\)](#), and [L4\\_TSC\\_INIT\\_AUTO](#).

Here is the call graph for this function:



### 16.13.2.4 l4\_get\_hz()

```
l4_uint32_t l4_get_hz (
 void)
```

Get CPU frequency in Hz.

#### Returns

frequency in Hz

### 16.13.2.5 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 |
|-----------|--------------|

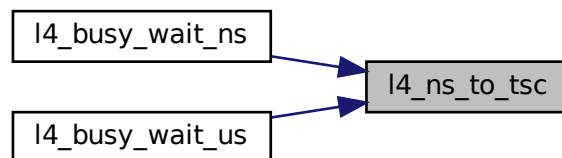
**Returns**

CPU ticks

Definition at line 305 of file [rdtsc.h](#).

Referenced by [l4\\_busy\\_wait\\_ns\(\)](#), and [l4\\_busy\\_wait\\_us\(\)](#).

Here is the caller graph for this function:

**16.13.2.6 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 207 of file [rdtsc.h](#).



### 16.13.2.7 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 229 of file [rdtsc.h](#).

### 16.13.2.8 l4\_rdtsc()

```
l4_cpu_time_t l4_rdtsc (
 void) [inline]
```

Read current value of CPU-internal time stamp counter.

#### Returns

64-bit time stamp

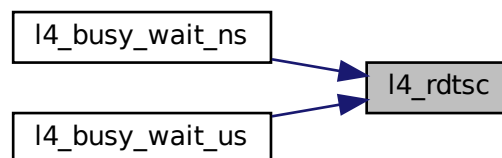
#### Examples

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

Definition at line 187 of file [rdtsc.h](#).

Referenced by [l4\\_busy\\_wait\\_ns\(\)](#), and [l4\\_busy\\_wait\\_us\(\)](#).

Here is the caller graph for this function:



### 16.13.2.9 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 248 of file [rdtsc.h](#).

### 16.13.2.10 l4\_tsc\_init()

```
l4_uint32_t l4_tsc_init (
 int constraint,
 l4_kernel_info_t * kip)
```

Initialize scaler for TSC calibrations.

Initialize the scalers needed by [l4\\_tsc\\_to\\_ns\(\)](#)/[l4\\_ns\\_to\\_tsc\(\)](#) and so on. Current versions of Fiasco export these scalers from kernel into userland. The programmer may decide whether he allows to use these scalers or if an calibration should be performed.

#### Parameters

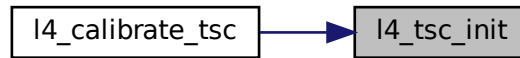
|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>constraint</i> | <p>programmers constraint:</p> <ul style="list-style-type: none"> <li>• <a href="#">L4_TSC_INIT_AUTO</a> if the kernel exports the scalers then use them. If not, perform calibration using channel 0 of the PIT (i8254). The latter case may lead into short (unpredictable) periods where interrupts are disabled.</li> <li>• <a href="#">L4_TSC_INIT_KERNEL</a> depend on retrieving the scalers from kernel. If the scalers are not available, return 0.</li> <li>• <a href="#">L4_TSC_INIT_CALIBRATE</a> Ignore possible scalers exported by the scaler, instead insist on calibration using the PIT.</li> </ul> |
| <i>kip</i>        | KIP pointer                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

#### Returns

0 on error (no scalers exported by kernel, calibrating failed ...) 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:



#### 16.13.2.11 l4\_tsc\_to\_ns()

```
l4_uint64_t l4_tsc_to_ns (
 l4_cpu_time_t tsc) [inline]
```

Convert time stamp to ns value.

##### Parameters

|                  |                         |
|------------------|-------------------------|
| <code>tsc</code> | time value in CPU ticks |
|------------------|-------------------------|

##### Returns

time value in ns

##### Examples

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

Definition at line 262 of file `rdtsc.h`.

#### 16.13.2.12 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 time stamp to s.ns value.

##### Parameters

|                  |                         |
|------------------|-------------------------|
| <code>tsc</code> | time value in CPU ticks |
|------------------|-------------------------|

## Return values

|           |              |
|-----------|--------------|
| <i>s</i>  | seconds      |
| <i>ns</i> | nano seconds |

Definition at line 290 of file [rdtsc.h](#).

**16.13.2.13 l4\_tsc\_to\_us()**

```
l4_uint64_t l4_tsc_to_us (
 l4_cpu_time_t tsc) [inline]
```

Convert time stamp into micro seconds value.

## Parameters

|            |                         |
|------------|-------------------------|
| <i>tsc</i> | time value in CPU ticks |
|------------|-------------------------|

## Returns

time value in micro seconds

Definition at line 276 of file [rdtsc.h](#).

**16.14 rdtsc.h**

```
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 */
00038
00039 #define L4_TSC_INIT_AUTO 0
00040 #define L4_TSC_INIT_KERNEL 1
00041 #define L4_TSC_INIT_CALIBRATE 2
00042
00043 extern l4_uint32_t l4_scaler_tsc_to_ns;
00044 extern l4_uint32_t l4_scaler_tsc_to_us;
00045 extern l4_uint32_t l4_scaler_ns_to_tsc;
00046 extern l4_uint32_t l4_scaler_tsc_linux;
00047
00052 L4_INLINE l4_cpu_time_t
00053 l4_rdtsc (void);
00054
00060 L4_INLINE
```

```

00061 l4_uint32_t l4_rdtsc_32(void);
00062
00069 L4_INLINE l4_uint64_t
00070 l4_rdpmc (int ecx);
00071
00077 L4_INLINE
00078 l4_uint32_t l4_rdpmc_32(int ecx);
00079
00084 L4_INLINE l4_uint64_t
00085 l4_tsc_to_ns (l4_cpu_time_t tsc);
00086
00091 L4_INLINE l4_uint64_t
00092 l4_tsc_to_us (l4_cpu_time_t tsc);
00093
00099 L4_INLINE void
00100 l4_tsc_to_s_and_ns (l4_cpu_time_t tsc, l4_uint32_t *s, l4_uint32_t *ns);
00101
00107 L4_INLINE l4_cpu_time_t
00108 l4_ns_to_tsc (l4_uint64_t ns);
00109
00115 L4_INLINE void
00116 l4_busy_wait_ns (l4_uint64_t ns);
00117
00123 L4_INLINE void
00124 l4_busy_wait_us (l4_uint64_t us);
00125
00126 EXTERN_C_BEGIN
00127
00136 L4_INLINE l4_uint32_t
00137 l4_calibrate_tsc (l4_kernel_info_t *kip);
00138
00164 L4_CV l4_uint32_t
00165 l4_tsc_init (int constraint, l4_kernel_info_t *kip);
00166
00171 L4_CV l4_uint32_t
00172 l4_get_hz (void);
00173
00176 EXTERN_C_END
00177
00178 /* implementation */
00179
00180 L4_INLINE l4_uint32_t
00181 l4_calibrate_tsc (l4_kernel_info_t *kip)
00182 {
00183 return l4_tsc_init(L4_TSC_INIT_AUTO, kip);
00184 }
00185
00186 L4_INLINE l4_cpu_time_t
00187 l4_rdtsc (void)
00188 {
00189 l4_cpu_time_t v;
00190
00191 __asm__ __volatile__ (
00192 (".byte 0x0f, 0x31\n\t"
00193 "mov $0xffffffff, %%rcx\n\t" /* clears the upper 32 bits! */
00194 "and %%rcx, %%rax\n\t"
00195 "shlq $32, %%rdx\n\t"
00196 "orq %%rdx, %%rax\n\t"
00197 :
00198 "=a" (v)
00199 : /* no inputs */
00200 : "rdx", "rcx"
00201);
00202
00203 return v;
00204 }
00205
00206 L4_INLINE l4_uint64_t
00207 l4_rdpmc (int ecx)
00208 {
00209 l4_cpu_time_t v;
00210 l4_uint64_t dummy;
00211
00212 __asm__ __volatile__ (
00213 "rdpmc\n\t"
00214 "mov $0xffffffff, %%rcx\n\t" /* clears the upper 32 bits! */
00215 "and %%rcx, %%rax\n\t"
00216 "shlq $32, %%rdx\n\t"
00217 "orq %%rdx, %%rax\n\t"
00218 :
00219 "=a" (v), "=c" (dummy)
00220 : "c" (ecx)
00221 : "rdx"
00222);
00223
00224 return v;
00225 }

```

```

00226
00227 /* the same, but only 32 bit. Useful for smaller differences */
00228 L4_INLINE
00229 l4_uint32_t l4_rdpmc_32(int ecx)
00230 {
00231 l4_uint32_t x;
00232 l4_uint64_t dummy;
00233
00234 __asm__ __volatile__ (
00235 "rdpmc\n\t"
00236 "mov $0xffffffff, %%rcx\n\t" /* clears the upper 32 bits! */
00237 "and %%rcx, %%rax\n\t"
00238 : "=a" (x), "=c" (dummy)
00239 : "c" (ecx)
00240 : "rdx");
00241
00242 return x;
00243 }
00244
00245 /* the same, but only 32 bit. Useful for smaller differences,
00246 needs less cycles. */
00247 L4_INLINE
00248 l4_uint32_t l4_rdtsc_32(void)
00249 {
00250 l4_uint32_t x;
00251
00252 __asm__ __volatile__ (
00253 ".byte 0x0f, 0x31\n\t" // rdtsc
00254 : "=a" (x)
00255 :
00256 : "rdx");
00257
00258 return x;
00259 }
00260
00261 L4_INLINE l4_uint64_t
00262 l4_tsc_to_ns (l4_cpu_time_t tsc)
00263 {
00264 l4_uint64_t ns, dummy;
00265
00266 __asm__
00267 ("
00268 \n\t"
00269 "mulq %3\n\t"
00270 "shrd $27, %%rdx, %%rax\n\t"
00271 : "=a" (ns), "=d" (dummy)
00272 : "a" (tsc), "r" ((l4_uint64_t)l4_scaler_tsc_to_ns)
00273);
00274
00275 return ns;
00276 }
00277
00278 L4_INLINE l4_uint64_t
00279 l4_tsc_to_us (l4_cpu_time_t tsc)
00280 {
00281 l4_uint64_t ns, dummy;
00282
00283 __asm__
00284 ("
00285 \n\t"
00286 "mulq %3\n\t"
00287 "shrd $32, %%rdx, %%rax\n\t"
00288 : "=a" (ns), "=d" (dummy)
00289 : "a" (tsc), "r" ((l4_uint64_t)l4_scaler_tsc_to_us)
00290);
00291
00292 return ns;
00293 }
00294
00295 L4_INLINE void
00296 l4_tsc_to_s_and_ns (l4_cpu_time_t tsc, l4_uint32_t *s, l4_uint32_t *ns)
00297 {
00298 __asm__
00299 ("
00300 \n\t"
00301 "mulq %3\n\t"
00302 "shrd $27, %%rdx, %%rax\n\t"
00303 "xorq %%rdx, %%rdx\n\t"
00304 "divq %4\n\t"
00305 : "=a" (*s), "=d" (*ns)
00306 : "a" (tsc), "r" ((l4_uint64_t)l4_scaler_tsc_to_ns),
00307 "rm" (1000000000ULL)
00308);
00309 }
00310
00311 L4_INLINE l4_cpu_time_t
00312 l4_ns_to_tsc (l4_uint64_t ns)
00313 {
00314 l4_uint64_t tsc, dummy;
00315
00316 __asm__
00317 ("
00318 \n\t"
00319 "mulq %3\n\t"
00320 "shrd $27, %%rdx, %%rax\n\t"
00321 : "=a" (tsc), "=d" (dummy)

```

```

00313 : "a" (ns), "r" ((l4_uint64_t)l4_scaler_ns_to_tsc)
00314);
00315 return tsc;
00316 }
00317
00318 L4_INLINE void
00319 l4_busy_wait_ns (l4_uint64_t ns)
00320 {
00321 l4_cpu_time_t stop = l4_rdtsc();
00322 stop += l4_ns_to_tsc(ns);
00323
00324 while (l4_rdtsc() < stop)
00325 ;
00326 }
00327
00328 L4_INLINE void
00329 l4_busy_wait_us (l4_uint64_t us)
00330 {
00331 l4_cpu_time_t stop = l4_rdtsc ();
00332 stop += l4_ns_to_tsc(us*1000ULL);
00333
00334 while (l4_rdtsc() < stop)
00335 ;
00336 }
00337
00338 EXTERN_C_END
00339
00340 #endif /* __l4_rdtsc_h */
00341

```

## 16.15 x86/l4/util/rdtsc.h File Reference

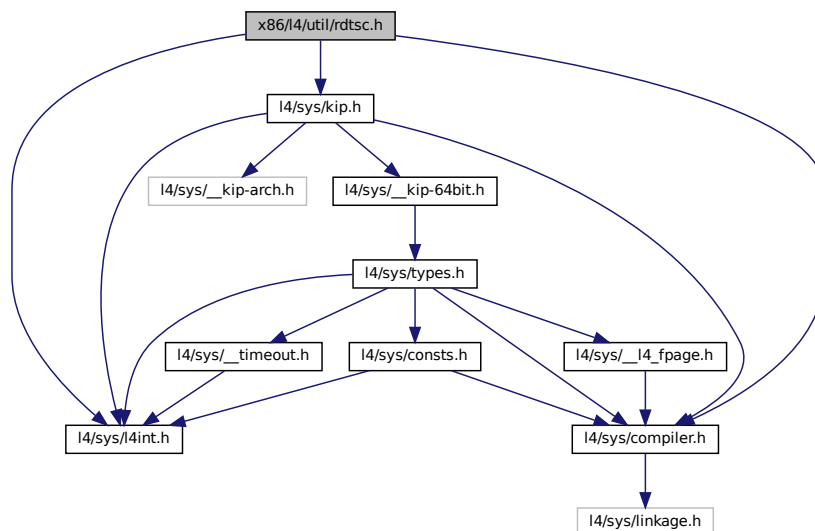
time stamp counter related functions

```

#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
#include <l4/sys/kip.h>

```

Include dependency graph for rdtsc.h:



## Macros

- `#define L4_TSC_INIT_AUTO 0`

- *Automatic init.*
- `#define L4_TSC_INIT_KERNEL 1`
- *Initialized by kernel.*
- `#define L4_TSC_INIT_CALIBRATE 2`
- *Initialized by user-level.*

## Functions

- `l4_cpu_time_t l4_rdtsc` (void)  
*Read current value of CPU-internal time stamp 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 time stamp to ns value.*
- `l4_uint64_t l4_tsc_to_us` (`l4_cpu_time_t` tsc)  
*Convert time stamp 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 time stamp 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` \*kip)  
*Calibrate scalers for time stamp calculations.*
- `l4_uint32_t l4_tsc_init` (int constraint, `l4_kernel_info_t` \*kip)  
*Initialize scaler for TSC calibrations.*
- `l4_uint32_t l4_get_hz` (void)  
*Get CPU frequency in Hz.*

### 16.15.1 Detailed Description

time stamp counter related functions

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file `rdtsc.h`.

### 16.15.2 Function Documentation

#### 16.15.2.1 `l4_busy_wait_ns()`

```
void l4_busy_wait_ns (
 l4_uint64_t ns) [inline]
```

Wait busy for a small amount of time.



## Parameters

|           |                      |
|-----------|----------------------|
| <i>ns</i> | nano seconds to wait |
|-----------|----------------------|

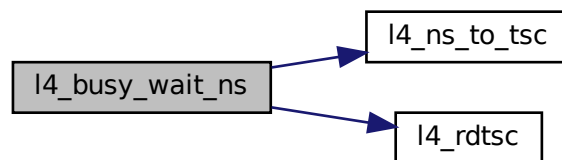
## Attention

Not intended for any use!

Definition at line 345 of file [rdtsc.h](#).

References [l4\\_ns\\_to\\_tsc\(\)](#), and [l4\\_rdtsc\(\)](#).

Here is the call graph for this function:



### 16.15.2.2 l4\_busy\_wait\_us()

```
void l4_busy_wait_us (
 l4_uint64_t us) [inline]
```

Wait busy for a small amount of time.

## Parameters

|           |                       |
|-----------|-----------------------|
| <i>us</i> | micro seconds to wait |
|-----------|-----------------------|

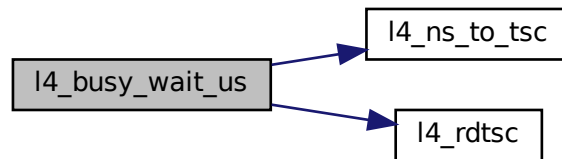
## Attention

Not intended for any use!

Definition at line 355 of file [rdtsc.h](#).

References [l4\\_ns\\_to\\_tsc\(\)](#), and [l4\\_rdtsc\(\)](#).

Here is the call graph for this function:



### 16.15.2.3 l4\_calibrate\_tsc()

```
l4_uint32_t l4_calibrate_tsc (
 l4_kernel_info_t * kip) [inline]
```

Calibrate scalers for time stamp calculations.

Determine some scalers to be able to convert between real time and CPU ticks. This test uses channel 0 of the PIT (i8254) or the kernel KIP, depending on availability. Just calls `l4_tsc_init(L4_TSC_INIT_AUTO)`.

Definition at line 183 of file `rdtsc.h`.

References [l4\\_tsc\\_init\(\)](#), and [L4\\_TSC\\_INIT\\_AUTO](#).

Here is the call graph for this function:



### 16.15.2.4 l4\_get\_hz()

```
l4_uint32_t l4_get_hz (
 void)
```

Get CPU frequency in Hz.

Returns

frequency in Hz

### 16.15.2.5 l4\_ns\_to\_tsc()

```
l4_cpu_time_t l4_ns_to_tsc (
 l4_uint64_t ns) [inline]
```

Convert nano seconds into CPU ticks.

#### Parameters

|           |              |
|-----------|--------------|
| <i>ns</i> | nano seconds |
|-----------|--------------|

#### Returns

CPU ticks

Definition at line 321 of file [rdtsc.h](#).

### 16.15.2.6 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 222 of file [rdtsc.h](#).

### 16.15.2.7 l4\_rdpmc\_32()

```
l4_uint32_t l4_rdpmc_32 (
 int ecx) [inline]
```

Return the least significant 32 bit of a performance counter.

Useful for smaller differences, needs less cycles.

Definition at line 239 of file [rdtsc.h](#).

### 16.15.2.8 l4\_rdtsc()

```
l4_cpu_time_t l4_rdtsc (
 void) [inline]
```

Read current value of CPU-internal time stamp counter.

#### Returns

64-bit time stamp

Definition at line 189 of file [rdtsc.h](#).

### 16.15.2.9 l4\_rdtsc\_32()

```
l4_uint32_t l4_rdtsc_32 (
 void) [inline]
```

Read the lest significant 32 bit of the TSC.

Useful for smaller differences, needs less cycles.

Definition at line 208 of file [rdtsc.h](#).

### 16.15.2.10 l4\_tsc\_init()

```
l4_uint32_t l4_tsc_init (
 int constraint,
 l4_kernel_info_t * kip)
```

Initialize scaler for TSC calibrations.

Initialize the scalers needed by [l4\\_tsc\\_to\\_ns\(\)](#)/[l4\\_ns\\_to\\_tsc\(\)](#) and so on. Current versions of Fiasco export these scalers from kernel into userland. The programmer may decide whether he allows to use these scalers or if an calibration should be performed.

#### Parameters

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>constraint</i> | <p>programmers constraint:</p> <ul style="list-style-type: none"> <li>• <a href="#">L4_TSC_INIT_AUTO</a> if the kernel exports the scalers then use them. If not, perform calibration using channel 0 of the PIT (i8254). The latter case may lead into short (unpredictable) periods where interrupts are disabled.</li> <li>• <a href="#">L4_TSC_INIT_KERNEL</a> depend on retrieving the scalers from kernel. If the scalers are not available, return 0.</li> <li>• <a href="#">L4_TSC_INIT_CALIBRATE</a> Ignore possible scalers exported by the scaler, instead insist on calibration using the PIT.</li> </ul> |
| <i>kip</i>        | KIP pointer                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

### Returns

0 on error (no scalars exported by kernel, calibrating failed ...) 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.

#### 16.15.2.11 l4\_tsc\_to\_ns()

```
l4_uint64_t l4_tsc_to_ns (
 l4_cpu_time_t tsc) [inline]
```

Convert time stamp to ns value.

### Parameters

|            |                         |
|------------|-------------------------|
| <i>tsc</i> | time value in CPU ticks |
|------------|-------------------------|

### Returns

time value in ns

Definition at line 253 of file [rdtsc.h](#).

#### 16.15.2.12 l4\_tsc\_to\_s\_and\_ns()

```
void l4_tsc_to_s_and_ns (
 l4_cpu_time_t tsc,
 l4_uint32_t * s,
 l4_uint32_t * ns) [inline]
```

Convert time stamp to s.ns value.

### Parameters

|            |                         |
|------------|-------------------------|
| <i>tsc</i> | time value in CPU ticks |
|------------|-------------------------|

### Return values

|           |              |
|-----------|--------------|
| <i>s</i>  | seconds      |
| <i>ns</i> | nano seconds |

Definition at line 299 of file [rdtsc.h](#).

### 16.15.2.13 l4\_tsc\_to\_us()

```
l4_uint64_t l4_tsc_to_us (
 l4_cpu_time_t tsc) [inline]
```

Convert time stamp into micro seconds value.

#### Parameters

|            |                         |
|------------|-------------------------|
| <i>tsc</i> | time value in CPU ticks |
|------------|-------------------------|

#### Returns

time value in micro seconds

Definition at line 277 of file [rdtsc.h](#).

## 16.16 rdtsc.h

```
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 */
00040
00041 #define L4_TSC_INIT_AUTO 0
00042 #define L4_TSC_INIT_KERNEL 1
00043 #define L4_TSC_INIT_CALIBRATE 2
00044
00045 extern l4_uint32_t l4_scaler_tsc_to_ns;
00046 extern l4_uint32_t l4_scaler_tsc_to_us;
00047 extern l4_uint32_t l4_scaler_ns_to_tsc;
00048 extern l4_uint32_t l4_scaler_tsc_linux;
00049
00054 L4_INLINE l4_cpu_time_t
00055 l4_rdtsc (void);
00056
00062 L4_INLINE
00063 l4_uint32_t l4_rdtsc_32(void);
00064
00071 L4_INLINE l4_uint64_t
00072 l4_rdpmc (int ecx);
00073
00079 L4_INLINE
00080 l4_uint32_t l4_rdpmc_32(int ecx);
00081
00086 L4_INLINE l4_uint64_t
00087 l4_tsc_to_ns (l4_cpu_time_t tsc);
00088
00093 L4_INLINE l4_uint64_t
00094 l4_tsc_to_us (l4_cpu_time_t tsc);
00095
00101 L4_INLINE void
00102 l4_tsc_to_s_and_ns (l4_cpu_time_t tsc, l4_uint32_t *s, l4_uint32_t *ns);
```

```

00103
00109 L4_INLINE l4_cpu_time_t
00110 l4_ns_to_tsc (l4_uint64_t ns);
00111
00117 L4_INLINE void
00118 l4_busy_wait_ns (l4_uint64_t ns);
00119
00125 L4_INLINE void
00126 l4_busy_wait_us (l4_uint64_t us);
00127
00128 EXTERN_C_BEGIN
00129
00138 L4_INLINE l4_uint32_t
00139 l4_calibrate_tsc (l4_kernel_info_t *kip);
00140
00166 L4_CV l4_uint32_t
00167 l4_tsc_init (int constraint, l4_kernel_info_t *kip);
00168
00173 L4_CV l4_uint32_t
00174 l4_get_hz (void);
00175
00178 EXTERN_C_END
00179
00180 /* implementaion */
00181
00182 L4_INLINE l4_uint32_t
00183 l4_calibrate_tsc (l4_kernel_info_t *kip)
00184 {
00185 return l4_tsc_init(L4_TSC_INIT_AUTO, kip);
00186 }
00187
00188 L4_INLINE l4_cpu_time_t
00189 l4_rdtsc (void)
00190 {
00191 l4_cpu_time_t v;
00192
00193 __asm__ __volatile__ (
00194 (" \n\t"
00195 ".byte 0x0f, 0x31 \n\t"
00196 /*"rdtsc\n\t"*/
00197 :
00198 "=A" (v)
00199 : /* no inputs */
00200);
00201
00202 return v;
00203 }
00204
00205 /* the same, but only 32 bit. Useful for smaller differences,
00206 needs less cycles. */
00207 L4_INLINE
00208 l4_uint32_t l4_rdtsc_32(void)
00209 {
00210 l4_uint32_t x;
00211
00212 __asm__ __volatile__ (
00213 ".byte 0x0f, 0x31\n\t" // rdtsc
00214 : "=a" (x)
00215 :
00216 : "edx");
00217
00218 return x;
00219 }
00220
00221 L4_INLINE l4_uint64_t
00222 l4_rdpmc (int ecx)
00223 {
00224 l4_cpu_time_t v;
00225
00226 __asm__ __volatile__ (
00227 "rdpmc \n\t"
00228 :
00229 "=A" (v)
00230 : "c" (ecx)
00231);
00232
00233 return v;
00234 }
00235
00236 /* the same, but only 32 bit. Useful for smaller differences,
00237 needs less cycles. */
00238 L4_INLINE
00239 l4_uint32_t l4_rdpmc_32(int ecx)
00240 {
00241 l4_uint32_t x;
00242
00243 __asm__ __volatile__ (

```

```

00244 "rdpmc \n\t"
00245 : "=a" (x)
00246 : "c" (ecx)
00247 : "edx");
00248
00249 return x;
00250 }
00251
00252 L4_INLINE l4_uint64_t
00253 l4_tsc_to_ns (l4_cpu_time_t tsc)
00254 {
00255 l4_uint32_t dummy;
00256 l4_uint64_t ns;
00257 __asm__
00258 (
00259 "movl %%edx, %%ecx \n\t"
00260 "mull %3 \n\t"
00261 "movl %%ecx, %%eax \n\t"
00262 "movl %%edx, %%ecx \n\t"
00263 "mull %3 \n\t"
00264 "addl %%ecx, %%eax \n\t"
00265 "adcl $0, %%edx \n\t"
00266 "shld $5, %%eax, %%edx \n\t"
00267 "shll $5, %%eax \n\t"
00268 : "=A" (ns),
00269 "=&c" (dummy)
00270 : "0" (tsc),
00271 "g" (l4_scaler_tsc_to_ns)
00272);
00273 return ns;
00274 }
00275
00276 L4_INLINE l4_uint64_t
00277 l4_tsc_to_us (l4_cpu_time_t tsc)
00278 {
00279 l4_uint32_t dummy;
00280 l4_uint64_t us;
00281 __asm__
00282 (
00283 "movl %%edx, %%ecx \n\t"
00284 "mull %3 \n\t"
00285 "movl %%ecx, %%eax \n\t"
00286 "movl %%edx, %%ecx \n\t"
00287 "mull %3 \n\t"
00288 "addl %%ecx, %%eax \n\t"
00289 "adcl $0, %%edx \n\t"
00290 : "=A" (us),
00291 "=&c" (dummy)
00292 : "0" (tsc),
00293 "g" (l4_scaler_tsc_to_us)
00294);
00295 return us;
00296 }
00297
00298 L4_INLINE void
00299 l4_tsc_to_s_and_ns (l4_cpu_time_t tsc, l4_uint32_t *s, l4_uint32_t *ns)
00300 {
00301 l4_uint32_t dummy;
00302 __asm__
00303 (
00304 "movl %%edx, %%ecx \n\t"
00305 "mull %4 \n\t"
00306 "movl %%ecx, %%eax \n\t"
00307 "movl %%edx, %%ecx \n\t"
00308 "mull %4 \n\t"
00309 "addl %%ecx, %%eax \n\t"
00310 "adcl $0, %%edx \n\t"
00311 "movl $1000000000, %%ecx \n\t"
00312 "shld $5, %%eax, %%edx \n\t"
00313 "shll $5, %%eax \n\t"
00314 "divl %%ecx \n\t"
00315 : "=a" (*s), "=d" (*ns), "=&c" (dummy)
00316 : "A" (tsc), "g" (l4_scaler_tsc_to_ns)
00317);
00318 }
00319
00320 L4_INLINE l4_cpu_time_t
00321 l4_ns_to_tsc (l4_uint64_t ns)
00322 {
00323 l4_uint32_t dummy;
00324 l4_cpu_time_t tsc;
00325 __asm__
00326 (
00327 "movl %%edx, %%ecx \n\t"
00328 "mull %3 \n\t"
00329 "movl %%ecx, %%eax \n\t"
00330 "movl %%edx, %%ecx \n\t"

```



```

00331 "mull %3 \n\t"
00332 "addl %%ecx, %%eax \n\t"
00333 "adcl $0, %%edx \n\t"
00334 "shld $5, %%eax, %%edx \n\t"
00335 "shll $5, %%eax \n\t"
00336 : "=A" (tsc),
00337 "=&c" (dummy)
00338 : "0" (ns),
00339 "g" (l4_scaler_ns_to_tsc)
00340);
00341 return tsc;
00342 }
00343
00344 L4_INLINE void
00345 l4_busy_wait_ns (l4_uint64_t ns)
00346 {
00347 l4_cpu_time_t stop = l4_rdtsc();
00348 stop += l4_ns_to_tsc(ns);
00349
00350 while (l4_rdtsc() < stop)
00351 ;
00352 }
00353
00354 L4_INLINE void
00355 l4_busy_wait_us (l4_uint64_t us)
00356 {
00357 l4_cpu_time_t stop = l4_rdtsc ();
00358 stop += l4_ns_to_tsc(us*1000ULL);
00359
00360 while (l4_rdtsc() < stop)
00361 ;
00362 }
00363
00364 EXTERN_C_END
00365
00366 #endif /* __l4_rdtsc_h */
00367

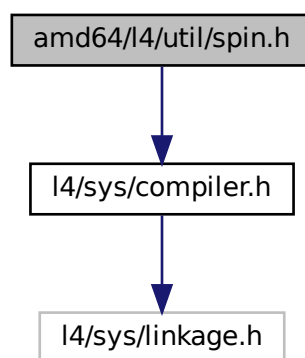
```

## 16.17 amd64/l4/util/spin.h File Reference

Spinning for amd64.

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

Include dependency graph for spin.h:



### 16.17.1 Detailed Description

Spinning for amd64.

Definition in file [spin.h](#).

## 16.18 spin.h

```

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

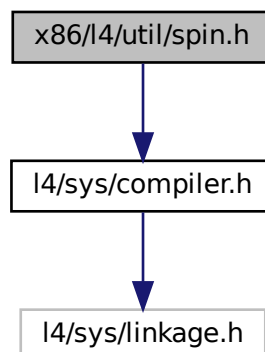
```

## 16.19 x86/l4/util/spin.h File Reference

Spinning for x86.

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

Include dependency graph for spin.h:



## 16.19.1 Detailed Description

Spinning for x86.

Definition in file [spin.h](#).

## 16.20 spin.h

```

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

```

## 16.21 amd64/l4/util/util.h File Reference

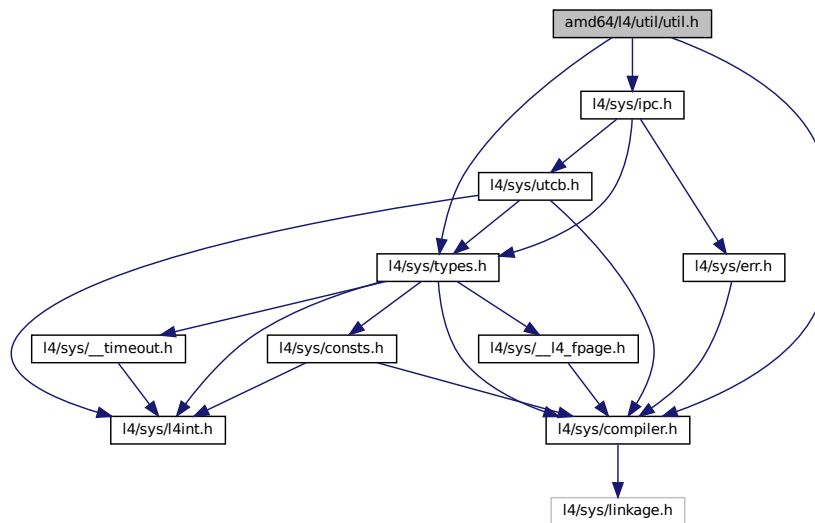
Utilities, amd64 version.

```

#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.

### 16.21.1 Detailed Description

Utilities, amd64 version.

Definition in file [util.h](#).

### 16.21.2 Function Documentation

#### 16.21.2.1 l4\_sleep()

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

Suspend thread for a period of *ms* milliseconds.

## Parameters

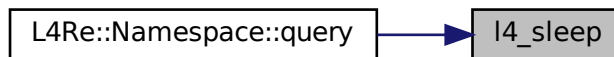
|           |                      |
|-----------|----------------------|
| <i>ms</i> | Time in milliseconds |
|-----------|----------------------|

## 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:



## 16.21.2.2 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"> <li>• 0 -&gt; timeout 0</li> <li>• ~0U -&gt; timeout NEVER</li> </ul> |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

the corresponding l4\_timeout value

## 16.22 util.h

```
00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00008 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
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 #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
00029 L4_CV l4_timeout_s l4util_micros2l4to(unsigned int mus) L4_NOTHROW;
00030
00034 L4_CV void l4_sleep(int ms) L4_NOTHROW;
00035
00036 /* Suspend thread for a period of \a us microseconds.
00037 * \param us Time in microseconds
00038 * WARNING: This function is mostly bogus since the timer resolution of
00039 * current L4 implementations is about lms! */
00040 L4_CV void l4_usleep(int us) L4_NOTHROW;
00041
00047 L4_INLINE void l4_sleep_forever(void) L4_NOTHROW __attribute__((noreturn));
00048
00049 L4_INLINE void
00050 l4_sleep_forever(void) L4_NOTHROW
00051 {
00052 for (;;)
00053 l4_ipc_sleep(L4_IPC_NEVER);
00054 }
00055
00057 static inline void
00058 l4_touch_ro(const void*addr, unsigned size) L4_NOTHROW
00059 {
00060 const char *bptr, *eptr;
00061
00062 bptr = (const char*)((l4_addr_t)addr) & L4_PAGEMASK;
00063 eptr = (const char*)((l4_addr_t)addr+size-1) & L4_PAGEMASK;
00064 for(;bptr<=eptr;bptr+=L4_PAGESIZE){
00065 asm volatile("or %0,%rax \n"
00066 :
00067 : "m" (*(const unsigned*)bptr)
00068 : "rax");
00069 }
00070 }
00071
00072
00074 static inline void
00075 l4_touch_rw(const void*addr, unsigned size) L4_NOTHROW
00076 {
00077 const char *bptr, *eptr;
00078
00079 bptr = (const char*)((l4_addr_t)addr) & L4_PAGEMASK;
00080 eptr = (const char*)((l4_addr_t)addr+size-1) & L4_PAGEMASK;
00081 for(;bptr<=eptr;bptr+=L4_PAGESIZE){
00082 asm volatile("orb $0,%0 \n"
00083 :
00084 : "m" (*(const unsigned*)bptr)
00085);
00086 }
00087 }
00088
00089 EXTERN_C_END
00090
00091 #endif
00092

```

## 16.23 x86/l4/util/util.h File Reference

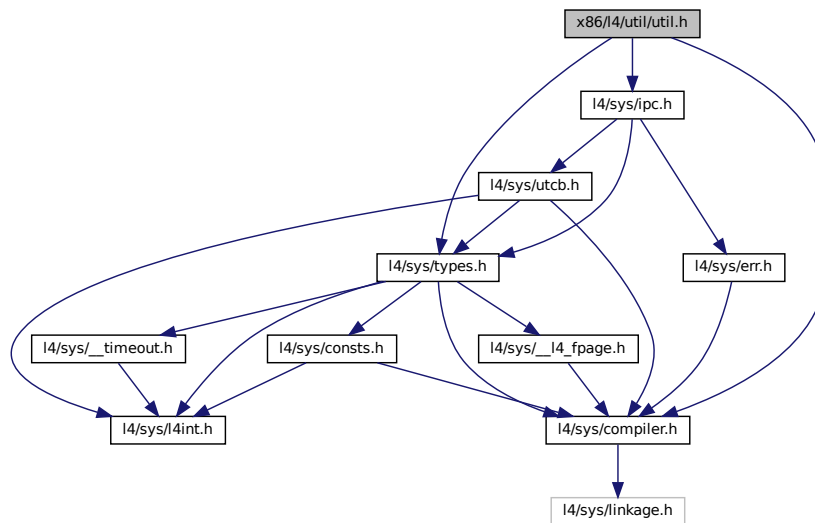
Utilities for x86.

```

#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.

### 16.23.1 Detailed Description

Utilities for x86.

Definition in file [util.h](#).

### 16.23.2 Function Documentation

#### 16.23.2.1 l4\_sleep()

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

Suspend thread for a period of *ms* milliseconds.

## Parameters

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

**16.23.2.2 l4util\_micros2l4to()**

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

Calculate l4 timeouts.

## Parameters

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

## Returns

the corresponding l4\_timeout value

**16.24 util.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 * Jork Löser <jork@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 __UTIL_H
00013 #define __UTIL_H
00014
00015 #include <l4/sys/types.h>
00016 #include <l4/sys/compiler.h>
00017 #include <l4/sys/ipc.h>
00018
00019 EXTERN_C_BEGIN
00020
00021 L4_INLINE l4_timeout_s l4util_micros2l4to(unsigned int mus) L4_NOTHROW;
00022
00023 L4_INLINE void l4_sleep(int ms) L4_NOTHROW;
00024
00025 /* Suspend thread for a period of \a us microseconds.
00026 * \param us Time in microseconds
00027 * WARNING: This function is mostly bogus since the timer resolution of
00028 * current L4 implementations is about 1ms! */
00029 L4_INLINE void l4_usleep(int us) L4_NOTHROW;
00030
00031 L4_INLINE void l4_sleep_forever(void) L4_NOTHROW __attribute__((noreturn));
00032
00033 L4_INLINE void
00034 l4_sleep_forever(void) L4_NOTHROW
00035 {
00036 for (;;)
00037
```



```

00054 l4_ipc_sleep(L4_IPC_NEVER);
00055 }
00056
00057 static inline void
00058 l4_touch_ro(const void*addr, unsigned size) L4_NOTHROW
00059 {
00060 const char *bptr, *eptr;
00061 bptr = (const char*)((l4_addr_t)addr & L4_PAGEMASK);
00062 eptr = (const char*)((l4_addr_t)addr+size-1 & L4_PAGEMASK);
00063 for(;bptr<=eptr;bptr+=L4_PAGESIZE){
00064 asm volatile("or %0,%eax \n"
00065 : "m" (*(const unsigned*)bptr)
00066 : "eax");
00067 }
00068 }
00069
00070 static inline void
00071 l4_touch_rw(const void*addr, unsigned size) L4_NOTHROW
00072 {
00073 const char *bptr, *eptr;
00074 bptr = (const char*)((l4_addr_t)addr & L4_PAGEMASK);
00075 eptr = (const char*)((l4_addr_t)addr+size-1 & L4_PAGEMASK);
00076 for(;bptr<=eptr;bptr+=L4_PAGESIZE){
00077 asm volatile("orb $0,%0 \n"
00078 : "m" (*(const unsigned*)bptr)
00079 :);
00080 }
00081 }
00082
00083 EXTERN_C_END
00084 #endif
00085
00086

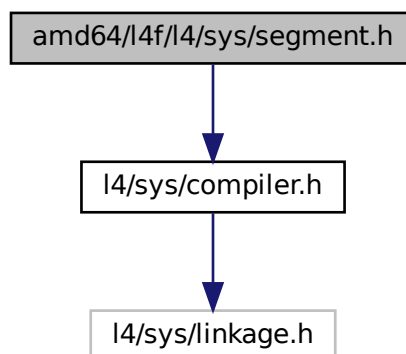
```

## 16.25 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.*

### 16.25.1 Detailed Description

l4f specific fs/gs manipulation

Definition in file [segment.h](#).

### 16.25.2 Function Documentation

#### 16.25.2.1 `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 of the caller.                                     |

#### Return values

|                         |                                                            |
|-------------------------|------------------------------------------------------------|
| <code>L4_EOK</code>     | Success.                                                   |
| <code>-L4_EINVAL</code> | Invalid base address (base).                               |
| <code>-L4_ENOSYS</code> | 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](#).

## 16.25.2.2 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 <a href="#">L4_sys_segment</a> ).                  |
| <i>base</i>   | Base address.                                                                |
| <i>utcb</i>   | UTCB of the caller.                                                          |

## 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](#).

## 16.26 segment.h

```
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 }
00041
```

```

00042 L4_INLINE long
00043 fiasco_amd64_set_segment_base(l4_cap_idx_t thread, enum L4_sys_segment segr,
00044 l4_umword_t base, l4_utcb_t *utcb)
00045 {
00046 l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_AMD64_SET_SEGMENT_BASE_OP | ((l4_umword_t)segr << 16);
00047 l4_utcb_mr_u(utcb)->mr[1] = base;
00048 return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0), L4_IPC_NEVER),
00049 utcb);
00049 }
00050
00051 L4_INLINE long
00052 fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
00053 unsigned int entry_number_start, l4_utcb_t *utcb)
00054 {
00055 l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
00056 l4_utcb_mr_u(utcb)->mr[1] = entry_number_start;
00057 __builtin_memcpy(&l4_utcb_mr_u(utcb)->mr[2], desc, size);
00058 return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2 + (size / 8), 0, 0),
00059 L4_IPC_NEVER), utcb);
00059 }
00060
00061 #endif /* ! __L4_SYS_ARCH_X86__L4API_L4F__SEGMENT_H__ */

```

## 16.27 amd64/l4/sys/segment.h File Reference

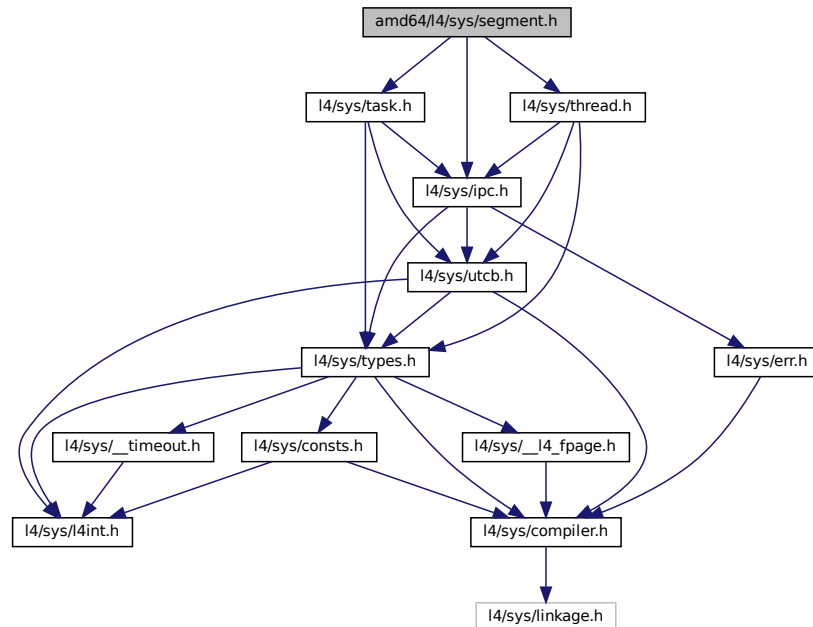
Segment handling.

```

#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` , `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.*

### 16.27.1 Detailed Description

Segment handling.

Definition in file [segment.h](#).

### 16.27.2 Enumeration Type Documentation

#### 16.27.2.1 L4\_sys\_segment

enum [L4\\_sys\\_segment](#)

Constants for identifying segments.

Enumerator

|                                     |                                      |
|-------------------------------------|--------------------------------------|
| <a href="#">L4_AMD64_SEGMENT_FS</a> | Constant identifying the FS segment. |
| <a href="#">L4_AMD64_SEGMENT_GS</a> | Constant identifying the GS segment. |

Definition at line [107](#) of file [segment.h](#).

#### 16.27.2.2 L4\_task\_ldt\_x86\_consts

enum [L4\\_task\\_ldt\\_x86\\_consts](#)

Contants 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. |
| 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 76 of file [segment.h](#).

## 16.27.3 Function Documentation

### 16.27.3.1 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. |
| in  | <i>utcb</i>      | UTCB of the caller.         |

## Returns

System call error

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

### 16.27.3.2 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 of the caller.                                     |

## 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](#).

## 16.27.3.3 fiasco\_amd64\_set\_segment\_base()

```
long fiasco_amd64_set_segment_base (
 l4_cap_idx_t thread,
 enum L4_sys_segment segr,
 l4_umword_t base,
 l4_utcb_t * utcb) [inline]
```

Set the base address for a segment.

## Parameters

|               |                                                                              |
|---------------|------------------------------------------------------------------------------|
| <i>thread</i> | Thread for which the base address of the selected segment shall be modified. |
| <i>segr</i>   | Segment to modify (one of <a href="#">L4_sys_segment</a> ).                  |
| <i>base</i>   | Base address.                                                                |
| <i>utcb</i>   | UTCB of the caller.                                                          |

## 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](#).

## 16.28 segment.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 /*****
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
00043 L4_INLINE long
00044 fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int num_desc,
00045 unsigned int entry_number_start, l4_utcb_t *utcb);
00046
00060 L4_INLINE long
00061 fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
00062 unsigned int entry_number_start, l4_utcb_t *utcb);
00063
00070 L4_INLINE unsigned
00071 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread, l4_utcb_t *utcb);
00072
00076 enum L4_task_ldt_x86_consts
00077 {
00079 L4_TASK_LDT_X86_ENTRY_SIZE = 8,
00081 L4_TASK_LDT_X86_MAX_ENTRIES
00082 = (L4_UTCB_GENERIC_DATA_SIZE - 2)
00083 / (L4_TASK_LDT_X86_ENTRY_SIZE / (L4_MWORD_BITS / 8)),
00084 };
00085
00101 L4_INLINE long
00102 fiasco_amd64_set_fs(l4_cap_idx_t thread, l4_umword_t base, l4_utcb_t *utcb);
00103
00107 enum L4_sys_segment
00108 {
00110 L4_AMD64_SEGMENT_FS = 0,
00112 L4_AMD64_SEGMENT_GS = 1
00113 };
00114
00128 L4_INLINE long
00129 fiasco_amd64_set_segment_base(l4_cap_idx_t thread, enum L4_sys_segment segr,
00130 l4_umword_t base, l4_utcb_t *utcb);
00131
00141 L4_INLINE long
00142 fiasco_amd64_segment_info(l4_cap_idx_t thread, unsigned *user_ds,
00143 unsigned *user_cs, unsigned *user32_cs,
00144 l4_utcb_t *utcb);
00145
00146 /*****
00147 *** Implementation
00148 *****/
00149
00150 #include <l4/sys/task.h>
00151 #include <l4/sys/thread.h>
00152
00153 L4_INLINE long
00154 fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int num_desc,
00155 unsigned int entry_number_start, l4_utcb_t *utcb)
00156 {
00157 if (num_desc > L4_TASK_LDT_X86_MAX_ENTRIES)
00158 return -L4_EINVAL;
00159 l4_utcb_mr_u(utcb)->mr[0] = L4_TASK_LDT_SET_X86_OP;
00160 l4_utcb_mr_u(utcb)->mr[1] = entry_number_start;
00161 __builtin_memcpy(&l4_utcb_mr_u(utcb)->mr[2], ldt,
00162 num_desc * L4_TASK_LDT_X86_ENTRY_SIZE);
00163 return l4_error_u(l4_ipc_call(task, utcb, l4_msgtag(L4_PROTO_TASK, 2 + num_desc * 2, 0, 0),
00164 L4_IPC_NEVER), utcb);
00165 }
00166
00166 L4_INLINE unsigned
00167 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread, l4_utcb_t *utcb)

```



```

00168 {
00169 l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
00170 if (l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER), utcb))
00171 return -1;
00172 return l4_utcb_mr_u(utcb)->mr[0];
00173 }
00174
00175 L4_INLINE long
00176 fiasco_amd64_segment_info(l4_cap_idx_t thread, unsigned *user_ds,
00177 unsigned *user_cs, unsigned *user32_cs,
00178 l4_utcb_t *utcb)
00179 {
00180 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00181 int r;
00182
00183 m->mr[0] = L4_THREAD_AMD64_GET_SEGMENT_INFO_OP;
00184
00185 r = l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0),
00186 L4_IPC_NEVER), utcb);
00187 if (r < 0)
00188 return r;
00189
00190 *user_ds = m->mr[0];
00191 *user_cs = m->mr[1];
00192 *user32_cs = m->mr[2];
00193
00194 return 0;
00195 }
00196
00197 #endif /* ! __L4_SYS__ARCH_X86__SEGMENT_H__ */

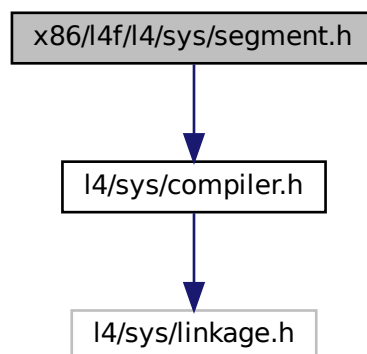
```

## 16.29 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.*

## 16.29.1 Detailed Description

l4f specific segment manipulation

Definition in file [segment.h](#).

## 16.30 segment.h

```

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 /***** Implementation *****/
00031
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__ */

```

## 16.31 x86/l4/sys/segment.h File Reference

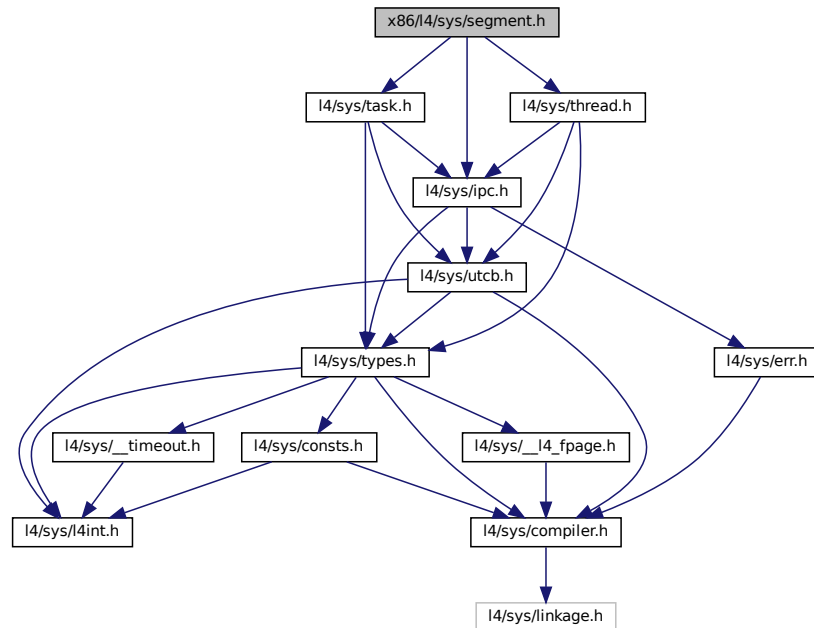
Segment handling.

```

#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](#) , [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.*

### 16.31.1 Detailed Description

Segment handling.

Definition in file [segment.h](#).

## 16.31.2 Enumeration Type Documentation

### 16.31.2.1 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. |
| 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 76 of file [segment.h](#).

## 16.32 segment.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 /*****
00024 * #ifndef __L4_SYS_ARCH_X86_SEGMENT_H__
00025 * #define __L4_SYS_ARCH_X86_SEGMENT_H__
00026 *
00027 * #ifndef L4API_14f
00028 * #error This header file can only be used with a L4API version!
00029 * #endif
00030 *
00031 * #include <l4/sys/ipc.h>
00032 *
00043 * L4_INLINE long
00044 * fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int size,
00045 * unsigned int entry_number_start, l4_utcb_t *utcb);
00046 *
00060 * L4_INLINE long
00061 * fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
00062 * unsigned int entry_number_start, l4_utcb_t *utcb);
00063 *
00070 * L4_INLINE unsigned
00071 * fiasco_gdt_get_entry_offset(l4_cap_idx_t thread, l4_utcb_t *utcb);
00072 *
00076 * enum L4_task_ldt_x86_consts
00077 * {
00079 * L4_TASK_LDT_X86_ENTRY_SIZE = 8,
00081 * L4_TASK_LDT_X86_MAX_ENTRIES
00082 * = (L4_UTCB_GENERIC_DATA_SIZE - 2)

```

```

00083 / (L4_TASK_LDT_X86_ENTRY_SIZE / (L4_MWORD_BITS / 8)),
00084 };
00085
00086 /*****
00087 *** Implementation
00088 *****/
00089
00090 #include <l4/sys/task.h>
00091 #include <l4/sys/thread.h>
00092
00093 L4_INLINE long
00094 fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int num_desc,
00095 unsigned int entry_number_start, l4_utcb_t *utcb)
00096 {
00097 if (num_desc > L4_TASK_LDT_X86_MAX_ENTRIES)
00098 return -L4_EINVAL;
00099 l4_utcb_mr_u(utcb)->mr[0] = L4_TASK_LDT_SET_X86_OP;
00100 l4_utcb_mr_u(utcb)->mr[1] = entry_number_start;
00101 __builtin_memcpy(&l4_utcb_mr_u(utcb)->mr[2], ldt,
00102 num_desc * L4_TASK_LDT_X86_ENTRY_SIZE);
00103 return l4_error_u(l4_ipc_call(task, utcb, l4_msgtag(L4_PROTO_TASK, 2 + num_desc * 2, 0, 0),
00104 L4_IPC_NEVER), utcb);
00105 }
00106
00107 L4_INLINE unsigned
00108 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread, l4_utcb_t *utcb)
00109 {
00110 l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
00111 if (l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER), utcb))
00112 return -1;
00113 return l4_utcb_mr_u(utcb)->mr[0];
00114 }
00115 #endif /* ! __L4_SYS_ARCH_X86_SEGMENT_H__ */

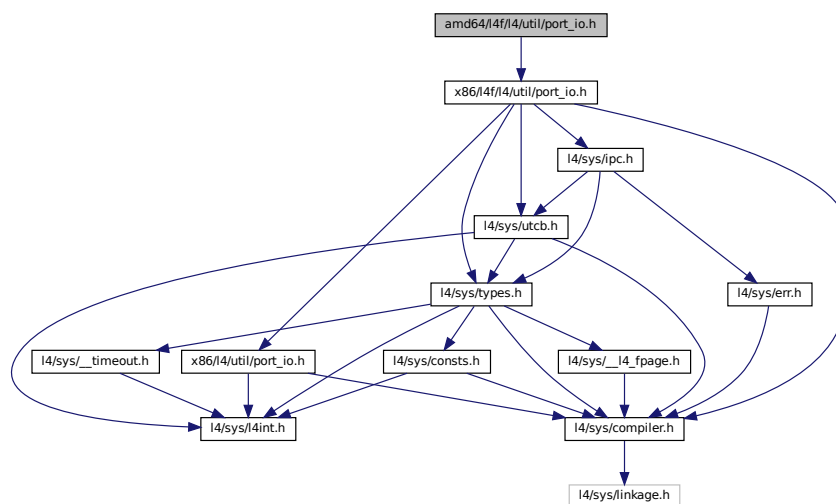
```

## 16.33 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:



### 16.33.1 Detailed Description

Port I/O functions.

Definition in file [port\\_io.h](#).

## 16.34 port\_io.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU Lesser General Public License 2.1.
00010 * Please see the COPYING-LGPL-2.1 file for details.
00011 */
00012 #include <x86/l4/l4/util/port_io.h>

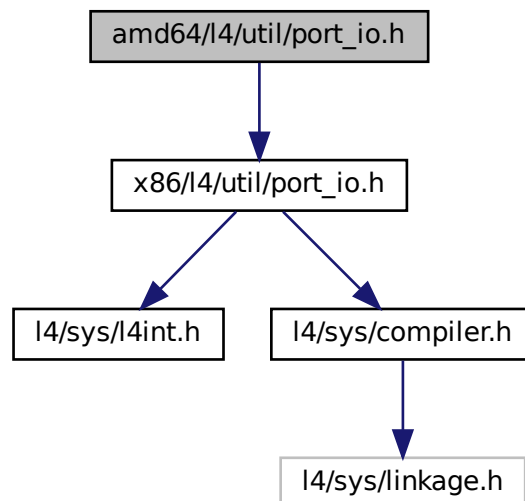
```

## 16.35 amd64/l4/util/port\_io.h File Reference

Port I/O functions.

```
#include <x86/l4/util/port_io.h>
```

Include dependency graph for port\_io.h:



### 16.35.1 Detailed Description

Port I/O functions.

Definition in file [port\\_io.h](#).

## 16.36 port\_io.h

```

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>

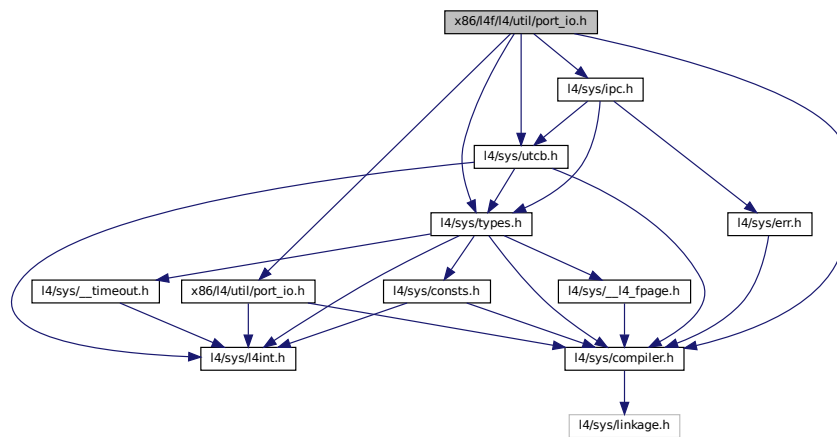
```

## 16.37 x86/l4f/l4/util/port\_io.h File Reference

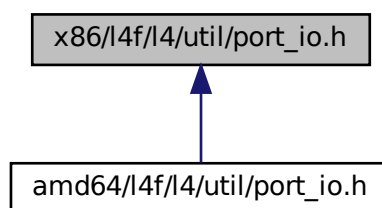
port I/O functions

```
#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <x86/l4/util/port_io.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>
```

Include dependency graph for port\_io.h:



This graph shows which files directly or indirectly include this file:



### Functions

- int [l4util\\_ioport\\_map](#) (l4\_cap\_idx\_t sigma0id, unsigned port\_start, unsigned log2size)  
Map a range of I/O ports.

## 16.37.1 Detailed Description

port I/O functions

Date

06/2003

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [port\\_io.h](#).

## 16.37.2 Function Documentation

### 16.37.2.1 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

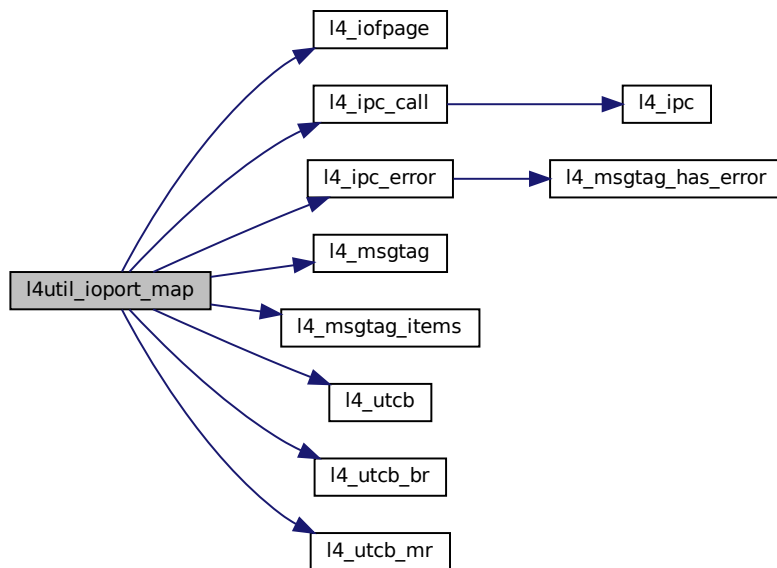
IPC result: 0 if the range could be successfully mapped on error: IPC failure, or -L4\_ENOENT if nothing mapped

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\\_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:



## 16.38 port\_io.h

```

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_iofp_page(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

```

## 16.39 x86/l4/util/port\_io.h File Reference

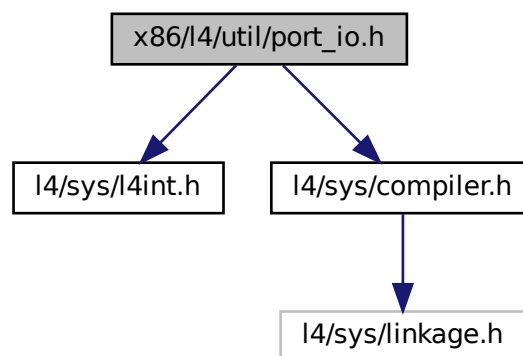
x86 port I/O

```

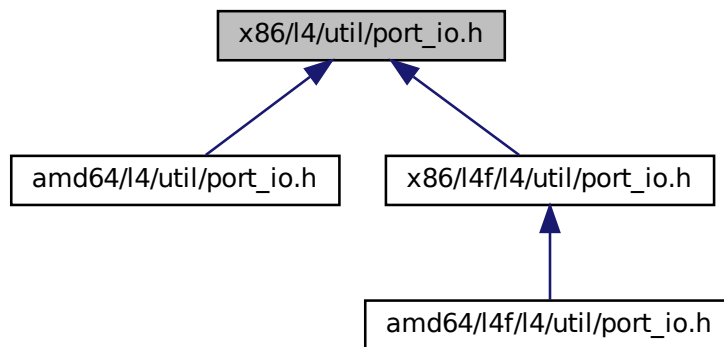
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>

```

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*

### 16.39.1 Detailed Description

x86 port I/O

Date

06/2003

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [port\\_io.h](#).

### 16.39.2 Function Documentation

#### 16.39.2.1 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](#).

#### 16.39.2.2 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](#).

**16.39.2.3 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 |
|-------------|------------------|

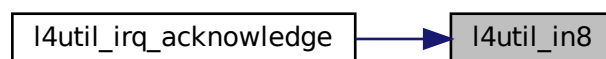
**Returns**

value

Definition at line 172 of file [port\\_io.h](#).

Referenced by [l4util\\_irq\\_acknowledge\(\)](#).

Here is the caller graph for this function:

**16.39.2.4 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](#).

**16.39.2.5 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](#).

**16.39.2.6 l4util\_ins8()**

```
void l4util_ins8 (
 l4_uint16_t port,
 l4_umword_t addr,
 l4_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](#).

### 16.39.2.7 l4util\_out16()

```
void l4util_out16 (
 l4_uint16_t value,
 l4_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](#).

### 16.39.2.8 l4util\_out32()

```
void l4util_out32 (
 l4_uint32_t value,
 l4_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](#).

### 16.39.2.9 l4util\_out8()

```
void l4util_out8 (
 l4_uint8_t value,
 l4_uint16_t port) [inline]
```

Write byte to I/O port.

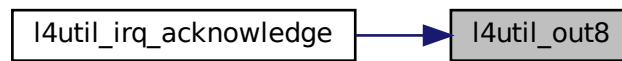
#### Parameters

|              |                  |
|--------------|------------------|
| <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:



#### 16.39.2.10 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](#).

#### 16.39.2.11 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](#).



**16.39.2.12 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](#).

**16.40 port\_io.h**

```
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
00022 /* L4 includes */
00023 #include <l4/sys/l4int.h>
00024 #include <l4/sys/compiler.h>
00025
00026 /*****
00027 *** Prototypes
00028 *****/
00029
00030 EXTERN_C_BEGIN
00031 L4_INLINE l4_uint8_t
00032 l4util_in8(l4_uint16_t port);
00033
00034 L4_INLINE l4_uint16_t
00035 l4util_in16(l4_uint16_t port);
00036
00037 L4_INLINE l4_uint32_t
00038 l4util_in32(l4_uint16_t port);
00039
00040 L4_INLINE void
00041 l4util_ins8(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00042
00043 L4_INLINE void
00044 l4util_ins16(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00045
00046 L4_INLINE void
00047 l4util_ins32(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00048
00049 L4_INLINE void
00050 l4util_out8(l4_uint8_t value, l4_uint16_t port);
00051
00052 L4_INLINE void
00053 l4util_out16(l4_uint16_t value, l4_uint16_t port);
00054
00055 L4_INLINE void
00056 l4util_out32(l4_uint32_t value, l4_uint16_t port);
00057
00058 L4_INLINE void
00059 l4util_outs8(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00060
00061
```

```

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 /***** Implementation *****/
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_idelay(void)
00269 {
00270 asm volatile ("outb %a1,$0x80");
00271 }
00272
00273 #endif

```

## 16.41 arm/l4/sys/linkage.h File Reference

Linkage.

### Macros

- `#define L4_CV`  
Define calling convention.

### 16.41.1 Detailed Description

Linkage.

Definition in file [linkage.h](#).

## 16.42 linkage.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 #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__ */

```

## 16.43 amd64/l4/sys/linkage.h File Reference

Linkage.

### Macros

- `#define L4_CV`  
*Define calling convention.*

### 16.43.1 Detailed Description

Linkage.

Definition in file [linkage.h](#).

## 16.44 linkage.h

```

00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00009 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #ifndef __L4__SYS__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__ */

```

## 16.45 x86/I4/sys/linkage.h File Reference

Linkage.

### Macros

- `#define L4_CV __attribute__((regparm(0)))`  
Define calling convention.

### 16.45.1 Detailed Description

Linkage.

Definition in file [linkage.h](#).

## 16.46 linkage.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 * 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__ */

```

## 16.47 arm/l4/sys/mem\_op.h File Reference

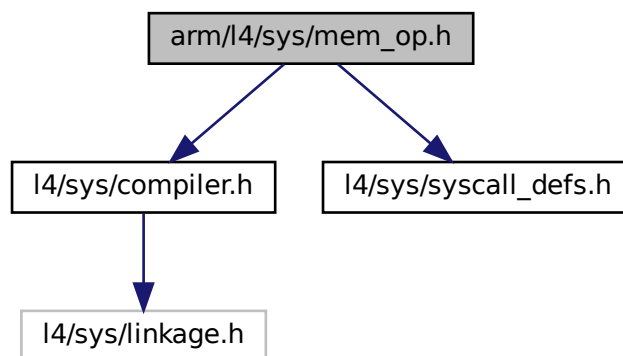
Memory access functions (ARM specific)

```

#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.*

## 16.47.1 Detailed Description

Memory access functions (ARM specific)

### Date

2010-10

### Author

Adam Lackorzynski [adam@os.inf.tu-dresden.de](mailto:adam@os.inf.tu-dresden.de)

Definition in file [mem\\_op.h](#).

## 16.48 mem\_op.h

```

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__ */

```

## 16.49 arm/l4/sys/vm.h File Reference

ARM virtualization interface.

### Data Structures

- struct [l4\\_vm\\_tz\\_state](#)  
*state structure for TrustZone VMs*

### 16.49.1 Detailed Description

ARM virtualization interface.

Definition in file [vm.h](#).



## 16.50 vm.h

```

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 cpl0_fpexc;
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 }
```

## 16.51 arm/l4/util/bitops\_arch.h File Reference

ARM specific implementation of bitops functions.

### 16.51.1 Detailed Description

ARM specific implementation of bitops functions.

Definition in file [bitops\\_arch.h](#).

## 16.52 bitops\_arch.h

```
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__ */
```

## 16.53 amd64/l4/util/bitops\_arch.h File Reference

amd64 bit manipulation functions

### 16.53.1 Detailed Description

amd64 bit manipulation functions

#### Date

07/03/2001

#### Author

Lars Reuther [reuther@os.inf.tu-dresden.de](mailto:reuther@os.inf.tu-dresden.de) Torsten Frenzel [frenzel@os.inf.tu-dresden.de](mailto:frenzel@os.inf.tu-dresden.de)  
Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [bitops\\_arch.h](#).

## 16.54 bitops\_arch.h

```

00001 /*****
00011 */
00012 * (c) 2000-2009 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 /*****
00020 #ifndef __L4UTIL__INCLUDE__ARCH_AMD64__BITOPS_ARCH_H__
00021 #define __L4UTIL__INCLUDE__ARCH_AMD64__BITOPS_ARCH_H__
00022
00023 EXTERN_C_BEGIN
00024
00025 /*****
00026 *** Implementation
00027 *****/
00028
00029 #define __L4UTIL_BITOPS_HAVE_ARCH_SET_BIT
00030 L4_INLINE void
00031 l4util_set_bit(int b, volatile l4_umword_t * dest)
00032 {
00033 __asm__ __volatile__
00034 (
00035 "lock; bts %1,%0 \n\t"
00036 :
00037 :
00038 "m" (*dest), /* 0 mem, destination operand */
00039 "Ir" (b) /* 1, bit number */
00040 :
00041 "memory", "cc"
00042);
00043 }
00044
00045 /* clear bit */
00046 #define __L4UTIL_BITOPS_HAVE_ARCH_CLEAR_BIT
00047 L4_INLINE void
00048 l4util_clear_bit(int b, volatile l4_umword_t * dest)
00049 {
00050 __asm__ __volatile__
00051 (
00052 "lock; btr %1,%0 \n\t"
00053 :
00054 :
00055 "m" (*dest), /* 0 mem, destination operand */
00056 "Ir" (b) /* 1, bit number */
00057 :
00058 "memory", "cc"
00059);
00060 }
00061
00062 /* change bit */
00063 #define __L4UTIL_BITOPS_HAVE_ARCH_COMPLEMENT_BIT
00064 L4_INLINE void
00065 l4util_complement_bit(int b, volatile l4_umword_t * dest)
00066 {
00067 __asm__ __volatile__
00068 (
00069 "lock; btc %1,%0 \n\t"
00070 :
00071 :
00072 "m" (*dest), /* 0 mem, destination operand */
00073 "Ir" (b) /* 1, bit number */
00074 :
00075 "memory", "cc"
00076);
00077 }
00078
00079 /* test bit */
00080 #define __L4UTIL_BITOPS_HAVE_ARCH_TEST_BIT
00081 L4_INLINE int
00082 l4util_test_bit(int b, const volatile l4_umword_t * dest)
00083 {
00084 l4_int8_t bit;
00085
00086 __asm__ __volatile__
00087 (
00088 "bt %2,%1 \n\t"
00089 "setc %0 \n\t"
00090 :
00091 "r" (bit) /* 0, old bit value */
00092 :
00093 "m" (*dest), /* 1 mem, destination operand */
00094 "Ir" (b) /* 2, bit number */

```

```

00095 :
00096 "memory", "cc"
00097);
00098
00099 return (int)bit;
00100 }
00101
00102
00103 /* bit test and set */
00104 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_SET
00105 L4_INLINE int
00106 l4util_bts(int b, volatile l4_umword_t * dest)
00107 {
00108 l4_int8_t bit;
00109
00110 __asm__ __volatile__
00111 (
00112 "lock; bts %2,%1 \n\t"
00113 "setc %0 \n\t"
00114 :
00115 "=r" (bit) /* 0, old bit value */
00116 :
00117 "m" (*dest), /* 1 mem, destination operand */
00118 "Ir" (b) /* 2, bit number */
00119 :
00120 "memory", "cc"
00121);
00122
00123 return (int)bit;
00124 }
00125
00126 /* bit test and reset */
00127 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_RESET
00128 L4_INLINE int
00129 l4util_btr(int b, volatile l4_umword_t * dest)
00130 {
00131 l4_int8_t bit;
00132
00133 __asm__ __volatile__
00134 (
00135 "lock; btr %2,%1 \n\t"
00136 "setc %0 \n\t"
00137 :
00138 "=r" (bit) /* 0, old bit value */
00139 :
00140 "m" (*dest), /* 1 mem, destination operand */
00141 "Ir" (b) /* 2, bit number */
00142 :
00143 "memory", "cc"
00144);
00145
00146 return (int)bit;
00147 }
00148
00149 /* bit test and complement */
00150 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_COMPLEMENT
00151 L4_INLINE int
00152 l4util_btc(int b, volatile l4_umword_t * dest)
00153 {
00154 l4_int8_t bit;
00155
00156 __asm__ __volatile__
00157 (
00158 "lock; btc %2,%1 \n\t"
00159 "setc %0 \n\t"
00160 :
00161 "=r" (bit) /* 0, old bit value */
00162 :
00163 "m" (*dest), /* 1 mem, destination operand */
00164 "Ir" ((l4_umword_t)b) /* 2, bit number */
00165 :
00166 "memory", "cc"
00167);
00168
00169 return (int)bit;
00170 }
00171
00172 /* bit scan reverse */
00173 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_REVERSE
00174 L4_INLINE int
00175 l4util_bsr(l4_umword_t word)
00176 {
00177 l4_umword_t tmp;
00178
00179 if (L4_UNLIKELY(word == 0))
00180 return -1;
00181

```

```

00182 __asm__ __volatile__
00183 (
00184 "bsr %1,%0 \n\t"
00185 :
00186 "=r" (tmp) /* 0, index of most significant set bit */
00187 :
00188 "r" (word) /* 1, argument */
00189);
00190
00191 return tmp;
00192 }
00193
00194 /* bit scan forward */
00195 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_FORWARD
00196 L4_INLINE int
00197 l4util_bsfl(l4_umword_t word)
00198 {
00199 l4_umword_t tmp;
00200
00201 if (L4_UNLIKELY(word == 0))
00202 return -1;
00203
00204 __asm__ __volatile__
00205 (
00206 "bsf %1,%0 \n\t"
00207 :
00208 "=r" (tmp) /* 0, index of least significant set bit */
00209 :
00210 "r" (word) /* 1, argument */
00211);
00212
00213 return tmp;
00214 }
00215
00216 #define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_SET_BIT
00217 L4_INLINE int
00218 l4util_find_first_set_bit(const void * dest, l4_size_t size)
00219 {
00220 l4_mword_t dummy0, dummy1, res;
00221
00222 __asm__ __volatile__
00223 (
00224 "xor %%rax,%%rax \n\t"
00225 "repe; scasl \n\t"
00226 "jz 1f \n\t"
00227 "lea -4(%%rdi),%%rdi \n\t"
00228 "bsfq (%%rdi),%%rax \n\t"
00229 "1: \n\t"
00230 "sub %%rbx,%%rdi \n\t"
00231 "shl $3,%%rdi \n\t"
00232 "add %%rdi,%%rax \n\t"
00233 :
00234 "=a" (res), "=&c" (dummy0), "&D" (dummy1)
00235 :
00236 "1" ((size + 31) >> 5), "2" (dest), "b" (dest)
00237 :
00238 "cc", "memory");
00239
00240 return res;
00241 }
00242
00243 #define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_ZERO_BIT
00244 L4_INLINE int
00245 l4util_find_first_zero_bit(const void * dest, l4_size_t size)
00246 {
00247 l4_mword_t dummy0, dummy1, dummy2, res;
00248
00249 if (!size)
00250 return 0;
00251
00252 __asm__ __volatile__
00253 (
00254 "mov $-1,%%rax \n\t"
00255 "xor %%rdx,%%rdx \n\t"
00256 "repe; scasl \n\t"
00257 "je 1f \n\t"
00258 "xor -4(%%rdi),%%rax \n\t"
00259 "sub $4,%%rdi \n\t"
00260 "bsf %%rax,%%rdx \n\t"
00261 "1: \n\t"
00262 "sub %[dest],%%rdi \n\t"
00263 "shl $3,%%rdi \n\t"
00264 "add %%rdi,%%rdx \n\t"
00265 :
00266 "=d" (res), "=&c" (dummy0), "&D" (dummy1), "=&a" (dummy2)
00267 :
00268 "1" ((size + 31) >> 5), "2" (dest), [dest] "S" (dest)

```

```

00269 :
00270 "cc", "memory");
00271
00272 return res;
00273 }
00274
00275 EXTERN_C_END
00276
00277 #endif /* ! __L4UTIL__INCLUDE__ARCH_AMD64__BITOPS_ARCH_H__ */

```

## 16.55 x86/l4/util/bitops\_arch.h File Reference

x86 bit manipulation functions

### 16.55.1 Detailed Description

x86 bit manipulation functions

#### Date

07/03/2001

#### Author

Lars Reuther [reuther@os.inf.tu-dresden.de](mailto:reuther@os.inf.tu-dresden.de)

Definition in file [bitops\\_arch.h](#).

## 16.56 bitops\_arch.h

```

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__ARCH_X86__BITOPS_ARCH_H__
00019 #define __L4UTIL__INCLUDE__ARCH_X86__BITOPS_ARCH_H__
00020
00021 /*****
00022 *** Implementation
00023 *****/
00024
00025 EXTERN_C_BEGIN
00026
00027 /* set bit */
00028 #define __L4UTIL_BITOPS_HAVE_ARCH_SET_BIT
00029 L4_INLINE void
00030 l4util_set_bit(int b, volatile l4_umword_t * dest)
00031 {
00032 __asm__ __volatile__
00033 (
00034 "lock; btsl %1,%0 \n\t"
00035 :
00036 :
00037 "m" (*dest), /* 0 mem, destination operand */
00038 "Ir" (b) /* 1, bit number */
00039 :
00040 "memory", "cc"
00041);
00042 }

```

```

00043
00044 /* clear bit */
00045 #define __L4UTIL_BITOPS_HAVE_ARCH_CLEAR_BIT
00046 L4_INLINE void
00047 l4util_clear_bit(int b, volatile l4_umword_t * dest)
00048 {
00049 __asm__ __volatile__
00050 (
00051 "lock; btrl %1,%0 \n\t"
00052 :
00053 :
00054 "m" (*dest), /* 0 mem, destination operand */
00055 "Ir" (b) /* 1, bit number */
00056 :
00057 "memory", "cc"
00058);
00059 }
00060
00061 /* change bit */
00062 #define __L4UTIL_BITOPS_HAVE_ARCH_COMPLEMENT_BIT
00063 L4_INLINE void
00064 l4util_complement_bit(int b, volatile l4_umword_t * dest)
00065 {
00066 __asm__ __volatile__
00067 (
00068 "lock; btc1 %1,%0 \n\t"
00069 :
00070 :
00071 "m" (*dest), /* 0 mem, destination operand */
00072 "Ir" (b) /* 1, bit number */
00073 :
00074 "memory", "cc"
00075);
00076 }
00077
00078 /* test bit */
00079 #define __L4UTIL_BITOPS_HAVE_ARCH_TEST_BIT
00080 L4_INLINE int
00081 l4util_test_bit(int b, const volatile l4_umword_t * dest)
00082 {
00083 l4_int8_t bit;
00084
00085 __asm__ __volatile__
00086 (
00087 "btl %2,%1 \n\t"
00088 "setc %0 \n\t"
00089 :
00090 "=q" (bit) /* 0, old bit value */
00091 :
00092 "m" (*dest), /* 1 mem, destination operand */
00093 "Ir" (b) /* 2, bit number */
00094 :
00095 "memory", "cc"
00096);
00097
00098 return (int)bit;
00099 }
00100
00101 /* bit test and set */
00102 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_SET
00103 L4_INLINE int
00104 l4util_bts(int b, volatile l4_umword_t * dest)
00105 {
00106 l4_int8_t bit;
00107
00108 __asm__ __volatile__
00109 (
00110 "lock; btsl %2,%1 \n\t"
00111 "setc %0 \n\t"
00112 :
00113 "=q" (bit) /* 0, old bit value */
00114 :
00115 "m" (*dest), /* 1 mem, destination operand */
00116 "Ir" (b) /* 2, bit number */
00117 :
00118 "memory", "cc"
00119);
00120
00121 return (int)bit;
00122 }
00123
00124 /* bit test and reset */
00125 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_RESET
00126 L4_INLINE int
00127 l4util_btr(int b, volatile l4_umword_t * dest)
00128 {
00129 l4_int8_t bit;

```

```

00130
00131 __asm__ __volatile__
00132 (
00133 "lock; btrl %2,%1 \n\t"
00134 "setc %0 \n\t"
00135 :
00136 "=q" (bit) /* 0, old bit value */
00137 :
00138 "m" (*dest), /* 1 mem, destination operand */
00139 "Ir" (b) /* 2, bit number */
00140 :
00141 "memory", "cc"
00142);
00143
00144 return (int)bit;
00145 }
00146
00147 /* bit test and complement */
00148 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_COMPLEMENT
00149 L4_INLINE int
00150 l4util_btc(int b, volatile l4_umword_t * dest)
00151 {
00152 l4_int8_t bit;
00153
00154 __asm__ __volatile__
00155 (
00156 "lock; btcl %2,%1 \n\t"
00157 "setc %0 \n\t"
00158 :
00159 "=q" (bit) /* 0, old bit value */
00160 :
00161 "m" (*dest), /* 1 mem, destination operand */
00162 "Ir" (b) /* 2, bit number */
00163 :
00164 "memory", "cc"
00165);
00166
00167 return (int)bit;
00168 }
00169
00170 /* bit scan reverse */
00171 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_REVERSE
00172 L4_INLINE int
00173 l4util_bsr(l4_umword_t word)
00174 {
00175 int tmp;
00176
00177 if (L4_UNLIKELY(word == 0))
00178 return -1;
00179
00180 __asm__ __volatile__
00181 (
00182 "bsrl %1,%0 \n\t"
00183 :
00184 "=r" (tmp) /* 0, index of most significant set bit */
00185 :
00186 "r" (word) /* 1, argument */
00187);
00188
00189 return tmp;
00190 }
00191
00192 /* bit scan forward */
00193 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_FORWARD
00194 L4_INLINE int
00195 l4util_bsf(l4_umword_t word)
00196 {
00197 int tmp;
00198
00199 if (L4_UNLIKELY(word == 0))
00200 return -1;
00201
00202 __asm__ __volatile__
00203 (
00204 "bsfl %1,%0 \n\t"
00205 :
00206 "=r" (tmp) /* 0, index of least significant set bit */
00207 :
00208 "r" (word) /* 1, argument */
00209);
00210
00211 return tmp;
00212 }
00213
00214 #define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_SET_BIT
00215 L4_INLINE int
00216 l4util_find_first_set_bit(const void * dest, l4_size_t size)

```



```

00217 {
00218 l4_mword_t dummy0, dummy1, res;
00219
00220 __asm__ __volatile__
00221 (
00222 "repe; scasl \n\t"
00223 "jz lf \n\t"
00224 "leal -4(%edi),%edi \n\t"
00225 "bsfl (%edi),%eax \n\t"
00226 "1: \n\t"
00227 "subl %%esi,%edi \n\t"
00228 "shll $3,%edi \n\t"
00229 "addl %%edi,%eax \n\t"
00230 :
00231 "=a" (res), "=c" (dummy0), "=D" (dummy1)
00232 :
00233 "a"(0), "c" ((size+31) >> 5), "D" (dest), "S" (dest)
00234 :
00235 "cc", "memory");
00236
00237 return res;
00238 }
00239
00240 #define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_ZERO_BIT
00241 L4_INLINE int
00242 l4util_find_first_zero_bit(const void * dest, l4_size_t size)
00243 {
00244 l4_mword_t dummy0, dummy1, dummy2, res;
00245
00246 if (!size)
00247 return 0;
00248
00249 __asm__ __volatile__
00250 (
00251 "repe; scasl \n\t"
00252 "je lf \n\t"
00253 "xorl -4(%edi),%eax \n\t"
00254 "subl $4,%edi \n\t"
00255 "bsfl %%eax,%edx \n\t"
00256 "1: \n\t"
00257 "subl %%esi,%edi \n\t"
00258 "shll $3,%edi \n\t"
00259 "addl %%edi,%edx \n\t"
00260 :
00261 "=d" (res), "=c" (dummy0), "=D" (dummy1), "=a" (dummy2)
00262 :
00263 "a" (~0), "c" ((size+31) >> 5), "d"(0), "D" (dest), "S" (dest)
00264 :
00265 "cc", "memory");
00266
00267 return res;
00268 }
00269
00270 EXTERN_C_END
00271
00272 #endif /* ! __L4UTIL__INCLUDE__ARCH_X86__BITOPS_ARCH_H__ */

```

## 16.57 arm/l4/util/cpu.h File Reference

CPU related functions.

### 16.57.1 Detailed Description

CPU related functions.

#### Author

Adam Lackorzynski [adam@os.inf.tu-dresden.de](mailto:adam@os.inf.tu-dresden.de)

Definition in file [cpu.h](#).

## 16.58 cpu.h

```

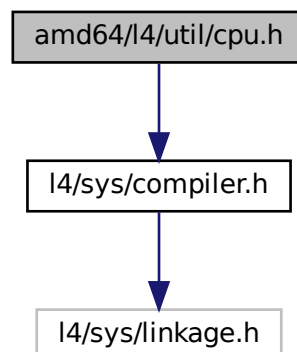
00001
00008 /*
00009 * (c) 2004-2009 Author(s)
00010 * economic rights: Technische Universität Dresden (Germany)
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU Lesser General Public License 2.1.
00013 * Please see the COPYING-LGPL-2.1 file for details.
00014 */
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__ */

```

## 16.59 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.*

## 16.59.1 Detailed Description

CPU related functions.

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [cpu.h](#).

## 16.59.2 Function Documentation

### 16.59.2.1 l4util\_cpu\_capabilities()

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

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

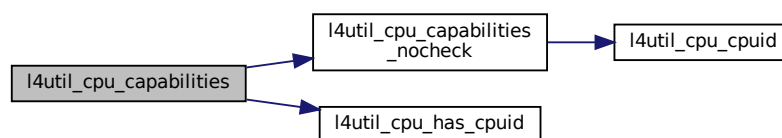
Returns

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:



### 16.59.2.2 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](#).

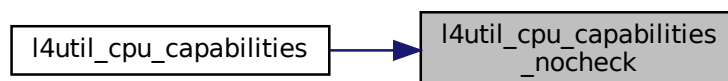
References [l4util\\_cpu\\_cpuid\(\)](#).

Referenced by [l4util\\_cpu\\_capabilities\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.59.2.3 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:

**16.60 cpu.h**

```

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
00027
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 return 1;
00069 }
00070
00071 L4_INLINE void
00072 l4util_cpu_cpuid(unsigned long mode,
00073 unsigned long *eax, unsigned long *ebx,
00074 unsigned long *ecx, unsigned long *edx)
00075 {
00076 asm volatile("cpuid"
00077 : "=a" (*eax),
00078 "=b" (*ebx),
00079 "=c" (*ecx),
00080 "=d" (*edx)
00081 : "a" (mode)
00082);

```

```

00083 }
00084
00085 L4_INLINE unsigned int
00086 l4util_cpu_capabilities_nocheck(void)
00087 {
00088 unsigned long dummy, capability;
00089
00090 /* get CPU capabilities */
00091 l4util_cpu_cpuid(1, &dummy, &dummy, &dummy, &capability);
00092
00093 return capability;
00094 }
00095
00096 L4_INLINE unsigned int
00097 l4util_cpu_capabilities(void)
00098 {
00099 if (!l4util_cpu_has_cpuid())
00100 return 0; /* CPU has not cpuid instruction */
00101
00102 return l4util_cpu_capabilities_nocheck();
00103 }
00104
00105 EXTERN_C_END
00106
00107 #endif
00108

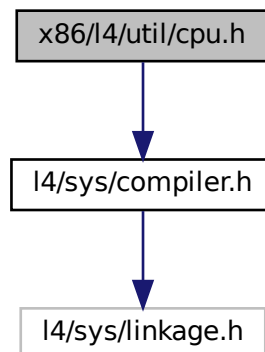
```

## 16.61 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.*

## 16.61.1 Detailed Description

CPU related functions.

### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [cpu.h](#).

## 16.61.2 Function Documentation

### 16.61.2.1 l4util\_cpu\_capabilities()

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

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

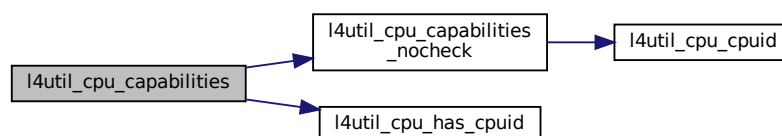
### Returns

CPU capabilities if the "cpuid" instruction is available, 0 if the "cpuid" instruction is not supported.

Definition at line 118 of file [cpu.h](#).

References [l4util\\_cpu\\_capabilities\\_nocheck\(\)](#), and [l4util\\_cpu\\_has\\_cpuid\(\)](#).

Here is the call graph for this function:



### 16.61.2.2 l4util\_cpu\_capabilities\_nocheck()

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

Returns the CPU capabilities.

#### Returns

CPU capabilities.

Definition at line 107 of file [cpu.h](#).

References [l4util\\_cpu\\_cpuid\(\)](#).

Here is the call graph for this function:



### 16.61.2.3 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](#).



## 16.62 cpu.h

```

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
00027
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 return l4util_cpu_capabilities_nocheck();
00123 }
00124
00125
00126 EXTERN_C_END
00127
00128 #endif
00129

```

## 16.63 arm/l4/util/l4\_macros.h File Reference

Main function.

### 16.63.1 Detailed Description

Main function.

#### Date

08/29/2000

#### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [l4\\_macros.h](#).

## 16.64 l4\_macros.h

```

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 */

```

## 16.65 amd64/l4/util/l4\_macros.h File Reference

Main function.

## 16.65.1 Detailed Description

Main function.

### Date

08/29/2000

### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [l4\\_macros.h](#).

## 16.66 l4\_macros.h

```
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 */
```

## 16.67 l4/util/l4\_macros.h File Reference

Some useful generic macros, L4f version.

### 16.67.1 Detailed Description

Some useful generic macros, L4f version.

### Date

11/12/2002

### Author

Lars Reuther [reuther@os.inf.tu-dresden.de](mailto:reuther@os.inf.tu-dresden.de)

Definition in file [l4\\_macros.h](#).

## 16.68 l4\_macros.h

```

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

```

## 16.69 x86/l4/util/l4\_macros.h File Reference

Main function.

### 16.69.1 Detailed Description

Main function.

#### Date

08/29/2000

#### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [l4\\_macros.h](#).

## 16.70 l4\_macros.h

```

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 */

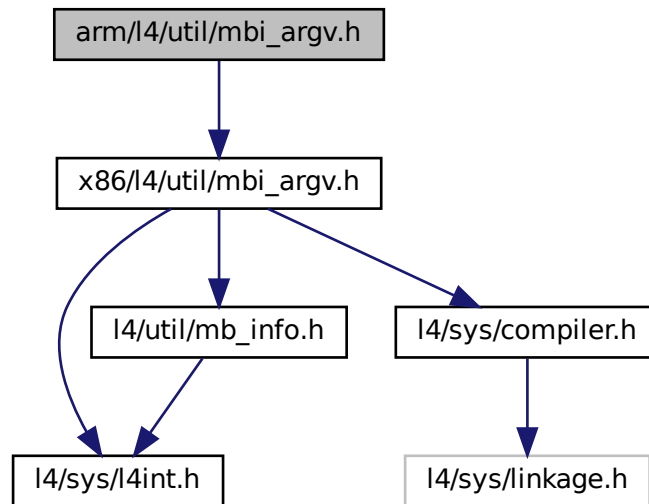
```

## 16.71 arm/l4/util/mbi\_argv.h File Reference

Multiboot.

```
#include <x86/l4/util/mbi_argv.h>
```

Include dependency graph for mbi\_argv.h:



### 16.71.1 Detailed Description

Multiboot.

Definition in file [mbi\\_argv.h](#).

## 16.72 mbi\_argv.h

```

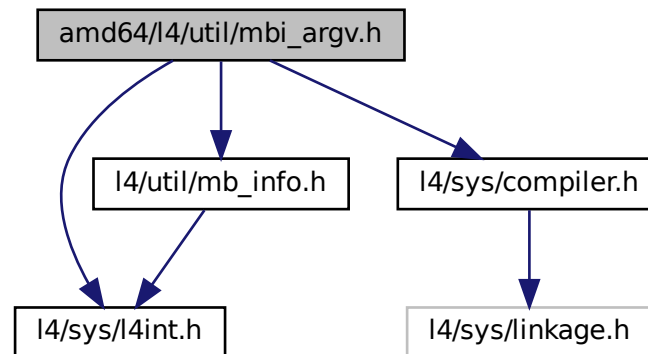
00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU Lesser General Public License 2.1.
00010 * Please see the COPYING-LGPL-2.1 file for details.
00011 */
00012 /* If this persists, move it to a generic place */
00013 #include <x86/l4/util/mbi_argv.h>

```

## 16.73 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:
```



### 16.73.1 Detailed Description

command line handling

Date

2003

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [mbi\\_argv.h](#).

## 16.74 mbi\_argv.h

```
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 L4UTIL_MBI_ARGV
00011 #define L4UTIL_MBI_ARGV
00012
```

```

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

```

## 16.75 x86/l4/util/mbi\_argv.h File Reference

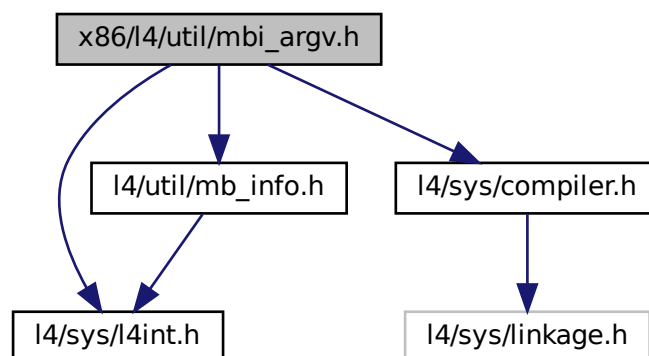
command line handling

```

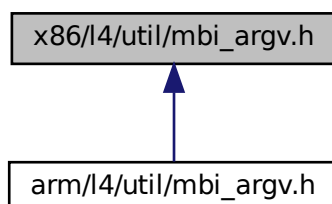
#include <l4/sys/l4int.h>
#include <l4/util/mb_info.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for mbi\_argv.h:



This graph shows which files directly or indirectly include this file:



## 16.75.1 Detailed Description

command line handling

Date

2003

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [mbi\\_argv.h](#).

## 16.76 mbi\_argv.h

```

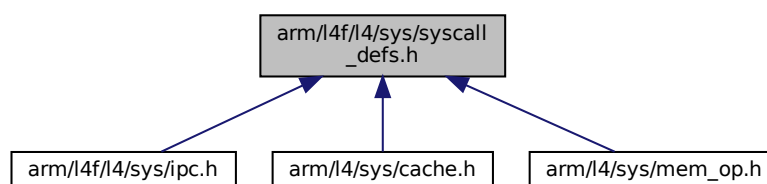
00001
00008 /*
00009 * (c) 2003-2009 Author(s)
00010 * economic rights: Technische Universität Dresden (Germany)
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU Lesser General Public License 2.1.
00013 * Please see the COPYING-LGPL-2.1 file for details.
00014 */
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

```

## 16.77 arm/l4f/l4/sys/syscall\_defs.h File Reference

Syscall entry definitions.

This graph shows which files directly or indirectly include this file:





## 16.77.1 Detailed Description

Syscall entry definitions.

Definition in file [syscall\\_defs.h](#).

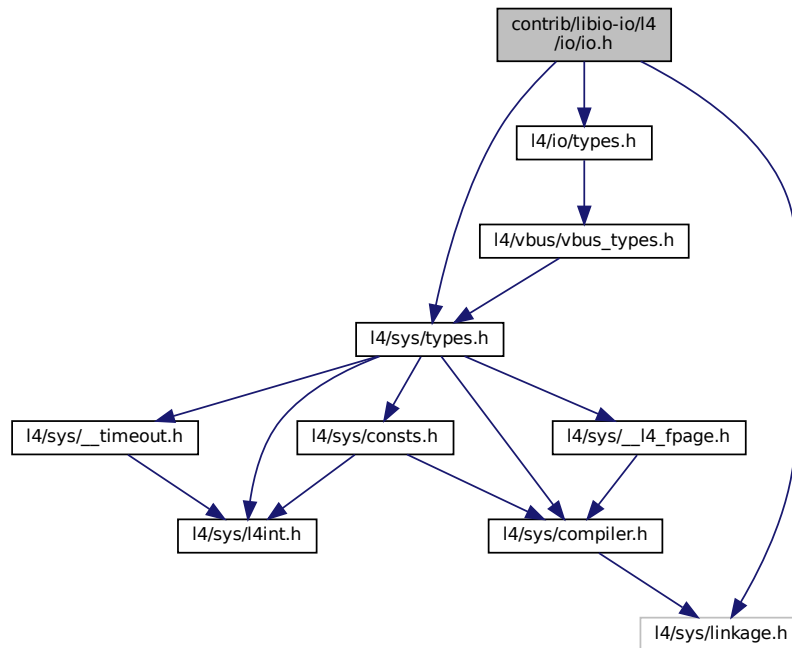
## 16.78 syscall\_defs.h

```
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 #define L4_SYSCALL_DEBUGGER (-0x0000000C-L4_SYSCALL_MAGIC_OFFSET)
00031
00032 #endif /* __L4SYS__ARCH_ARM__L4API_L4F__SYSCALL_DEFS_H__ */
```

## 16.79 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.*

## 16.79.1 Function Documentation

### 16.79.1.1 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 255 of file [io.h](#).

### 16.79.1.2 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

|                  |                                     |
|------------------|-------------------------------------|
| <i>devhandle</i> | Device handle to start iterating at |
|------------------|-------------------------------------|

#### Return values

|                  |                                 |
|------------------|---------------------------------|
| <i>devhandle</i> | Next device handle              |
| <i>dev</i>       | Device information, may be NULL |
| <i>reshandle</i> | Resource handle.                |

**Returns**

0 on success, error code otherwise

**16.79.1.3 l4io\_request\_all\_ioports()**

```
void l4io_request_all_ioports (
 void(*) (l4vbus_resource_t const *res) res_cb)
```

Request all available IO-port resources.

**Parameters**

|               |                                                                                    |
|---------------|------------------------------------------------------------------------------------|
| <i>res_cb</i> | Callback function called for every port resource found, give NULL for no callback. |
|---------------|------------------------------------------------------------------------------------|

**16.79.1.4 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.

**16.80 io.h**

```
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 irq_cap);
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
00171 L4_CV int L4_EXPORT
00172 l4io_iterate_devices(l4io_device_handle_t *devhandle,
00173 l4io_device_t *dev, l4io_resource_handle_t *reshandle);
00174
00186 L4_CV int L4_EXPORT
00187 l4io_lookup_device(const char *devname,
00188 l4io_device_handle_t *dev_handle,
00189 l4io_device_t *dev, l4io_resource_handle_t *res_handle);
00190
00205 L4_CV int L4_EXPORT
00206 l4io_lookup_resource(l4io_device_handle_t devhandle,
00207 enum l4io_resource_types_t type,
00208 l4io_resource_handle_t *reshandle,
00209 l4io_resource_t *res);
00210
00211
00212 /* ----- Convenience functions ----- */
00213
00226 L4_CV l4_addr_t L4_EXPORT
00227 l4io_request_resource_iomem(l4io_device_handle_t devhandle,
00228 l4io_resource_handle_t *reshandle);
00229
00236 L4_CV void L4_EXPORT
00237 l4io_request_all_ioports(void (*res_cb)(l4vbus_resource_t const *res));
00238
00247 L4_CV int L4_EXPORT
00248 l4io_has_resource(enum l4io_resource_types_t type,
00249 l4vbus_paddr_t start, l4vbus_paddr_t end);
00250
00251 /* ----- Implementations ----- */
00252 /* Implementations */
00253
00254 L4_INLINE
00255 l4io_device_handle_t l4io_get_root_device(void)
00256 { return 0; }
00257
00258 EXTERN_C_END

```

## 16.81 l4/cxx/alloc.h File Reference

Alloc list.

### Data Structures

- class [L4::Alloc\\_list](#)  
A simple list-based allocator.

### Namespaces

- [L4](#)  
*L4 low-level kernel interface.*

## 16.81.1 Detailed Description

Alloc list.

Definition in file [alloc.h](#).

## 16.82 alloc.h

```

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 }
```

## 16.83 l4/cxx/avl\_map File Reference

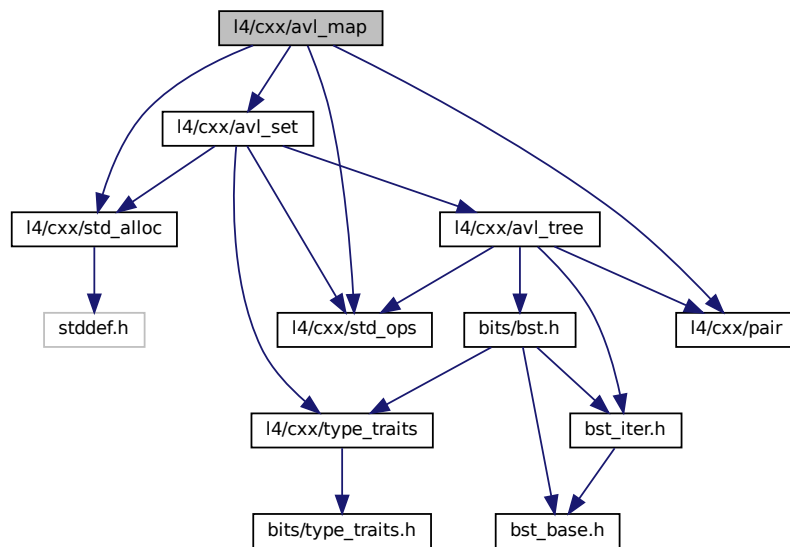
AVL map.

```

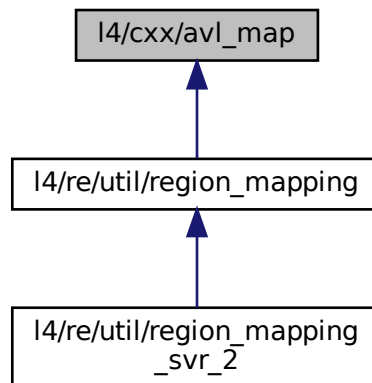
#include <l4/cxx/std_alloc>
#include <l4/cxx/std_ops>
#include <l4/cxx/pair>
```

```
#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

- [cxx](#)

*Our C++ library.*

- [cxx::Bits](#)

*Internal helpers for the cxx package.*

### 16.83.1 Detailed Description

AVL map.

Definition in file [avl\\_map](#).

## 16.84 avl\_map

```

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 <l4/cxx/std_alloc>
00027 #include <l4/cxx/std_ops>
00028 #include <l4/cxx/pair>
00029 #include <l4/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

```

## 16.85 l4/cxx/avl\_set File Reference

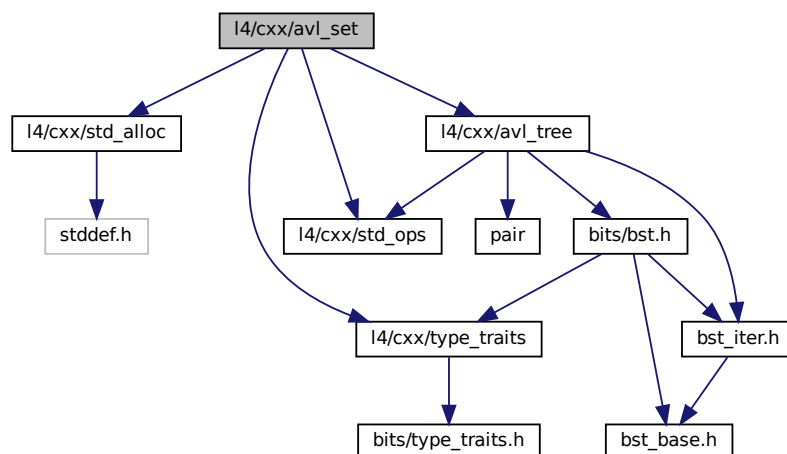
AVL set.

```

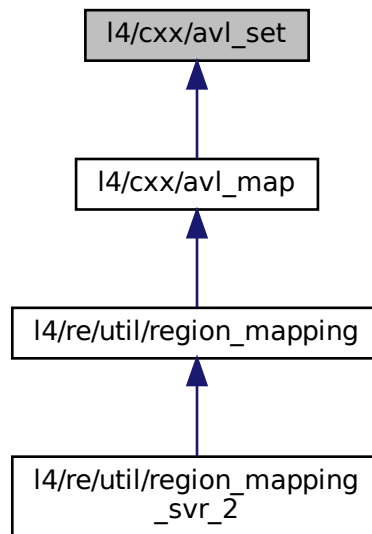
#include <l4/cxx/std_alloc>
#include <l4/cxx/std_ops>
#include <l4/cxx/type_traits>
#include <l4/cxx/avl_tree>

```

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

- `cxx`  
*Our C++ library.*
- `cxx::Bits`  
*Internal helpers for the cxx package.*

### 16.85.1 Detailed Description

AVL set.

Definition in file [avl\\_set](#).

## 16.86 avl\_set

```

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 * () { if (_n) return &_n->item; else return 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
00296 int remove(Key_type const &item)
00297 {
00298 _Node *n = _tree.remove(item);
00299 if ((long)n == -E_inval)
00300 return -E_inval;
00301
00302 if (n)

```

```

00303 {
00304 n->~_Node();
00305 _alloc.free(n);
00306 return 0;
00307 }
00308
00309 return -E_noent;
00310 }
00311
00316 int erase(Key_type const &item)
00317 { return remove(item); }
00318
00327 Node find_node(Key_type const &item) const
00328 { return Node(_tree.find_node(item)); }
00329
00338 Node lower_bound_node(Key_type const &key) const
00339 { return Node(_tree.lower_bound_node(key)); }
00340
00341 Node lower_bound_node(Key_type &&key) const
00342 { return Node(_tree.lower_bound_node(key)); }
00343
00348 Const_iterator begin() const { return _tree.begin(); }
00353 Const_iterator end() const { return _tree.end(); }
00354
00359 Iterator begin() { return _tree.begin(); }
00364 Iterator end() { return _tree.end(); }
00365
00370 Const_rev_iterator rbegin() const { return _tree.rbegin(); }
00375 Const_rev_iterator rend() const { return _tree.rend(); }
00376
00381 Rev_iterator rbegin() { return _tree.rbegin(); }
00386 Rev_iterator rend() { return _tree.rend(); }
00387
00388 Const_iterator find(Key_type const &item) const
00389 { return _tree.find(item); }
00390 };
00391
00392
00393 //-----
00394 /* Implementation of AVL Tree */
00395
00396 /* Create a copy */
00397 template< typename Item, class Compare, template<typename A> class Alloc, typename KEY_TYPE>
00398 Base_avl_set<Item, Compare, Alloc, KEY_TYPE>::Base_avl_set(Base_avl_set const &o)
00399 : _tree(), _alloc(o._alloc)
00400 {
00401 for (Const_iterator i = o.begin(); i != o.end(); ++i)
00402 insert(*i);
00403 }
00404
00405 /* Insert new _Node. */
00406 template< typename Item, class Compare, template< typename A > class Alloc, typename KEY_TYPE>
00407 Pair<typename Base_avl_set<Item, Compare, Alloc, KEY_TYPE>::Iterator, int>
00408 Base_avl_set<Item, Compare, Alloc, KEY_TYPE>::insert(Item const &item)
00409 {
00410 _Node *n = _alloc.alloc();
00411 if (!n)
00412 return cxx::pair(end(), -E_nomem);
00413
00414 new (n, Nothrow()) _Node(item);
00415 Pair<_Node *, bool> err = _tree.insert(n);
00416 if (!err.second)
00417 {
00418 n->~_Node();
00419 _alloc.free(n);
00420 }
00421
00422 return cxx::pair(Iterator(typename Tree::Iterator(err.first, err.first)), err.second ? 0 :
-E_exist);
00423 }
00424
00425 } // namespace Bits
00426
00438 template< typename ITEM_TYPE, class COMPARE = Lt_functor<ITEM_TYPE>,
00439 template<typename A> class ALLOC = New_allocator>
00440 class Avl_set :
00441 public Bits::Base_avl_set<ITEM_TYPE, COMPARE, ALLOC,
00442 Bits::Avl_set_get_key<ITEM_TYPE> >
00443 {
00444 private:
00445 typedef Bits::Base_avl_set<ITEM_TYPE, COMPARE, ALLOC,
00446 Bits::Avl_set_get_key<ITEM_TYPE> > Base;
00447
00448 public:
00449 typedef typename Base::Node_allocator Node_allocator;
00450 Avl_set() = default;
00451 Avl_set(Node_allocator const &alloc)

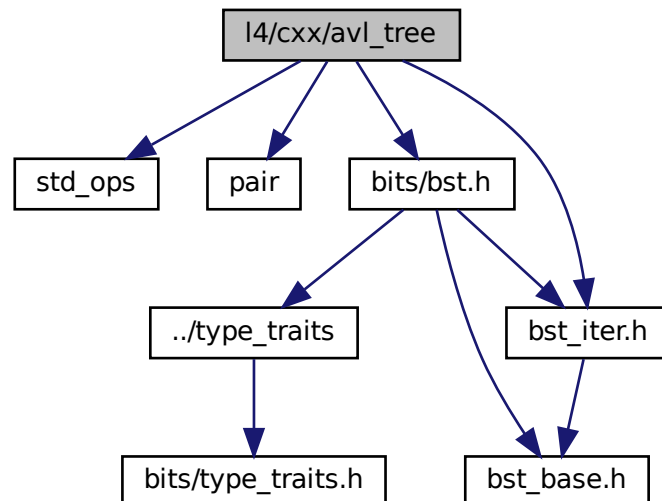
```

```
00452 : Base(alloc)
00453 {}
00454 };
00455
00456 } // namespace cxx
```

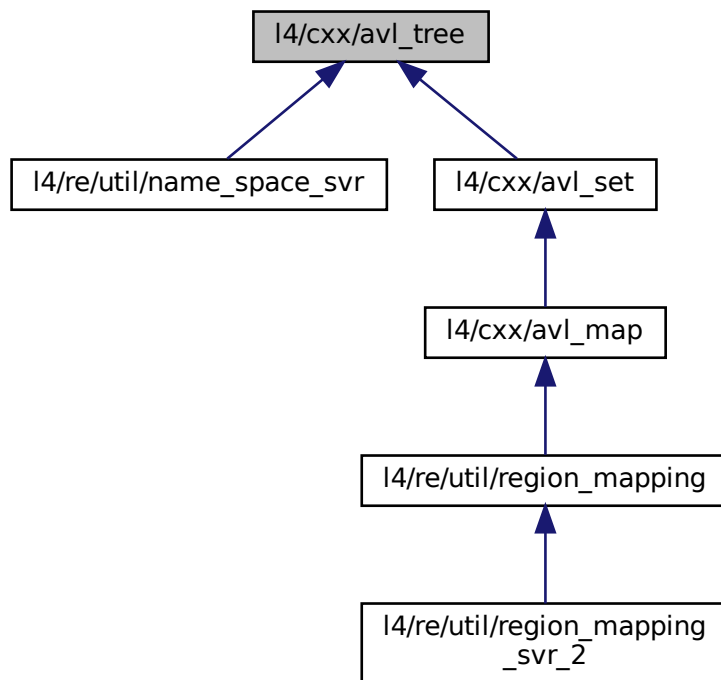
## 16.87 l4/cxx/avl\_tree File Reference

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

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

### 16.87.1 Detailed Description

AVL tree.

Definition in file [avl\\_tree](#).

## 16.88 avl\_tree

```

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
00068 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
00136 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
 pfx)
00371 {
00372 typedef Avl_tree_node A;
00373
00374 if (!n)
00375 return true;
00376
00377 int dpx[2] = {depth, depth};
00378 bool res = true;
00379
00380 res = rec_dump(n->next<A>(Dir::R), depth + 1, dpx + 1, print, '/');
00381
00382 if (print)
00383 {
00384 fprintf(stderr, "%2d: [%8p] b=%1d: ", depth, n, (int)n->balance().d);
00385
00386 for (int i = 0; i < depth; ++i)
00387 std::cerr << " ";
00388
00389 std::cerr << pfx << (static_cast<Node*>(n)->item) << std::endl;
00390 }
00391
00392 res = res & rec_dump(n->next<A>(Dir::L), depth + 1, dpx, print, '\\');
00393
00394 int b = dpx[1] - dpx[0];
00395
00396 if (b < 0)
00397 *dp = dpx[0];
00398 else
00399 *dp = dpx[1];
00400
00401 Bal x = n->balance();
00402 if ((b < -1 || b > 1) ||
00403 (b == 0 && x != Bal::N) ||
00404 (b == -1 && x != Bal::L) ||
00405 (b == 1 && x != Bal::R))
00406 {
00407 if (print)
00408 fprintf(stderr, "%2d: [%8p] b=%1d: balance error %d\n", depth, n, (int)n->balance().d, b);
00409 return false;
00410 }
00411 return res;
00412 }

```

```

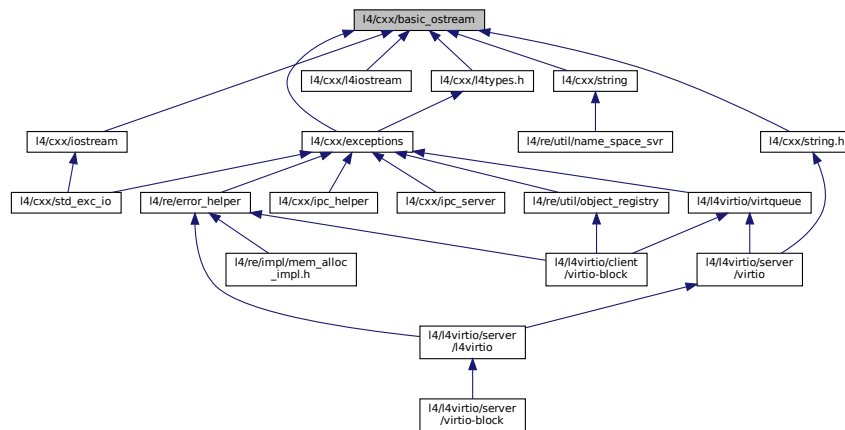
00413 #endif
00414
00415 }
00416

```

## 16.89 I4/cxx/basic\_ostream File Reference

Basic IO stream.

This graph shows which files directly or indirectly include this file:



### Data Structures

- class [L4::IOModifier](#)  
*Modifier class for the IO stream.*

### Namespaces

- [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.*

#### 16.89.1 Detailed Description

Basic IO stream.

Definition in file [basic\\_ostream](#).

## 16.90 basic\_ostream

```

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 }

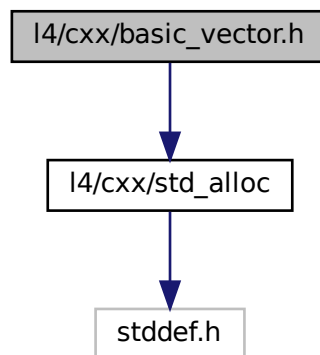
```

```
00288
00289 inline
00290 L4::BasicOStream &
00291 operator « (L4::BasicOStream &o, L4::IONumFmt const &n)
00292 { return n.print(o); }
```

## 16.91 l4/cxx/basic\_vector.h File Reference

Basic vector.

```
#include <l4/cxx/std_alloc>
Include dependency graph for basic_vector.h:
```



### Namespaces

- [cxx](#)

*Our C++ library.*

#### 16.91.1 Detailed Description

Basic vector.

Definition in file [basic\\_vector.h](#).



## 16.92 basic\_vector.h

```

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 };

```

## 16.93 l4/cxx/bits/bst.h File Reference

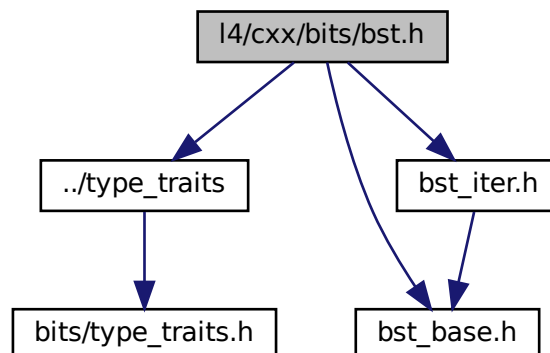
AVL tree.

```

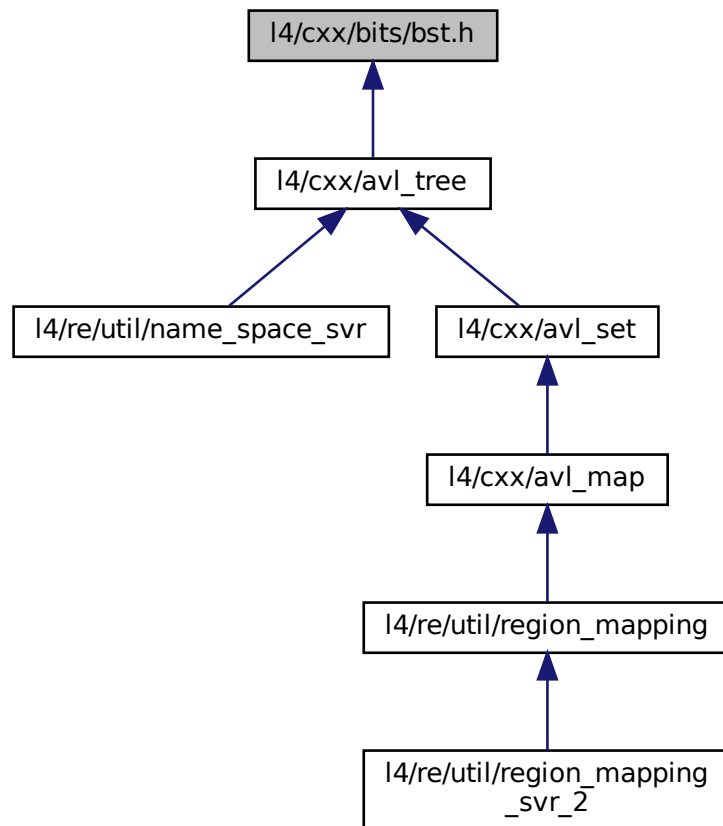
#include "../type_traits"
#include "bst_base.h"
#include "bst_iter.h"

```

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

- [cxx](#)  
*Our C++ library.*
- [cxx::Bits](#)  
*Internal helpers for the cxx package.*

### 16.93.1 Detailed Description

AVL tree.

Definition in file [bst.h](#).

## 16.94 bst.h

```

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 #pragma once
00025
00026 #include "../type_traits"
00027 #include "bst_base.h"
00028 #include "bst_iter.h"
00029
00030 namespace cxx { namespace Bits {
00031
00032 template< typename Node, typename Get_key, typename Compare >
00033 class Bst
00034 {
00035 private:
00036 typedef Direction Dir;
00037 struct Fwd
00038 {
00039 static Node *child(Node const *n, Direction d)
00040 { return Bst_node::next<Node>(n, d); }
00041
00042 static bool cmp(Node const *l, Node const *r)
00043 { return Compare()(Get_key::key_of(l), Get_key::key_of(r)); }
00044 };
00045
00046 struct Rev
00047 {
00048 static Node *child(Node const *n, Direction d)
00049 { return Bst_node::next<Node>(n, !d); }
00050
00051 static bool cmp(Node const *l, Node const *r)
00052 { return Compare()(Get_key::key_of(r), Get_key::key_of(l)); }
00053 };
00054
00055 public:
00056 typedef typename Get_key::Key_type Key_type;
00057 typedef typename Type_traits<Key_type>::Param_type Key_param_type;
00058
00059 typedef Fwd Fwd_iter_ops;
00060 typedef Rev Rev_iter_ops;
00061
00062 typedef __Bst_iter<Node, Node, Fwd> Iterator;
00063 typedef __Bst_iter<Node, Node const, Fwd> Const_iterator;
00064 typedef __Bst_iter<Node, Node, Rev> Rev_iterator;
00065 typedef __Bst_iter<Node, Node const, Rev> Const_rev_iterator;
00066
00067 protected:
00068 Bst_node *_head;
00069
00070 Bst() : _head(0) {}
00071
00072 Node *head() const { return static_cast<Node*>(_head); }
00073
00074 static Key_type k(Bst_node const *n)
00075 { return Get_key::key_of(static_cast<Node const *>(n)); }
00076
00077 static Dir dir(Key_param_type l, Key_param_type r)
00078 {
00079 Compare cmp;
00080 Dir d(cmp(r, l));
00081 if (d == Direction::L && !cmp(l, r))
00082 return Direction::N;
00083 return d;
00084 }
00085
00086 static Dir dir(Key_param_type l, Bst_node const *r)
00087 { return dir(l, k(r)); }
00088

```

```

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(); }
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 }

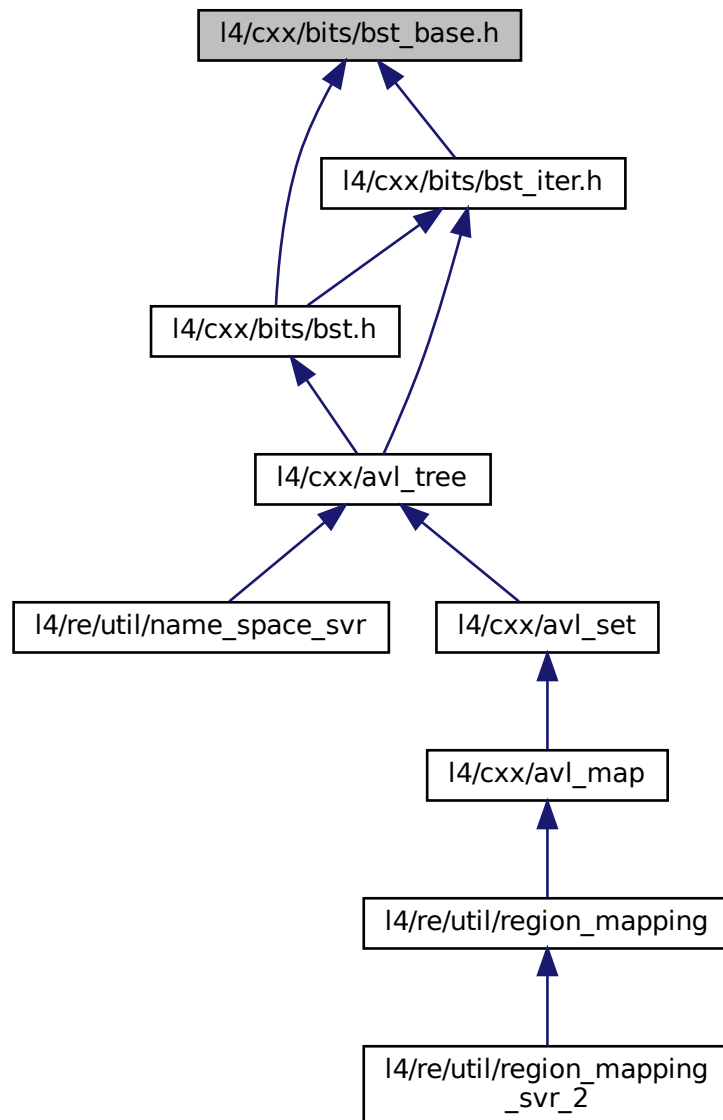
```

```
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 }
```

## 16.95 I4/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

- [CXX](#)

*Our C++ library.*

- [cxx::Bits](#)

*Internal helpers for the cxx package.*

## 16.95.1 Detailed Description

AVL tree.

Definition in file [bst\\_base.h](#).

## 16.96 bst\_base.h

```

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 search 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
00065 static Bst_node *next(Bst_node const *p, Direction d)
00066 { return p->c[d.d]; }
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

```

```

00098 static void next(Bst_node *p, Direction d, Bst_node *n)
00099 { p->_c[d.d] = n; }
00100
00102 static Bst_node **next_p(Bst_node *p, Direction d)
00103 { return &p->_c[d.d]; }
00104
00106 template< typename Node > static
00107 Node *next(Bst_node const *p, Direction d)
00108 { return static_cast<Node *>(p->_c[d.d]); }
00109
00111 static void rotate(Bst_node **t, Direction idir);
00114 private:
00115 Bst_node *_c[2];
00116
00117 protected:
00119 Bst_node() {}
00120
00122 explicit Bst_node(bool) { _c[0] = _c[1] = 0; }
00123 };
00124
00125 inline
00126 void
00127 Bst_node::rotate(Bst_node **t, Direction idir)
00128 {
00129 Bst_node *tmp = *t;
00130 *t = next(tmp, idir);
00131 next(tmp, idir, next(*t, !idir));
00132 next(*t, !idir, tmp);
00133 }
00134
00135 }}

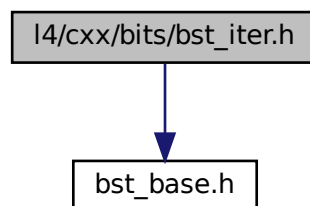
```

## 16.97 l4/cxx/bits/bst\_iter.h File Reference

AVL tree.

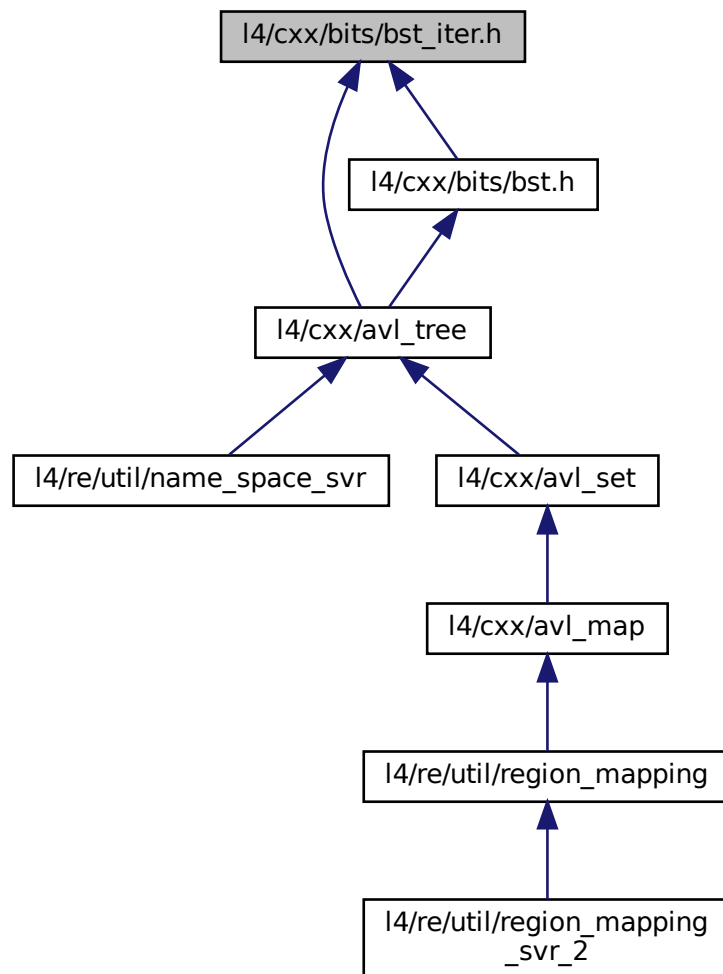
```
#include "bst_base.h"
```

Include dependency graph for bst\_iter.h:





This graph shows which files directly or indirectly include this file:



## Namespaces

- [cxx](#)  
*Our C++ library.*
- [cxx::Bits](#)  
*Internal helpers for the cxx package.*

### 16.97.1 Detailed Description

AVL tree.

Definition in file [bst\\_iter.h](#).

## 16.98 bst\_iter.h

```

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 //-----
00086 /* IMPLEMENTATION: __Bst_iter_b */
00087
00088 template< typename Node, typename Node_op>
00089 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 }

```

## 16.99 l4/cxx/exceptions File Reference

Base exceptions.

```

#include <l4/cxx/l4types.h>
#include <l4/cxx/basic_ostream>
#include <l4/sys/err.h>
#include <l4/sys/capability>
#include <l4/util/backtrace.h>

```



*Exception for an unknown condition.*

- class [L4::Element\\_not\\_found](#)

*Exception for a failed lookup (element not found).*

- class [L4::Invalid\\_capability](#)

*Indicates that an invalid object was invoked.*

- class [L4::Com\\_error](#)

*Error conditions during IPC.*

- class [L4::Bounds\\_error](#)

*Access out of bounds.*

## Namespaces

- [L4](#)

*L4 low-level kernel interface.*

## Macros

- `#define L4\_CXX\_EXCEPTION\_BACKTRACE 20`

*Number of instruction pointers in backtrace.*

### 16.99.1 Detailed Description

Base exceptions.

Definition in file [exceptions](#).

## 16.100 exceptions

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00007 /*
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 * Alexander Warg <warg@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #pragma once
00026
00027 #include <l4/cxx/l4types.h>
00028 #include <l4/cxx/basic_ostream>
00029 #include <l4/sys/err.h>
00030 #include <l4/sys/capability>
00031
00032
00033
00034
00039 #ifndef L4_CXX_NO_EXCEPTION_BACKTRACE
00040 # define L4_CXX_EXCEPTION_BACKTRACE 20
00041 #endif
00042
00043 #if defined(L4_CXX_EXCEPTION_BACKTRACE)
00044 #include <l4/util/backtrace.h>

```

```

00045 #endif
00046
00048 namespace L4
00049 {
00062 class Exception_tracer
00063 {
00064 #if defined(L4_CXX_EXCEPTION_BACKTRACE)
00065 private:
00066 void *_pc_array[L4_CXX_EXCEPTION_BACKTRACE];
00067 int _frame_cnt;
00068
00069 protected:
00073 #if defined(__PIC__)
00074 Exception_tracer() throw() : _frame_cnt(0) {}
00075 #else
00076 Exception_tracer() throw()
00077 : _frame_cnt(l4util_backtrace(_pc_array, L4_CXX_EXCEPTION_BACKTRACE)) {}
00078 #endif
00079
00080 public:
00084 void const *const *_pc_array() const throw() { return _pc_array; }
00088 int frame_count() const throw() { return _frame_cnt; }
00089 #else
00090 protected:
00094 Exception_tracer() throw() {}
00095
00096 public:
00100 void const *const *_pc_array() const throw() { return 0; }
00104 int frame_count() const throw() { return 0; }
00105 #endif
00106 };
00107
00116 class Base_exception : public Exception_tracer
00117 {
00118 protected:
00120 Base_exception() throw() {}
00121
00122 public:
00126 virtual char const *str() const throw () = 0;
00127
00129 virtual ~Base_exception() throw () {}
00130 };
00131
00139 class Runtime_error : public Base_exception
00140 {
00141 private:
00142 long _errno;
00143 char _extra[80];
00144
00145 public:
00152 explicit Runtime_error(long err_no, char const *extra = 0) throw ()
00153 : _errno(err_no)
00154 {
00155 if (!extra)
00156 _extra[0] = 0;
00157 else
00158 {
00159 unsigned i = 0;
00160 for (; i < sizeof(_extra) && extra[i]; ++i)
00161 _extra[i] = extra[i];
00162 _extra[i < sizeof(_extra) ? i : sizeof(_extra) - 1] = 0;
00163 }
00164 }
00165 char const *str() const throw ()
00166 { return l4sys_errtostr(_errno); }
00167
00173 char const *extra_str() const { return _extra; }
00174 ~Runtime_error() throw () {}
00175
00181 long err_no() const throw() { return _errno; }
00182 };
00183
00188 class Out_of_memory : public Runtime_error
00189 {
00190 public:
00192 explicit Out_of_memory(char const *extra = "") throw()
00193 : Runtime_error(-L4_ENOMEM, extra) {}
00195 ~Out_of_memory() throw() {}
00196 };
00197
00198
00203 class Element_already_exists : public Runtime_error
00204 {
00205 public:
00206 explicit Element_already_exists(char const *e = "") throw()
00207 : Runtime_error(-L4_EEXIST, e) {}
00208 ~Element_already_exists() throw() {}

```

```

00209 };
00210
00219 class Unknown_error : public Base_exception
00220 {
00221 public:
00222 Unknown_error() throw() {}
00223 char const *str() const throw() { return "unknown error"; }
00224 ~Unknown_error() throw() {}
00225 };
00226
00231 class Element_not_found : public Runtime_error
00232 {
00233 public:
00234 explicit Element_not_found(char const *e = "") throw()
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) throw() : _o(o) {}
00256 template< typename T>
00257 explicit Invalid_capability(Cap<T> const &o) throw() : _o(o.cap()) {}
00258 char const *str() const throw() { return "invalid object"; }
00259
00264 Cap<void> const &cap() const throw() { return _o; }
00265 ~Invalid_capability() throw() {}
00266 };
00267
00274 class Com_error : public Runtime_error
00275 {
00276 public:
00281 explicit Com_error(long err) throw() : Runtime_error(err) {}
00282
00283 ~Com_error() throw() {}
00284 };
00285
00289 class Bounds_error : public Runtime_error
00290 {
00291 public:
00292 explicit Bounds_error(char const *e = "") throw()
00293 : Runtime_error(-L4_ERANGE, e) {}
00294 ~Bounds_error() throw() {}
00295 };
00297 };
00298
00299 inline
00300 L4::BasicOStream &
00301 operator << (L4::BasicOStream &o, L4::Base_exception const &e)
00302 {
00303 o << "Exception: " << e.str() << ", backtrace ...\n";
00304 for (int i = 0; i < e.frame_count(); ++i)
00305 o << L4::n_hex(l4_addr_t(e.pc_array()[i])) << '\n';
00306
00307 return o;
00308 }
00309
00310 inline
00311 L4::BasicOStream &
00312 operator << (L4::BasicOStream &o, L4::Runtime_error const &e)
00313 {
00314 o << "Exception: " << e.str() << ": ";
00315 if (e.extra_str())
00316 o << e.extra_str() << ": ";
00317 o << "backtrace ...\n";
00318 for (int i = 0; i < e.frame_count(); ++i)
00319 o << L4::n_hex(l4_addr_t(e.pc_array()[i])) << '\n';
00320
00321 return o;
00322 }

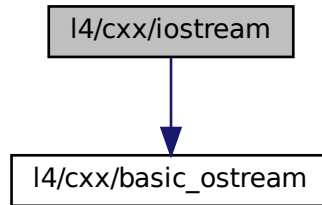
```

## 16.101 l4/cxx/iostream File Reference

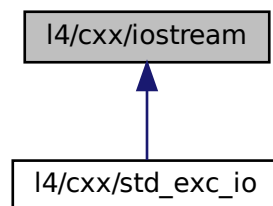
IO Stream.

```
#include <l4/cxx/basic_ostream>
```

Include dependency graph for iostream:



This graph shows which files directly or indirectly include this file:



## Namespaces

- [L4](#)  
*L4 low-level kernel interface.*

## Variables

- BasicOSTream [L4::cout](#)  
*Standard output stream.*
- BasicOSTream [L4::cerr](#)  
*Standard error stream.*

### 16.101.1 Detailed Description

IO Stream.

Definition in file [iostream](#).



## 16.102 iostream

```

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 };

```

## 16.103 l4/cxx/ipc\_helper File Reference

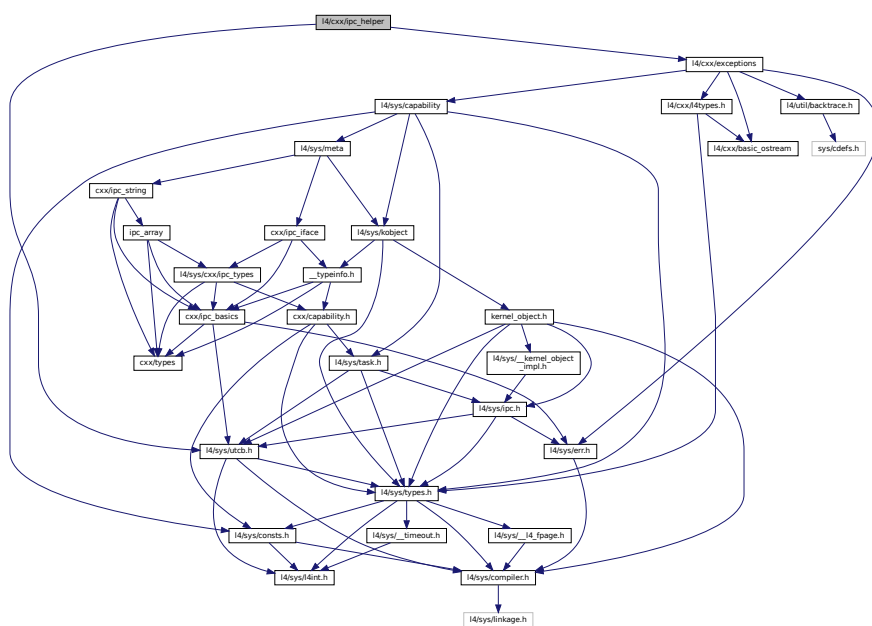
IPC helper.

```

#include <l4/cxx/exceptions>
#include <l4/sys/utcb.h>

```

Include dependency graph for ipc\_helper:



## Namespaces

- [L4](#)

*[L4](#) low-level kernel interface.*

## Functions

- void [L4::throw\\_ipc\\_exception](#) ([L4::Cap](#)< void > const &o, [l4\\_msgtag\\_t](#) const &err, [l4\\_utcb\\_t](#) \*utcb)  
*Throw an [L4](#) IPC error as exception.*
- void [L4::throw\\_ipc\\_exception](#) (void const \*o, [l4\\_msgtag\\_t](#) const &err, [l4\\_utcb\\_t](#) \*utcb)  
*Throw an [L4](#) IPC error as exception.*

### 16.103.1 Detailed Description

IPC helper.

Definition in file [ipc\\_helper](#).

### 16.104 ipc\_helper

```

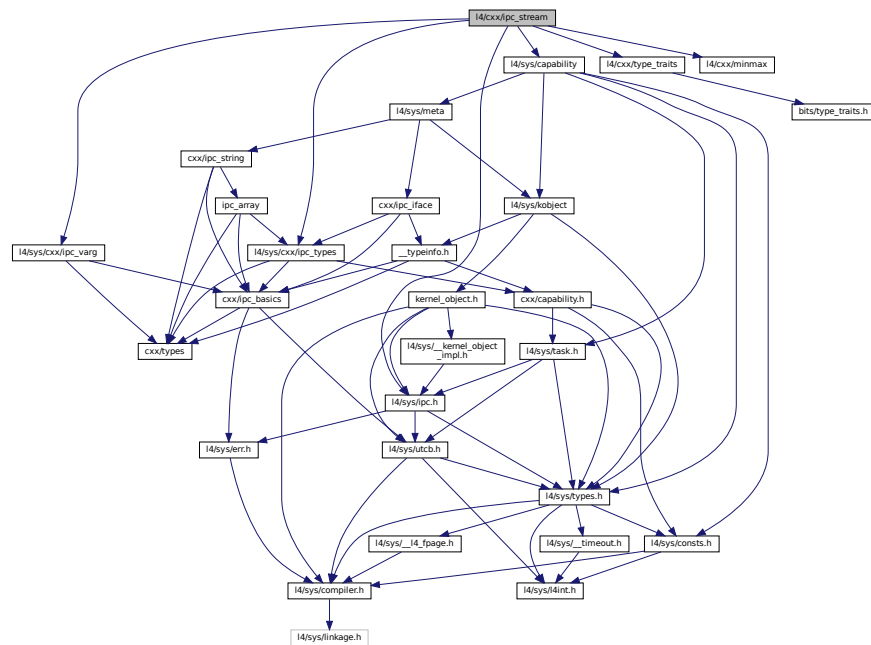
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #pragma once
00025
00026 #include <l4/cxx/exceptions>
00027 #include <l4/sys/utcb.h>
00028
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 }
```

## 16.105 l4/cxx/ipc\_stream File Reference

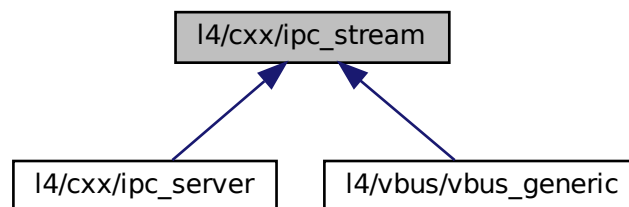
IPC stream.

```
#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>
```

Include dependency graph for ipc\_stream:



This graph shows which files directly or indirectly include this file:



### Data Structures

- class [L4::ipc::Str\\_cp\\_in< T >](#)

- *Abstraction for extracting a zero-terminated string from an [lpc::lstream](#).*
- class [L4::lpc::Msg\\_ptr< T >](#)  
*Pointer to an element of type *T* in an [lpc::lstream](#).*
- class [L4::lpc::lstream](#)  
*Input stream for IPC unmarshalling.*
- class [L4::lpc::Ostream](#)  
*Output stream for IPC marshalling.*
- class [L4::lpc::lostream](#)  
*Input/Output stream for IPC [un]marshalling.*

## Namespaces

- [L4](#)  
*[L4](#) low-level kernel interface.*
- [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::lstream](#).*
- `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::lstream & operator>> (L4::lpc::lstream &s, bool &v)`  
*Extract one element of type *T* from the stream *s*.*
- `L4::lpc::lstream & operator>> (L4::lpc::lstream &s, l4\_msgtag\_t &v)`  
*Extract the [L4](#) message tag from the stream *s*.*
- `template<typename T >`  
`L4::lpc::lstream & operator>> (L4::lpc::lstream &s, L4::lpc::Internal::Buf\_in< T > const &v)`  
*Extract an array of *T* elements from the stream *s*.*
- `template<typename T >`  
`L4::lpc::lstream & operator>> (L4::lpc::lstream &s, L4::lpc::Msg\_ptr< T > const &v)`  
*Extract an element of type *T* from the stream *s*.*
- `template<typename T >`  
`L4::lpc::lstream & operator>> (L4::lpc::lstream &s, L4::lpc::Internal::Buf\_cp\_in< T > const &v)`  
*Extract an array of *T* elements from the stream *s*.*
- `template<typename T >`  
`L4::lpc::lstream & operator>> (L4::lpc::lstream &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.*

## 16.105.1 Detailed Description

IPC stream.

Definition in file [ipc\\_stream](#).

## 16.105.2 Function Documentation

### 16.105.2.1 [operator<<\(\)](#) [1/4]

```
L4::lpc::Ostream& operator<< (
 L4::lpc::Ostream & s,
 bool v) [inline]
```

Insert an element to type `T` into the stream `s`.

#### Parameters

|                |                                                   |
|----------------|---------------------------------------------------|
| <code>s</code> | The stream to insert the element <code>v</code> . |
| <code>v</code> | The element to insert.                            |

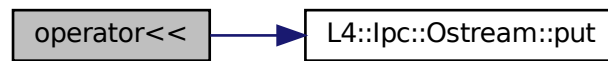
#### Returns

The stream `s`.

Definition at line 1204 of file [ipc\\_stream](#).

References [L4::lpc::Ostream::put\(\)](#).

Here is the call graph for this function:



### 16.105.2.2 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 1276 of file [ipc\\_stream](#).

References [L4::Ipc::Ostream::put\(\)](#).

Here is the call graph for this function:



### 16.105.2.3 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

|   |                                              |
|---|----------------------------------------------|
| s | The stream to insert the array v.            |
| v | The array to insert (see lpc::Buf_cp_out()). |

#### Returns

The stream s.

Definition at line 1255 of file [ipc\\_stream](#).

References [L4::lpc::Ostream::put\(\)](#).

Here is the call graph for this function:



### 16.105.2.4 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

|   |                                               |
|---|-----------------------------------------------|
| s | The stream to insert the tag v.               |
| v | The <a href="#">L4</a> message tag to insert. |

**Returns**

The stream `s`.

**Note**

Only one message tag can be inserted into a stream. Multiple insertions simply overwrite previous insertions.

Definition at line 1239 of file `ipc_stream`.

References [L4::ipc::Ostream::tag\(\)](#).

Here is the call graph for this function:

**16.105.2.5 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**

|                  |                |                             |
|------------------|----------------|-----------------------------|
|                  | <code>s</code> | The stream to extract from. |
| <code>out</code> | <code>v</code> | Extracted value.            |

**Returns**

The stream `s`.

Definition at line 1051 of file `ipc_stream`.

References [L4::ipc::Istream::get\(\)](#).



Here is the call graph for this function:



### 16.105.2.6 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

|     |   |                                                              |
|-----|---|--------------------------------------------------------------|
|     | s | The stream to extract from.                                  |
| out | v | Buffer description to copy the array to (lpc::Buf_cp_out()). |

#### Returns

The stream s.

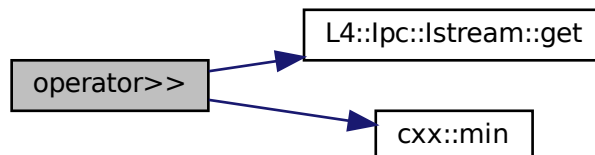
This operator does a copy out of the data into the given buffer.

See lpc::Buf\_in, lpc::Buf\_cp\_in, and lpc::Buf\_cp\_out.

Definition at line 1158 of file [lpc\\_stream](#).

References [L4::lpc::Istream::get\(\)](#), and [cxx::min\(\)](#).

Here is the call graph for this function:



**16.105.2.7 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**

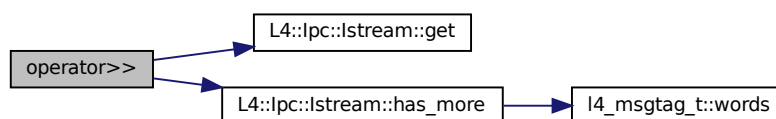
If array does not fit into transmitted words size will be set to zero. Client has to implement check against zero.

See `lpc::Buf_in`, `lpc::Buf_cp_in`, and `lpc::Buf_cp_out`.

Definition at line 1110 of file `ipc_stream`.

References `L4::lpc::Istream::get()`, and `L4::lpc::Istream::has_more()`.

Here is the call graph for this function:

**16.105.2.8 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**

|     |          |                                   |
|-----|----------|-----------------------------------|
|     | <i>s</i> | The stream to extract from.       |
| out | <i>v</i> | Pointer to the extracted element. |

**Returns**

The stream *s*.

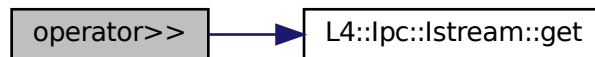
This operator actually does not copy out the data, but returns a pointer into the message buffer itself. This means that the data is only valid as long as there is no new data inserted into the stream.

See `Msg_ptr`.

Definition at line 1137 of file `ipc_stream`.

References [L4::Ipc::Istream::get\(\)](#).

Here is the call graph for this function:

**16.105.2.9 operator>>() [5/6]**

```

template<typename T >
L4::Ipc::Istream& operator>> (
 L4::Ipc::Istream & s,
 L4::Ipc::Str_cp_in< T > const & v) [inline]

```

Extract a zero-terminated string from the stream.

**Parameters**

|     |          |                                                                             |
|-----|----------|-----------------------------------------------------------------------------|
|     | <i>s</i> | The stream to extract from.                                                 |
| out | <i>v</i> | Buffer description to copy the array to ( <code>Ipc::Str_cp_out()</code> ). |

**Returns**

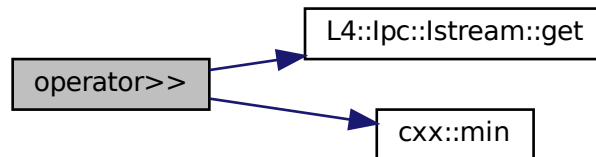
The stream *s*.

This operator does a copy out of the data into the given buffer.

Definition at line 1179 of file [ipc\\_stream](#).

References [L4::Ipc::Istream::get\(\)](#), and [cxx::min\(\)](#).

Here is the call graph for this function:



#### 16.105.2.10 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 1085 of file [ipc\\_stream](#).

References [L4::Ipc::Istream::tag\(\)](#).

Here is the call graph for this function:



## 16.106 ipc\_stream

```

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 #define L4_CXX_IPC_BACKWARD_COMPAT
00035
00036 namespace L4 {
00037 namespace Ipc {
00038
00039 class Ostream;
00040 class Istream;
00041
00042 namespace Internal {
00060 template< typename T >
00061 class Buf_cp_out
00062 {
00063 public:
00070 Buf_cp_out(T const *v, unsigned long size) : _v(v), _s(size) {}
00071
00079 unsigned long size() const { return _s; }
00080
00088 T const *buf() const { return _v; }
00089
00090 private:
00091 friend class Ostream;
00092 T const *_v;
00093 unsigned long _s;
00094 };
00095 }
00096
00112 template< typename T >
00113 Internal::Buf_cp_out<T> buf_cp_out(T const *v, unsigned long size)
00114 { return Internal::Buf_cp_out<T>(v, size); }
00115
00116 namespace Internal {
00130 template< typename T >
00131 class Buf_cp_in
00132 {
00133 public:
00142 Buf_cp_in(T *v, unsigned long &size) : _v(v), _s(&size) {}
00143
00144 unsigned long &size() const { return *_s; }
00145 T *buf() const { return _v; }
00146
00147 private:
00148 friend class Istream;
00149 T *_v;
00150 unsigned long *_s;
00151 };
00152 }
00153
00171 template< typename T >
00172 Internal::Buf_cp_in<T> buf_cp_in(T *v, unsigned long &size)
00173 { return Internal::Buf_cp_in<T>(v, size); }
00174
00190 template< typename T >
00191 class Str_cp_in
00192 {
00193 public:

```

```

00202 Str_cp_in(T *v, unsigned long &size) : _v(v), _s(&size) {}
00203
00204 unsigned long &size() const { return *_s; }
00205 T *buf() const { return _v; }
00206
00207 private:
00208 friend class Istream;
00209 T *_v;
00210 unsigned long *_s;
00211 };
00212
00225 template< typename T >
00226 Str_cp_in<T> str_cp_in(T *v, unsigned long &size)
00227 { return Str_cp_in<T>(v, size); }
00228
00241 template< typename T >
00242 class Msg_ptr
00243 {
00244 private:
00245 T **_p;
00246 public:
00253 explicit Msg_ptr(T *p) : _p(&p) {}
00254 void set(T *p) const { *_p = p; }
00255 };
00256
00264 template< typename T >
00265 Msg_ptr<T> msg_ptr(T *p)
00266 { return Msg_ptr<T>(p); }
00267
00268
00269 namespace Internal {
00283 template< typename T >
00284 class Buf_in
00285 {
00286 public:
00293 Buf_in(T *v, unsigned long &size) : _v(&v), _s(&size) {}
00294
00295 void set_size(unsigned long s) const { *_s = s; }
00296 T *buf() const { return *_v; }
00297
00298 private:
00299 friend class Istream;
00300 T **_v;
00301 unsigned long *_s;
00302 };
00303 }
00304
00322 template< typename T >
00323 Internal::Buf_in<T> buf_in(T *v, unsigned long &size)
00324 { return Internal::Buf_in<T>(v, size); }
00325
00326 namespace Utc_b_stream_check
00327 {
00328 static bool check_utc_b_data_offset(unsigned sz)
00329 { return sz > sizeof(l4_umword_t) * L4_UTCB_GENERIC_DATA_SIZE; }
00330 }
00331
00332
00347 class Istream
00348 {
00349 public:
00361 Istream(l4_utcb_t *utcb)
00362 : _tag(), _utcb(utcb),
00363 _current_msg(reinterpret_cast<char*>(l4_utcb_mr_u(utcb)->mr)),
00364 _pos(0), _current_buf(0)
00365 {}
00366
00371 void reset()
00372 {
00373 _pos = 0;
00374 _current_buf = 0;
00375 _current_msg = reinterpret_cast<char*>(l4_utcb_mr_u(_utcb)->mr);
00376 }
00377
00381 template< typename T >
00382 bool has_more(unsigned long count = 1)
00383 {
00384 auto const max_bytes = L4_UTCB_GENERIC_DATA_SIZE * sizeof(l4_umword_t);
00385 unsigned apos = cxx::Type_traits<T>::align(_pos);
00386 return (count <= max_bytes / sizeof(T))
00387 && (apos + (sizeof(T) * count)
00388 <= _tag.words() * sizeof(l4_umword_t));
00389 }
00390
00395
00406 template< typename T >
00407 unsigned long get(T *buf, unsigned long elems)

```

```

00408 {
00409 if (L4_UNLIKELY(!has_more<T>(elems)))
00410 return 0;
00411
00412 unsigned long size = elems * sizeof(T);
00413 _pos = cxx::Type_traits<T>::align(_pos);
00414
00415 __builtin_memcpy(buf, _current_msg + _pos, size);
00416 _pos += size;
00417 return elems;
00418 }
00419
00420
00426 template< typename T >
00427 void skip(unsigned long elems)
00428 {
00429 if (L4_UNLIKELY(!has_more<T>(elems)))
00430 return;
00431
00432 unsigned long size = elems * sizeof(T);
00433 _pos = cxx::Type_traits<T>::align(_pos);
00434 _pos += size;
00435 }
00436
00451 template< typename T >
00452 unsigned long get(Msg_ptr<T> const &buf, unsigned long elems = 1)
00453 {
00454 if (L4_UNLIKELY(!has_more<T>(elems)))
00455 return 0;
00456
00457 unsigned long size = elems * sizeof(T);
00458 _pos = cxx::Type_traits<T>::align(_pos);
00459
00460 buf.set(reinterpret_cast<T*>(_current_msg + _pos));
00461 _pos += size;
00462 return elems;
00463 }
00464
00465
00476 template< typename T >
00477 bool get(T &v)
00478 {
00479 if (L4_UNLIKELY(!has_more<T>()))
00480 {
00481 v = T();
00482 return false;
00483 }
00484
00485 _pos = cxx::Type_traits<T>::align(_pos);
00486 v = *(reinterpret_cast<T*>(_current_msg + _pos));
00487 _pos += sizeof(T);
00488 return true;
00489 }
00490
00491
00492 bool get(Ipc::Varg *va)
00493 {
00494 Ipc::Varg::Tag t;
00495 if (!has_more<Ipc::Varg::Tag>())
00496 {
00497 va->tag(0);
00498 return 0;
00499 }
00500 get(t);
00501 va->tag(t);
00502 char const *d;
00503 get(msg_ptr(d), va->length());
00504 va->data(d);
00505
00506 return 1;
00507 }
00508
00518 l4_msgtag_t tag() const { return _tag; }
00519
00520
00530 l4_msgtag_t &tag() { return _tag; }
00531
00533
00538 inline bool put(Buf_item const &);
00539
00544 inline bool put(Small_buf const &);
00545
00546
00551
00561 inline l4_msgtag_t wait(l4_umword_t *src)
00562 { return wait(src, L4_IPC_NEVER); }
00563

```

```

00574 inline l4_msgtag_t wait(l4_umword_t *src, l4_timeout_t timeout);
00575
00585 inline l4_msgtag_t receive(l4_cap_idx_t src)
00586 { return receive(src, L4_IPC_NEVER); }
00587 inline l4_msgtag_t receive(l4_cap_idx_t src, l4_timeout_t timeout);
00588
00590
00594 inline l4_utcb_t *utcb() const { return _utcb; }
00595
00596 protected:
00597 l4_msgtag_t _tag;
00598 l4_utcb_t *_utcb;
00599 char *_current_msg;
00600 unsigned _pos;
00601 unsigned char _current_buf;
00602 };
00603
00604 class Istream_copy : public Istream
00605 {
00606 private:
00607 l4_msg_regs_t _mrs;
00608
00609 public:
00610 Istream_copy(Istream const &o) : Istream(o), _mrs(*l4_utcb_mr_u(o.utcb()))
00611 {
00612 // do some reverse mr to utcb trickery
00613 _utcb = (l4_utcb_t *)((l4_addr_t)&_mrs - (l4_addr_t)l4_utcb_mr_u((l4_utcb_t *)0));
00614 _current_msg = reinterpret_cast<char*>(l4_utcb_mr_u(_utcb)->mr);
00615 }
00616
00617 };
00618
00634 class Ostream
00635 {
00636 public:
00640 Ostream(l4_utcb_t *utcb)
00641 : _tag(), _utcb(utcb),
00642 _current_msg(reinterpret_cast<char *>(l4_utcb_mr_u(_utcb)->mr)),
00643 _pos(0), _current_item(0)
00644 {}
00645
00649 void reset()
00650 {
00651 _pos = 0;
00652 _current_item = 0;
00653 _current_msg = reinterpret_cast<char*>(l4_utcb_mr_u(_utcb)->mr);
00654 }
00655
00663
00670 template< typename T >
00671 bool put(T *buf, unsigned long size)
00672 {
00673 size *= sizeof(T);
00674 _pos = cxx::Type_traits<T>::align(_pos);
00675 if (Utc_stream_check::check_utcb_data_offset(_pos + size))
00676 return false;
00677
00678 __builtin_memcpy(_current_msg + _pos, buf, size);
00679 _pos += size;
00680 return true;
00681 }
00682
00688 template< typename T >
00689 bool put(T const &v)
00690 {
00691 _pos = cxx::Type_traits<T>::align(_pos);
00692 if (Utc_stream_check::check_utcb_data_offset(_pos + sizeof(T)))
00693 return false;
00694
00695 *(reinterpret_cast<T*>(_current_msg + _pos)) = v;
00696 _pos += sizeof(T);
00697 return true;
00698 }
00699
00700 int put(Varg const &va)
00701 {
00702 put(va.tag());
00703 put(va.data(), va.length());
00704
00705 return 0;
00706 }
00707
00708 template< typename T >
00709 int put(Varg_t<T> const &va)
00710 { return put(static_cast<Varg const &>(va)); }
00711
00717 l4_msgtag_t tag() const { return _tag; }

```



```

00718
00724 l4_msgtag_t &tag() { return _tag; }
00725
00727
00732 inline bool put_snd_item(Snd_item const &);
00733
00734
00739
00749 inline l4_msgtag_t send(l4_cap_idx_t dst, long proto = 0, unsigned flags = 0);
00750
00752
00756 inline l4_utcb_t *utcb() const { return _utcb; }
00757 #if 0
00761 unsigned long tell() const
00762 {
00763 unsigned w = (_pos + sizeof(l4_umword_t)-1) / sizeof(l4_umword_t);
00764 w -= _current_item * 2;
00765 _tag = l4_msgtag(0, w, _current_item, 0);
00766 }
00767 #endif
00768 public:
00769 l4_msgtag_t prepare_ipc(long proto = 0, unsigned flags = 0)
00770 {
00771 unsigned w = (_pos + sizeof(l4_umword_t) - 1) / sizeof(l4_umword_t);
00772 w -= _current_item * 2;
00773 return l4_msgtag(proto, w, _current_item, flags);
00774 }
00775
00776 // XXX: this is a hack for <l4/sys/cxx/ipc_server> adaption
00777 void set_ipc_params(l4_msgtag_t tag)
00778 {
00779 _pos = (tag.words() + tag.items() * 2) * sizeof(l4_umword_t);
00780 _current_item = tag.items();
00781 }
00782 protected:
00783 l4_msgtag_t _tag;
00784 l4_utcb_t *_utcb;
00785 char *_current_msg;
00786 unsigned _pos;
00787 unsigned char _current_item;
00788 };
00789
00790
00802 class Iostream : public Istream, public Ostream
00803 {
00804 public:
00805
00814 explicit Iostream(l4_utcb_t *utcb)
00815 : Istream(utcb), Ostream(utcb)
00816 {}
00817
00818 // disambiguate those functions
00819 l4_msgtag_t tag() const { return Istream::tag(); }
00820 l4_msgtag_t &tag() { return Istream::tag(); }
00821 l4_utcb_t *utcb() const { return Istream::utcb(); }
00822
00828 void reset()
00829 {
00830 Istream::reset();
00831 Ostream::reset();
00832 }
00833
00834
00842
00843 using Istream::get;
00844 using Istream::put;
00845 using Ostream::put;
00846
00848
00853
00869 inline l4_msgtag_t call(l4_cap_idx_t dst, l4_timeout_t timeout, long proto = 0);
00870 inline l4_msgtag_t call(l4_cap_idx_t dst, long proto = 0);
00871
00887 inline l4_msgtag_t reply_and_wait(l4_umword_t *src_dst, long proto = 0)
00888 { return reply_and_wait(src_dst, L4_IPC_SEND_TIMEOUT_0, proto); }
00889
00890 inline l4_msgtag_t send_and_wait(l4_cap_idx_t dest, l4_umword_t *src,
00891 long proto = 0)
00892 { return send_and_wait(dest, src, L4_IPC_SEND_TIMEOUT_0, proto); }
00893
00910 inline l4_msgtag_t reply_and_wait(l4_umword_t *src_dst,
00911 l4_timeout_t timeout, long proto = 0);
00912 inline l4_msgtag_t send_and_wait(l4_cap_idx_t dest, l4_umword_t *src,
00913 l4_timeout_t timeout, long proto = 0);
00914 inline l4_msgtag_t reply(l4_timeout_t timeout, long proto = 0);
00915 inline l4_msgtag_t reply(long proto = 0)
00916 { return reply(L4_IPC_SEND_TIMEOUT_0, proto); }

```

```

00917
00918 };
00919
00920
00921
00922 inline bool
00923 Ostream::put_snd_item(Snd_item const &v)
00924 {
00925 typedef Snd_item T;
00926 _pos = cxx::Type_traits<Snd_item>::align(_pos);
00927 if (Utc_b_stream_check::check_utcb_data_offset(_pos + sizeof(T)))
00928 return false;
00929
00930 *(reinterpret_cast<T*>(_current_msg + _pos)) = v;
00931 _pos += sizeof(T);
00932 ++_current_item;
00933 return true;
00934 }
00935
00936
00937 inline bool
00938 Istream::put(Buf_item const &item)
00939 {
00940 if (_current_buf >= L4_UTCB_GENERIC_BUFFERS_SIZE - 3)
00941 return false;
00942
00943 l4_utcb_br_u(_utcb)->bdr &= ~L4_BDR_OFFSET_MASK;
00944
00945 reinterpret_cast<Buf_item&>(l4_utcb_br_u(_utcb)->br[_current_buf]) = item;
00946 _current_buf += 2;
00947 return true;
00948 }
00949
00950
00951 inline bool
00952 Istream::put(Small_buf const &item)
00953 {
00954 if (_current_buf >= L4_UTCB_GENERIC_BUFFERS_SIZE - 2)
00955 return false;
00956
00957 l4_utcb_br_u(_utcb)->bdr &= ~L4_BDR_OFFSET_MASK;
00958
00959 reinterpret_cast<Small_buf&>(l4_utcb_br_u(_utcb)->br[_current_buf]) = item;
00960 _current_buf += 1;
00961 return true;
00962 }
00963
00964
00965 inline l4_msgtag_t
00966 Ostream::send(l4_cap_idx_t dst, long proto, unsigned flags)
00967 {
00968 l4_msgtag_t tag = prepare_ipc(proto, L4_MSGTAG_FLAGS & flags);
00969 return l4_ipc_send(dst, _utcb, tag, L4_IPC_NEVER);
00970 }
00971
00972 inline l4_msgtag_t
00973 Iostream::call(l4_cap_idx_t dst, l4_timeout_t timeout, long label)
00974 {
00975 l4_msgtag_t tag = prepare_ipc(label);
00976 tag = l4_ipc_call(dst, Ostream::_utcb, tag, timeout);
00977 Istream::tag() = tag;
00978 Istream::_pos = 0;
00979 return tag;
00980 }
00981
00982 inline l4_msgtag_t
00983 Iostream::call(l4_cap_idx_t dst, long label)
00984 { return call(dst, L4_IPC_NEVER, label); }
00985
00986
00987 inline l4_msgtag_t
00988 Iostream::reply_and_wait(l4_umword_t *src_dst, l4_timeout_t timeout, long proto)
00989 {
00990 l4_msgtag_t tag = prepare_ipc(proto);
00991 tag = l4_ipc_reply_and_wait(Ostream::_utcb, tag, src_dst, timeout);
00992 Istream::tag() = tag;
00993 Istream::_pos = 0;
00994 return tag;
00995 }
00996
00997
00998 inline l4_msgtag_t
00999 Iostream::send_and_wait(l4_cap_idx_t dest, l4_umword_t *src,
01000 l4_timeout_t timeout, long proto)
01001 {
01002 l4_msgtag_t tag = prepare_ipc(proto);
01003 tag = l4_ipc_send_and_wait(dest, Ostream::_utcb, tag, src, timeout);
01004 Istream::tag() = tag;

```

```

01005 Istream::_pos = 0;
01006 return tag;
01007 }
01008
01009 inline l4_msgtag_t
01010 Iostream::reply(l4_timeout_t timeout, long proto)
01011 {
01012 l4_msgtag_t tag = prepare_ipc(proto);
01013 tag = l4_ipc_send(L4_INVALID_CAP | L4_SYSF_REPLY, Ostream::_utcb, tag, timeout);
01014 Istream::tag() = tag;
01015 Istream::_pos = 0;
01016 return tag;
01017 }
01018
01019 inline l4_msgtag_t
01020 Istream::wait(l4_umword_t *src, l4_timeout_t timeout)
01021 {
01022 l4_msgtag_t res;
01023 res = l4_ipc_wait(_utcb, src, timeout);
01024 tag() = res;
01025 _pos = 0;
01026 return res;
01027 }
01028
01029
01030 inline l4_msgtag_t
01031 Istream::receive(l4_cap_idx_t src, l4_timeout_t timeout)
01032 {
01033 l4_msgtag_t res;
01034 res = l4_ipc_receive(src, _utcb, timeout);
01035 tag() = res;
01036 _pos = 0;
01037 return res;
01038 }
01039
01040 } // namespace Ipc
01041 } // namespace L4
01042
01051 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, bool &v) { s.get(v); return s; }
01052 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, int &v) { s.get(v); return s; }
01053 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, long int &v) { s.get(v); return s; }
01054 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, long long int &v) { s.get(v); return s; }
01055 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned int &v) { s.get(v); return s; }
01056 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned long int &v) { s.get(v); return s; }
01057 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned long long int &v) { s.get(v);
 return s; }
01058 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, short int &v) { s.get(v); return s; }
01059 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned short int &v) { s.get(v); return s; }
01060 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, char &v) { s.get(v); return s; }
01061 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned char &v) { s.get(v); return s; }
01062 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, signed char &v) { s.get(v); return s; }
01063 inline L4::Ipc::Istream &operator « (L4::Ipc::Istream &s, L4::Ipc::Buf_item const &v) { s.put(v);
 return s; }
01064 inline L4::Ipc::Istream &operator « (L4::Ipc::Istream &s, L4::Ipc::Small_buf const &v) { s.put(v);
 return s; }
01065 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, L4::Ipc::Snd_item &v)
01066 {
01067 l4_umword_t b, d;
01068 s » b » d;
01069 v = L4::Ipc::Snd_item(b, d);
01070 return s;
01071 }
01072 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, L4::Ipc::Varg &v)
01073 { s.get(&v); return s; }
01074
01075
01084 inline
01085 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, l4_msgtag_t &v)
01086 {
01087 v = s.tag();
01088 return s;
01089 }
01090
01108 template< typename T >
01109 inline
01110 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s,
 L4::Ipc::Internal::Buf_in<T> const &v)
01111 {
01112 unsigned long si;
01113 if (s.get(si) && s.has_more<T>(si))
01114 v.set_size(s.get(L4::Ipc::Msg_ptr<T>(v.buf()), si));
01115 else
01116 v.set_size(0);
01117 return s;
01118 }
01119 }

```

```

01120
01135 template< typename T >
01136 inline
01137 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s,
01138 L4::Ipc::Msg_ptr<T> const &v)
01139 {
01140 s.get(v);
01141 return s;
01142 }
01143
01156 template< typename T >
01157 inline
01158 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s,
01159 L4::Ipc::Internal::Buf_cp_in<T> const &v)
01160 {
01161 unsigned long sz;
01162 s.get(sz);
01163 v.size() = s.get(v.buf(), cxx::min(v.size(), sz));
01164 return s;
01165 }
01166
01177 template< typename T >
01178 inline
01179 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s,
01180 L4::Ipc::Str_cp_in<T> const &v)
01181 {
01182 unsigned long sz;
01183 s.get(sz);
01184 unsigned long rsz = s.get(v.buf(), cxx::min(v.size(), sz));
01185 if (rsz < v.size() && v.buf()[rsz - 1])
01186 ++rsz; // add the zero termination behind the received data
01187 if (rsz != 0)
01188 v.buf()[rsz - 1] = 0;
01189 v.size() = rsz;
01190 return s;
01191 }
01192
01193 }
01194
01195
01204 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, bool v) { s.put(v); return s; }
01205 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, int v) { s.put(v); return s; }
01206 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, long int v) { s.put(v); return s; }
01207 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, long long int v) { s.put(v); return s; }
01208 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned int v) { s.put(v); return s; }
01209 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned long int v) { s.put(v); return s; }
01210 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned long long int v) { s.put(v); return
 s; }
01211 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, short int v) { s.put(v); return s; }
01212 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned short int v) { s.put(v); return s;
 }
01213 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, char v) { s.put(v); return s; }
01214 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned char v) { s.put(v); return s; }
01215 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, signed char v) { s.put(v); return s; }
01216 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, L4::Ipc::Snd_item const &v) {
 s.put_snd_item(v); return s; }
01217 template< typename T >
01218 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, L4::Cap<T> const &v)
01219 { s « L4::Ipc::Snd_fpage(v.fpage()); return s; }
01220
01221 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, L4::Ipc::Varg const &v)
01222 { s.put(v); return s; }
01223 template< typename T >
01224 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, L4::Ipc::Varg_t<T> const &v)
01225 { s.put(v); return s; }
01226
01238 inline
01239 L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, l4_msgtag_t const &v)
01240 {
01241 s.tag() = v;
01242 return s;
01243 }
01244
01253 template< typename T >
01254 inline
01255 L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s,
01256 L4::Ipc::Internal::Buf_cp_out<T> const &v)
01257 {
01258 s.put(v.size());
01259 s.put(v.buf(), v.size());
01260 return s;
01261 }
01262
01276 inline
01277 L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, char const *v)
01278 {
 unsigned long l = __builtin_strlen(v) + 1;

```

```

01279 s.put(1);
01280 s.put(v, 1);
01281 return s;
01282 }
01283
01284
01285 #ifdef L4_CXX_IPC_BACKWARD_COMPAT
01286 namespace L4 {
01287
01288 #if 0
01289 template< typename T > class Ipc_buf_cp_out : public Ipc::Buf_cp_out<T> {};
01290 template< typename T > class Ipc_buf_cp_in : public Ipc::Buf_cp_in<T> {};
01291 template< typename T > class Ipc_buf_in : public Ipc::Buf_in<T> {};
01292 template< typename T > class Msg_ptr : public Ipc::Msg_ptr<T> {};
01293 #endif
01294
01295 template< typename T >
01296 Ipc::Internal::Buf_cp_out<T> ipc_buf_cp_out(T *v, unsigned long size)
01297 L4_DEPRECATED("Use L4::Ipc::buf_cp_out() now");
01298
01299 template< typename T >
01300 Ipc::Internal::Buf_cp_out<T> ipc_buf_cp_out(T *v, unsigned long size)
01301 { return Ipc::Internal::Buf_cp_out<T>(v, size); }
01302
01303
01304 template< typename T >
01305 Ipc::Internal::Buf_cp_in<T> ipc_buf_cp_in(T *v, unsigned long &size)
01306 L4_DEPRECATED("Use L4::Ipc::buf_cp_in() now");
01307
01308 template< typename T >
01309 Ipc::Internal::Buf_cp_in<T> ipc_buf_cp_in(T *v, unsigned long &size)
01310 { return Ipc::Internal::Buf_cp_in<T>(v, size); }
01311
01312
01313 template< typename T >
01314 Ipc::Internal::Buf_in<T> ipc_buf_in(T *v, unsigned long &size)
01315 L4_DEPRECATED("Use L4::Ipc::buf_in() now");
01316
01317 template< typename T >
01318 Ipc::Internal::Buf_in<T> ipc_buf_in(T *v, unsigned long &size)
01319 { return Ipc::Internal::Buf_in<T>(v, size); }
01320
01321
01322 template< typename T >
01323 Ipc::Msg_ptr<T> msg_ptr(T *p)
01324 L4_DEPRECATED("Use L4::Ipc::msg_ptr() now");
01325
01326 template< typename T >
01327 Ipc::Msg_ptr<T> msg_ptr(T *p)
01328 { return Ipc::Msg_ptr<T>(p); }
01329
01330 typedef Ipc::Istream Ipc_istream L4_DEPRECATED("Use L4::Ipc::Istream now");
01331 typedef Ipc::Ostream Ipc_ostream L4_DEPRECATED("Use L4::Ipc::Ostream now");
01332 typedef Ipc::Iostream Ipc_iostream L4_DEPRECATED("Use L4::Ipc::Iostream now");
01333 typedef Ipc::Snd_fpage Snd_fpage L4_DEPRECATED("Use L4::Ipc::Snd_fpage now");
01334 typedef Ipc::Rcv_fpage Rcv_fpage L4_DEPRECATED("Use L4::Ipc::Rcv_fpage now");
01335 typedef Ipc::Small_buf Small_buf L4_DEPRECATED("Use L4::Ipc::Small_buf now");
01336
01337
01338 namespace Ipc {
01339 template< typename T > class Buf_cp_in : public Internal::Buf_cp_in<T>
01340 {
01341 public:
01342 Buf_cp_in(T *v, unsigned long &size) : Internal::Buf_cp_in<T>(v, size) {}
01343 };
01344
01345 template< typename T >
01346 class Buf_cp_out : public Internal::Buf_cp_out<T>
01347 {
01348 public:
01349 Buf_cp_out(T const *v, unsigned long size) : Internal::Buf_cp_out<T>(v, size) {}
01350 };
01351
01352 template< typename T >
01353 class Buf_in : public Internal::Buf_in<T>
01354 {
01355 public:
01356 Buf_in(T *v, unsigned long &size) : Internal::Buf_in<T>(v, size) {}
01357 };
01358 } // namespace Ipc
01359 } // namespace L4
01360
01361 template< typename T >
01362 inline
01363 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, L4::Ipc::Buf_cp_in<T> const &v)
01364 L4_DEPRECATED("Use L4::Ipc::buf_cp_in() now");
01365

```

```

01366 template< typename T >
01367 inline
01368 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, L4::Ipc::Buf_cp_in<T> const &v)
01369 { return operator»(s, static_cast<L4::Ipc::Internal::Buf_cp_in<T> >(v)); }
01370
01371 template< typename T >
01372 inline
01373 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, L4::Ipc::Buf_in<T> const &v)
01374 { L4_DEPRECATED("Use L4::Ipc::buf_in() now");
01375
01376 template< typename T >
01377 inline
01378 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, L4::Ipc::Buf_in<T> const &v)
01379 { return operator»(s, static_cast<L4::Ipc::Internal::Buf_in<T> >(v)); }
01380
01381 template< typename T >
01382 inline
01383 L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, L4::Ipc::Buf_cp_out<T> const &v)
01384 { L4_DEPRECATED("Use L4::Ipc::buf_cp_out() now");
01385
01386 template< typename T >
01387 inline
01388 L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, L4::Ipc::Buf_cp_out<T> const &v)
01389 { return operator«(s, static_cast<L4::Ipc::Internal::Buf_cp_out<T> >(v)); }
01390 #endif
01391
01392 namespace L4 { namespace Ipc {
01402 template< typename T >
01403 inline
01404 T read(Istream &s) { T t; s » t; return t; }
01405
01406 } // namespace Ipc
01407 } // namespace L4

```

## 16.107 l4/cxx/l4iostream File Reference

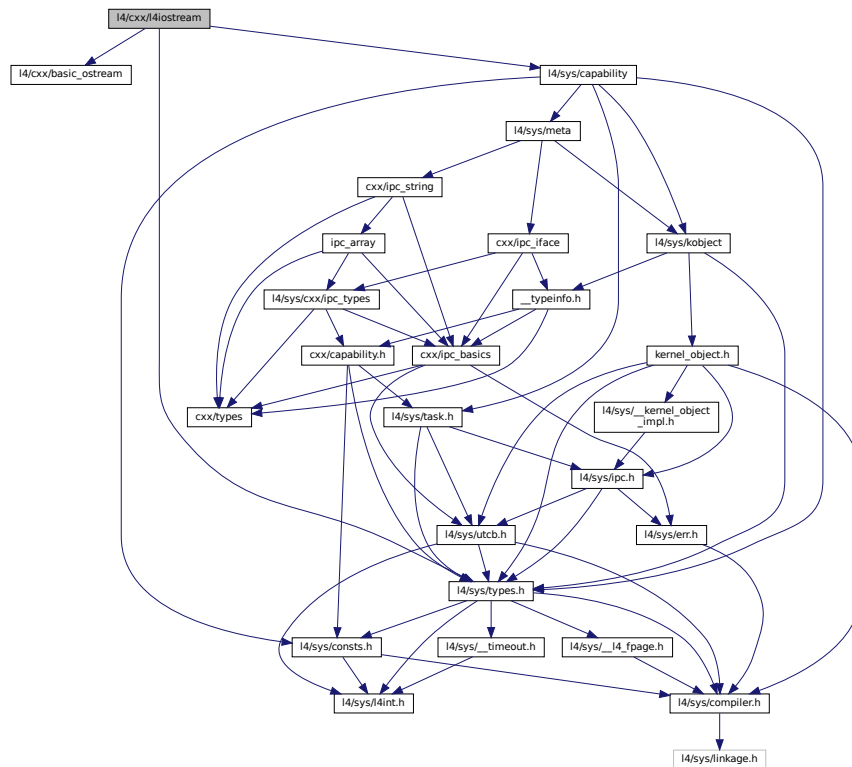
L4 IO stream.

```

#include <l4/cxx/basic_ostream>
#include <l4/sys/types.h>
#include <l4/sys/capability>

```

Include dependency graph for l4iostream:



### 16.107.1 Detailed Description

[L4](#) IO stream.

Definition in file [l4iostream](#).

## 16.108 l4iostream

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004 * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00005 * economic rights: Technische Universität Dresden (Germany)
00006 *
00007 * This file is part of TUD:OS and distributed under the terms of the
00008 * GNU General Public License 2.
00009 * Please see the COPYING-GPL-2 file for details.
00010 *
00011 * As a special exception, you may use this file as part of a free software
00012 * library without restriction. Specifically, if other files instantiate
00013 * templates or use macros or inline functions from this file, or you compile
00014 * this file and link it with other files to produce an executable, this
00015 * file does not by itself cause the resulting executable to be covered by
00016 * the GNU General Public License. This exception does not however
00017 * invalidate any other reasons why the executable file might be covered by
00018 * the GNU General Public License.
00019 */
00020 #pragma once
00021 #include <l4/cxx/basic_ostream>
00022 #include <l4/sys/types.h>
00023 #include <l4/sys/capability>
00024

```

```

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 }

```

## 16.109 l4/cxx/l4types.h File Reference

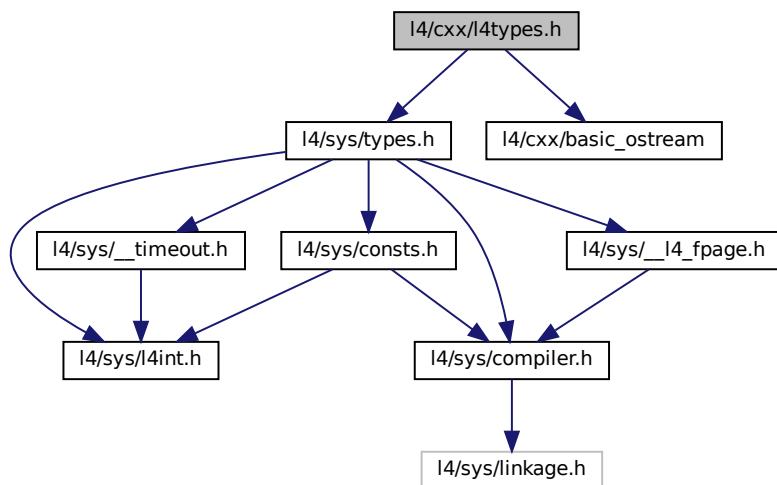
[L4](#) Types.

```

#include <l4/sys/types.h>
#include <l4/cxx/basic_ostream>

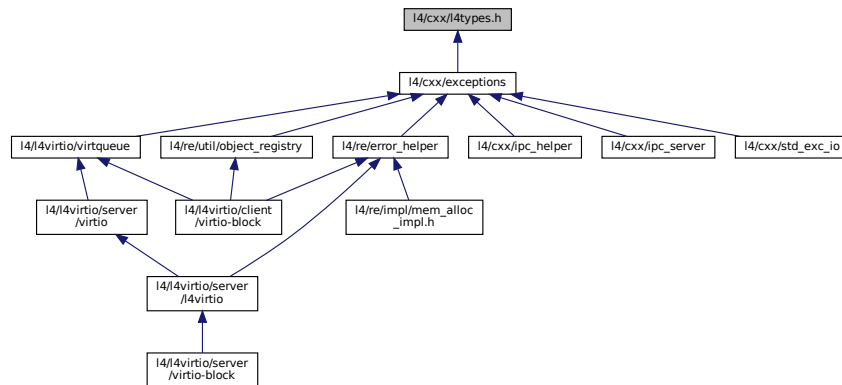
```

Include dependency graph for l4types.h:





This graph shows which files directly or indirectly include this file:



## 16.109.1 Detailed Description

[L4](#) Types.

Definition in file [l4types.h](#).

## 16.110 l4types.h

```

00001
00005 /*
00006 * (c) 2004-2009 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 #include <l4/cxx/basic_ostream>

```

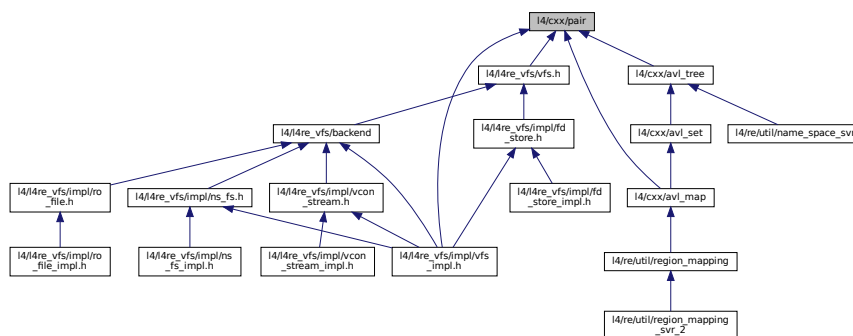
## 16.111 l4/cxx/main\_thread File Reference

Main thread.



### 16.113 I4/cxx/pair File Reference

This graph shows which files directly or indirectly include this file:



- struct `cxx::Pair< First, Second >`  
*Pair of two values.*
- class `cxx::Pair_first_compare< Cmp, Typ >`  
*Comparison functor for `Pair`.*

- CXX

*Our C++ library.*

### 16.113.1 Detailed Description

Pair implementation.

Definition in file [pair](#).

## 16.114 pair

```

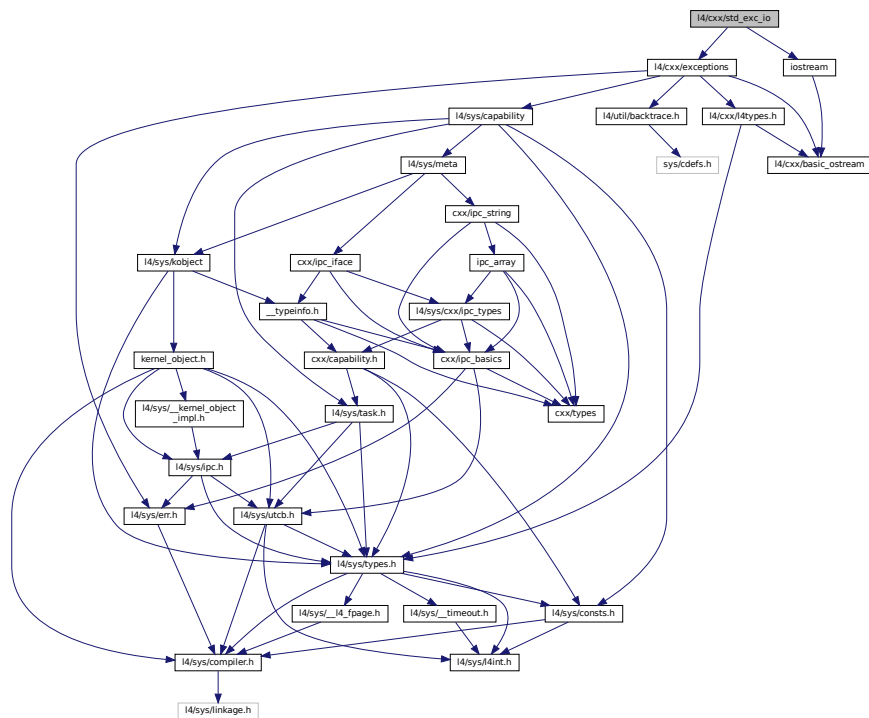
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
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

```

## 16.115 I4/cxx/std\_exc\_io File Reference

Base exceptions std stream operator.

```
#include <l4/cxx/exceptions>
#include <iostream>
Include dependency graph for std_exc_io:
```



### 16.115.1 Detailed Description

Base exceptions std stream operator.

Definition in file `std_exc_io`.

## 16.116 std\_exc\_io

```
00001 // vi:set ft=cpp: -- Mode: C++ --
00002 /*
00003 * (c) 2011 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004 * Alexander Warg <warg@os.inf.tu-dresden.de>
00005 * economic rights: Technische Universität Dresden (Germany)
00006 *
00007 * This file is part of TUD:OS and distributed under the terms of the
00008 * GNU General Public License 2.
00009 * Please see the COPYING-GPL-2 file for details.
00010 *
00011 * As a special exception, you may use this file as part of a free software
00012 * library without restriction. Specifically, if other files instantiate
00013 * templates or use macros or inline functions from this file, or you compile
00014 * this file and link it with other files to produce an executable, this
00015 * file does not by itself cause the resulting executable to be covered by
00016 * the GNU General Public License. This exception does not however
00017 * invalidate any other reasons why the executable file might be covered by
00018 * the GNU General Public License.
00019 */
00020 #pragma once
00021 #include <14/cxx/exceptions>
00022 #include <iostream>
```

```

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 }

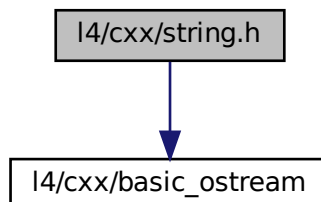
```

## 16.117 l4/cxx/string.h File Reference

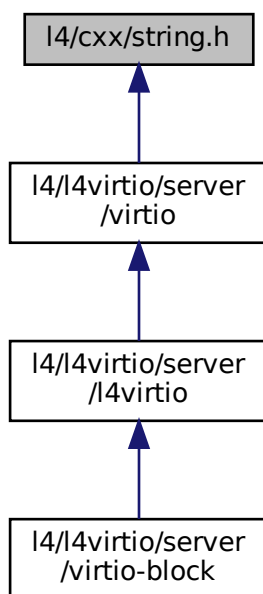
String.

```
#include <l4/cxx/basic_ostream>
```

Include dependency graph for string.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4::String](#)  
*A null-terminated string container class.*

## Namespaces

- [L4](#)  
*L4 low-level kernel interface.*

### 16.117.1 Detailed Description

String.

Definition in file [string.h](#).

## 16.118 string.h

```

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 }

```

## 16.119 l4/irq/irq.h File Reference

IRQ handling routines.

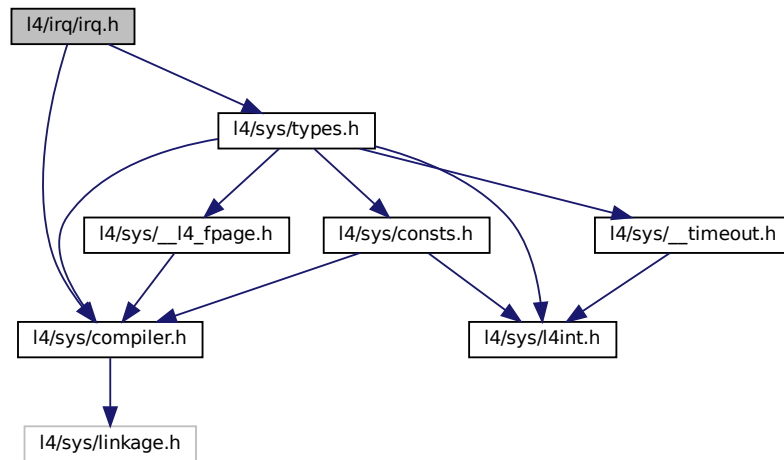
```

#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.*

### 16.119.1 Detailed Description

IRQ handling routines.

Definition in file [irq.h](#).

## 16.120 irq.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Henning Schild <hschild@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 */
00014 #pragma once
00015
00016 #include <l4/sys/compiler.h>
00017 #include <l4/sys/types.h>
00018
00019 __BEGIN_DECLS
00020
00028 struct l4irq_t;
00029 typedef struct l4irq_t l4irq_t;
00030
00041 L4_CV l4irq_t *
00042 l4irq_attach(int irqnum);
00043
00055 L4_CV l4irq_t *
00056 l4irq_attach_ft(int irqnum, unsigned mode);
00057
00068 L4_CV l4irq_t *
00069 l4irq_attach_thread(int irqnum, l4_cap_idx_t to_thread);
00070
00082 L4_CV l4irq_t *
00083 l4irq_attach_thread_ft(int irqnum, l4_cap_idx_t to_thread,
00084 unsigned mode);
00085
00093 L4_CV long
00094 l4irq_wait(l4irq_t *irq);
00095
00104 L4_CV long
00105 l4irq_unmask_and_wait_any(l4irq_t *unmask_irq, l4irq_t **ret_irq);
00106
00114 L4_CV long
00115 l4irq_wait_any(l4irq_t **irq);
00116
00127 L4_CV long
00128 l4irq_unmask(l4irq_t *irq);
00129
00137 L4_CV long
00138 l4irq_detach(l4irq_t *irq);
00139
00140
00141
00142 /*****
00164 L4_CV l4irq_t *
00165 l4irq_request(int irqnum, void (*isr_handler)(void *), void *isr_data,
00166 int irq_thread_prio, unsigned mode);
00167
00175 L4_CV long
00176 l4irq_release(l4irq_t *irq);
00177
00178

```

```

00179
00180 /*****
00181 /*****
00182
00198 L4_CV l4irq_t *
00199 l4irq_attach_cap(l4_cap_idx_t irqcap);
00200
00212 L4_CV l4irq_t *
00213 l4irq_attach_cap_ft(l4_cap_idx_t irqcap, unsigned mode);
00214
00225 L4_CV l4irq_t *
00226 l4irq_attach_thread_cap(l4_cap_idx_t irqcap, l4_cap_idx_t to_thread);
00227
00239 L4_CV l4irq_t *
00240 l4irq_attach_thread_cap_ft(l4_cap_idx_t irqcap, l4_cap_idx_t to_thread,
00241 unsigned mode);
00242
00243 /*****
00264 L4_CV l4irq_t *
00265 l4irq_request_cap(l4_cap_idx_t irqcap,
00266 void (*isr_handler)(void *), void *isr_data,
00267 int irq_thread_prio, unsigned mode);
00268
00269 __END_DECLS

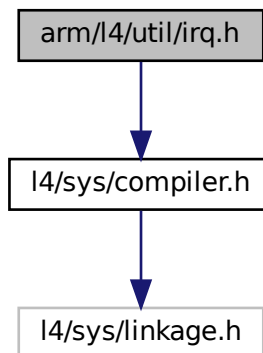
```

## 16.121 arm/l4/util/irq.h File Reference

ARM specific implementation of irq functions.

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

Include dependency graph for irq.h:



### 16.121.1 Detailed Description

ARM specific implementation of irq functions.

Do not use.

Definition in file [irq.h](#).

## 16.122 irq.h

```

00001
00007 /*
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00010 * Frank Mehnert <fm3@os.inf.tu-dresden.de>
00011 * economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016 #ifndef __L4UTIL__ARCH_ARCH__IRQ_H__
00017 #define __L4UTIL__ARCH_ARCH__IRQ_H__
00018
00019 #ifdef __GNUC__
00020
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00025 L4_INLINE void l4util_cli (void);
00026 L4_INLINE void l4util_sti (void);
00027 L4_INLINE void l4util_flags_save(l4_umword_t *flags);
00028 L4_INLINE void l4util_flags_restore(l4_umword_t *flags);
00029
00030 L4_INLINE
00031 void
00032 l4util_cli(void)
00033 {
00034 extern void __do_not_use_l4util_cli(void);
00035 __do_not_use_l4util_cli();
00036 }
00037
00038
00039 L4_INLINE
00040 void
00041 l4util_sti(void)
00042 {
00043 extern void __do_not_use_l4util_sti(void);
00044 __do_not_use_l4util_sti();
00045 }
00046
00047
00048 L4_INLINE
00049 void
00050 l4util_flags_save(l4_umword_t *flags)
00051 {
00052 (void) flags;
00053 extern void __do_not_use_l4util_flags_save(void);
00054 __do_not_use_l4util_flags_save();
00055 }
00056
00057 L4_INLINE
00058 void
00059 l4util_flags_restore(l4_umword_t *flags)
00060 {
00061 (void) flags;
00062 extern void __do_not_use_l4util_flags_restore(void);
00063 __do_not_use_l4util_flags_restore();
00064 }
00065
00066 EXTERN_C_END
00067
00068 #endif //__GNUC__
00069
00070 #endif /* ! __L4UTIL__ARCH_ARCH__IRQ_H__ */

```

## 16.123 amd64/l4/util/irq.h File Reference

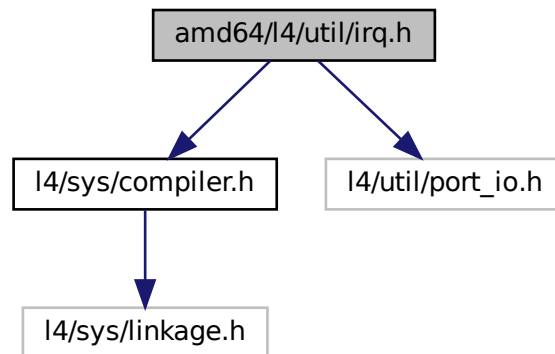
some PIC and hardware interrupt related functions

```

#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.*

### 16.123.1 Detailed Description

some PIC and hardware interrupt related functions

#### Date

2003

#### Author

Jork Loeser [jork.loeser@inf.tu-dresden.de](mailto:jork.loeser@inf.tu-dresden.de) Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [irq.h](#).

### 16.123.2 Function Documentation

#### 16.123.2.1 l4util\_irq\_acknowledge()

```
void l4util_irq_acknowledge (
 unsigned int irq) [inline]
```

Acknowledge IRQ at PIC in fully special nested mode.

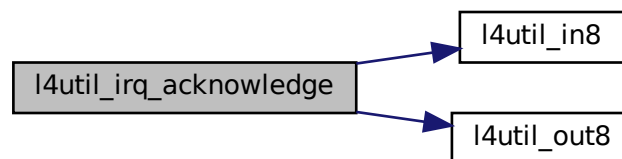
## Parameters

|            |                                    |
|------------|------------------------------------|
| <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:



## 16.124 irq.h

```

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

```

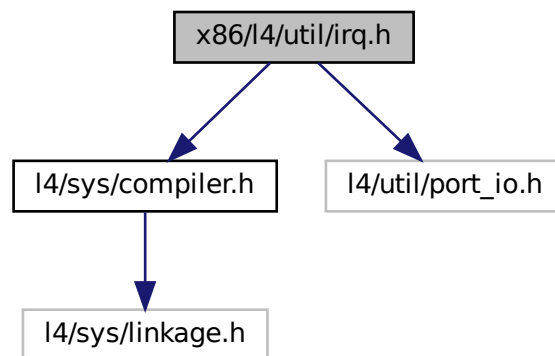
## 16.125 x86/I4/util/irq.h File Reference

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.*

### 16.125.1 Detailed Description

some PIC and hardware interrupt related functions

Date

2003

Author

Jork Loeser [jork.loeser@inf.tu-dresden.de](mailto:jork.loeser@inf.tu-dresden.de) Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file `irq.h`.

## 16.125.2 Function Documentation

### 16.125.2.1 l4util\_irq\_acknowledge()

```
void l4util_irq_acknowledge (
 unsigned int irq) [inline]
```

Acknowledge IRQ at PIC in fully special nested mode.

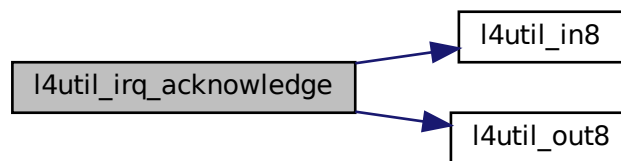
#### Parameters

|            |                                    |
|------------|------------------------------------|
| <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:



## 16.126 irq.h

```
00001
00010 /*
00011 * (c) 2003-2009 Author(s)
00012 * economic rights: Technische Universität Dresden (Germany)
00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU Lesser General Public License 2.1.
00015 * Please see the COPYING-LGPL-2.1 file for details.
00016 */
00017
00018 #ifndef __L4_IRQ_H__
00019 #define __L4_IRQ_H__
00020
00021 #include <l4/sys/compiler.h>
00022 #include <l4/util/port_io.h>
00023
00024 EXTERN_C_BEGIN
00025
00029 L4_INLINE void
00030 l4util_irq_acknowledge(unsigned int irq);
00031
00034 static inline void
00035 l4util_cli (void)
00036 {
00037 __asm__ __volatile__ ("cli" : : : "memory");
```



```

00038 }
00039
00042 static inline void
00043 l4util_sti (void)
00044 {
00045 __asm__ __volatile__ ("sti" : : : "memory");
00046 }
00047
00051 static inline void
00052 l4util_flags_save (l4_umword_t *flags)
00053 {
00054 __asm__ __volatile__ ("pushfl ; popl %0 " : "=g" (*flags) : : "memory");
00055 }
00056
00059 static inline void
00060 l4util_flags_restore (l4_umword_t *flags)
00061 {
00062 __asm__ __volatile__ ("pushl %0 ; popfl" : : "g" (*flags) : "memory");
00063 }
00064
00065 L4_INLINE void
00066 l4util_irq_acknowledge(unsigned int irq)
00067 {
00068 if (irq > 7)
00069 {
00070 l4util_out8(0x60+(irq & 7), 0xA0);
00071 l4util_out8(0x0B, 0xA0);
00072 if (l4util_in8(0xA0) == 0)
00073 l4util_out8(0x60 + 2, 0x20);
00074 }
00075 else
00076 l4util_out8(0x60+irq, 0x20); /* acknowledge the irq */
00077 };
00078
00079 EXTERN_C_END
00080
00081 #endif

```

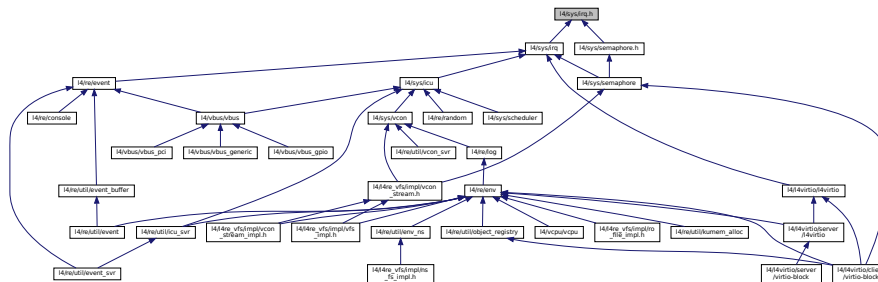
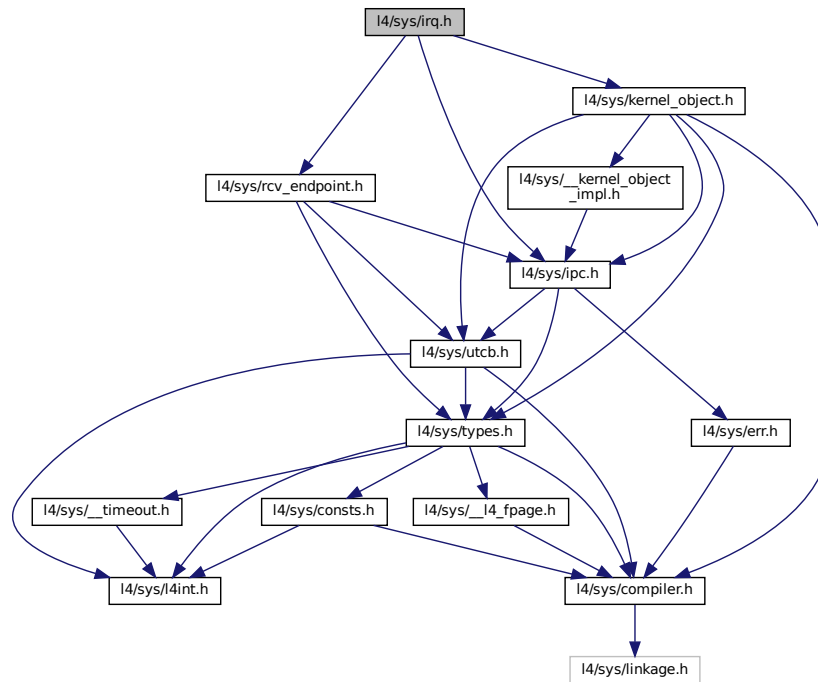
## 16.127 l4/sys/irq.h File Reference

C Irq interface.

```

#include <l4/sys/kernel_object.h>
#include <l4/sys/ipc.h>
#include <l4/sys/rcv_endpoint.h>

```



- `l4_msgtag_t l4_irq_mux_chain (l4_cap_idx_t irq, l4_cap_idx_t slave) L4_NOTHROW`

*Attach an IRQ to this multiplexer.*

- `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.*

- `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.*

## 16.127.1 Detailed Description

C irq interface.

Definition in file [irq.h](#).

## 16.128 irq.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 * 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
00071 L4_INLINE l4_msgtag_t
00072 l4_irq_mux_chain(l4_cap_idx_t irq, l4_cap_idx_t slave) L4_NOTHROW;
00073
00080 L4_INLINE l4_msgtag_t
00081 l4_irq_mux_chain_u(l4_cap_idx_t irq, l4_cap_idx_t slave,
00082 l4_utcb_t *utcb) L4_NOTHROW;
00083
00092 L4_INLINE l4_msgtag_t
00093 l4_irq_detach(l4_cap_idx_t irq) L4_NOTHROW;
00094
00101 L4_INLINE l4_msgtag_t
00102 l4_irq_detach_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW;
00103
00104
00120 L4_INLINE l4_msgtag_t
00121 l4_irq_trigger(l4_cap_idx_t irq) L4_NOTHROW;
```

```

00122
00129 L4_INLINE l4_msgtag_t
00130 l4_irq_trigger_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW;
00131
00141 L4_INLINE l4_msgtag_t
00142 l4_irq_receive(l4_cap_idx_t irq, l4_timeout_t to) L4_NOTHROW;
00143
00150 L4_INLINE l4_msgtag_t
00151 l4_irq_receive_u(l4_cap_idx_t irq, l4_timeout_t timeout, l4_utcb_t *utcb) L4_NOTHROW;
00152
00163 L4_INLINE l4_msgtag_t
00164 l4_irq_wait(l4_cap_idx_t irq, l4_umword_t *label,
00165 l4_timeout_t to) L4_NOTHROW;
00166
00173 L4_INLINE l4_msgtag_t
00174 l4_irq_wait_u(l4_cap_idx_t irq, l4_umword_t *label,
00175 l4_timeout_t timeout, l4_utcb_t *utcb) L4_NOTHROW;
00176
00187 L4_INLINE l4_msgtag_t
00188 l4_irq_unmask(l4_cap_idx_t irq) L4_NOTHROW;
00189
00196 L4_INLINE l4_msgtag_t
00197 l4_irq_unmask_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW;
00198
00202 enum l4_irq_sender_op
00203 {
00204 L4_IRQ_SENDER_OP_RESERVED1 = 0, // Ex ATTACH
00205 L4_IRQ_SENDER_OP_DETACH = 1
00206 };
00207
00211 enum l4_irq_mux_op
00212 {
00213 L4_IRQ_MUX_OP_CHAIN = 0
00214 };
00215
00219 enum l4_irq_op
00220 {
00221 L4_IRQ_OP_TRIGGER = 2,
00222 L4_IRQ_OP_EOI = 4
00223 };
00224
00225 /*****
00226 * Implementations
00227 */
00228
00229 L4_INLINE l4_msgtag_t
00230 l4_irq_mux_chain_u(l4_cap_idx_t irq, l4_cap_idx_t slave,
00231 l4_utcb_t *utcb) L4_NOTHROW
00232 {
00233 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00234 m->mr[0] = L4_IRQ_MUX_OP_CHAIN;
00235 m->mr[1] = l4_map_obj_control(0, 0);
00236 m->mr[2] = l4_obj_fpage(slave, 0, L4_CAP_FPAGE_RWS).raw;
00237 return l4_ipc_call(irq, utcb, l4_msgtag(L4_PROTO_IRQ_MUX, 1, 1, 0),
00238 L4_IPC_NEVER);
00239 }
00240
00241 L4_INLINE l4_msgtag_t
00242 l4_irq_detach_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW
00243 {
00244 l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_SENDER_OP_DETACH;
00245 return l4_ipc_call(irq, utcb, l4_msgtag(L4_PROTO_IRQ_SENDER, 1, 0, 0),
00246 L4_IPC_NEVER);
00247 }
00248
00249 L4_INLINE l4_msgtag_t
00250 l4_irq_trigger_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW
00251 {
00252 return l4_ipc_send(irq, utcb, l4_msgtag(L4_PROTO_IRQ, 0, 0, 0),
00253 L4_IPC_BOTH_TIMEOUT_0);
00254 }
00255
00256 L4_INLINE l4_msgtag_t
00257 l4_irq_receive_u(l4_cap_idx_t irq, l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW
00258 {
00259 l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_OP_EOI;
00260 return l4_ipc_call(irq, utcb, l4_msgtag(L4_PROTO_IRQ, 1, 0, 0), to);
00261 }
00262
00263 L4_INLINE l4_msgtag_t
00264 l4_irq_wait_u(l4_cap_idx_t irq, l4_umword_t *label,
00265 l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW
00266 {
00267 l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_OP_EOI;
00268 return l4_ipc_send_and_wait(irq, utcb, l4_msgtag(L4_PROTO_IRQ, 1, 0, 0),
00269 label, to);
00269 }
00270 }

```

```

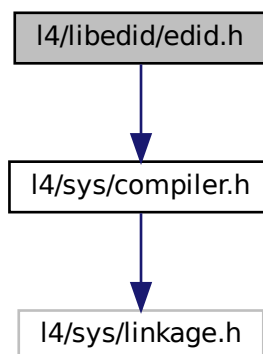
00271
00272 L4_INLINE l4_msgtag_t
00273 l4_irq_unmask_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW
00274 {
00275 l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_OP_EOI;
00276 return l4_ipc_send(irq, utcb, l4_msgtag(L4_PROTO_IRQ, 1, 0, 0), L4_IPC_NEVER);
00277 }
00278
00279
00280 L4_INLINE l4_msgtag_t
00281 l4_irq_mux_chain(l4_cap_idx_t irq, l4_cap_idx_t slave) L4_NOTHROW
00282 {
00283 return l4_irq_mux_chain_u(irq, slave, l4_utcb());
00284 }
00285
00286 L4_INLINE l4_msgtag_t
00287 l4_irq_detach(l4_cap_idx_t irq) L4_NOTHROW
00288 {
00289 return l4_irq_detach_u(irq, l4_utcb());
00290 }
00291
00292 L4_INLINE l4_msgtag_t
00293 l4_irq_trigger(l4_cap_idx_t irq) L4_NOTHROW
00294 {
00295 return l4_irq_trigger_u(irq, l4_utcb());
00296 }
00297
00298 L4_INLINE l4_msgtag_t
00299 l4_irq_receive(l4_cap_idx_t irq, l4_timeout_t to) L4_NOTHROW
00300 {
00301 return l4_irq_receive_u(irq, to, l4_utcb());
00302 }
00303
00304 L4_INLINE l4_msgtag_t
00305 l4_irq_wait(l4_cap_idx_t irq, l4_umword_t *label,
00306 l4_timeout_t to) L4_NOTHROW
00307 {
00308 return l4_irq_wait_u(irq, label, to, l4_utcb());
00309 }
00310
00311 L4_INLINE l4_msgtag_t
00312 l4_irq_unmask(l4_cap_idx_t irq) L4_NOTHROW
00313 {
00314 return l4_irq_unmask_u(irq, l4_utcb());
00315 }
00316

```

## 16.129 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_prefered_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.*

## 16.130 edid.h

```

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
00036 int libedid_check_header(const unsigned char *edid);
00037
00044 int libedid_checksum(const unsigned char *edid);
00045
00052 unsigned libedid_version(const unsigned char *edid);
00053
00060 unsigned libedid_revision(const unsigned char *edid);
00061
00068 void libedid_pnp_id(const unsigned char *edid, unsigned char *id);
00069
00077 void libedid_prefered_resolution(const unsigned char *edid,
00078 unsigned *w, unsigned *h);
00079
00086 unsigned libedid_num_ext_blocks(const unsigned char *edid);
00087
00094 unsigned libedid_dump_standard_timings(const unsigned char *edid);
00095
00101 void libedid_dump(const unsigned char *edid);
00102
00105 __END_DECLS

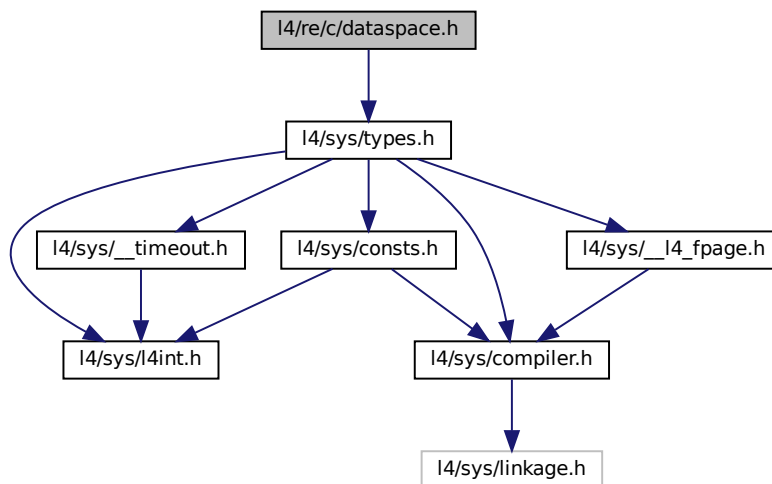
```

## 16.131 I4/re/c/dataspace.h File Reference

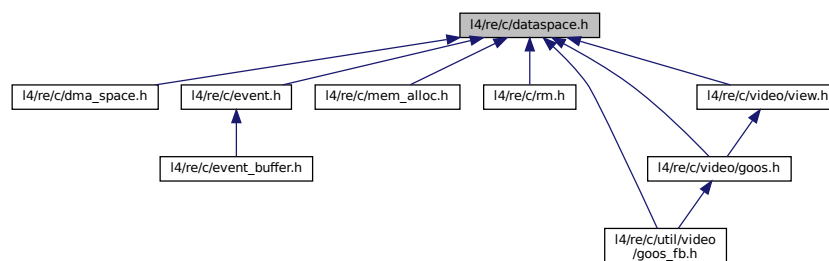
Data space C interface.

```
#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 `I4re_ds_stats_t`  
*Information about the data space.*

### Typedefs

- typedef `I4_cap_idx_t I4re_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](#)
- long [l4re\\_ds\\_allocate](#) ([l4re\\_ds\\_t](#) ds, [l4re\\_ds\\_offset\\_t](#) offset, [l4re\\_ds\\_size\\_t](#) size) [L4\\_NOTHROW](#)
- 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](#)
- [l4re\\_ds\\_size\\_t](#) [l4re\\_ds\\_size](#) ([l4re\\_ds\\_t](#) ds) [L4\\_NOTHROW](#)
- [l4re\\_ds\\_flags\\_t](#) [l4re\\_ds\\_flags](#) ([l4re\\_ds\\_t](#) ds) [L4\\_NOTHROW](#)
- int [l4re\\_ds\\_info](#) ([l4re\\_ds\\_t](#) ds, [l4re\\_ds\\_stats\\_t](#) \*stats) [L4\\_NOTHROW](#)

### 16.131.1 Detailed Description

Data space C interface.

Definition in file [dataspace.h](#).

## 16.132 dataspace.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
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,

```



```

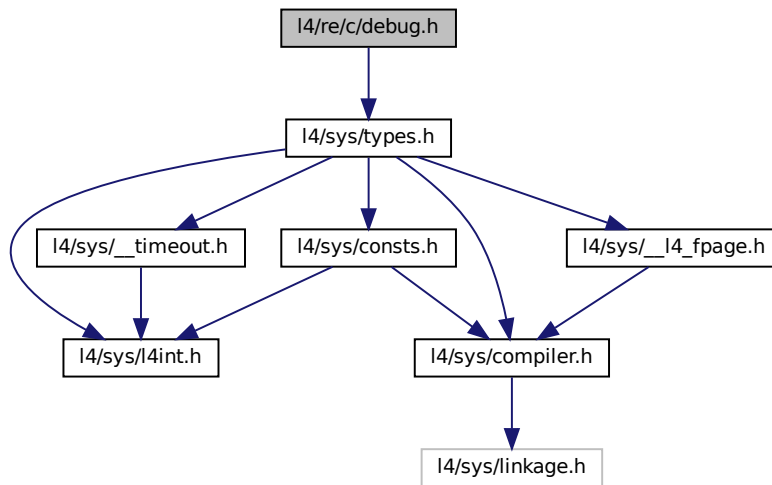
00067
00068 L4RE_DS_F_NORMAL = 0x00,
00069 L4RE_DS_F_CACHEABLE = L4RE_DS_F_NORMAL,
00070 L4RE_DS_F_BUFFERABLE = 0x10,
00071 L4RE_DS_F_UNCACHEABLE = 0x20,
00072 L4RE_DS_F_CACHING_MASK = 0x30,
00073 L4RE_DS_F_CACHING_SHIFT = 4,
00074 };
00075
00081 L4_CV int
00082 l4re_ds_map(l4re_ds_t ds,
00083 l4re_ds_offset_t 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
00116 L4_CV long
00117 l4re_ds_allocate(l4re_ds_t ds,
00118 l4re_ds_offset_t offset,
00119 l4re_ds_size_t size) L4_NOTHROW;
00120
00126 L4_CV int
00127 l4re_ds_copy_in(l4re_ds_t ds, l4re_ds_offset_t dst_offs,
00128 l4re_ds_t src, l4re_ds_offset_t src_offs,
00129 l4re_ds_size_t size) L4_NOTHROW;
00130
00136 L4_CV l4re_ds_size_t
00137 l4re_ds_size(l4re_ds_t ds) L4_NOTHROW;
00138
00143 L4_CV l4re_ds_flags_t
00144 l4re_ds_flags(l4re_ds_t ds) L4_NOTHROW;
00145
00150 L4_CV int
00151 l4re_ds_info(l4re_ds_t ds, l4re_ds_stats_t *stats) L4_NOTHROW;
00152
00153 EXTERN_C_END

```

## 16.133 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.

### 16.133.1 Detailed Description

Debug C interface.

Definition in file [debug.h](#).

### 16.134 debug.h

```

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 #pragma once
00020
00021 #include <l4/sys/types.h>
00022
00023 EXTERN_C_BEGIN
00024
00025 L4_CV long
00026 l4re_debug_obj_debug(l4_cap_idx_t srv, unsigned long function) L4_NOTHROW;
00027
00028 EXTERN_C_END

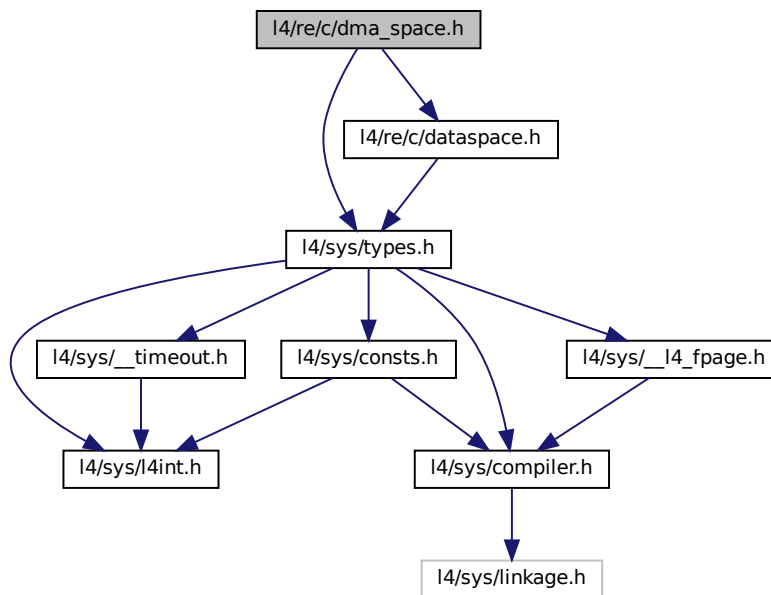
```

## 16.135 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 DMA task for a device to this Dma\_space.*

- long [l4re\\_dma\\_space\\_disassociate](#) ([l4re\\_dma\\_space\\_t](#) dma)

*Disassociate the DMA task from this Dma\_space.*

### 16.135.1 Detailed Description

DMA space C interface.

Definition in file [dma\\_space.h](#).

### 16.135.2 Enumeration Type Documentation

#### 16.135.2.1 l4re\_dma\_space\_direction

enum [l4re\\_dma\\_space\\_direction](#)

Direction of the DMA transfers.

Enumerator

|                                              |                                       |
|----------------------------------------------|---------------------------------------|
| <a href="#">L4RE_DMA_SPACE_BIDIRECTIONAL</a> | device reads and writes to the memory |
| <a href="#">L4RE_DMA_SPACE_TO_DEVICE</a>     | device reads the memory               |
| <a href="#">L4RE_DMA_SPACE_FROM_DEVICE</a>   | device writes to the memory           |
| <a href="#">L4RE_DMA_SPACE_NONE</a>          | device is coherently connected        |

Definition at line 37 of file [dma\\_space.h](#).

#### 16.135.2.2 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\_attribs

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](#).

## 16.136 dma\_space.h

```

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_attribs
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
00069 L4_CV long
00070 l4re_dma_space_map(l4re_dma_space_t dma, l4re_ds_t src,
00071 l4re_ds_offset_t offset,
00072 l4_size_t * size, unsigned long attrs,
00073 enum l4re_dma_space_direction dir,
00074 l4re_dma_space_dma_addr_t *dma_addr) L4_NOTHROW;
00075
00076
00082 L4_CV long
00083 l4re_dma_space_unmap(l4re_dma_space_t dma, l4re_dma_space_dma_addr_t dma_addr,
00084 l4_size_t size, unsigned long attrs,
00085 enum l4re_dma_space_direction dir) L4_NOTHROW;
00086
00092 L4_CV long
00093 l4re_dma_space_associate(l4re_dma_space_t dma, l4_cap_idx_t dma_task,
00094 unsigned long attr) L4_NOTHROW;
00095

```

```

00101 L4_CV long
00102 l4re_dma_space_disassociate(l4re_dma_space_t dma);
00103
00104
00105 EXTERN_C_END

```

## 16.137 l4/re/c/event.h File Reference

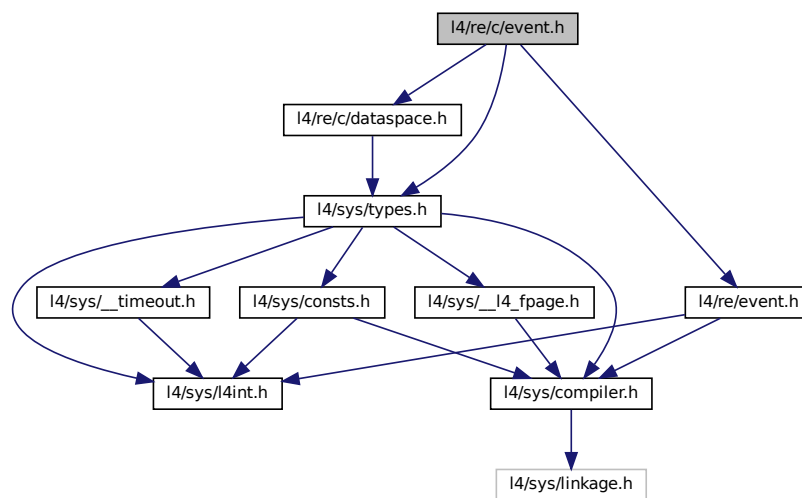
Event C interface.

```

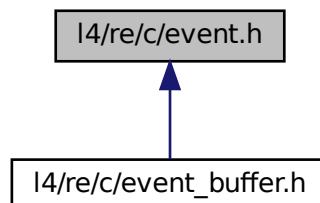
#include <l4/sys/types.h>
#include <l4/re/c/dataspace.h>
#include <l4/re/event.h>

```

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.*

### 16.137.1 Detailed Description

Event C interface.

Definition in file [event.h](#).

## 16.138 event.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
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

```

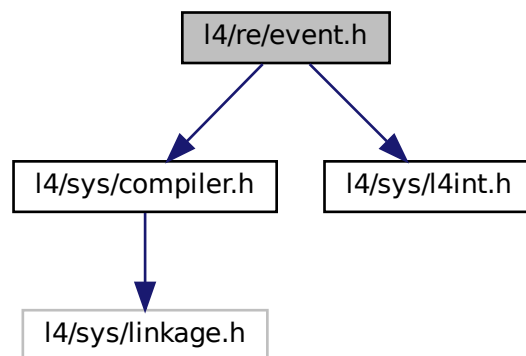
## 16.139 l4/re/event.h File Reference

Events.

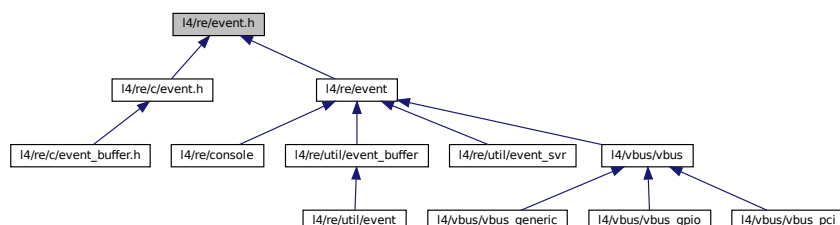
```
#include <l4/sys/compiler.h>
```

```
#include <l4/sys/l4int.h>
```

Include dependency graph for event.h:



This graph shows which files directly or indirectly include this file:





## 16.139.1 Detailed Description

Events.

Definition in file [event.h](#).

## 16.140 event.h

```

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

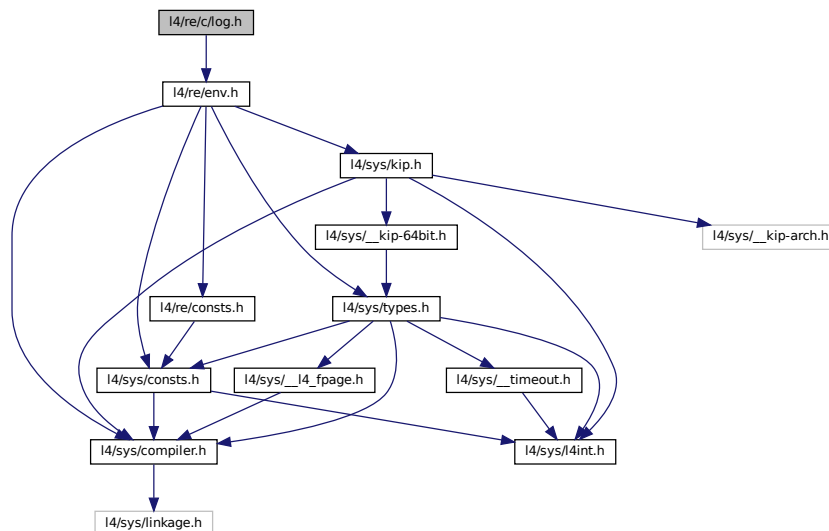
```

## 16.141 I4/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.

### 16.141.1 Detailed Description

Log C interface.

Definition in file [log.h](#).

## 16.142 log.h

```

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
00044 L4_CV L4_INLINE void
00045 l4re_log_print(char const *string) L4_NOTHROW;
00046
00058 L4_CV L4_INLINE void
00059 l4re_log_printn(char const *string, int len) L4_NOTHROW;
00060
00061
00062
00063
00075 L4_CV void
00076 l4re_log_print_srv(const l4_cap_idx_t logcap,
00077 char const *string) L4_NOTHROW;
00078
00091 L4_CV void
00092 l4re_log_printn_srv(const l4_cap_idx_t logcap,
00093 char const *string, int len) L4_NOTHROW;
00094
00095
00096 /***** Implementations *****/
00097
00098 L4_CV L4_INLINE void
00099 l4re_log_print(char const *string) L4_NOTHROW
00100 {
00101 l4re_log_print_srv(l4re_global_env->log, string);
00102 }
00103
00104 L4_CV L4_INLINE void
00105 l4re_log_printn(char const *string, int len) L4_NOTHROW
00106 {
00107 l4re_log_printn_srv(l4re_global_env->log, string, len);
00108 }
00109
00110 EXTERN_C_END

```

## 16.143 l4/re/c/mem\_alloc.h File Reference

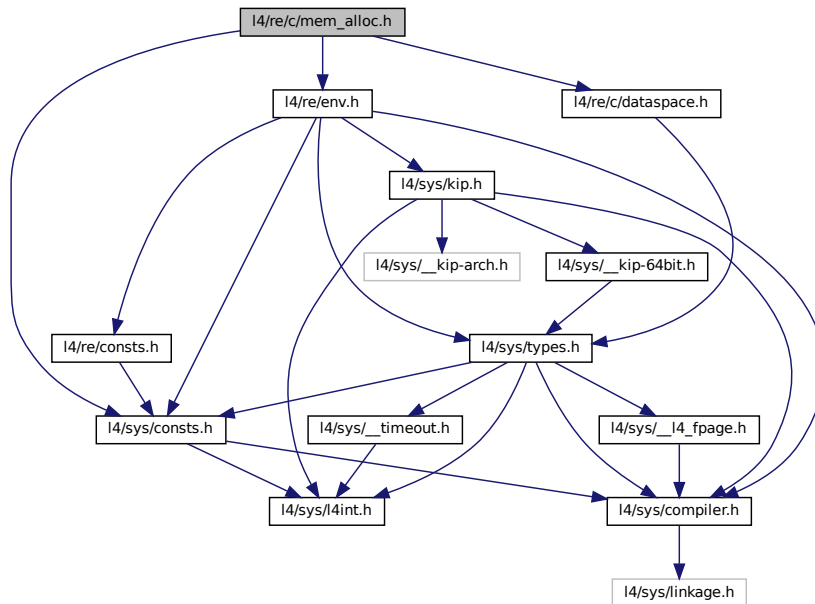
Memory allocator C interface.

```

#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\\_free](#) ([l4re\\_ds\\_t](#) const mem) [L4\\_NOTHROW](#)  
*Free 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.*
- long [l4re\\_ma\\_free\\_srv](#) ([l4\\_cap\\_idx\\_t](#) srv, [l4re\\_ds\\_t](#) const mem) [L4\\_NOTHROW](#)  
*Free memory.*

### 16.143.1 Detailed Description

Memory allocator C interface.

Definition in file [mem\\_alloc.h](#).

## 16.144 mem\_alloc.h

```

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
00131 /* Deprecation message added Q4 2016.
00132 * Remove together with L4Re::Mem_alloc::free() */
00133 L4_CV L4_INLINE long
00134 l4re_ma_free(l4re_ds_t const mem) L4_NOTHROW
00135 L4_DEPRECATED("This function is an empty stub and remains for backward compatibility only. See
 documentation for replacement options.");
00136
00137
00138
00139
00158 L4_CV long
00159 l4re_ma_alloc_align_srv(l4_cap_idx_t srv, long size,
00160 l4re_ds_t const mem, unsigned long flags,
00161 unsigned long align) L4_NOTHROW;
00162
00175 /* Deprecation message added Q4 2016.
00176 * Remove together with L4Re::Mem_alloc::free() */
00177 L4_CV long
00178 l4re_ma_free_srv(l4_cap_idx_t srv, l4re_ds_t const mem) L4_NOTHROW
00179 L4_DEPRECATED("This function is an empty stub and remains for backward compatibility only. See
 documentation for replacement options.");
00180
00181
00182
00183
00184
00185 /***** Implementation *****/
00186
00187 /* Just warn actual users, but not for internal implementations */
00188 #pragma GCC diagnostic push
00189 #pragma GCC diagnostic ignored "-Wdeprecated-declarations"
00190
00191 L4_CV L4_INLINE long
00192 l4re_ma_alloc(long size, l4re_ds_t const mem,
00193 unsigned long flags) L4_NOTHROW
00194 {
00195 return l4re_ma_alloc_align_srv(l4re_global_env->mem_alloc, size, mem,
00196 flags, 0);
00197 }
00198
00199 L4_CV L4_INLINE long
00200 l4re_ma_alloc_align(long size, l4re_ds_t const mem,
00201 unsigned long flags, unsigned long align) L4_NOTHROW

```

```

00202 {
00203 return l4re_ma_alloc_align_srv(l4re_global_env->mem_alloc, size, mem,
00204 flags, align);
00205 }
00206
00207 L4_CV L4_INLINE long
00208 l4re_ma_free(l4re_ds_t const mem) L4_NOTHROW
00209 {
00210 return l4re_ma_free_srv(l4re_global_env->mem_alloc, mem);
00211 }
00212
00213 #pragma GCC diagnostic pop
00214
00215 EXTERN_C_END

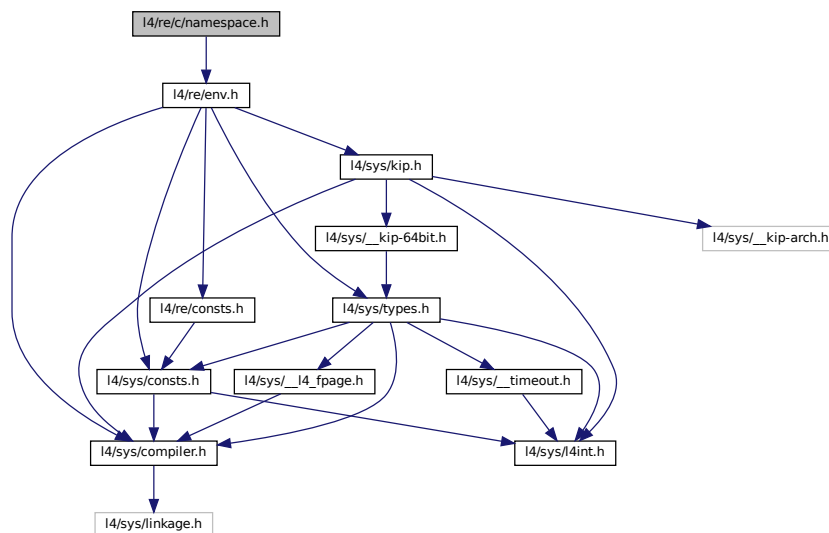
```

## 16.145 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.*

### 16.145.1 Detailed Description

Namespace functions, C interface.

Definition in file [namespace.h](#).

## 16.146 namespace.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
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

```

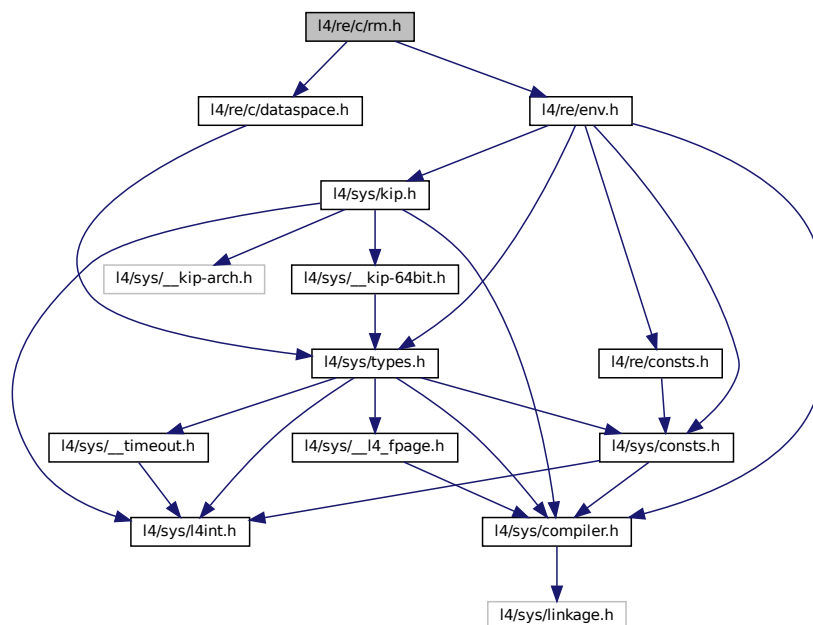
## 16.147 l4/re/c/rm.h File Reference

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](#)
- int [l4re\\_rm\\_free\\_area](#) ([l4\\_addr\\_t](#) addr) [L4\\_NOTHROW](#)
- 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](#)
- int [l4re\\_rm\\_detach](#) (void \*addr) [L4\\_NOTHROW](#)  
*Detach and unmap in current task.*
- int [l4re\\_rm\\_detach\\_ds](#) (void \*addr, [l4re\\_ds\\_t](#) \*ds) [L4\\_NOTHROW](#)  
*Detach, unmap and return affected dataspace in 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](#)
- 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.*

### 16.147.1 Detailed Description

Region map interface, C interface.

Definition in file [rm.h](#).

## 16.148 rm.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #pragma once
00024
00031 #include <l4/re/env.h>

```

```

00032 #include <l4/re/c/dataspace.h>
00033
00034 EXTERN_C_BEGIN
00035
00040 enum l4re_rm_flags_values {
00041 L4RE_RM_F_R = L4RE_DS_F_R,
00042 L4RE_RM_F_W = L4RE_DS_F_W,
00043 L4RE_RM_F_X = L4RE_DS_F_X,
00044 L4RE_RM_F_RX = L4RE_DS_F_RX,
00045 L4RE_RM_F_RW = L4RE_DS_F_RW,
00046 L4RE_RM_F_RWX = L4RE_DS_F_RWX,
00047
00048 L4RE_RM_F_NO_ALIAS = 0x200,
00049 L4RE_RM_F_PAGER = 0x400,
00050 L4RE_RM_F_RESERVED = 0x800,
00052 L4RE_RM_CACHING_SHIFT = 4,
00055 L4RE_RM_F_CACHING = L4RE_DS_F_CACHING_MASK,
00056
00057 L4RE_RM_REGION_FLAGS = 0xffff,
00060 L4RE_RM_F_CACHE_NORMAL = L4RE_DS_F_NORMAL,
00061
00063 L4RE_RM_F_CACHE_BUFFERED = L4RE_DS_F_BUFFERABLE,
00064
00066 L4RE_RM_F_CACHE_UNCACHED = L4RE_DS_F_UNCACHEABLE,
00067
00068 L4RE_RM_F_SEARCH_ADDR = 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
00084 L4_CV L4_INLINE int
00085 l4re_rm_reserve_area(l4_addr_t *start, unsigned long size,
00086 l4re_rm_flags_t flags, unsigned char align) L4_NOTHROW;
00087
00095 L4_CV L4_INLINE int
00096 l4re_rm_free_area(l4_addr_t addr) L4_NOTHROW;
00097
00106 L4_CV L4_INLINE int
00107 l4re_rm_attach(void **start, unsigned long size, l4re_rm_flags_t flags,
00108 l4re_ds_t mem,
00109 l4re_rm_offset_t offs,
00110 unsigned char align) L4_NOTHROW;
00111
00112
00123 L4_CV L4_INLINE int
00124 l4re_rm_detach(void *addr) L4_NOTHROW;
00125
00138 L4_CV L4_INLINE int
00139 l4re_rm_detach_ds(void *addr, l4re_ds_t *ds) L4_NOTHROW;
00140
00152 L4_CV L4_INLINE int
00153 l4re_rm_detach_unmap(l4_addr_t addr, l4_cap_idx_t task) L4_NOTHROW;
00154
00167 L4_CV L4_INLINE int
00168 l4re_rm_detach_ds_unmap(void *addr, l4re_ds_t *ds,
00169 l4_cap_idx_t task) L4_NOTHROW;
00170
00171
00177 L4_CV L4_INLINE int
00178 l4re_rm_find(l4_addr_t *addr, unsigned long *size,
00179 l4re_rm_offset_t *offset,
00180 l4re_rm_flags_t *flags, l4re_ds_t *m) L4_NOTHROW;
00181
00188 L4_CV L4_INLINE void
00189 l4re_rm_show_lists(void) L4_NOTHROW;
00190
00191
00192 /*
00193 * Variants of functions that also take a capability of the region map
00194 * service.
00195 */
00196
00197
00202 L4_CV int
00203 l4re_rm_reserve_area_srv(l4_cap_idx_t rm, l4_addr_t *start, unsigned long size,
00204 l4re_rm_flags_t flags, unsigned char align) L4_NOTHROW;
00205
00210 L4_CV int
00211 l4re_rm_free_area_srv(l4_cap_idx_t rm, l4_addr_t addr) L4_NOTHROW;
00212
00217 L4_CV int
00218 l4re_rm_attach_srv(l4_cap_idx_t rm, void **start, unsigned long size,
00219 l4re_rm_flags_t flags, l4re_ds_t mem,

```

```

00220 l4re_rm_offset_t offs,
00221 unsigned char align) L4_NOTHROW;
00222
00223
00228 L4_CV int
00229 l4re_rm_detach_srv(l4_cap_idx_t rm, l4_addr_t addr,
00230 l4re_ds_t *ds, l4_cap_idx_t task) L4_NOTHROW;
00231
00232
00237 L4_CV int
00238 l4re_rm_find_srv(l4_cap_idx_t rm, l4_addr_t *addr,
00239 unsigned long *size, l4re_rm_offset_t *offset,
00240 l4re_rm_flags_t *flags, l4re_ds_t *m) L4_NOTHROW;
00241
00246 L4_CV void
00247 l4re_rm_show_lists_srv(l4_cap_idx_t rm) L4_NOTHROW;
00248
00249
00250 /***** Implementations *****/
00251
00252 L4_CV L4_INLINE int
00253 l4re_rm_reserve_area(l4_addr_t *start, unsigned long size,
00254 l4re_rm_flags_t flags, unsigned char align) L4_NOTHROW
00255 {
00256 return l4re_rm_reserve_area_srv(l4re_global_env->rm, start, size,
00257 flags, align);
00258 }
00259
00260 L4_CV L4_INLINE int
00261 l4re_rm_free_area(l4_addr_t addr) L4_NOTHROW
00262 {
00263 return l4re_rm_free_area_srv(l4re_global_env->rm, addr);
00264 }
00265
00266 L4_CV L4_INLINE int
00267 l4re_rm_attach(void **start, unsigned long size, l4re_rm_flags_t flags,
00268 l4re_ds_t mem, l4re_rm_offset_t offs,
00269 unsigned char align) L4_NOTHROW
00270 {
00271 return l4re_rm_attach_srv(l4re_global_env->rm, start, size,
00272 flags, mem, offs, align);
00273 }
00274
00275
00276 L4_CV L4_INLINE int
00277 l4re_rm_detach(void *addr) L4_NOTHROW
00278 {
00279 return l4re_rm_detach_srv(l4re_global_env->rm,
00280 (l4_addr_t)addr, 0, L4_BASE_TASK_CAP);
00281 }
00282
00283 L4_CV L4_INLINE int
00284 l4re_rm_detach_unmap(l4_addr_t addr, l4_cap_idx_t task) L4_NOTHROW
00285 {
00286 return l4re_rm_detach_srv(l4re_global_env->rm, addr, 0, task);
00287 }
00288
00289 L4_CV L4_INLINE int
00290 l4re_rm_detach_ds(void *addr, l4re_ds_t *ds) L4_NOTHROW
00291 {
00292 return l4re_rm_detach_srv(l4re_global_env->rm, (l4_addr_t)addr,
00293 ds, L4_BASE_TASK_CAP);
00294 }
00295
00296 L4_CV L4_INLINE int
00297 l4re_rm_detach_ds_unmap(void *addr, l4re_ds_t *ds, l4_cap_idx_t task) L4_NOTHROW
00298 {
00299 return l4re_rm_detach_srv(l4re_global_env->rm, (l4_addr_t)addr,
00300 ds, task);
00301 }
00302
00303 L4_CV L4_INLINE int
00304 l4re_rm_find(l4_addr_t *addr, unsigned long *size,
00305 l4re_rm_offset_t *offset,
00306 l4re_rm_flags_t *flags, l4re_ds_t *m) L4_NOTHROW
00307 {
00308 return l4re_rm_find_srv(l4re_global_env->rm, addr, size, offset, flags, m);
00309 }
00310
00311 L4_CV L4_INLINE void
00312 l4re_rm_show_lists(void) L4_NOTHROW
00313 {
00314 l4re_rm_show_lists_srv(l4re_global_env->rm);
00315 }
00316
00317 EXTERN_C_END

```

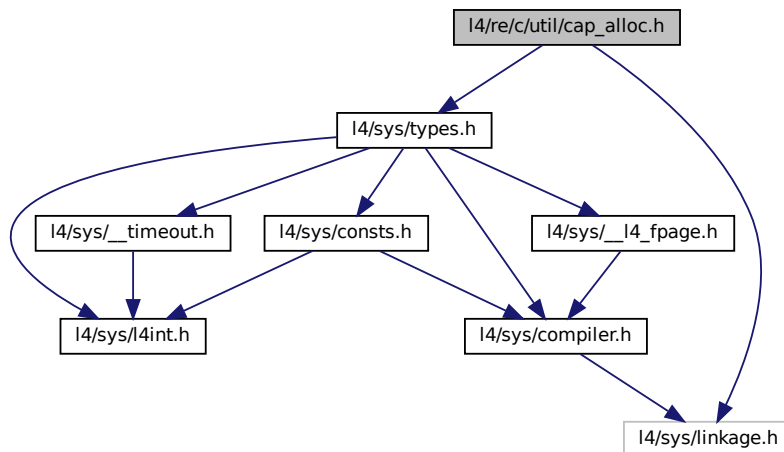
## 16.149 l4/re/c/util/cap\_alloc.h File Reference

Capability allocator C interface.

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

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

Include dependency graph for cap\_alloc.h:



### Functions

- `l4_cap_idx_t l4re_util_cap_alloc (void) L4_NOTHROW`  
Get free capability index at capability allocator.
- `void l4re_util_cap_free (l4_cap_idx_t cap) L4_NOTHROW`  
Return capability index to capability allocator.
- `void l4re_util_cap_free_um (l4_cap_idx_t cap) L4_NOTHROW`  
Return capability index to capability allocator, and unmaps the object.
- `long l4re_util_cap_last (void) L4_NOTHROW`  
Return last capability index the allocator can return.

### 16.149.1 Detailed Description

Capability allocator C interface.

Definition in file `cap_alloc.h`.

## 16.150 cap\_alloc.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
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

```

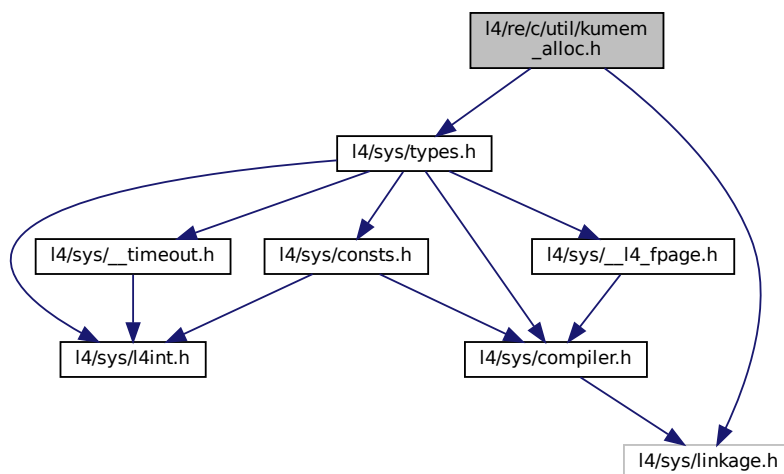
## 16.151 l4/re/c/util/kumem\_alloc.h File Reference

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.*

### 16.151.1 Detailed Description

Kumem allocator utility C interface.

Definition in file [kumem\\_alloc.h](#).

### 16.151.2 Function Documentation

#### 16.151.2.1 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](#).

## 16.152 kumem\_alloc.h

```

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

```

## 16.153 l4/re/c/util/video/goos\_fb.h File Reference

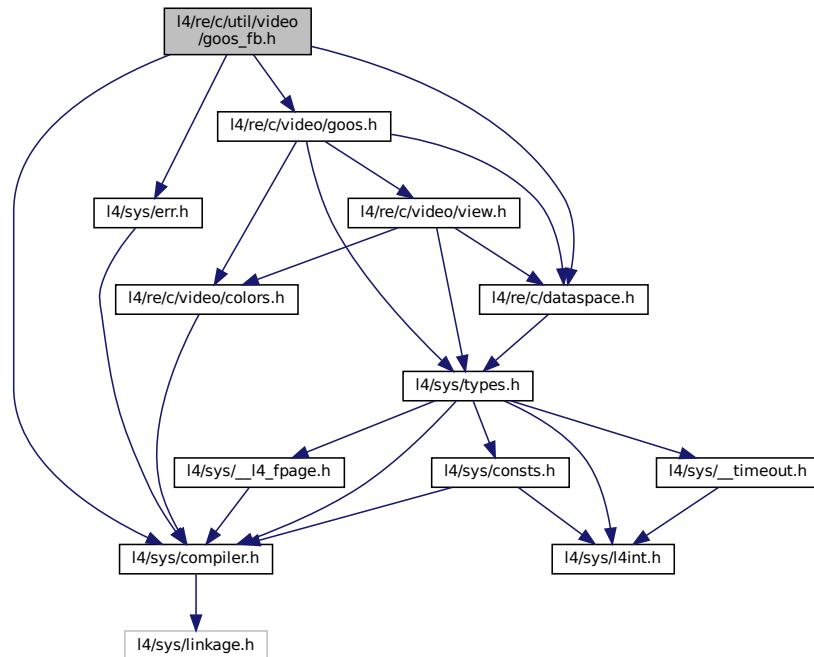
Framebuffer utility functionality.

```

#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`:



### 16.153.1 Detailed Description

Framebuffer utility functionality.

Definition in file [goos\\_fb.h](#).

## 16.154 goos\_fb.h

```

00001
00002 /*
00003 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00004 * economic rights: Technische Universität Dresden (Germany)
00005 *
00006 * This file is part of TUD:OS and distributed under the terms of the
00007 * GNU General Public License 2.
00008 * Please see the COPYING-GPL-2 file for details.
00009 *
00010 * As a special exception, you may use this file as part of a free software
00011 * library without restriction. Specifically, if other files instantiate
00012 * templates or use macros or inline functions from this file, or you compile
00013 * this file and link it with other files to produce an executable, this
00014 * file does not by itself cause the resulting executable to be covered by
00015 * the GNU General Public License. This exception does not however
00016 * invalidate any other reasons why the executable file might be covered by
00017 * the GNU General Public License.
00018 */
00019 #pragma once
00020
00021 #include <l4/sys/compiler.h>
00022 #include <l4/re/c/video/goos.h>
00023 #include <l4/sys/err.h>
00024 #include <l4/re/c/dataspace.h>
00025
00026 EXTERN_C_BEGIN
00027
00028 typedef struct

```



```

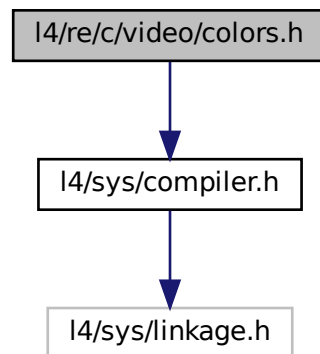
00032 {
00033 unsigned long _obj_buf[6];
00034 } l4re_util_video_goos_fb_t;
00035
00036 L4_CV int
00037 l4re_util_video_goos_fb_setup_name(l4re_util_video_goos_fb_t *goosfb,
00038 char const *name) L4_NOTHROW;
00039
00040 L4_CV void
00041 l4re_util_video_goos_fb_destroy(l4re_util_video_goos_fb_t *goosfb) 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

```

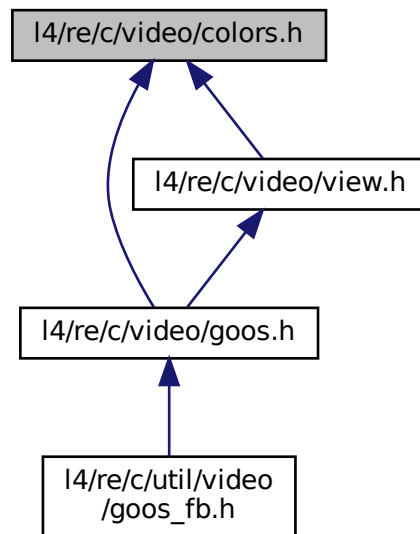
## 16.155 l4/re/c/video/colors.h File Reference

#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.*

### 16.155.1 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](#).

## 16.156 colors.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/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

```

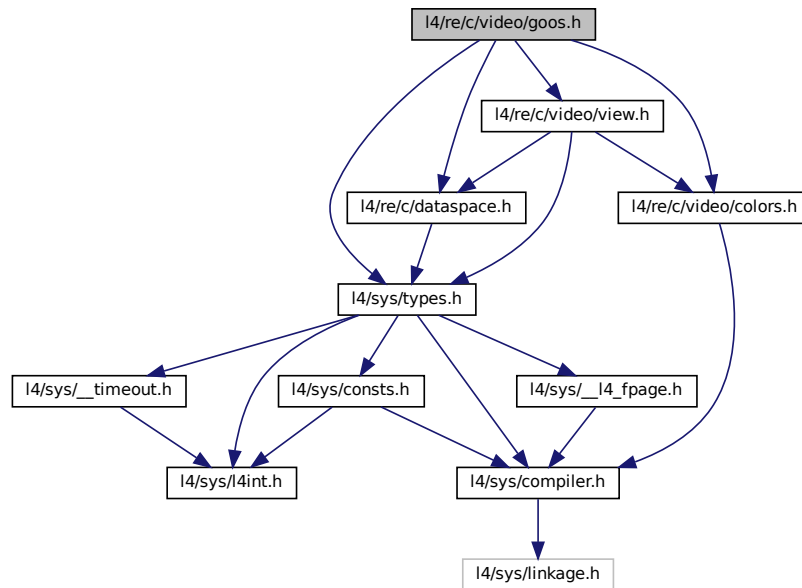
## 16.157 l4/re/c/video/goos.h File Reference

```

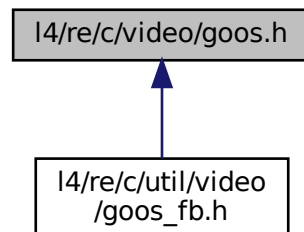
#include <l4/sys/types.h>
#include <l4/re/c/dataspace.h>
#include <l4/re/c/video/colors.h>
#include <l4/re/c/video/view.h>

```

Include dependency graph for `goos.h`:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `l4re_video_goos_info_t`  
*Goos information structure.*

## Typedefs

- typedef `l4_cap_idx_t l4re_video_goos_t`  
*Goos object type.*

## Enumerations

- enum `l4re_video_goos_info_flags_t` { `F_l4re_video_goos_auto_refresh` = 0x01 , `F_l4re_video_goos_pointer` = 0x02 , `F_l4re_video_goos_dynamic_views` = 0x04 , `F_l4re_video_goos_dynamic_buffers` = 0x08 }

*Flags of information on the goos.*

## Functions

- int `l4re_video_goos_info` (`l4re_video_goos_t` goos, `l4re_video_goos_info_t` \*ginfo) `L4_NOTHROW`  
*Get information on a goos.*
- int `l4re_video_goos_refresh` (`l4re_video_goos_t` goos, int x, int y, int w, int h) `L4_NOTHROW`  
*Flush a rectangle of pixels of the goos screen.*
- int `l4re_video_goos_create_buffer` (`l4re_video_goos_t` goos, unsigned long size, `l4_cap_idx_t` buffer) `L4_NOTHROW`  
*Create a new buffer (memory buffer) for pixel data.*
- int `l4re_video_goos_delete_buffer` (`l4re_video_goos_t` goos, unsigned idx) `L4_NOTHROW`  
*Delete a pixel buffer.*
- int `l4re_video_goos_get_static_buffer` (`l4re_video_goos_t` goos, unsigned idx, `l4_cap_idx_t` buffer) `L4_NOTHROW`  
*Get the data-space capability of the static pixel buffer.*
- int `l4re_video_goos_create_view` (`l4re_video_goos_t` goos, `l4re_video_view_t` \*view) `L4_NOTHROW`  
*Create a new view (.*
- int `l4re_video_goos_delete_view` (`l4re_video_goos_t` goos, `l4re_video_view_t` \*view) `L4_NOTHROW`  
*Delete a view.*
- int `l4re_video_goos_get_view` (`l4re_video_goos_t` goos, unsigned idx, `l4re_video_view_t` \*view) `L4_NOTHROW`  
*Get a view for the given index.*

### 16.157.1 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](#).

## 16.158 goos.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/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
00170 L4_CV int
00171 l4re_video_goos_get_view(l4re_video_goos_t goos, unsigned idx,
00172 l4re_video_view_t *view) L4_NOTHROW;
00173
00174 EXTERN_C_END

```

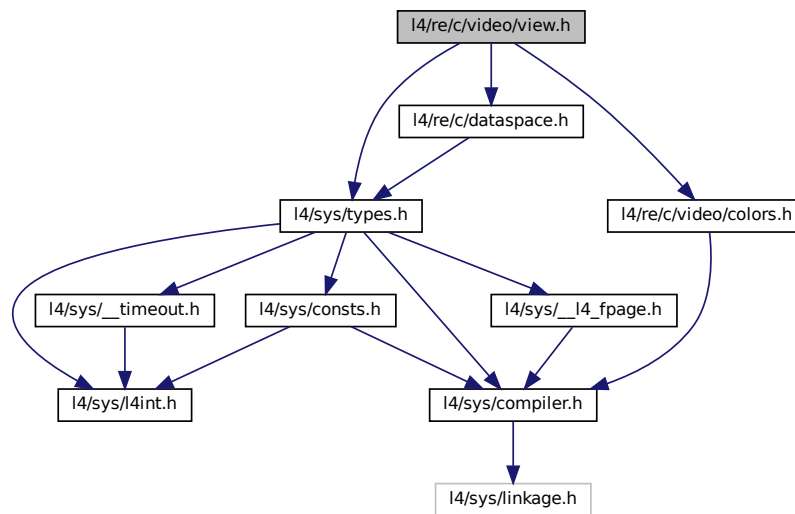
## 16.159 l4/re/c/video/view.h File Reference

```

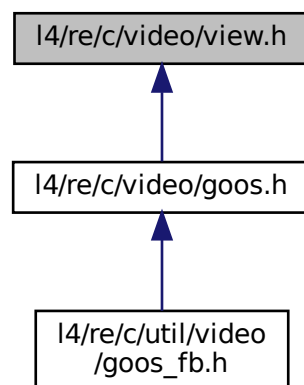
#include <l4/sys/types.h>
#include <l4/re/c/dataspace.h>
#include <l4/re/c/video/colors.h>

```

Include dependency graph for view.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [l4re\\_video\\_view\\_info\\_t](#)  
View information structure.
- struct [l4re\\_video\\_view\\_t](#)  
C representation of a goos view.

## Typedefs

- typedef struct [l4re\\_video\\_view\\_info\\_t](#) [l4re\\_video\\_view\\_info\\_t](#)  
*View information structure.*
- typedef struct [l4re\\_video\\_view\\_t](#) [l4re\\_video\\_view\\_t](#)  
*C representation of a goos view.*

## Enumerations

- enum [l4re\\_video\\_view\\_info\\_flags\\_t](#) {  
[F\\_l4re\\_video\\_view\\_none](#) = 0x00 , [F\\_l4re\\_video\\_view\\_set\\_buffer](#) = 0x01 , [F\\_l4re\\_video\\_view\\_set\\_buffer\\_offset](#) = 0x02 , [F\\_l4re\\_video\\_view\\_set\\_bytes\\_per\\_line](#) = 0x04 ,  
[F\\_l4re\\_video\\_view\\_set\\_pixel](#) = 0x08 , [F\\_l4re\\_video\\_view\\_set\\_position](#) = 0x10 , [F\\_l4re\\_video\\_view\\_dyn\\_allocated](#) = 0x20 , [F\\_l4re\\_video\\_view\\_set\\_background](#) = 0x40 ,  
[F\\_l4re\\_video\\_view\\_set\\_flags](#) = 0x80 , [F\\_l4re\\_video\\_view\\_fully\\_dynamic](#) , [F\\_l4re\\_video\\_view\\_above](#) = 0x01000 , [F\\_l4re\\_video\\_view\\_flags\\_mask](#) = 0xff000 }  
*Flags of information on a view.*

## Functions

- int [l4re\\_video\\_view\\_refresh](#) ([l4re\\_video\\_view\\_t](#) \*view, int x, int y, int w, int h) [L4\\_NOTHROW](#)  
*Flush the given rectangle of pixels of the given view.*
- int [l4re\\_video\\_view\\_get\\_info](#) ([l4re\\_video\\_view\\_t](#) \*view, [l4re\\_video\\_view\\_info\\_t](#) \*info) [L4\\_NOTHROW](#)  
*Retrieve information about the given view.*
- int [l4re\\_video\\_view\\_set\\_info](#) ([l4re\\_video\\_view\\_t](#) \*view, [l4re\\_video\\_view\\_info\\_t](#) \*info) [L4\\_NOTHROW](#)  
*Set properties of the view.*
- int [l4re\\_video\\_view\\_set\\_viewport](#) ([l4re\\_video\\_view\\_t](#) \*view, int x, int y, int w, int h, unsigned long bofs) [L4\\_NOTHROW](#)  
*Set the viewport parameters of a view.*
- int [l4re\\_video\\_view\\_stack](#) ([l4re\\_video\\_view\\_t](#) \*view, [l4re\\_video\\_view\\_t](#) \*pivot, int behind) [L4\\_NOTHROW](#)  
*Change the stacking order in the stack of visible views.*

### 16.159.1 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](#).



## 16.160 view.h

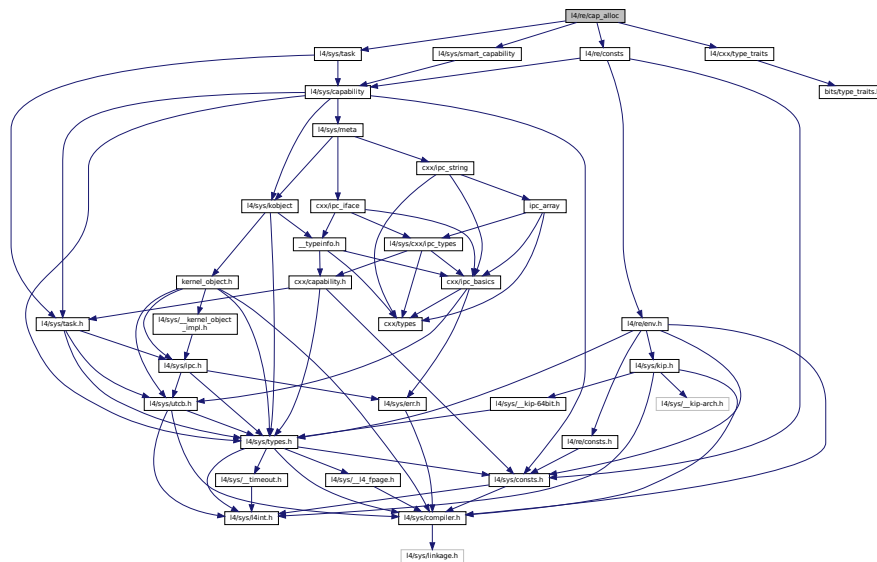
```

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/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
00096 L4_CV int
00097 l4re_video_view_refresh(l4re_video_view_t *view, int x, int y, int w,
00098 int h) L4_NOTHROW;
00099
00106 L4_CV int
00107 l4re_video_view_get_info(l4re_video_view_t *view,
00108 l4re_video_view_info_t *info) L4_NOTHROW;
00109
00120 L4_CV int
00121 l4re_video_view_set_info(l4re_video_view_t *view,
00122 l4re_video_view_info_t *info) L4_NOTHROW;
00123
00139 L4_CV int
00140 l4re_video_view_set_viewport(l4re_video_view_t *view, int x, int y, int w,
00141 int h, unsigned long bofs) L4_NOTHROW;
00142
00152 L4_CV int
00153 l4re_video_view_stack(l4re_video_view_t *view, l4re_video_view_t *pivot,

```

## 16.161 l4/re/cap\_alloc File Reference

```
#include <l4/sys/task>
#include <l4/sys/smart_capability>
#include <l4/re/consts>
#include <l4/cxx/type_traits>
Include dependency graph for cap_alloc:
```

[illegible]

## 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

- [L4Re](#)  
*L4Re C++ Interfaces.*

### 16.161.1 Detailed Description

Abstract capability-allocator interface.

Definition in file [cap\\_alloc](#).

## 16.162 cap\_alloc

```

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
00146 template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00147 class Smart_cap_auto
00148 {
00149 private:
00150 Cap_alloc *_ca;
00151
00152 public:
00153 Smart_cap_auto() : _ca(0) {}
00154 Smart_cap_auto(Cap_alloc *ca) : _ca(ca) {}
00155
00156 void free(L4::Cap_base &c)
00157 {
00158 if (c.is_valid() && _ca)
00159 _ca->free(L4::Cap<void>(c.cap()), This_task, Unmap_flags);
00160
00161 invalidate(c);
00162 }
00163
00164 static void invalidate(L4::Cap_base &c)
00165 {
00166 if (c.is_valid())
00167 c.invalidate();
00168 }
00169
00170 static L4::Cap_base copy(L4::Cap_base const &src)

```

```

00171 {
00172 L4::Cap_base r = src;
00173 invalidate(const_cast<L4::Cap_base &>(src));
00174 return r;
00175 }
00176 };
00177
00181 template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00182 class Smart_count_cap
00183 {
00184 private:
00185 Cap_alloc *_ca;
00186 public:
00187 Smart_count_cap() : _ca(nullptr) {}
00188 Smart_count_cap(Cap_alloc *ca) : _ca(ca) {}
00194 void free(L4::Cap_base &c) noexcept
00195 {
00196 if (c.is_valid())
00197 {
00198 if (_ca && _ca->release(L4::Cap<void>(c.cap()), This_task, Unmap_flags))
00199 c.invalidate();
00200 }
00201 }
00202
00206 static void invalidate(L4::Cap_base &c) noexcept
00207 {
00208 if (c.is_valid())
00209 c.invalidate();
00210 }
00211
00215 L4::Cap_base copy(L4::Cap_base const &src)
00216 {
00217 if (src.is_valid())
00218 _ca->take(L4::Cap<void>(src.cap()));
00219 return src;
00220 }
00221 };
00224 }

```

## 16.163 l4/re/util/cap\_alloc File Reference

Capability allocator.

```

#include <l4/re/util/cap_alloc_impl.h>
#include <l4/sys/smart_capability>
#include <l4/sys/task>
#include <l4/re/consts>

```

[illegible]

- 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.*

- [L4Re](#)  
*L4Re C++ Interfaces.*
- [L4Re::Util](#)  
*Documentation of the L4 Runtime Environment utility functionality in C++.*

## Functions

- `template<typename T>`  
`Ref_cap< T >::Cap L4Re::Util::make_ref_cap ()`  
*Allocate a capability slot and wrap it in a [Ref\\_cap](#).*
- `template<typename T>`  
`Ref_del_cap< T >::Cap L4Re::Util::make_ref_del_cap ()`  
*Allocate a capability slot and wrap it in a [Ref\\_del\\_cap](#).*

## Variables

- `_Cap_alloc & L4Re::Util::cap_alloc`  
*Capability allocator.*

### 16.163.1 Detailed Description

Capability allocator.

Definition in file [cap\\_alloc](#).

## 16.164 cap\_alloc

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 * Alexander Warg <warg@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025
00026 #pragma once
00027
00028 #include <l4/re/util/cap_alloc_impl.h>
00029 #include <l4/sys/smart_capability>
00030 #include <l4/sys/task>
00031 #include <l4/re/consts>
00032
00033 namespace L4Re { namespace Util {
00034
00053 extern _Cap_alloc &cap_alloc;
00054
00058 template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00059 class Smart_cap_auto
00060 {
00061 public:
00065 static void free(L4::Cap_base &c)
00066 {
00067 if (c.is_valid())
00068 {
00069 cap_alloc.free(L4::Cap<void>(c.cap()), This_task, Unmap_flags);
00070 c.invalidate();
00071 }
00072 }
00073
00077 static void invalidate(L4::Cap_base &c)
```

```

00078 {
00079 if (c.is_valid())
00080 c.invalidate();
00081 }
00082
00086 static L4::Cap_base copy(L4::Cap_base const &src)
00087 {
00088 L4::Cap_base r = src;
00089 invalidate(const_cast<L4::Cap_base &>(src));
00090 return r;
00091 }
00092 };
00093
00094
00098 template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00099 class Smart_count_cap
00100 {
00101 public:
00106 static void free(L4::Cap_base &c) noexcept
00107 {
00108 if (c.is_valid())
00109 {
00110 if (cap_alloc.release(L4::Cap<void>(c.cap()), This_task, Unmap_flags))
00111 c.invalidate();
00112 }
00113 }
00114
00118 static void invalidate(L4::Cap_base &c) noexcept
00119 {
00120 if (c.is_valid())
00121 c.invalidate();
00122 }
00123
00127 static L4::Cap_base copy(L4::Cap_base const &src)
00128 {
00129 cap_alloc.take(L4::Cap<void>(src.cap()));
00130 return src;
00131 }
00132 };
00133
00134
00164 template< typename T >
00165 struct Ref_cap
00166 {
00167 typedef L4::Smart_cap<T, Smart_count_cap<L4_FP_ALL_SPACES> > Cap;
00168 };
00169
00205 template< typename T >
00206 struct Ref_del_cap
00207 {
00208 typedef L4::Smart_cap<T, Smart_count_cap<L4_FP_DELETE_OBJ> > Cap;
00209 };
00210
00216 template< typename T >
00217 typename Ref_cap<T>::Cap
00218 make_ref_cap() { return typename Ref_cap<T>::Cap(cap_alloc.alloc<T>()); }
00219
00225 template< typename T >
00226 typename Ref_del_cap<T>::Cap
00227 make_ref_del_cap()
00228 { return typename Ref_del_cap<T>::Cap(cap_alloc.alloc<T>()); }
00229
00232 }}
00233

```

## 16.165 l4/re/dataspace File Reference

Dataspace interface.

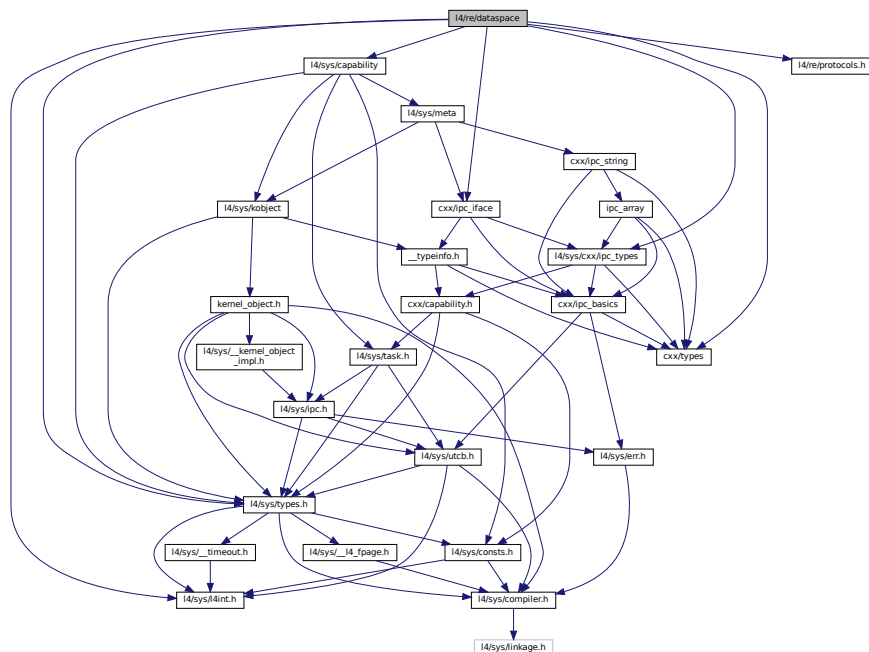
```

#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 dependency graph for dataspace:



- class `L4Re::Dataspace`  
*Interface for memory-like objects.*
- struct `L4Re::Dataspace::F`  
*`Dataspace` flags definitions.*
- struct `L4Re::Dataspace::Stats`  
*Information about the dataspace.*

- L4Re

Definition in file [dataspace](#).

## 16.166 dataspace

```

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
00077 enum Flags
00078 {
00080 R = L4_FPAGE_RO,
00082 Ro = L4_FPAGE_RO,
00084 RW = L4_FPAGE_RW,
00086 W = L4_FPAGE_W,
00087 X = L4_FPAGE_X,
00088 RX = L4_FPAGE_RX,
00089 RWX = L4_FPAGE_RWX,
00091 Rights_mask = 0x0f,
00092
00094 Normal = 0x00,
00096 Cacheable = Normal,
00098 Bufferable = 0x10,
00100 Uncacheable = 0x20,
00102 Caching_mask = 0x30,
00103 };
00104
00105 L4_TYPES_FLAGS_OPS_DEF(Flags);
00106 };
00107
00108 struct Flags : L4::Types::Flags_ops_t<Flags>
00109 {
00110 unsigned long raw;
00111 Flags() = default;
00112 explicit constexpr Flags(unsigned long f) : raw(f) {}
00113 constexpr Flags(F::Flags f) : raw(f) {}
00114 constexpr bool r() const { return raw & L4_FPAGE_RO; }
00115 constexpr bool w() const { return raw & L4_FPAGE_W; }
00116 constexpr bool x() const { return raw & L4_FPAGE_X; }
00117
00118 constexpr unsigned long fpage_rights() const
00119 { return raw & 0xf; }

```

```

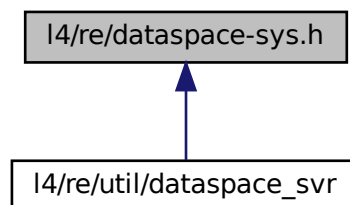
00120 };
00121
00122 typedef l4_uint64_t Size;
00123 typedef l4_uint64_t Offset;
00124 typedef l4_uint64_t Map_addr;
00125
00129 struct Stats
00130 {
00131 Size size;
00132 Flags flags;
00133 };
00134
00135
00158 long map(Offset offset, Flags flags, Map_addr local_addr,
00159 Map_addr min_addr, Map_addr max_addr) const noexcept;
00160
00186 long map_region(Offset offset, Flags flags,
00187 Map_addr min_addr, Map_addr max_addr) const noexcept;
00188
00206 L4_RPC(long, clear, (Offset offset, Size size));
00207
00227 L4_RPC(long, allocate, (Offset offset, Size size));
00228
00246 L4_RPC(long, copy_in, (Offset dst_offs, L4::Ipc::Cap<Dataspace> src,
00247 Offset src_offs, Size size));
00248
00254 Size size() const noexcept;
00255
00264 Flags flags() const noexcept;
00265
00274 L4_RPC(long, info, (Stats *stats));
00275
00276 L4_RPC_NF(long, map, (Offset offset, Map_addr spot,
00277 Flags flags, L4::Ipc::Rcv_fpage r,
00278 L4::Ipc::Snd_fpage &fp));
00279
00280 private:
00281
00282 long __map(Offset offset, unsigned char *order, Flags flags,
00283 Map_addr local_addr) const noexcept;
00284
00285 public:
00286 typedef L4::Typeid::Rpc<map_t, clear_t, info_t, copy_in_t,
00287 allocate_t> Rpc;
00288
00289 };
00290
00291 }
00292

```

## 16.167 I4/re/dataspace-sys.h File Reference

Dataspace protocol definition.

This graph shows which files directly or indirectly include this file:



## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

## Enumerations

- enum [L4Re::Dataspace\\_::Opcodes](#)  
*Data-space communication-protocol opcodes.*

### 16.167.1 Detailed Description

Dataspace protocol definition.

Definition in file [dataspace-sys.h](#).

### 16.168 dataspace-sys.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
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

```

### 16.169 l4/re/debug File Reference

Debug interface.

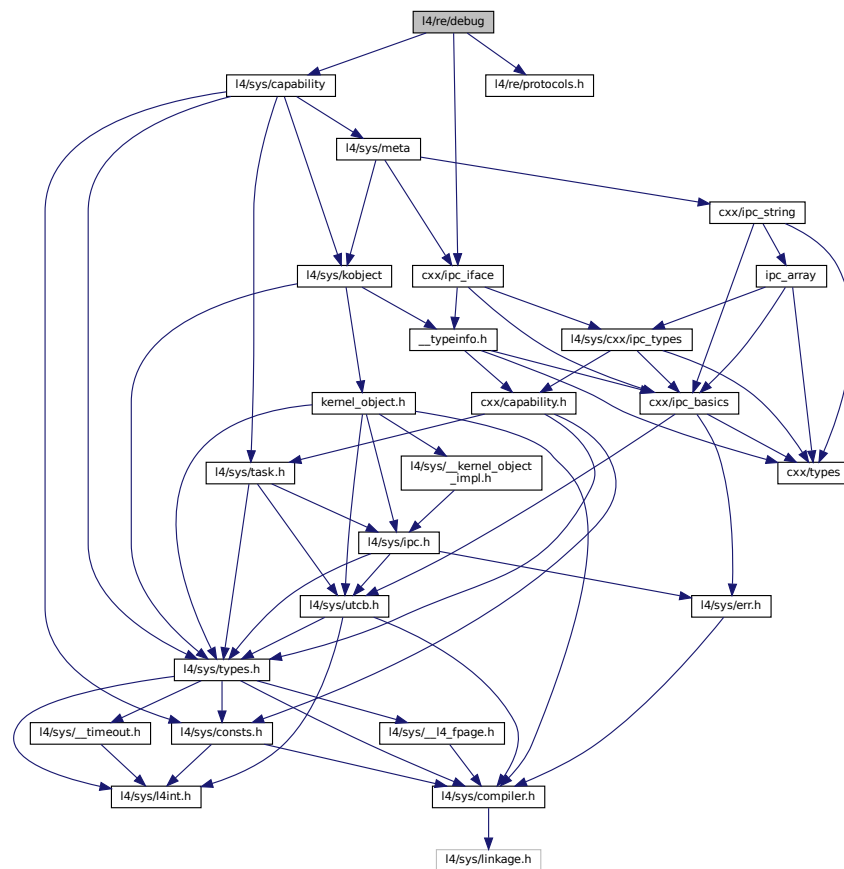
```

#include <l4/sys/capability>
#include <l4/re/protocols.h>

```

```
#include <l4/sys/cxx/ipc_iface>
```

Include dependency graph for debug:



## Data Structures

- class [L4Re::Debug\\_obj](#)  
*Debug interface.*

## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

### 16.169.1 Detailed Description

Debug interface.

Definition in file [debug](#).

## 16.170 debug

```

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
00063 L4_INLINE_RPC(long, debug, (unsigned long function));
00064 typedef L4::Typeid::Rpc_nocode<debug_t> Rpccs;
00065 };
00066
00067 template<typename BASE>
00068 class Debug_obj_t :
00069 public L4::Kobject_2t<Debug_obj_t<BASE>, BASE, Debug_obj, L4::PROTO_EMPTY>
00070 {
00071 typedef L4::Typeid::Rpccs<> Rpccs;
00072 };
00073 }

```

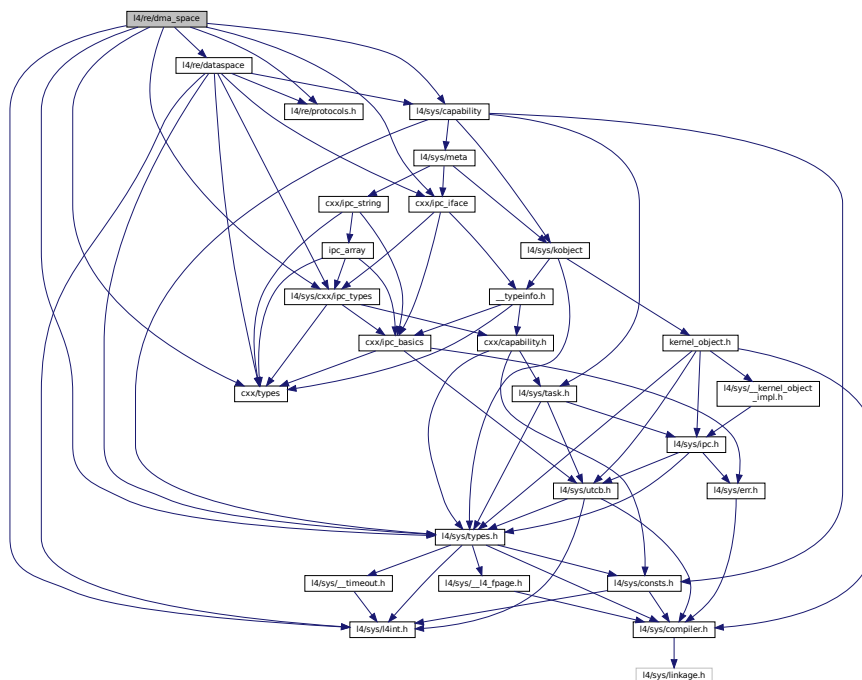
## 16.171 l4/re/dma\_space File Reference

```

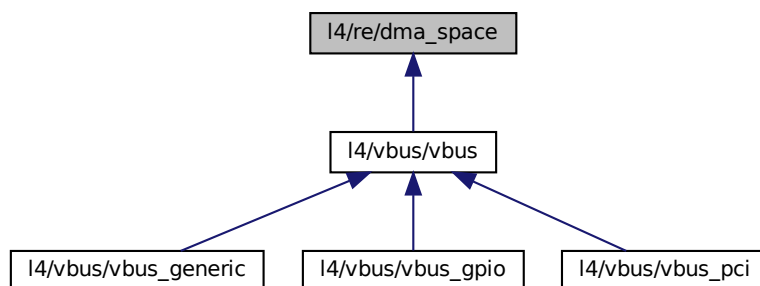
#include <l4/sys/types.h>
#include <l4/sys/l4int.h>
#include <l4/sys/capability>
#include <l4/re/dataspace>
#include <l4/re/protocols.h>
#include <l4/sys/cxx/types>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>

```

Include dependency graph for dma\_space:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `L4Re::Dma_space`  
*DMA Address Space.*

## Namespaces

- L4Re
- L4Re C++ Interfaces.*

## 16.172 dma\_space

```

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
00057 class Dma_space :
00058 public L4::Kobject_0t< Dma_space,
00059 L4RE_PROTO_DMA_SPACE,
00060 L4::Type_info::Demand_t<1> >
00061 {
00062 public:
00064 typedef l4_uint64_t Dma_addr;
00065
00069 enum Direction
00070 {
00071 Bidirectional,
00072 To_device,
00073 From_device,
00074 None
00075 };
00076
00081 enum Attribute
00082 {
00094 No_sync
00095 };
00096
00102 typedef L4::Types::Flags<Attribute> Attributes;
00103
00109 enum Space_attrib
00110 {
00117 Coherent,
00118
00123 Phys_space
00124 };
00125
00127 typedef L4::Types::Flags<Space_attrib> Space_attribs;
00128
00153 L4_INLINE_RPC(
00154 long, map, (L4::Ipc::Cap<L4Re::Dataspace> src,
00155 L4Re::Dataspace::Offset offset,
00156 L4::Ipc::In_out<l4_size_t *> size,
00157 Attributes attrs, Direction dir,
00158 Dma_addr *dma_addr));
00159
00168 L4_INLINE_RPC(
00169 long, unmap, (Dma_addr dma_addr,
00170 l4_size_t size, Attributes attrs, Direction dir));
00171
00185 L4_INLINE_RPC(
00186 long, associate, (L4::Ipc::Opt<L4::Ipc::Cap<L4::Task> > dma_task,
00187 Space_attribs attr),
00188 L4::Ipc::Call_t<L4_CAP_FPAGE_RW>);
00189
00195 L4_INLINE_RPC(
00196 long, disassociate, (),
00197 L4::Ipc::Call_t<L4_CAP_FPAGE_RW>);
00198

```



```

00199 typedef L4::Typeid::Rpc<map_t, unmap_t, associate_t, disassociate_t> Rpc;
00200 };
00201
00202 }

```

## 16.173 l4/re/elf\_aux.h File Reference

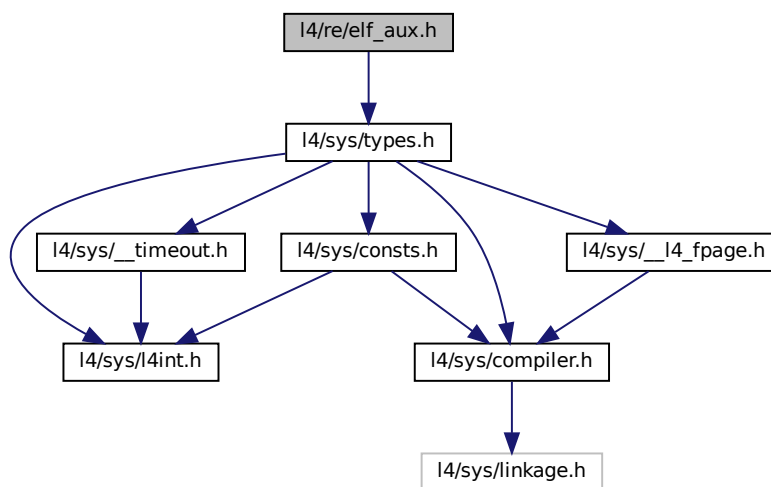
Auxiliary information for binaries.

```

#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](#) }

### 16.173.1 Detailed Description

Auxiliary information for binaries.

Definition in file [elf\\_aux.h](#).

### 16.173.2 Macro Definition Documentation

#### 16.173.2.1 L4RE\_ELF\_AUX\_ELEM

```
#define L4RE_ELF_AUX_ELEM const __attribute__((used, section(".ro_l4re_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](#).

#### 16.173.2.2 L4RE\_ELF\_AUX\_ELEM\_T

```
#define L4RE_ELF_AUX_ELEM_T(
 type,
 id,
 tag,
 val...) static L4RE_ELF_AUX_ELEM type id = {tag, sizeof(type), val}
```

Define an auxiliary vector element.

## Parameters

|             |                                                                                                                        |
|-------------|------------------------------------------------------------------------------------------------------------------------|
| <i>type</i> | is the data type for the element (e.g., <a href="#">l4re_elf_aux_vma_t</a> )                                           |
| <i>id</i>   | is the identifier (variable name) for the declaration (the variable is defined with <code>static</code> storage class) |
| <i>tag</i>  | is the tag value for the element e.g., <a href="#">L4RE_ELF_AUX_T_VMA</a>                                              |
| <i>val</i>  | are the values to be set in the descriptor                                                                             |

## Usage:

```
L4RE_ELF_AUX_ELEM_T(l4re_elf_aux_vma_t, decl_name, L4RE_ELF_AUX_T_VMA, 0x2000, 0x4000);
```

Definition at line 67 of file [elf\\_aux.h](#).

## 16.173.3 Enumeration Type Documentation

### 16.173.3.1 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](#).

## 16.174 elf\_aux.h

```
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 <14/sys/types.h>
```

```

00025
00026
00039
00052 #define L4RE_ELF_AUX_ELEM const __attribute__((used, section(".ro14re_elf_aux"),
 aligned(sizeof(l4_umword_t))))
00053
00067 #define L4RE_ELF_AUX_ELEM_T(type, id, tag, val...) \
00068 static L4RE_ELF_AUX_ELEM type id = {tag, sizeof(type), val}
00069
00070 enum
00071 {
00075 L4RE_ELF_AUX_T_NONE = 0,
00076
00080 L4RE_ELF_AUX_T_VMA,
00081
00086 L4RE_ELF_AUX_T_STACK_SIZE,
00087
00092 L4RE_ELF_AUX_T_STACK_ADDR,
00093
00098 L4RE_ELF_AUX_T_KIP_ADDR,
00099 };
00100
00104 typedef struct l4re_elf_aux_t
00105 {
00106 l4_umword_t type;
00107 l4_umword_t length;
00108 } l4re_elf_aux_t;
00109
00113 typedef struct l4re_elf_aux_vma_t
00114 {
00115 l4_umword_t type;
00116 l4_umword_t length;
00117 l4_umword_t start;
00118 l4_umword_t end;
00119 } l4re_elf_aux_vma_t;
00120
00124 typedef struct l4re_elf_aux_mword_t
00125 {
00126 l4_umword_t type;
00127 l4_umword_t length;
00128 l4_umword_t value;
00129 } l4re_elf_aux_mword_t;
00130

```

## 16.175 l4/re/env File Reference

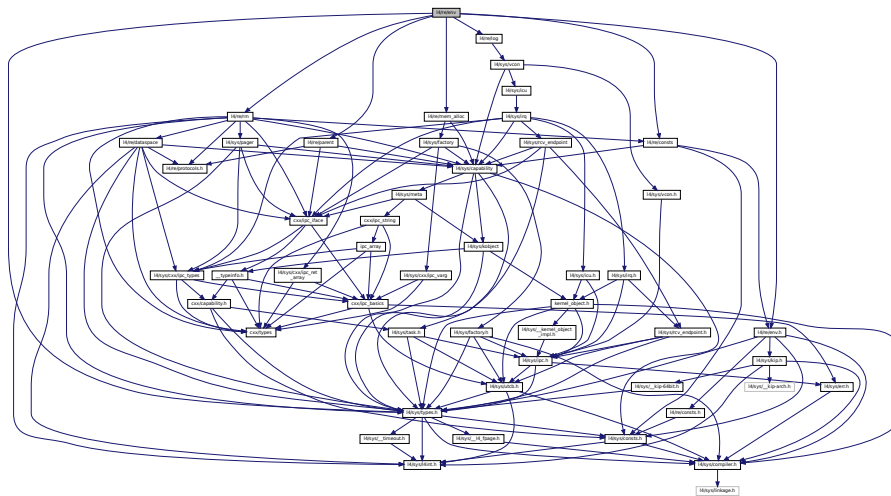
Environment interface.

```

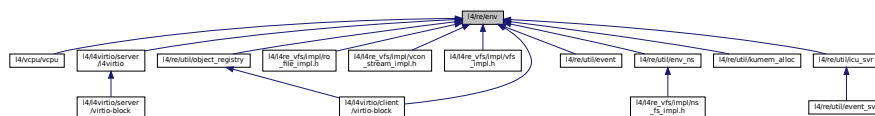
#include <l4/sys/types.h>
#include <l4/re/rm>
#include <l4/re/parent>
#include <l4/re/mem_alloc>
#include <l4/re/log>
#include <l4/re/consts>
#include <l4/re/env.h>

```

Include dependency graph for env:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4Re::Env](#)

*C++ interface of the initial environment that is provided to an [L4](#) task.*

## Namespaces

- [L4](#)  
*L4 low-level kernel interface.*
- [L4Re](#)  
*L4Re C++ Interfaces.*

## 16.175.1 Detailed Description

Environment interface.

Definition in file [env](#).

## 16.176 env

```

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/types.h>
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
00045 namespace L4Re
00046 {
00085 class L4_EXPORT Env
00086 {
00087 private:
00088 l4re_env_t _env;
00089 public:
00094 typedef l4re_env_cap_entry_t Cap_entry;
00095
00103 static Env const *env() noexcept
00104 { return reinterpret_cast<Env*>(l4re_global_env); }
00105
00110 L4::Cap<Parent> parent() const noexcept
00111 { return L4::Cap<Parent>(_env.parent); }
00116 L4::Cap<Mem_alloc> mem_alloc() const noexcept
00117 { return L4::Cap<Mem_alloc>(_env.mem_alloc); }
00121 L4::Cap<L4::Factory> user_factory() const noexcept
00122 { return L4::Cap<L4::Factory>(_env.mem_alloc); }
00127 L4::Cap<Rm> rm() const noexcept
00128 { return L4::Cap<Rm>(_env.rm); }
00133 L4::Cap<Log> log() const noexcept
00134 { return L4::Cap<Log>(_env.log); }
00139 L4::Cap<L4::Thread> main_thread() const noexcept
00140 { return L4::Cap<L4::Thread>(_env.main_thread); }
00145 L4::Cap<L4::Task> task() const noexcept
00146 { return L4::Cap<L4::Task>(L4RE_THIS_TASK_CAP); }
00151 L4::Cap<L4::Factory> factory() const noexcept
00152 { return L4::Cap<L4::Factory>(_env.factory); }
00159 l4_cap_idx_t first_free_cap() const noexcept
00160 { return _env.first_free_cap; }
00165 l4_fpage_t utcb_area() const noexcept
00166 { return _env.utcb_area; }
00174 l4_addr_t first_free_utcb() const noexcept
00175 { return _env.first_free_utcb; }
00176
00181 Cap_entry const *initial_caps() const noexcept
00182 { return _env.caps; }
00183
00192 Cap_entry const *get(char const *name, unsigned l) const noexcept
00193 { return l4re_env_get_cap_l(name, l, &_env); }
00194
00203 template< typename T >
00204 L4::Cap<T> get_cap(char const *name, unsigned l) const noexcept
00205 {
00206 if (Cap_entry const *e = get(name, l))
00207 return L4::Cap<T>(e->cap);
00208 }

```

```

00209 return L4::Cap<T> (~L4_ENOENT);
00210 }
00211
00218 template< typename T >
00219 L4::Cap<T> get_cap(char const *name) const noexcept
00220 { return get_cap<T>(name, __builtin_strlen(name)); }
00221
00226 void parent(L4::Cap<Parent> const &c) noexcept
00227 { _env.parent = c.cap(); }
00232 void mem_alloc(L4::Cap<Mem_alloc> const &c) noexcept
00233 { _env.mem_alloc = c.cap(); }
00238 void rm(L4::Cap<Rm> const &c) noexcept
00239 { _env.rm = c.cap(); }
00244 void log(L4::Cap<Log> const &c) noexcept
00245 { _env.log = c.cap(); }
00250 void main_thread(L4::Cap<L4::Thread> const &c) noexcept
00251 { _env.main_thread = c.cap(); }
00256 void factory(L4::Cap<L4::Factory> const &c) noexcept
00257 { _env.factory = c.cap(); }
00262 void first_free_cap(l4_cap_idx_t c) noexcept
00263 { _env.first_free_cap = c; }
00268 void utcb_area(l4_fpage_t utcb) noexcept
00269 { _env.utcb_area = utcb; }
00274 void first_free_utcb(l4_addr_t u) noexcept
00275 { _env.first_free_utcb = u; }
00276
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 };

```

## 16.177 I4/re/env.h File Reference

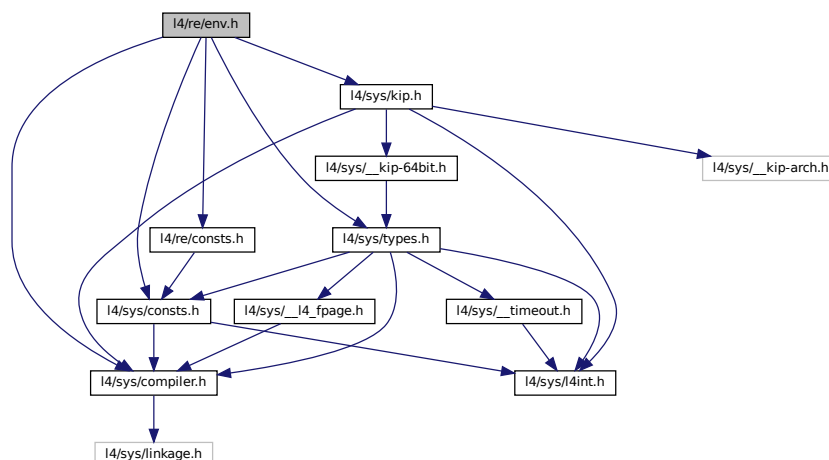
Environment interface.

```

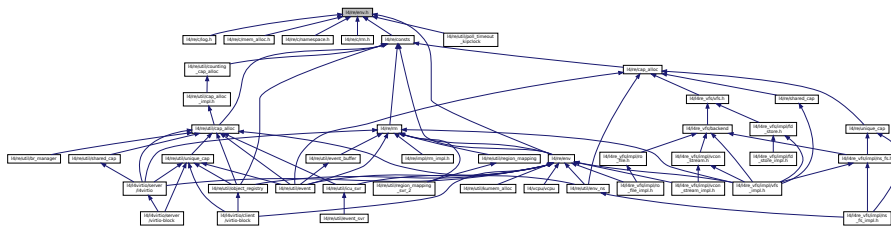
#include <l4/sys/consts.h>
#include <l4/sys/types.h>
#include <l4/sys/kip.h>
#include <l4/sys/compiler.h>
#include <l4/re/consts.h>

```

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 \\* 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.*

### 16.177.1 Detailed Description

Environment interface.

Definition in file [env.h](#).

### 16.177.2 Typedef Documentation



## 16.177.2.1 l4re\_env\_t

```
typedef struct l4re_env_t l4re_env_t
```

Initial environment data structure.

See also

[Initial environment](#)

## 16.178 env.h

```
00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
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 {
00052 l4_cap_idx_t cap;
00053 l4_umword_t flags;
00054 char name[16];
00055 #ifdef __cplusplus
00056 l4re_env_cap_entry_t() L4_NOTHROW : cap(L4_INVALID_CAP), flags(~0) {}
00057 l4re_env_cap_entry_t(char const *n, l4_cap_idx_t c, l4_umword_t f = 0) L4_NOTHROW
00058 : cap(c), flags(f)
00059 {
00060 for (unsigned i = 0; n && i < sizeof(name); ++i, ++n)
00061 {
00062 name[i] = *n;
00063 if (!*n)
00064 break;
00065 }
00066 }
00067 static bool is_valid_name(char const *n) L4_NOTHROW
00068 {
00069 for (unsigned i = 0; *n; ++i, ++n)
00070 if (i > sizeof(name))
00071 return false;
00072 return true;
00073 }
00074 #endif
00075 } l4re_env_cap_entry_t;
00076
00077 typedef struct l4re_env_t
00078 {
00079 l4_cap_idx_t parent;
00080 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;
00121 l4re_env_cap_entry_t *caps;
00122 } l4re_env_t;
00123
00129 extern l4re_env_t *l4re_global_env;
00130
00131
00137 L4_INLINE l4re_env_t *l4re_env(void) L4_NOTHROW;
00138
00139 /*
00140 * FIXME: this seems to be at the wrong place here
00141 */
00147 L4_INLINE l4_kernel_info_t *l4re_kip(void) L4_NOTHROW;
00148
00149
00157 L4_INLINE l4_cap_idx_t
00158 l4re_env_get_cap(char const *name) L4_NOTHROW;
00159
00168 L4_INLINE l4_cap_idx_t
00169 l4re_env_get_cap_e(char const *name, l4re_env_t const *e) L4_NOTHROW;
00170
00181 L4_INLINE l4re_env_cap_entry_t const *
00182 l4re_env_get_cap_l(char const *name, unsigned l, l4re_env_t const *e) L4_NOTHROW;
00183
00184 L4_INLINE
00185 l4re_env_t *l4re_env() L4_NOTHROW
00186 { return l4re_global_env; }
00187
00188 L4_INLINE
00189 l4_kernel_info_t *l4re_kip() L4_NOTHROW
00190 {
00191 extern char __L4_KIP_ADDR__[];
00192 return (l4_kernel_info_t *)__L4_KIP_ADDR__;
00193 }
00194
00195 L4_INLINE l4re_env_cap_entry_t const *
00196 l4re_env_get_cap_l(char const *name, unsigned l, l4re_env_t const *e) L4_NOTHROW
00197 {
00198 l4re_env_cap_entry_t const *c = e->caps;
00199 for (; c && c->flags != ~0UL; ++c)
00200 {
00201 unsigned i;
00202 for (i = 0;
00203 i < sizeof(c->name) && i < l && c->name[i] && name[i] && name[i] == c->name[i];
00204 ++i)
00205 ;
00206 if (i == l && (i == sizeof(c->name) || !c->name[i]))
00207 return c;
00208 }
00209 return NULL;
00210 }
00211
00212 L4_INLINE l4_cap_idx_t
00213 l4re_env_get_cap_e(char const *name, l4re_env_t const *e) L4_NOTHROW
00214 {
00215 unsigned l;
00216 l4re_env_cap_entry_t const *r;
00217 for (l = 0; name[l]; ++l) ;
00218 r = l4re_env_get_cap_l(name, l, e);
00219 if (r)
00220 return r->cap;
00221 return L4_INVALID_CAP;
00222 }
00223
00224 L4_INLINE l4_cap_idx_t
00225 l4re_env_get_cap(char const *name) L4_NOTHROW
00226 { return l4re_env_get_cap_e(name, l4re_env()); }
00227
00228

```

## 16.179 l4/re/error\_helper File Reference

Error helper.

Include dependency graph for error\_helper:



- 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.*

### 16.179.1 Detailed Description

Error helper.

Definition in file [error\\_helper](#).

## 16.180 error\_helper

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00009 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #pragma once
00026
00027 #include <l4/sys/types.h>
00028 #include <l4/cxx/exceptions>
00029 #include <l4/cxx/type_traits>
00030 #include <l4/sys/err.h>
00031
00032 namespace L4Re {
00033
00034 #ifdef __EXCEPTIONS
00035
00045 [[noreturn]] inline void throw_error(long err, char const *extra = "")
00046 {
00047 switch (err)
00048 {
00049 case -L4_ENOENT: throw (L4::Element_not_found(extra));
00050 case -L4_ENOMEM: throw (L4::Out_of_memory(extra));
00051 case -L4_EEXIST: throw (L4::Element_already_exists(extra));
00052 case -L4_ERANGE: throw (L4::Bounds_error(extra));
00053 default: throw (L4::Runtime_error(err, extra));
00054 }
00055 }
00056
00067 inline

```

```

00068 long chksys(long err, char const *extra = "", long ret = 0)
00069 {
00070 if (L4_UNLIKELY(err < 0))
00071 throw_error(ret ? ret : err, extra);
00072
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 }

```

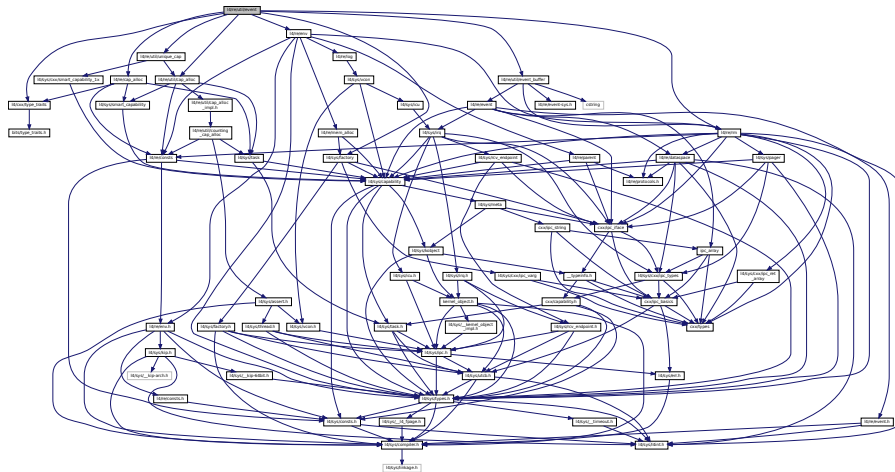
## 16.181 l4/re/util/event File Reference

```

#include <l4/re/cap_alloc>
#include <l4/re/util/cap_alloc>
#include <l4/re/util/unique_cap>
#include <l4/re/env>
#include <l4/re/rm>
#include <l4/re/util/event_buffer>
#include <l4/sys/factory>

```

```
#include <l4/cxx/type_traits>
Include dependency graph for event:
```



## Data Structures

- class [L4Re::Util::Event\\_t< PAYLOAD >](#)  
*Convenience wrapper for getting access to an event object.*

## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*
- [L4Re::Util](#)  
*Documentation of the L4 Runtime Environment utility functionality in C++.*

## 16.182 event

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #pragma once
00024
00025 #include <l4/re/cap_alloc>
00026 #include <l4/re/util/cap_alloc>
00027 #include <l4/re/util/unique_cap>
00028 #include <l4/re/env>
00029 #include <l4/re/rm>
00030 #include <l4/re/util/event_buffer>
00031 #include <l4/sys/factory>
```

```

00032 #include <l4/cxx/type_traits>
00033
00034 namespace L4Re { namespace Util {
00035
00042 template< typename PAYLOAD >
00043 class Event_t
00044 {
00045 public:
00049 enum Mode
00050 {
00051 Mode_irq,
00052 Mode_polling,
00053 };
00054
00069 template<typename IRQ_TYPE>
00070 int init(L4::Cap<L4Re::Event> event,
00071 L4Re::Env const *env = L4Re::Env::env(),
00072 L4Re::Cap_alloc *ca = L4Re::Cap_alloc::get_cap_alloc(L4Re::Util::cap_alloc))
00073 {
00074 Unique_cap<L4Re::Dataspace> ev_ds(ca->alloc<L4Re::Dataspace>());
00075 if (!ev_ds.is_valid())
00076 return -L4_ENOMEM;
00077
00078 int r;
00079
00080 Unique_del_cap<IRQ_TYPE> ev_irq(ca->alloc<IRQ_TYPE>());
00081 if (!ev_irq.is_valid())
00082 return -L4_ENOMEM;
00083
00084 if ((r = l4_error(env->factory()->create(ev_irq.get()))))
00085 return r;
00086
00087 if ((r = l4_error(event->bind(0, ev_irq.get()))))
00088 return r;
00089
00090 if ((r = event->get_buffer(ev_ds.get())))
00091 return r;
00092
00093 long sz = ev_ds->size();
00094 if (sz < 0)
00095 return sz;
00096
00097 Rm::Unique_region<void*> buf;
00098
00099 if ((r = env->rm()->attach(&buf, sz,
00100 L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00101 L4::Ipc::make_cap_rw(ev_ds.get()))))
00102 return r;
00103
00104 _ev_buffer = L4Re::Event_buffer_t<PAYLOAD>(buf.get(), sz);
00105 _ev_ds = cxx::move(ev_ds);
00106 _ev_irq = cxx::move(ev_irq);
00107 _buf = cxx::move(buf);
00108
00109 return 0;
00110 }
00111
00123 int init_poll(L4::Cap<L4Re::Event> event,
00124 L4Re::Env const *env = L4Re::Env::env(),
00125 L4Re::Cap_alloc *ca = L4Re::Cap_alloc::get_cap_alloc(L4Re::Util::cap_alloc))
00126 {
00127 Unique_cap<L4Re::Dataspace> ev_ds(ca->alloc<L4Re::Dataspace>());
00128 if (!ev_ds.is_valid())
00129 return -L4_ENOMEM;
00130
00131 int r;
00132
00133 if ((r = event->get_buffer(ev_ds.get())))
00134 return r;
00135
00136 long sz = ev_ds->size();
00137 if (sz < 0)
00138 return sz;
00139
00140 Rm::Unique_region<void*> buf;
00141
00142 if ((r = env->rm()->attach(&buf, sz,
00143 L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00144 L4::Ipc::make_cap_rw(ev_ds.get()))))
00145 return r;
00146
00147 _ev_buffer = L4Re::Event_buffer_t<PAYLOAD>(buf.get(), sz);
00148 _ev_ds = cxx::move(ev_ds);
00149 _buf = cxx::move(buf);
00150
00151 return 0;
00152 }

```





### 16.183.1 Detailed Description

Dataspace client stub implementation.

Definition in file [dataspace\\_impl.h](#).

## 16.184 dataspace\_impl.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #include <l4/re/dataspace>
00024 #include <l4/sys/cxx/ipc_client>
00025 #include <l4/sys/cxx/consts>
00026
00027 L4_RPC_DEF(L4Re::Dataspace::clear);
00028 L4_RPC_DEF(L4Re::Dataspace::allocate);
00029 L4_RPC_DEF(L4Re::Dataspace::copy_in);
00030 L4_RPC_DEF(L4Re::Dataspace::info);
00031
00032 namespace L4Re {
00033
00034
00035 long
00036 Dataspace::__map(Dataspace::Offset offset, unsigned char *size,
00037 Dataspace::Flags flags,
00038 Dataspace::Map_addr local_addr) const noexcept
00039 {
00040 Map_addr spot = local_addr & ~(~0ULL « L4_umword_t(*size));
00041 Map_addr base = local_addr & (~0ULL « L4_umword_t(*size));
00042 L4::Ipc::Rcv_fpage r;
00043 r = L4::Ipc::Rcv_fpage::mem(base, *size, 0);
00044
00045 L4::Ipc::Snd_fpage fp;
00046 long err = map_t::call(c(), offset, spot, flags, r, fp, L4_utcb());
00047 if (L4_UNLIKELY(err < 0))
00048 return err;
00049
00050 *size = fp.rcv_order();
00051 return err;
00052 }
00053
00054 long
00055 Dataspace::map_region(Dataspace::Offset offset, Dataspace::Flags flags,
00056 Dataspace::Map_addr min_addr,
00057 Dataspace::Map_addr max_addr) 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
00093 long
00094 Dataspace::map(Dataspace::Offset offset, Dataspace::Flags flags,
00095 Dataspace::Map_addr local_addr,
00096 Dataspace::Map_addr min_addr,
00097 Dataspace::Map_addr max_addr) const noexcept
00098 {
00099 min_addr = L4::trunc_page(min_addr);
00100 max_addr = L4::round_page(max_addr);
00101 local_addr = L4::trunc_page(local_addr);
00102 unsigned char order
00103 = L4::max_order(L4_LOG2_PAGESIZE, local_addr, min_addr, max_addr, local_addr);
00104
00105 return __map(offset, &order, flags, local_addr);
00106 }
00107
00108 Dataspace::Size
00109 Dataspace::size() const noexcept
00110 {
00111 Stats stats = Stats();
00112 int err = info(&stats);
00113 if (err < 0)
00114 return 0;
00115 return stats.size;
00116 }
00117
00118 Dataspace::Flags
00119 Dataspace::flags() const noexcept
00120 {
00121 Stats stats = Stats();
00122 int err = info(&stats);
00123 if (err < 0)
00124 return Flags(0);
00125 return stats.flags;
00126 }
00127
00128 };

```

## 16.185 l4/re/impl/mem\_alloc\_impl.h File Reference

Memory allocator client stub implementation.

```

#include <l4/re/mem_alloc>
#include <l4/re/mem_alloc-sys.h>
#include <l4/re/dataspace>
#include <l4/re/error_helper>
#include <l4/sys/factory>

```

- **L4Re**  
*L4Re C++ Interfaces.*

Definition in file [mem\\_alloc\\_impl.h](#).

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
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

```



## 16.187.1 Detailed Description

Namespace client stub implementation.

Definition in file [namespace\\_impl.h](#).

## 16.188 namespace\_impl.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #include <l4/re/namespace>
00024
00025 #include <l4/util/util.h>
00026 #include <l4/sys/cxx/ipc_client>
00027 #include <l4/sys/assert.h>
00028
00029 #include <cstring>
00030
00031 L4_RPC_DEF(L4Re::Namespace::query);
00032 L4_RPC_DEF(L4Re::Namespace::register_obj);
00033 L4_RPC_DEF(L4Re::Namespace::unlink);
00034
00035 namespace L4Re {
00036
00037 long
00038 Namespace::_query(char const *name, unsigned len,
00039 L4::Cap<void> const &target,
00040 l4_umword_t *local_id, bool iterate) const noexcept
00041 {
00042 l4_assert(target.is_valid());
00043 L4::Cap<Namespace> ns = c();
00044 L4::Ipc::Array<char const, unsigned long> _name(len, name);
00045 while (_name.length > 0)
00046 {
00047 L4::Ipc::Snd_fpage cap;
00048 L4::Opcode dummy;
00049 int err = query_t::call(ns, _name,
00050 L4::Ipc::Small_buf(target.cap(),
00051 local_id
00052 ? L4_RCV_ITEM_LOCAL_ID
00053 : 0),
00054 cap, dummy, _name);
00055 if (err < 0)
00056 return err;
00057 bool const partly = err & Partly_resolved;
00058 if (cap.id_received())
00059 {
00060 *local_id = cap.data();
00061 return _name.length;
00062 }
00063 if (partly && iterate)
00064 ns = L4::cap_cast<Namespace>(target);
00065 else
00066 return err;
00067 }
00068 return _name.length;
00069 }
00070
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 long ret;
00084 long rem = timeout;
00085 long to = 0;
00086
00087 if (rem)
00088 to = 10;
00089 do
00090 {
00091 ret = _query(name, len, target, local_id, iterate);
00092
00093 if (ret >= 0)
00094 return ret;
00095
00096 if (L4_UNLIKELY(ret != -L4_EAGAIN))
00097 return ret;
00098
00099 if (rem == to)
00100 return ret;
00101
00102 l4_sleep(to);
00103
00104 if (rem > 0)
00105 {
00106 rem -= to;
00107 if (to > rem)
00108 to = rem;
00109 }
00110
00111 if (to < 100)
00112 to += to;
00113 }
00114 while (486);
00115 }
00116
00117 long
00118 Namespace::query(char const *name, L4::Cap<void> const &target,
00119 int timeout, l4_umword_t *local_id,
00120 bool iterate) const noexcept
00121 {
00122 return query(name, __builtin_strlen(name), target,
00123 timeout, local_id, iterate);
00124 }
00125
00126 }

```

## 16.189 l4/re/impl/rm\_impl.h File Reference

Region map client stub implementation.

```

#include <l4/re/rm>
#include <l4/re/dataspace>
#include <l4/sys/cxx/ipc_client>
#include <l4/sys/task>
#include <l4/sys/err.h>

```



```

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 }

```

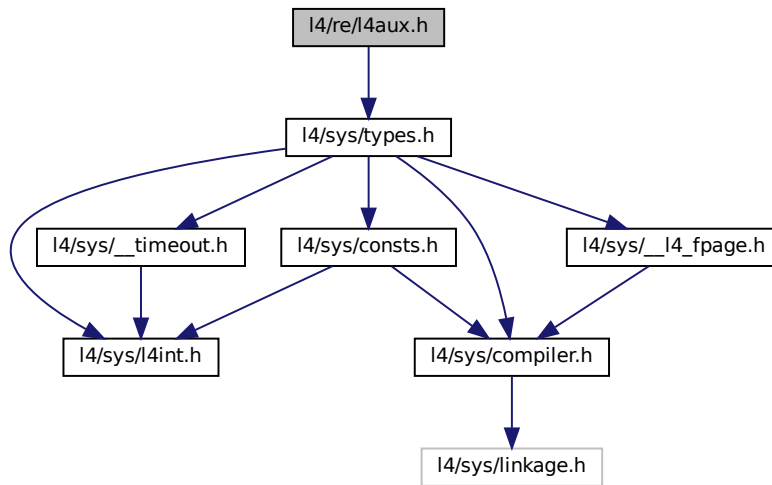
## 16.191 l4/re/l4aux.h File Reference

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.*

### 16.191.1 Detailed Description

Auxiliary definitions.

Definition in file [l4aux.h](#).

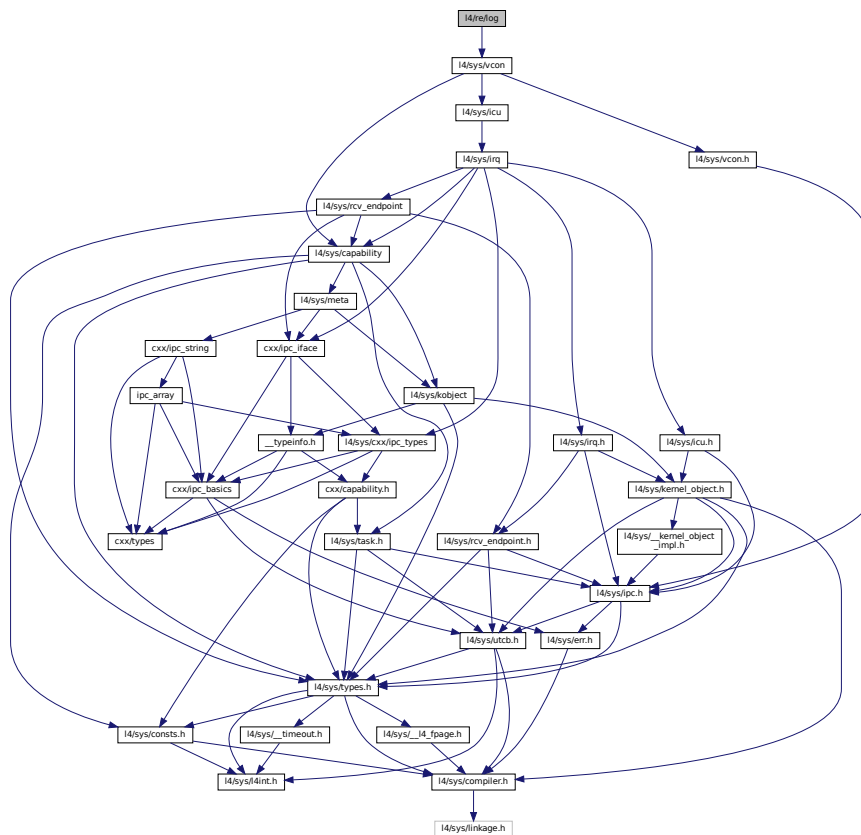
## 16.192 l4aux.h

```
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
```

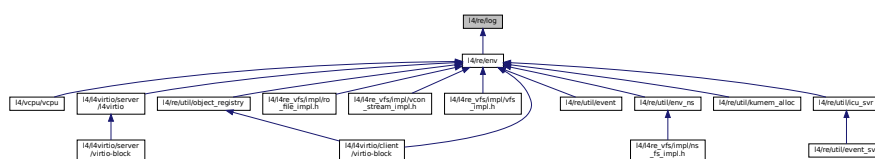
## 16.193 l4/re/log File Reference

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

- L4Re
- L4Re C++ Interfaces.*

## 16.193.1 Detailed Description

Log interface.

Definition in file [log](#).

## 16.194 log

```
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 printn(char const *string, int len) const noexcept;
00055
00061 void print(char const *string) const noexcept;
00062 };
00063 }
```

## 16.195 l4/re/log-sys.h File Reference

Log protocol definition.

### Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

### Enumerations

- enum [L4Re::Log\\_::Opcodes](#)  
*Logging-service communication-protocol opcodes.*

## 16.195.1 Detailed Description

Log protocol definition.

Definition in file [log-sys.h](#).

## 16.196 log-sys.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
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 };

```

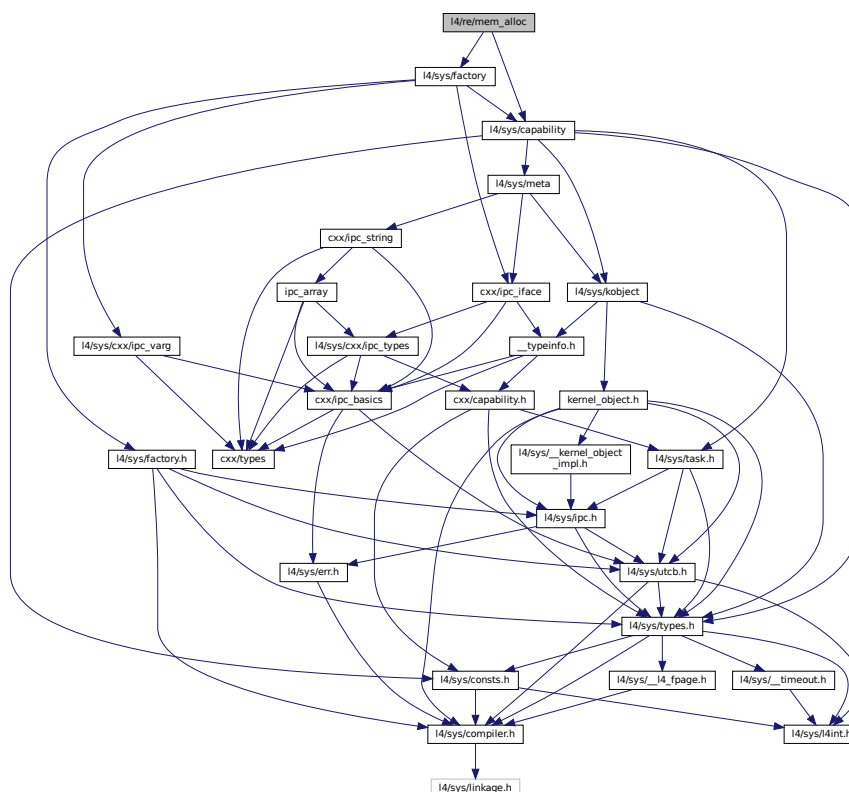
### 16.197 I4/re/mem\_alloc File Reference

Memory allocator interface.

```
#include <linux/sys/capability>
```

```
#include <l4/sys/factory>
```

Include dependency graph for mem alloc:





```

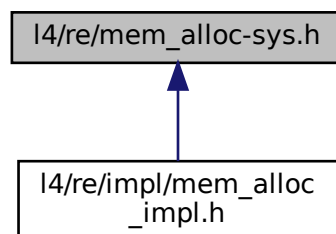
00040 // * shall we support superpages in noncont memory?
00041
00061 class L4_EXPORT Mem_alloc :
00062 public L4::Kobject_t<Mem_alloc, L4::Factory, L4::PROTO_EMPTY>
00063 {
00064 public:
00071 enum Mem_alloc_flags
00072 {
00073 Continuous = 0x01,
00074 Pinned = 0x02,
00075 Super_pages = 0x04,
00076 };
00077
00104 long alloc(long size, L4::Cap<Dataspace> mem,
00105 unsigned long flags = 0, unsigned long align = 0) const noexcept;
00106
00119 /* Deprecation message added Q4 2016 */
00120 long free(L4::Cap<Dataspace> mem) const noexcept
00121 L4_DEPRECATED("This function is an empty stub and remains for backward compatibility only. Check
00122 documentation for details.");
00123 };
00124
00125 };

```

## 16.199 l4/re/mem\_alloc-sys.h File Reference

Memory allocator protocol definitions.

This graph shows which files directly or indirectly include this file:



## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

## Enumerations

- enum [L4Re::Mem\\_alloc\\_::Opcodes](#)  
*Memory-allocator communication-protocol opcodes.*

### 16.199.1 Detailed Description

Memory allocator protocol definitions.

Definition in file [mem\\_alloc-sys.h](#).

## 16.200 mem\_alloc-sys.h

```
00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
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 Mem_alloc_
00028 {
00034 enum Opcodes { Alloc, Free };
00035 };
00036 };
```

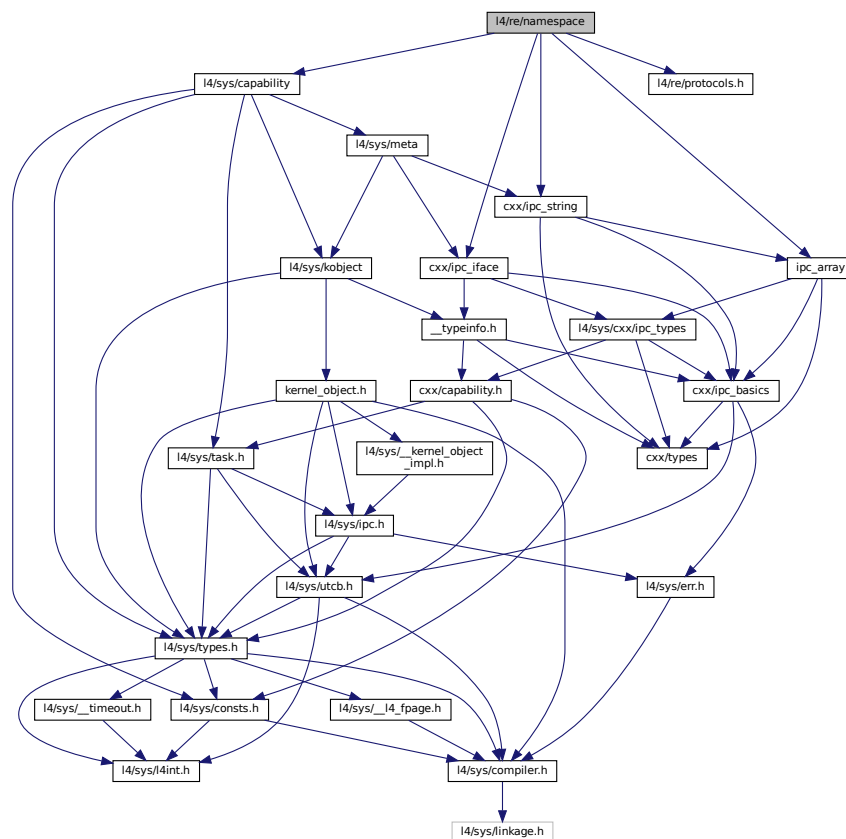
## 16.201 l4/re/namespace File Reference

Namespace interface.

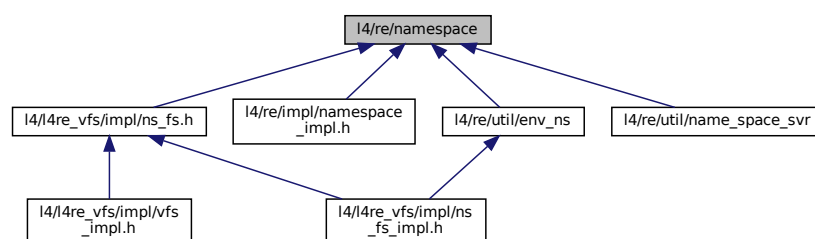
```
#include <l4/sys/capability>
#include <l4/re/protocols.h>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/ipc_array>
#include <l4/sys/cxx/ipc_string>
```



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

- [L4Re](#)  
*L4Re C++ Interfaces.*

## 16.201.1 Detailed Description

Namespace interface.

Definition in file [namespace](#).

## 16.202 namespace

```

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<l> >
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
00093 enum Query_timeout
00094 {
00095 To_default = 3600000,
00096 To_non_blocking = 0,
00097 To_forever = -1,
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
00202 long unlink(char const* name)
00203 {
00204 return unlink_t::call(c(), L4::Ipc::Array<char const, unsigned long>(
00205 __builtin_strlen(name), name));
00206 }
00207
00208 typedef L4::Typeid::Rpc<query_t, register_obj_t, unlink_t> Rpc;
00209
00210 private:
00211 long _query(char const *name, unsigned len,
00212 L4::Cap<void> const &target, l4_umword_t *local_id,
00213 bool iterate) const noexcept;
00214
00215 };
00216
00217 };

```

## 16.203 l4/re/namespace-sys.h File Reference

Namespace protocol definitions.

### Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

### Enumerations

- enum [L4Re::Namespace\\_::Opcodes](#)  
*Name-space communication-protocol opcodes.*

### 16.203.1 Detailed Description

Namespace protocol definitions.

Definition in file [namespace-sys.h](#).

## 16.204 namespace-sys.h

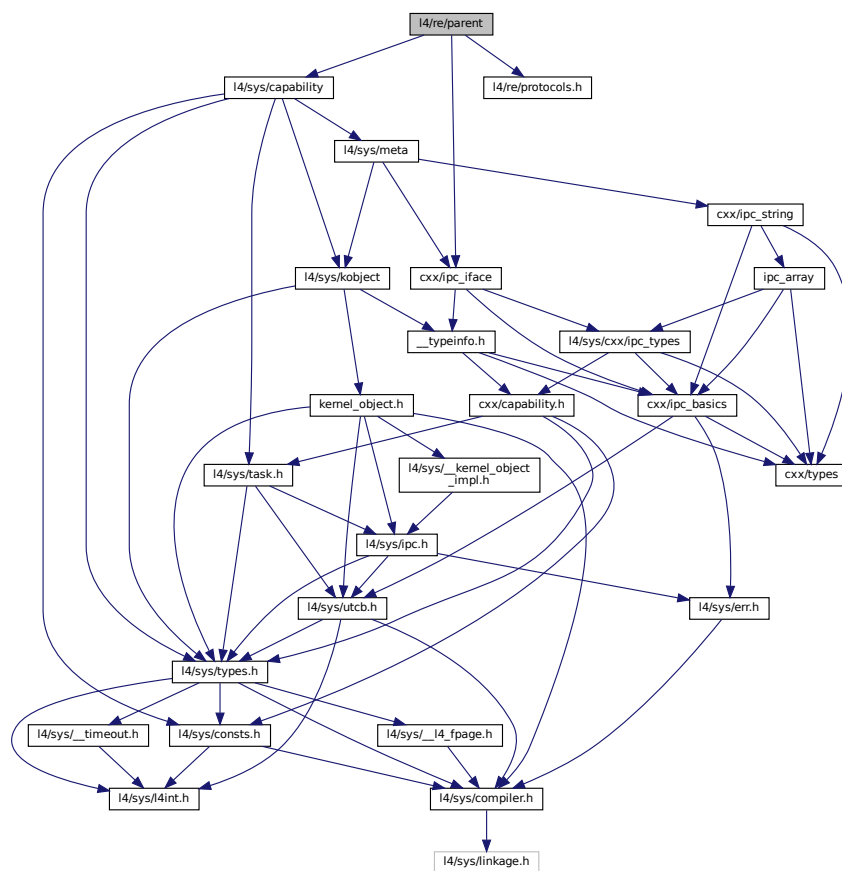
```
00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General 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 };
```

## 16.205 l4/re/parent File Reference

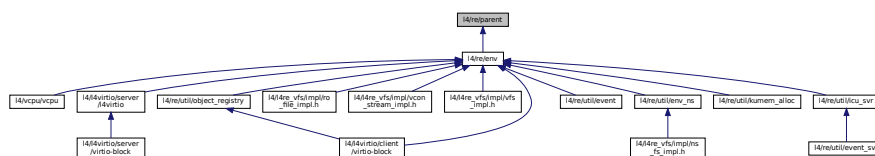
Parent interface.

```
#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

- L4Re
- L4Re C++ Interfaces.*

## 16.205.1 Detailed Description

Parent interface.

Definition in file [parent](#).

## 16.206 parent

```

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 #pragma once
00022
00023 #include <l4/sys/capability>
00024 #include <l4/re/protocols.h>
00025 #include <l4/sys/cxx/ipc_iface>
00026
00027 namespace L4Re {
00028
00029 class L4_EXPORT Parent {
00030 public:
00031 L4_INLINE_RPC(long, signal, (unsigned long sig, unsigned long val));
00032 typedef L4::Typeid::Rpc<signal_t> Rpc;
00033 };
00034
00035 }
```

## 16.207 l4/re/parent-sys.h File Reference

Parent protocol definition.

### Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

### Enumerations

- enum [L4Re::Parent\\_::Opcodes](#)  
*Parent communication-protocol opcodes.*

## 16.207.1 Detailed Description

Parent protocol definition.

Definition in file [parent-sys.h](#).

## 16.208 parent-sys.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
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 };

```

## 16.209 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 }

## 16.209.1 Detailed Description

[L4Re](#) Protocol Constants (C version)

Definition in file [protocols.h](#).

## 16.209.2 Enumeration Type Documentation

### 16.209.2.1 L4re\_protocols

enum [L4re\\_protocols](#)

Enumerator

|                       |                                               |
|-----------------------|-----------------------------------------------|
| L4RE_PROTO_DATASPACE  | ID for <a href="#">L4Re::Dataspace</a> RPCs   |
| L4RE_PROTO_NAMESPACE  | ID for <a href="#">L4Re::Namespace</a> RPCs   |
| L4RE_PROTO_PARENT     | ID for <a href="#">L4Re::Parent</a> RPCs      |
| L4RE_PROTO_GOOS       | ID for <a href="#">L4Re::Video::Goos</a> RPCs |
| L4RE_PROTO_RSVD_1     | Reserved ID                                   |
| L4RE_PROTO_RM         | ID for <a href="#">L4Re::Rm</a> RPCs          |
| L4RE_PROTO_EVENT      | ID for <a href="#">L4Re::Event</a> RPCs       |
| L4RE_PROTO_INHIBITOR  | ID for <a href="#">L4Re::Inhibitor</a> RPCs   |
| L4RE_PROTO_DMA_SPACE  | ID for <a href="#">L4Re::Dma_space</a> RPCs   |
| L4RE_PROTO_MMIO_SPACE | ID for <a href="#">L4Re::Mmio_space</a>       |
| L4RE_PROTO_DEBUG      | ID for debugging RPCs                         |

Definition at line 30 of file [protocols.h](#).

## 16.210 protocols.h

```

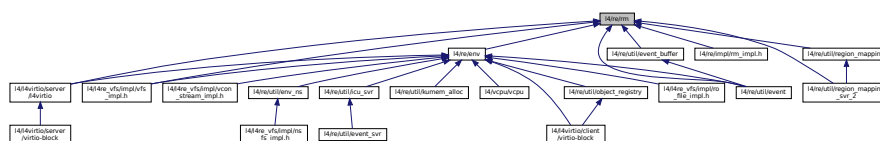
00001
00006 /*
00007 * (c) 2015 Alexander Warg <alexander.warg@kernkonzept.com>
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may 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,
```



## 16.211 l4/re/rm File Reference

```
#include <l4/sys/types.h>
#include <l4/sys/l4int.h>
#include <l4/sys/capability>
#include <l4/re/protocols.h>
#include <l4/sys/pager>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/ipc_ret_array>
#include <l4/sys/cxx/types>
#include <l4/re/consts>
#include <l4/re/dataspace>
```

**Include dependency graph for rm:**



## Data Structures

- class [L4Re::Rm](#)  
*Region map.*
- struct [L4Re::Rm::F](#)  
*Rm flags definitions.*
- struct [L4Re::Rm::Range](#)  
*A range of virtual addresses.*

## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

### 16.211.1 Detailed Description

Region mapper interface.

Definition in file [rm](#).

## 16.212 rm

```

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
00073 class L4_EXPORT Rm :
00074 public L4::Kobject_t<Rm, L4::Pager, L4RE_PROTO_RM,
00075 L4::Type_info::Demand_t<1> >
00076 {
00077 public:
00078 typedef L4Re::Dataspace::Offset Offset;
00079
00081 enum Detach_result
00082 {

```

```

00083 Detached_ds = 0,
00084 Kept_ds = 1,
00085 Split_ds = 2,
00086 Detach_result_mask = 3,
00087
00088 Detach_again = 4,
00089 };
00090
00091
00092 enum Region_flag_shifts
00093 {
00094 Caching_shift = Dataspace::F::Caching_shift,
00095 };
00096
00097
00098 struct F
00099 {
00100 enum Attach_flags : l4_uint32_t
00101 {
00102 Search_addr = 0x20000,
00103 In_area = 0x40000,
00104 Eager_map = 0x80000,
00105 Attach_mask = 0xf0000,
00106 };
00107
00108 L4_TYPES_FLAGS_OPS_DEF(Attach_flags);
00109
00110 enum Region_flags : l4_uint16_t
00111 {
00112 Rights_mask = 0x0f,
00113 R = Dataspace::F::R,
00114 W = Dataspace::F::W,
00115 X = Dataspace::F::X,
00116 RW = Dataspace::F::RW,
00117 RX = Dataspace::F::RX,
00118 RWX = Dataspace::F::RWX,
00119
00120 Detach_free = 0x200,
00121 Pager = 0x400,
00122 Reserved = 0x800,
00123
00124 Caching_mask = Dataspace::F::Caching_mask,
00125 Cache_normal = Dataspace::F::Normal,
00126 Cache_buffered = Dataspace::F::Bufferable,
00127 Cache_uncached = Dataspace::F::Uncacheable,
00128
00129 Ds_map_mask = 0xff,
00130
00131 Region_flags_mask = 0xffff,
00132 };
00133
00134 L4_TYPES_FLAGS_OPS_DEF(Region_flags);
00135
00136 friend constexpr Dataspace::Flags map_flags(Region_flags rf)
00137 {
00138 return Dataspace::Flags((l4_uint16_t)rf & Ds_map_mask);
00139 }
00140
00141 struct Flags : L4::Types::Flags_ops_t<Flags>
00142 {
00143 l4_uint32_t raw;
00144 Flags() = default;
00145 explicit constexpr Flags(l4_uint32_t f) : raw(f) {}
00146 constexpr Flags(Attach_flags rf) : raw((l4_uint32_t)rf) {}
00147 constexpr Flags(Region_flags rf) : raw((l4_uint32_t)rf) {}
00148
00149 friend constexpr Dataspace::Flags map_flags(Flags f)
00150 {
00151 return Dataspace::Flags(f.raw & Ds_map_mask);
00152 }
00153
00154 constexpr Region_flags region_flags() const
00155 {
00156 return Region_flags(raw & Region_flags_mask);
00157 }
00158
00159 constexpr Attach_flags attach_flags() const
00160 {
00161 return Attach_flags(raw & Attach_mask);
00162 }
00163
00164 constexpr bool r() const { return raw & L4_FPAGE_RO; }
00165 constexpr bool w() const { return raw & L4_FPAGE_W; }
00166 constexpr bool x() const { return raw & L4_FPAGE_X; }
00167 constexpr unsigned cap_rights() const
00168 {
00169 return w() ? L4_CAP_FPAGE_RW : L4_CAP_FPAGE_RO;
00170 }
00171 };
00172
00173 constexpr bool r() const { return raw & L4_FPAGE_RO; }
00174 constexpr bool w() const { return raw & L4_FPAGE_W; }
00175 constexpr bool x() const { return raw & L4_FPAGE_X; }
00176 constexpr unsigned cap_rights() const
00177 {
00178 return w() ? L4_CAP_FPAGE_RW : L4_CAP_FPAGE_RO;
00179 }
00180 };

```

```

00187
00188 friend constexpr Flags operator | (Region_flags l, Attach_flags r)
00189 { return Flags(l) | Flags(r); }
00190
00191 friend constexpr Flags operator | (Attach_flags l, Region_flags r)
00192 { return Flags(l) | Flags(r); }
00193 };
00194
00195 using Attach_flags = F::Attach_flags;
00196 using Region_flags = F::Region_flags;
00197 using Flags = F::Flags;
00198
00200 enum Detach_flags
00201 {
00211 Detach_exact = 1,
00221 Detach_overlap = 2,
00222
00230 Detach_keep = 4,
00231 };
00232
00258 long reserve_area(l4_addr_t *start, unsigned long size,
00259 Flags flags = Flags(0),
00260 unsigned char align = L4_PAGESHIFT) const noexcept
00261 { return reserve_area_t::call(c(), start, size, flags, align); }
00262
00263 L4_RPC_NF(long, reserve_area, (L4::Ipc::In_out<l4_addr_t *> start,
00264 unsigned long size,
00265 Flags flags,
00266 unsigned char align));
00267
00283 template< typename T >
00284 long reserve_area(T **start, unsigned long size,
00285 Flags flags = Flags(0),
00286 unsigned char align = L4_PAGESHIFT) const noexcept
00287 { return reserve_area_t::call(c(), (l4_addr_t*)start, size, flags, align); }
00288
00301 L4_RPC(long, free_area, (l4_addr_t addr));
00302
00303 L4_RPC_NF(long, attach, (L4::Ipc::In_out<l4_addr_t *> start,
00304 unsigned long size, Flags flags,
00305 L4::Ipc::Opt<L4::Ipc::Cap<Dataspace> > mem,
00306 Offset offs, unsigned char align,
00307 L4::Ipc::Opt<l4_cap_idx_t> client_cap));
00308
00309 L4_RPC_NF(long, detach, (l4_addr_t addr, unsigned long size, unsigned flags,
00310 l4_addr_t &start, l4_addr_t &rsz,
00311 l4_cap_idx_t &mem_cap));
00312
00361 long attach(l4_addr_t *start, unsigned long size, Flags flags,
00362 L4::Ipc::Cap<Dataspace> mem, Offset offs = 0,
00363 unsigned char align = L4_PAGESHIFT) const noexcept;
00364
00368 template< typename T >
00369 long attach(T **start, unsigned long size, Flags flags,
00370 L4::Ipc::Cap<Dataspace> mem, Offset offs = 0,
00371 unsigned char align = L4_PAGESHIFT) const noexcept
00372 {
00373 union X { l4_addr_t a; T* t; };
00374 X *x = reinterpret_cast<X*>(start);
00375 return attach(&x->a, size, flags, mem, offs, align);
00376 }
00377
00378 #if __cplusplus >= 201103L
00379 template< typename T >
00380 class Unique_region
00381 {
00382 private:
00383 T _addr;
00384 L4::Cap<Rm> _rm;
00385
00386 public:
00387 Unique_region(Unique_region const &) = delete;
00388 Unique_region &operator = (Unique_region const &) = delete;
00389
00390 Unique_region() noexcept
00391 : _addr(0), _rm(L4::Cap<Rm>::Invalid) {}
00392
00393 explicit Unique_region(T addr) noexcept
00394 : _addr(addr), _rm(L4::Cap<Rm>::Invalid) {}
00395
00396 Unique_region(T addr, L4::Cap<Rm> const &rm) noexcept
00397 : _addr(addr), _rm(rm) {}
00398
00399 Unique_region(Unique_region &&o) noexcept : _addr(o.get()), _rm(o._rm)
00400 { o.release(); }
00401
00402 Unique_region &operator = (Unique_region &&o) noexcept

```

```

00403 {
00404 if (&o != this)
00405 {
00406 if (_rm.is_valid())
00407 _rm->detach(l4_addr_t(_addr), 0);
00408 _rm = o._rm;
00409 _addr = o.release();
00410 }
00411 return *this;
00412 }
00413
00414 ~Unique_region() noexcept
00415 {
00416 if (_rm.is_valid())
00417 _rm->detach(l4_addr_t(_addr), 0);
00418 }
00419
00420 T get() const noexcept
00421 { return _addr; }
00422
00423 T release() noexcept
00424 {
00425 _rm = L4::Cap<Rm>::Invalid;
00426 return _addr;
00427 }
00428
00429 void reset(T addr, L4::Cap<Rm> const &rm) noexcept
00430 {
00431 if (_rm.is_valid())
00432 _rm->detach(l4_addr_t(_addr), 0);
00433
00434 _rm = rm;
00435 _addr = addr;
00436 }
00437
00438 void reset() noexcept
00439 { reset(0, L4::Cap<Rm>::Invalid); }
00440
00441 bool is_valid() const noexcept
00442 { return _rm.is_valid(); }
00443
00444 T operator * () const noexcept { return _addr; }
00445
00446 T operator -> () const noexcept { return _addr; }
00447 };
00448
00449
00450
00451 template< typename T >
00452 long attach(Unique_region<T> *start, unsigned long size, Flags flags,
00453 L4::Ipc::Cap<Dataspace> mem, Offset offs = 0,
00454 unsigned char align = L4_PAGESHIFT) const noexcept
00455 {
00456 l4_addr_t addr = (l4_addr_t)start->get();
00457
00458 long res = attach(&addr, size, flags, mem, offs, align);
00459 if (res < 0)
00460 return res;
00461
00462 start->reset((T)addr, L4::Cap<Rm>(cap()));
00463 return res;
00464 }
00465 #endif
00466
00467 int detach(l4_addr_t addr, L4::Cap<Dataspace> *mem,
00468 L4::Cap<L4::Task> const &task = This_task) const noexcept;
00469
00470 int detach(void *addr, L4::Cap<Dataspace> *mem,
00471 L4::Cap<L4::Task> const &task = This_task) const noexcept;
00472
00473 int detach(l4_addr_t start, unsigned long size, L4::Cap<Dataspace> *mem,
00474 L4::Cap<L4::Task> const &task) const noexcept;
00475
00476 int find(l4_addr_t *addr, unsigned long *size, Offset *offset,
00477 L4Re::Rm::Flags *flags, L4::Cap<Dataspace> *m) noexcept
00478 { return find_t::call(c(), addr, size, flags, offset, m); }
00479
00480 L4_RPC_NF(int, find, (L4::Ipc::In_out<l4_addr_t *> addr,
00481 L4::Ipc::In_out<unsigned long *> size,
00482 L4Re::Rm::Flags *flags, Offset *offset,
00483 L4::Ipc::As_value<L4::Cap<Dataspace> > *m));
00484
00485 struct Range
00486 {
00487 l4_addr_t start;
00488 l4_addr_t end;
00489 };
00490
00491 using Region = Range;

```

```

00585
00592 using Area = Range;
00593
00608 L4_RPC(long, get_regions, (l4_addr_t start, L4::Ipc::Ret_array<Range> regions));
00609
00625 L4_RPC(long, get_areas, (l4_addr_t start, L4::Ipc::Ret_array<Range> areas));
00626
00627 int detach(l4_addr_t start, unsigned long size, L4::Cap<Dataspace> *mem,
00628 L4::Cap<L4::Task> task, unsigned flags) const noexcept;
00629
00630 typedef L4::Typeid::Rpc<attach_t, detach_t, find_t,
00631 reserve_area_t, free_area_t,
00632 get_regions_t, get_areas_t> Rpc<
00633 >;
00634
00635
00636 inline int
00637 Rm::detach(l4_addr_t addr, L4::Cap<Dataspace> *mem,
00638 L4::Cap<L4::Task> const &task) const noexcept
00639 { return detach(addr, 1, mem, task, Detach_overlap); }
00640
00641 inline int
00642 Rm::detach(void *addr, L4::Cap<Dataspace> *mem,
00643 L4::Cap<L4::Task> const &task) const noexcept
00644 { return detach((l4_addr_t)addr, 1, mem, task, Detach_overlap); }
00645
00646 inline int
00647 Rm::detach(l4_addr_t addr, unsigned long size, L4::Cap<Dataspace> *mem,
00648 L4::Cap<L4::Task> const &task) const noexcept
00649 { return detach(addr, size, mem, task, Detach_exact); }
00650
00651 };

```

## 16.213 I4/re/rm-sys.h File Reference

Region mapper protocol definitions.

### Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

### Enumerations

- enum [L4Re::Rm\\_::Opcodes](#)  
*Region-map communication-protocol opcodes.*

#### 16.213.1 Detailed Description

Region mapper protocol definitions.

Definition in file [rm-sys.h](#).

## 16.214 rm-sys.h

```

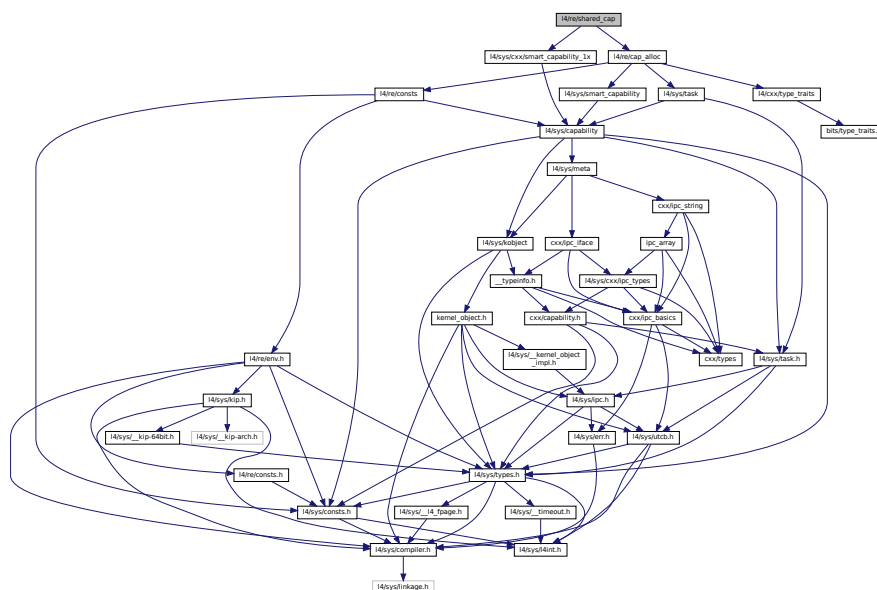
00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #pragma once
00024
00025 namespace L4Re
00026 {
00027 namespace Rm_
00028 {
00034 enum Opcodes
00035 {
00036 Attach, Detach, Find, Attach_area, Detach_area, Get_regions, Get_areas
00037 };
00038 };
00039 };

```

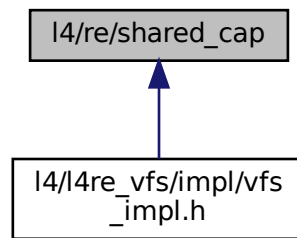
### 16.215 14/re/shared\_cap File Reference

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

- [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.*



### 16.215.1 Detailed Description

Shared\_cap / Shared\_del\_cap.

Definition in file [shared\\_cap](#).

## 16.216 shared\_cap

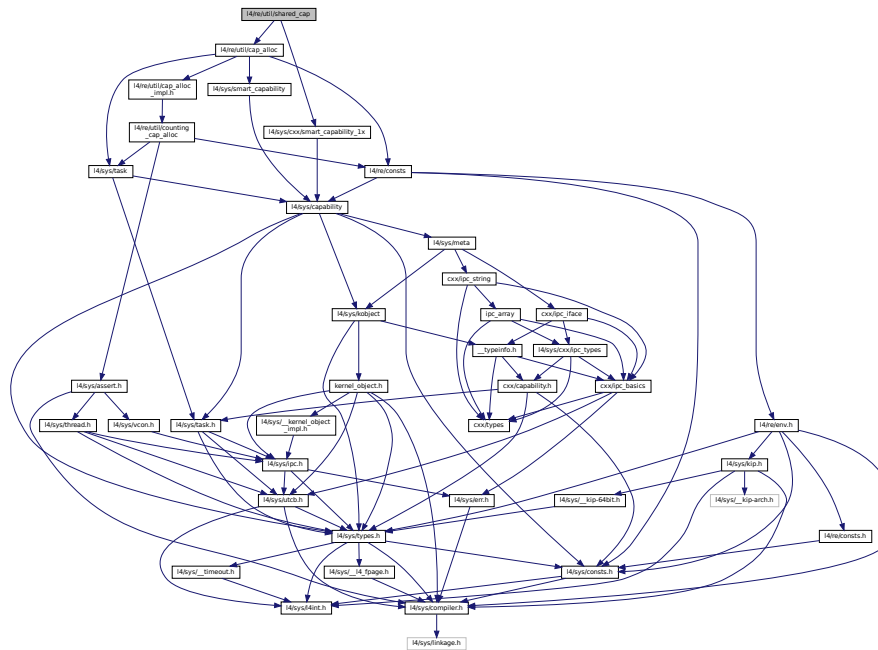
```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2018 Alexander Warg <alexander.warg@kernkonzept.com>
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022
00023 #pragma once
00024
00025 #include <l4/re/cap_alloc>
00026 #include <l4/sys/cxx/smart_capability_1x>
00027
00028 namespace L4Re {
00029
00043 template< typename T >
00044 using Shared_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>;
00046 template< typename T >
00047 using shared_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>;
00048
00058 template< typename T >
00059 Shared_cap<T>
00060 make_shared_cap(L4Re::Cap_alloc *ca)
00061 { return Shared_cap<T>(ca->alloc<T>(), ca); }
00062
00079 template< typename T >
00080 using Shared_del_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>;
00082 template<typename T>
00083 using shared_del_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>;
00084
00094 template< typename T >
00095 Shared_del_cap<T>
00096 make_shared_del_cap(L4Re::Cap_alloc *ca)
00097 { return Shared_del_cap<T>(ca->alloc<T>(), ca); }
00098
00099 } // namespace L4Re
```

## 16.217 l4/re/util/shared\_cap File Reference

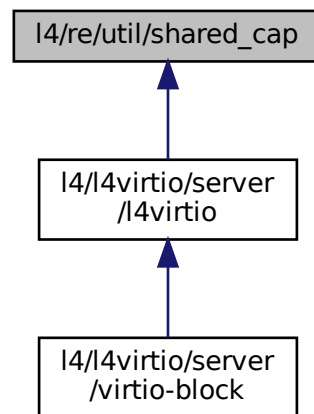
Shared\_cap / Shared\_del\_cap.

```
#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

- L4Re
  - L4Re C++ Interfaces.*
- 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.*

### 16.217.1 Detailed Description

Shared\_cap / Shared\_del\_cap.

Definition in file [shared\\_cap](#).

## 16.218 shared\_cap

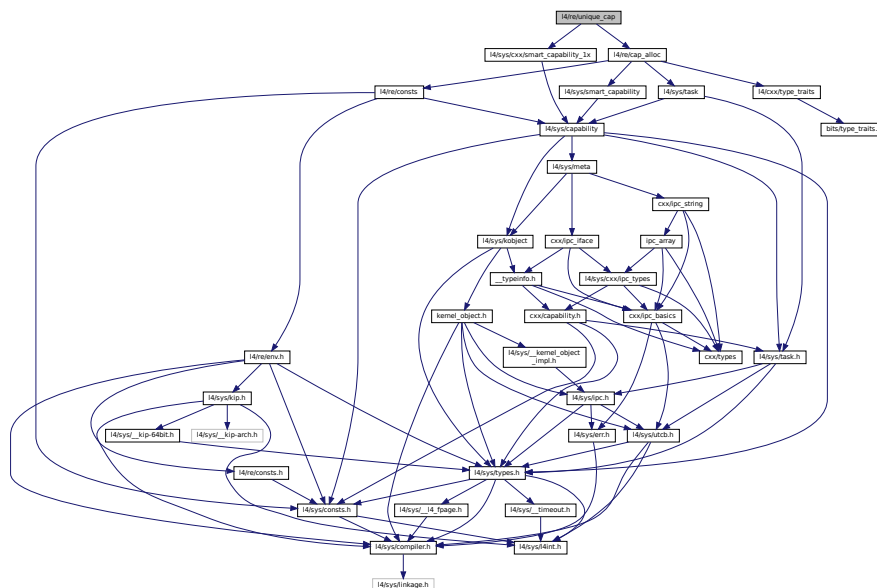
```

00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003 * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00004 *
00005 * This file is part of TUD:OS and distributed under the terms of the
00006 * GNU General Public License 2.
00007 * Please see the COPYING-GPL-2 file for details.
00008 *
00009 * As a special exception, you may use this file as part of a free software
00010 * library without restriction. Specifically, if other files instantiate
00011 * templates or use macros or inline functions from this file, or you compile
00012 * this file and link it with other files to produce an executable, this
00013 * file does not by itself cause the resulting executable to be covered by
00014 * the GNU General Public License. This exception does not however
00015 * invalidate any other reasons why the executable file might be covered by
00016 * the GNU General Public License.
00017 */
00018 #pragma once
00019 #include <l4/re/util/cap_alloc>
00020 #include <l4/sys/cxx/smart_capability_lx>
00021 namespace L4Re { namespace Util {

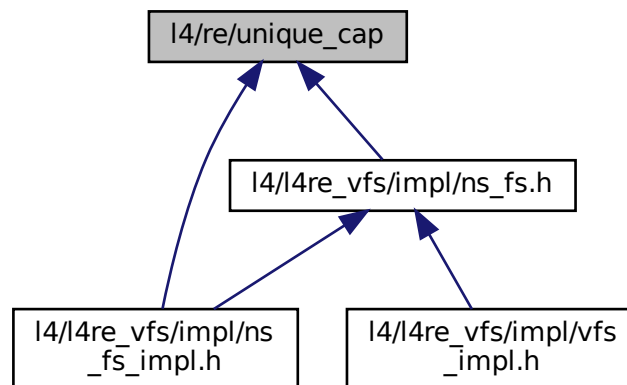
```

## 16.219 l4/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

- [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.*

## 16.219.1 Detailed Description

Unique\_cap / Unique\_del\_cap.

Definition in file [unique\\_cap](#).

## 16.220 unique\_cap

```

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 }

```

## 16.221 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>

```

The diagram illustrates the relationships between several components. At the top is a grey box labeled `I4/re/util/unique_cap`. Below it are three white boxes: `I4/I4virtio/server /I4virtio` on the left, `I4/re/util/event` in the center, and `I4/re/util/object_registry` on the right. At the bottom are two white boxes: `I4/I4virtio/server /virtio-block` on the left and `I4/I4virtio/client /virtio-block` on the right. Arrows point from `I4/I4virtio/server /I4virtio`, `I4/I4virtio/server /virtio-block`, `I4/I4virtio/client /virtio-block`, and `I4/re/util/object_registry` to `I4/re/util/unique_cap`. An arrow also points from `I4/re/util/event` to `I4/re/util/unique_cap`.

- **L4Re**  
*L4Re C++ Interfaces.*
- **L4Re::Util**

## 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.*

### 16.221.1 Detailed Description

Unique\_cap / Unique\_del\_cap.

Definition in file [unique\\_cap](#).

## 16.222 unique\_cap

```
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

```

## 16.223 l4/re/util/bitmap\_cap\_alloc File Reference

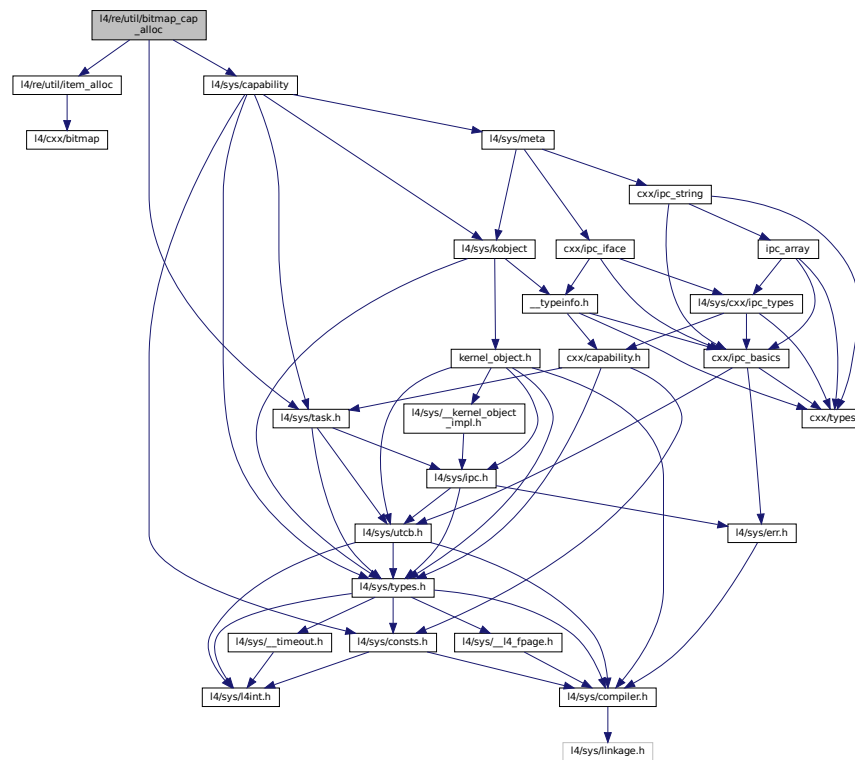
Bitmap capability allocator.

```

#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

- [L4Re](#)  
*L4Re C++ Interfaces.*
- [L4Re::Util](#)  
*Documentation of the L4 Runtime Environment utility functionality in C++.*

### 16.223.1 Detailed Description

Bitmap capability allocator.

Definition in file [bitmap\\_cap\\_alloc](#).

### 16.224 bitmap\_cap\_alloc

```

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 #pragma once
00022
00023 #include <l4/re/util/item_alloc>
00024 #include <l4/sys/capability>
00025 #include <l4/sys/task.h>
00026
00027 namespace L4Re { namespace Util {
00028
00029 class Cap_alloc_base
00030 {
00031 private:
00032 long _bias;
00033 Item_alloc_base _items;
00034 public:
00035 enum State { Free = 0, Allocated, Unknown };
00036 Cap_alloc_base(long max, void *mem, long bias = 0)
00037 noexcept : _bias(bias), _items(max, mem) {}
00038
00039 L4::Cap<void> alloc() noexcept
00040 {
00041 long cap = _items.alloc();
00042 if (cap < 0)
00043 return L4::Cap<void>::Invalid;
00044 return L4::Cap<void>((cap + _bias) « L4_CAP_SHIFT);
00045 }
00046
00047 long hint() const { return _items.hint(); }
00048
00049 template< typename T >
00050 L4::Cap<T> alloc() noexcept
00051 { return L4::Cap<T>(alloc().cap()); }
00052
00053 State is_allocated(L4::Cap<void> c) const noexcept
00054 {
00055 long idx = (c.cap() » L4_CAP_SHIFT);
00056 }

```

```

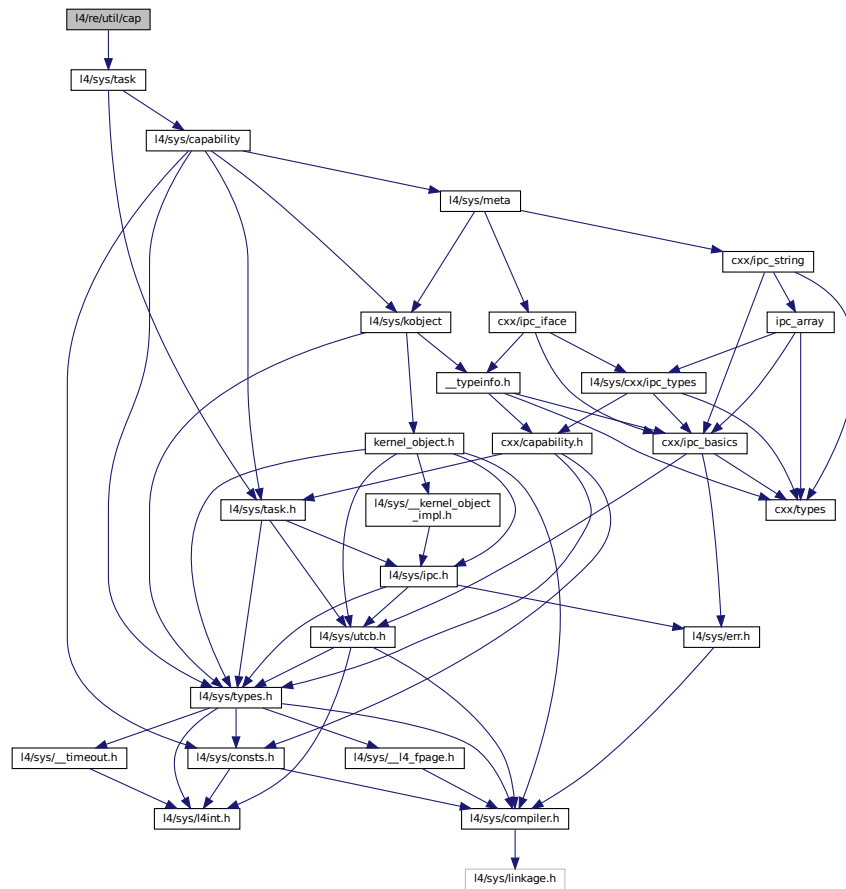
00071 if (idx < _bias)
00072 return Unknown;
00073
00074 idx -= _bias;
00075 return _items.is_allocated(idx) ? Allocated : Free;
00076 }
00077
00081 template< typename T>
00082 void free(L4::Cap<T> const &cap, l4_cap_idx_t task = L4_INVALID_CAP,
00083 l4_umword_t unmap_flags = L4_FP_ALL_SPACES) noexcept
00084 {
00085 long idx = (cap.cap() » L4_CAP_SHIFT);
00086 if (idx < _bias)
00087 return;
00088
00089 idx -= _bias;
00090
00091 _items.free(idx);
00092
00093 if (l4_is_valid_cap(task))
00094 l4_task_unmap(task, cap.fpage(), unmap_flags | 2);
00095 }
00096
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 }

```

## 16.225 l4/re/util/cap File Reference

Capability utility functions.

```
#include <l4/sys/task>
Include dependency graph for cap:
```



## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*
- [L4Re::Util](#)

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

### 16.225.1 Detailed Description

Capability utility functions.

Definition in file [cap](#).

**16.226 cap**

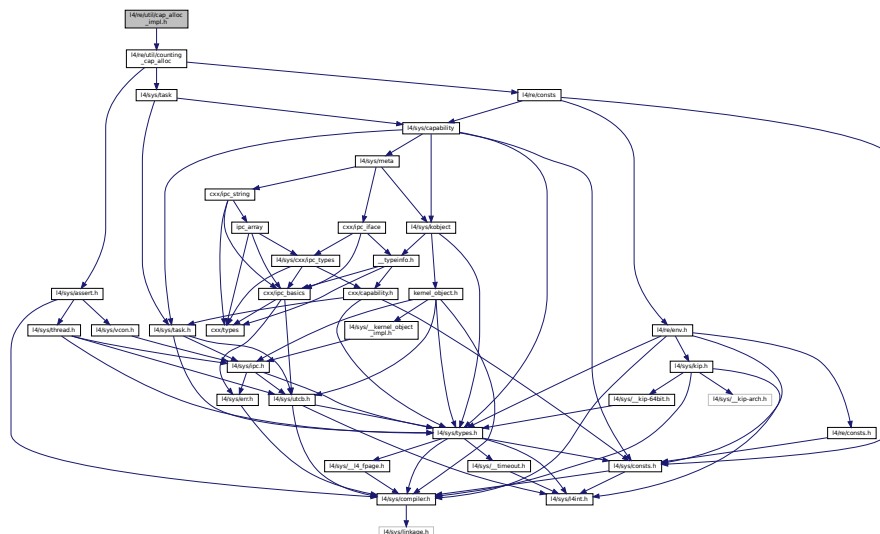
```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 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 }}
```

## 16.227 l4/re/util/cap\_alloc\_impl.h File Reference

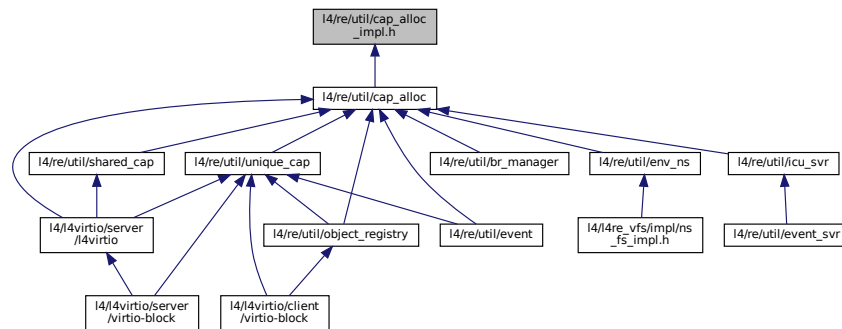
### Capability allocator implementation.

```
#include <linux/re/utl/counting_cap_alloc>
```

Include dependency graph for cap\_alloc\_impl.h:



This graph shows which files directly or indirectly include this file:



## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*
- [L4Re::Util](#)

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

### 16.227.1 Detailed Description

Capability allocator implementation.

Definition in file [cap\\_alloc\\_impl.h](#).

### 16.228 cap\_alloc\_impl.h

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008 * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024
00025 #pragma once
00026
00027 // #define L4RE_STATIC_CAP_ALLOC
00028 #if defined(L4RE_STATIC_CAP_ALLOC)
00029
00030 #include <I4/re/util/bitmap_cap_alloc>
00031
00032 namespace L4Re { namespace Util {
00033
00034 typedef Cap_alloc_base _Cap_alloc;
00035

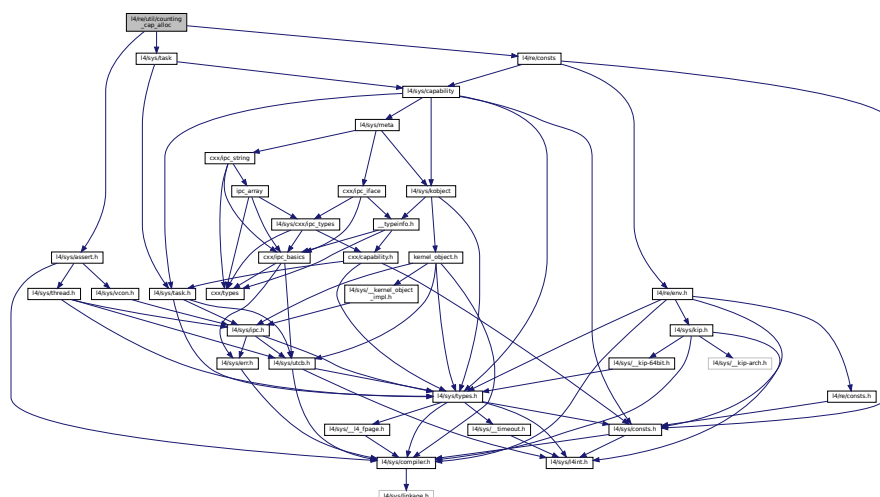
```

```
00036 }}
00037
00038 #else
00039 #include <l4/re/util/counting_cap_alloc>
00040
00041 namespace L4Re { namespace Util {
00042
00043 typedef Counting_cap_alloc<L4Re::Util::Counter<unsigned char> > _Cap_alloc;
00044
00045 }}
00046 #endif
00047
00048
```

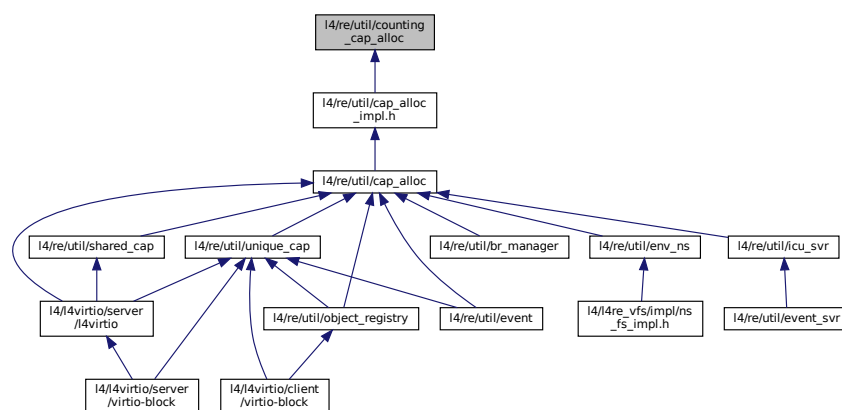
### 16.229 l4/re/util/counting\_cap\_alloc File Reference

Reference-counting capability allocator.

```
#include <l4/sys/task>
#include <l4/sys/assert.h>
#include <l4/re/consts>
Include dependency graph for counting_cap_alloc:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [L4Re::Util::Counter< COUNTER >](#)  
*Counter for [Counting\\_cap\\_alloc](#) with variable data width.*
- class [L4Re::Util::Counting\\_cap\\_alloc< COUNTERTYPE >](#)  
*Internal reference-counting cap allocator.*

## Namespaces

- [L4Re](#)  
*[L4Re](#) C++ Interfaces.*
- [L4Re::Util](#)  
*Documentation of the [L4](#) Runtime Environment utility functionality in C++.*

### 16.229.1 Detailed Description

Reference-counting capability allocator.

Definition in file [counting\\_cap\\_alloc](#).

## 16.230 counting\_cap\_alloc

```

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
00032 template< typename COUNTER = unsigned char >
00033 struct Counter
00034 {
00035 typedef COUNTER Type;
00036 Type _cnt;
00037
00038 static Type nil() { return 0; }
00039
00040 void free() { _cnt = 0; }
00041 bool is_free() const { return _cnt == 0; }
00042 void inc() { ++_cnt; }
00043 Type dec() { return --_cnt; }
00044 void alloc() { _cnt = 1; }
00045 };
00046
00047 template< typename COUNTERTYPE = L4Re::Util::Counter<unsigned char> >
00048 class Counting_cap_alloc
00049 {
00050 private:

```



```

00078 void operator = (Counting_cap_alloc const &) { }
00079 typedef COUNTERTYPE Counter;
00080
00081 COUNTERTYPE *_items;
00082 long _free_hint;
00083 long _bias;
00084 long _capacity;
00085
00086
00087 public:
00088
00089 template <unsigned COUNT>
00090 struct Counter_storage
00091 {
00092 COUNTERTYPE _buf[COUNT];
00093 typedef COUNTERTYPE Buf_type[COUNT];
00094 enum { Size = COUNT };
00095 };
00096
00097 protected:
00098
00104 Counting_cap_alloc() noexcept
00105 : _items(0), _free_hint(0), _bias(0), _capacity(0)
00106 {}
00107
00121 void setup(void *m, long capacity, long bias) noexcept
00122 {
00123 _items = (Counter*)m;
00124 _capacity = capacity;
00125 _bias = bias;
00126 }
00127
00128 public:
00135 L4::Cap<void> alloc() noexcept
00136 {
00137 for (long i = _free_hint; i < _capacity; ++i)
00138 {
00139 if (_items[i].is_free())
00140 {
00141 _items[i].alloc();
00142 _free_hint = i + 1;
00143
00144 return L4::Cap<void>((i + _bias) « L4_CAP_SHIFT);
00145 }
00146 }
00147
00148 return L4::Cap<void>::Invalid;
00149 }
00150
00152 template <typename T>
00153 L4::Cap<T> alloc() noexcept
00154 {
00155 return L4::cap_cast<T>(alloc());
00156 }
00157
00158
00168 void take(L4::Cap<void> cap) noexcept
00169 {
00170 long c = cap.cap() » L4_CAP_SHIFT;
00171 if (c < _bias)
00172 return;
00173
00174 c -= _bias;
00175 if (c >= _capacity)
00176 return;
00177
00178 _items[c].inc();
00179 }
00180
00181
00195 bool free(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
00196 unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept
00197 {
00198 long c = cap.cap() » L4_CAP_SHIFT;
00199 if (c < _bias)
00200 return false;
00201
00202 c -= _bias;
00203
00204 if (c >= _capacity)
00205 return false;
00206
00207 l4_assert(!_items[c].is_free());
00208
00209 if (l4_is_valid_cap(task))
00210 l4_task_unmap(task, cap.fpage(), unmap_flags);
00211

```

```

00212 if (c < _free_hint)
00213 _free_hint = c;
00214
00215 _items[c].free();
00216
00217 return true;
00218 }
00219
00238 bool release(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
00239 unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept
00240 {
00241 long c = cap.cap() » L4_CAP_SHIFT;
00242 if (c < _bias)
00243 return false;
00244
00245 c -= _bias;
00246
00247 if (c >= _capacity)
00248 return false;
00249
00250 l4_assert(!_items[c].is_free());
00251
00252 if (_items[c].dec() == Counter::nil())
00253 {
00254 if (task != L4_INVALID_CAP)
00255 l4_task_unmap(task, cap.fpage(), unmap_flags);
00256
00257 if (c < _free_hint)
00258 _free_hint = c;
00259
00260 return true;
00261 }
00262 return false;
00263 }
00264
00265
00269 long last() noexcept
00270 {
00271 return _capacity + _bias - 1;
00272 }
00273 };
00274
00275 }}
00276

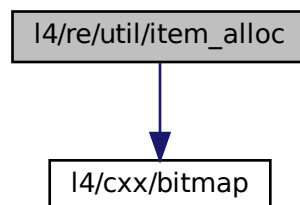
```

## 16.231 l4/re/util/item\_alloc File Reference

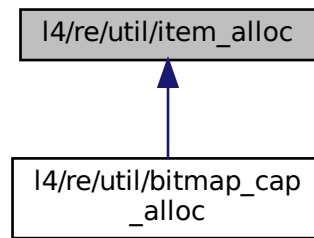
Item allocator.

```
#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

- [L4Re](#)  
*L4Re C++ Interfaces.*
- [L4Re::Util](#)  
*Documentation of the [L4](#) Runtime Environment utility functionality in C++.*

### 16.231.1 Detailed Description

Item allocator.

Definition in file [item\\_alloc](#).

## 16.232 item\_alloc

```

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 */

```

```

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 }}

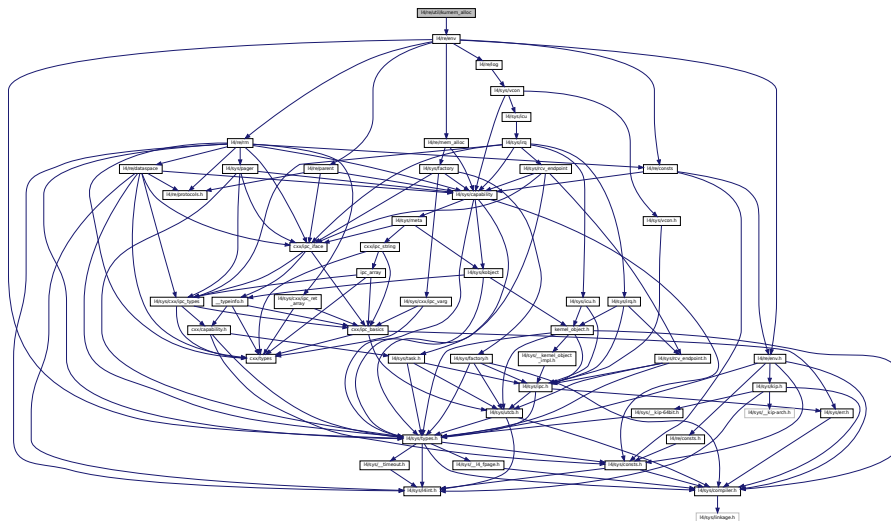
```

## 16.233 l4/re/util/kumem\_alloc File Reference

Kumem allocator helper.

```
#include <14/re/env>
```

Include dependency graph for kumem\_alloc:



## Namespaces

- L4Re
  - L4Re C++ Interfaces.*
- 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.

16.233.1 Detailed Description

Kumem allocator helper.

Definition in file [kumem alloc.](#)

16.234 kumem_alloc

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004  * (c) 2010 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
```

```

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
00037
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 }}

```

16.235 l4/re/util/region_mapping File Reference

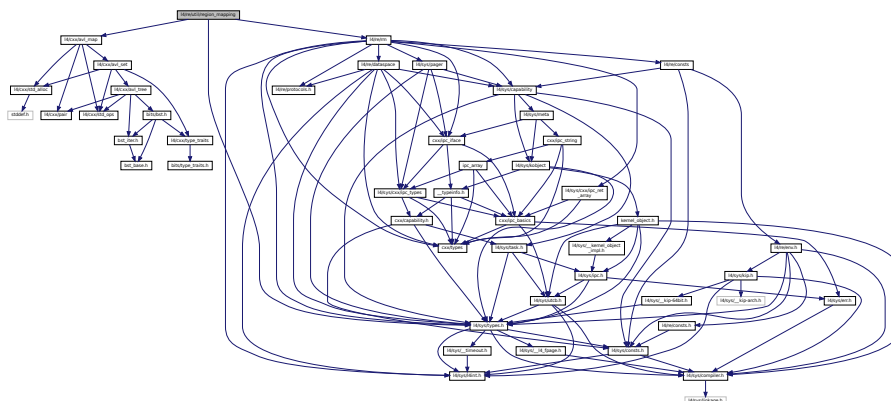
Region handling.

```

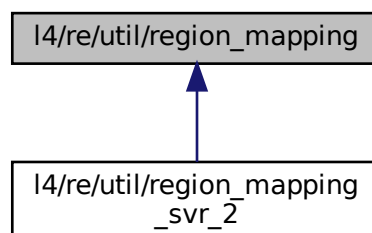
#include <l4/cxx/avl_map>
#include <l4/sys/types.h>
#include <l4/re/rm>

```

Include dependency graph for region_mapping:



This graph shows which files directly or indirectly include this file:



Namespaces

- [L4Re](#)
L4Re C++ Interfaces.
- [L4Re::Util](#)
Documentation of the L4 Runtime Environment utility functionality in C++.

16.235.1 Detailed Description

Region handling.

Definition in file [region_mapping](#).

16.236 region_mapping

```

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
00027 #pragma once
00028
00029 #include <l4/cxx/avl_map>
00030 #include <l4/sys/types.h>
00031 #include <l4/re/rm>
00032
00033 namespace L4Re { namespace Util {
00034 class Region
00035 {
00036 private:
00037     l4_addr_t _start, _end;
00038 public:
00039     Region() noexcept : _start(~0UL), _end(~0UL) {}
00040     Region(l4_addr_t addr) noexcept : _start(addr), _end(addr) {}
00041     Region(l4_addr_t start, l4_addr_t end) noexcept
00042         : _start(start), _end(end) {}
00043     l4_addr_t start() const noexcept { return _start; }
00044     l4_addr_t end() const noexcept { return _end; }
00045     unsigned long size() const noexcept { return end() - start() + 1; }
00046     bool invalid() const noexcept { return _start == ~0UL && _end == ~0UL; }
00047     bool operator < (Region const &o) const noexcept
00048     { return end() < o.start(); }
00049     bool contains(Region const &o) const noexcept
00050     { return o.start() >= start() && o.end() <= end(); }
00051     bool operator == (Region const &o) const noexcept
00052     { return o.start() == start() && o.end() == end(); }
00053     ~Region() noexcept {}
00054 };
00055
00056 template< typename DS, typename OPS >
00057 class Region_handler
00058 {
00059 private:
00060     L4Re::Rm::Offset _offs;
00061     DS _mem;

```

```

00064     l4_cap_idx_t _client_cap = L4_INVALID_CAP;
00065     L4Re::Rm::Region_flags _flags;
00066
00067 public:
00068     typedef DS Dataspace;
00069     typedef OPS Ops;
00070     typedef typename OPS::Map_result Map_result;
00071
00072     Region_handler() noexcept : _offs(0), _mem(), _flags() {}
00073     Region_handler(Dataspace const &mem, l4_cap_idx_t client_cap,
00074                   L4Re::Rm::Offset offset = 0,
00075                   L4Re::Rm::Region_flags flags = L4Re::Rm::Region_flags(0)) noexcept
00076         : _offs(offset), _mem(mem), _client_cap(client_cap), _flags(flags)
00077     {}
00078
00079     Dataspace const &memory() const noexcept
00080     {
00081         return _mem;
00082     }
00083
00084     l4_cap_idx_t client_cap_idx() const noexcept
00085     {
00086         return _client_cap;
00087     }
00088
00089     L4Re::Rm::Offset offset() const noexcept
00090     {
00091         return _offs;
00092     }
00093
00094     constexpr bool is_ro() const noexcept
00095     {
00096         return !(_flags & L4Re::Rm::F::W);
00097     }
00098
00099     L4Re::Rm::Region_flags caching() const noexcept
00100     {
00101         return _flags & L4Re::Rm::F::Caching_mask;
00102     }
00103
00104     L4Re::Rm::Region_flags flags() const noexcept
00105     {
00106         return _flags;
00107     }
00108
00109     Region_handler operator + (l4_int64_t offset) const noexcept
00110     {
00111         Region_handler n = *this; n._offs += offset; return n;
00112     }
00113
00114     void free(l4_addr_t start, unsigned long size) const noexcept
00115     {
00116         Ops::free(this, start, size);
00117     }
00118
00119     int map(l4_addr_t addr, Region const &r, bool writable,
00120            Map_result *result) const
00121     {
00122         return Ops::map(this, addr, r, writable, result);
00123     }
00124
00125 };
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
00239     if (_am.insert(c, Hdlr(typename Hdlr::Dataspace(), 0, 0, flags.region_flags()).second == 0)
00240         return addr;
00241
00242     return L4_INVALID_ADDR;
00243 }
00244
00245 bool detach_area(l4_addr_t addr) noexcept
00246 {
00247     if (_am.remove(addr))
00248         return false;
00249
00250     return true;
00251 }
00252
00253 void *attach(void *addr, unsigned long size, Hdlr const &hdlr,
00254             L4Re::Rm::Flags flags = L4Re::Rm::Flags(0),
00255             unsigned char align = L4_PAGESHIFT) noexcept
00256 {
00257     if (size < 2)
00258         return L4_INVALID_PTR;
00259
00260     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))
        && _am.find_node(Region(beg, beg + size - 1)))
00280         return L4_INVALID_PTR;
00281
00282     if (beg < min_addr() || beg + size - 1 > end)
00283         return L4_INVALID_PTR;
00284
00285     if (_rm.insert(Region(beg, beg + size - 1), hdlr).second == 0)
00286         return (void*)beg;
00287
00288     return L4_INVALID_PTR;
00289 }
00290
00291 int detach(void *addr, unsigned long sz, unsigned flags,
00292           Region *reg, Hdlr *hdlr) noexcept
00293 {
00294     Region dr((l4_addr_t)addr, (l4_addr_t)addr + sz - 1);
00295     Region res(~0UL, 0);
00296
00297     Node r = find(dr);
00298     if (!r)
00299         return -L4_ENOENT;
00300
00301     Region g = r->first;
00302     Hdlr const &h = r->second;
00303
00304     if (flags & L4Re::Rm::Detach_overlap || dr.contains(g))
00305     {
00306         if (_rm.remove(g))
00307             return -L4_ENOENT;
00308
00309         if (!(flags & L4Re::Rm::Detach_keep) && (h.flags() & L4Re::Rm::F::Detach_free))
00310             h.free(0, g.size());
00311
00312         if (hdlr) *hdlr = h;
00313         if (reg) *reg = g;
00314
00315         if (find(dr))
00316             return Rm::Detached_ds | Rm::Detach_again;
00317         else
00318             return Rm::Detached_ds;
00319     }
00320     else if (dr.start() <= g.start())
00321     {
00322         // move the start of a region
00323
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)))

```

```

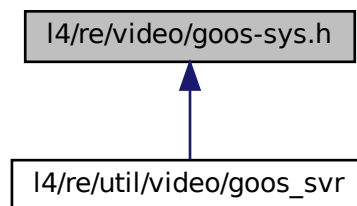
00412     {
00413         if (r->first.end() > end - size)
00414             return L4_INVALID_ADDR;
00415
00416         addr = l4_round_size(r->first.end() + 1, align);
00417         continue;
00418     }
00419     break;
00420 }
00421     else if (r->first.end() > end - size)
00422         return L4_INVALID_ADDR;
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 }

```

16.237 l4/re/video/goos-sys.h File Reference

Goos protocol definition.

This graph shows which files directly or indirectly include this file:



Namespaces

- [L4Re](#)
L4Re C++ Interfaces.

Enumerations

- enum [L4Re::Video::Goos_::Opcodes](#)
Frame buffer communication-protocol opcodes.

16.237.1 Detailed Description

Goos protocol definition.

Definition in file [goos-sys.h](#).

16.238 goos-sys.h

```

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 }}

```

16.239 l4/shmc/shmc.h File Reference

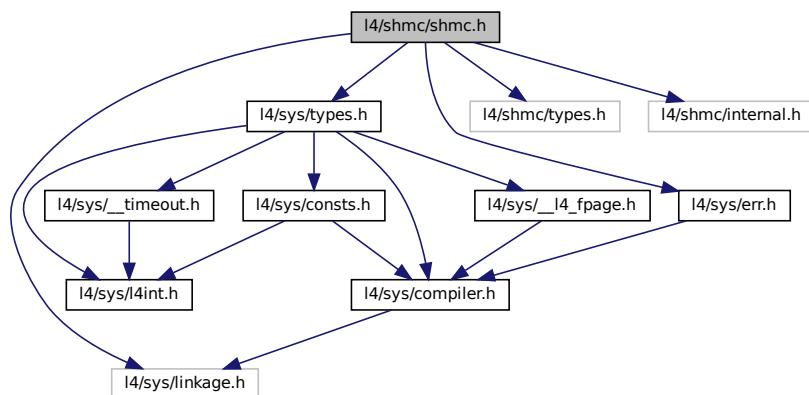
Shared memory library header file.

```

#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:



Functions

- long [l4shmc_create](#) (const char *shmc_name, [l4_umword_t](#) shm_size)
Create a shared memory area.
- long [l4shmc_attach](#) (const char *shmc_name, [l4shmc_area_t](#) *shmarea)
Attach to a shared memory area.
- long [l4shmc_attach_to](#) (const char *shmc_name, [l4_umword_t](#) timeout_ms, [l4shmc_area_t](#) *shmarea)
Attach to a shared memory area, with limited waiting.
- long [l4shmc_add_chunk](#) ([l4shmc_area_t](#) *shmarea, const char *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, const char *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_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, const char *chunk_name, [l4shmc_chunk_t](#) *chunk)
Get chunk out of shared memory area.
- long [l4shmc_get_chunk_to](#) ([l4shmc_area_t](#) *shmarea, const char *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](#) *shmarea, const char **chunk_name, long offs)
Iterate over names of all existing chunks.
- long [l4shmc_attach_signal](#) ([l4shmc_area_t](#) *shmarea, const char *signal_name, [l4_cap_idx_t](#) thread, [l4shmc_signal_t](#) *signal)
Attach to signal.
- long [l4shmc_attach_signal_to](#) ([l4shmc_area_t](#) *shmarea, const char *signal_name, [l4_cap_idx_t](#) thread, [l4_umword_t](#) timeout_ms, [l4shmc_signal_t](#) *signal)
Attach to signal, with timeout.
- long [l4shmc_get_signal_to](#) ([l4shmc_area_t](#) *shmarea, const char *signal_name, [l4_umword_t](#) timeout_ms, [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 *chunk)

Check whether data is available.
- long [l4shmc_is_chunk_clear](#) (l4shmc_chunk_t *chunk)

Check whether chunk is free.
- void * [l4shmc_chunk_ptr](#) (l4shmc_chunk_t *chunk)

Get data pointer to chunk.
- long [l4shmc_chunk_size](#) (l4shmc_chunk_t *chunk)

Get current size of a chunk.
- long [l4shmc_chunk_capacity](#) (l4shmc_chunk_t *chunk)

Get capacity of a chunk.
- l4shmc_signal_t * [l4shmc_chunk_signal](#) (l4shmc_chunk_t *chunk)

Get the signal of a chunk.
- [l4_cap_idx_t](#) [l4shmc_signal_cap](#) (l4shmc_signal_t *signal)

Get the signal capability of a signal.
- long [l4shmc_check_magic](#) (l4shmc_chunk_t *chunk)

Check magic value of a chunk.
- long [l4shmc_area_size](#) (l4shmc_area_t *shmarea)

Get size of shared memory area.
- long [l4shmc_area_size_free](#) (l4shmc_area_t *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.

16.239.1 Detailed Description

Shared memory library header file.

Definition in file [shmc.h](#).

16.240 shm.h

```

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
00084 L4_CV long
00085 l4shm_create(const char *shm_name, l4_umword_t shm_size);
00086
00098 L4_CV L4_INLINE long
00099 l4shm_attach(const char *shm_name, l4shm_area_t *shmarea);
00100
00113 L4_CV long
00114 l4shm_attach_to(const char *shm_name, l4_umword_t timeout_ms,
00115                l4shm_area_t *shmarea);
00116
00129 L4_CV long
00130 l4shm_add_chunk(l4shm_area_t *shmarea,
00131                const char *chunk_name,
00132                l4_umword_t chunk_capacity,
00133                l4shm_chunk_t *chunk);
00134
00146 L4_CV long
00147 l4shm_add_signal(l4shm_area_t *shmarea,
00148                 const char *signal_name,
00149                 l4shm_signal_t *signal);
00150
00160 L4_CV L4_INLINE long
00161 l4shm_trigger(l4shm_signal_t *signal);
00162
00172 L4_CV L4_INLINE long
00173 l4shm_chunk_try_to_take(l4shm_chunk_t *chunk);
00174
00185 L4_CV L4_INLINE long
00186 l4shm_chunk_ready(l4shm_chunk_t *chunk, l4_umword_t size);
00187
00198 L4_CV L4_INLINE long
00199 l4shm_chunk_ready_sig(l4shm_chunk_t *chunk, l4_umword_t size);
00200
00212 L4_CV L4_INLINE long
00213 l4shm_get_chunk(l4shm_area_t *shmarea,
00214                const char *chunk_name,
00215                l4shm_chunk_t *chunk);
00216
00230 L4_CV long
00231 l4shm_get_chunk_to(l4shm_area_t *shmarea,
00232                   const char *chunk_name,
00233                   l4_umword_t timeout_ms,
00234                   l4shm_chunk_t *chunk);
00235
00249 L4_CV long
00250 l4shm_iterate_chunk(l4shm_area_t *shmarea, const char **chunk_name,
00251                   long offs);
00252
00265 L4_CV L4_INLINE long
00266 l4shm_attach_signal(l4shm_area_t *shmarea,
00267                   const char *signal_name,
00268                   l4_cap_idx_t thread,
00269                   l4shm_signal_t *signal);
00270
00285 L4_CV long
00286 l4shm_attach_signal_to(l4shm_area_t *shmarea,
00287                      const char *signal_name,
00288                      l4_cap_idx_t thread,
00289                      l4_umword_t timeout_ms,
00290                      l4shm_signal_t *signal);
00291
00305 L4_CV long
00306 l4shm_get_signal_to(l4shm_area_t *shmarea,
00307                   const char *signal_name,
00308                   l4_umword_t timeout_ms,

```



```

00309             l4shmc_signal_t *signal);
00310
00311 L4_CV L4_INLINE long
00312 l4shmc_get_signal(l4shmc_area_t *shmarea,
00313                 const char *signal_name,
00314                 l4shmc_signal_t *signal);
00315
00316
00330 L4_CV long
00331 l4shmc_enable_signal(l4shmc_signal_t *signal);
00332
00346 L4_CV long
00347 l4shmc_enable_chunk(l4shmc_chunk_t *chunk);
00348
00358 L4_CV L4_INLINE long
00359 l4shmc_wait_any(l4shmc_signal_t **retsignal);
00360
00374 L4_CV L4_INLINE long
00375 l4shmc_wait_any_try(l4shmc_signal_t **retsignal);
00376
00391 L4_CV long
00392 l4shmc_wait_any_to(l4_timeout_t timeout, l4shmc_signal_t **retsignal);
00393
00403 L4_CV L4_INLINE long
00404 l4shmc_wait_signal(l4shmc_signal_t *signal);
00405
00416 L4_CV long
00417 l4shmc_wait_signal_to(l4shmc_signal_t *signal, l4_timeout_t timeout);
00418
00432 L4_CV L4_INLINE long
00433 l4shmc_wait_signal_try(l4shmc_signal_t *signal);
00434
00444 L4_CV L4_INLINE long
00445 l4shmc_wait_chunk(l4shmc_chunk_t *chunk);
00446
00461 L4_CV long
00462 l4shmc_wait_chunk_to(l4shmc_chunk_t *chunk, l4_timeout_t timeout);
00463
00477 L4_CV L4_INLINE long
00478 l4shmc_wait_chunk_try(l4shmc_chunk_t *chunk);
00479
00489 L4_CV L4_INLINE long
00490 l4shmc_chunk_consumed(l4shmc_chunk_t *chunk);
00491
00501 L4_CV long
00502 l4shmc_connect_chunk_signal(l4shmc_chunk_t *chunk,
00503                             l4shmc_signal_t *signal);
00504
00514 L4_CV L4_INLINE long
00515 l4shmc_is_chunk_ready(l4shmc_chunk_t *chunk);
00516
00526 L4_CV L4_INLINE long
00527 l4shmc_is_chunk_clear(l4shmc_chunk_t *chunk);
00528
00538 L4_CV L4_INLINE void *
00539 l4shmc_chunk_ptr(l4shmc_chunk_t *chunk);
00540
00550 L4_CV L4_INLINE long
00551 l4shmc_chunk_size(l4shmc_chunk_t *chunk);
00552
00562 L4_CV L4_INLINE long
00563 l4shmc_chunk_capacity(l4shmc_chunk_t *chunk);
00564
00574 L4_CV L4_INLINE l4shmc_signal_t *
00575 l4shmc_chunk_signal(l4shmc_chunk_t *chunk);
00576
00585 L4_CV L4_INLINE l4_cap_idx_t
00586 l4shmc_signal_cap(l4shmc_signal_t *signal);
00587
00597 L4_CV L4_INLINE long
00598 l4shmc_check_magic(l4shmc_chunk_t *chunk);
00599
00609 L4_CV L4_INLINE long
00610 l4shmc_area_size(l4shmc_area_t *shmarea);
00611
00623 L4_CV long
00624 l4shmc_area_size_free(l4shmc_area_t *shmarea);
00625
00632 L4_CV long
00633 l4shmc_area_overhead(void);
00634
00642 L4_CV long
00643 l4shmc_chunk_overhead(void);
00644
00645 #include <l4/shmc/internal.h>
00646
00647 __END_DECLS

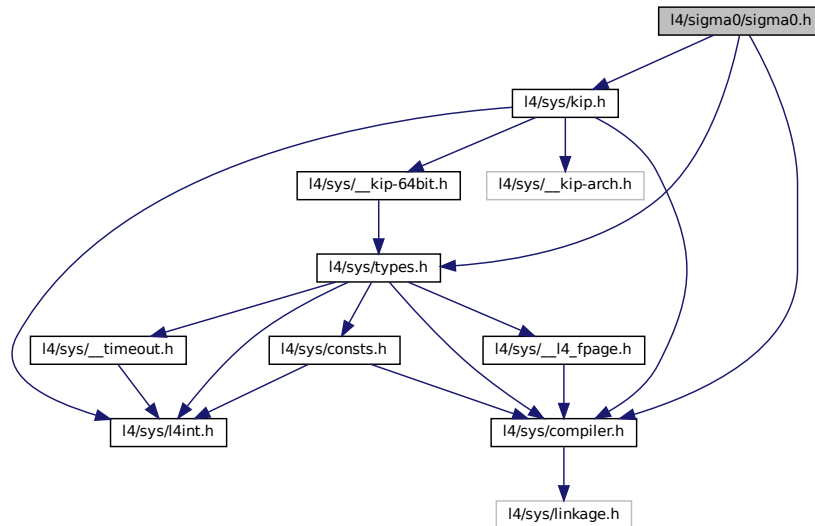
```

16.241 I4/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_REQ_ID_NEW_CLIENT 0xD0`
New client.

- `#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.
- `#define SIGMA0_REQ_NEW_CLIENT (SIGMA0_REQ(NEW_CLIENT))`
New client.

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.
- `int l4sigma0_new_client (l4_cap_idx_t sigma0, l4_cap_idx_t gate)`
Create a new IPC gate for a new Sigma0 client.
- `char const * l4sigma0_map_errstr (int err)`
Get user readable error messages for the return codes.

16.241.1 Detailed Description

Sigma0 interface.

Definition in file [sigma0.h](#).

16.241.2 Enumeration Type Documentation

16.241.2.1 l4sigma0_return_flags_t

```
enum l4sigma0_return_flags_t
```

Return flags of libsigma0 functions.

Enumerator

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

Definition at line 80 of file [sigma0.h](#).

16.241.3 Function Documentation

16.241.3.1 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.

16.241.3.2 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 $2^{\log2_map_size}$). 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

0	Success.
-L4SIGMA0_IPCERROR	IPC error.
-L4SIGMA0_NOFPAGE	No fpage received.

This function requests arbitrary free memory from sigma0. It should be used whenever spare memory is needed, instead of requesting specific physical memory with [l4sigma0_map_mem\(\)](#).

See [l4sigma0_map_errstr\(\)](#) to get a description of the return value.

16.241.3.3 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 214 of file [sigma0.h](#).

16.241.3.4 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.

16.241.3.5 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.

16.241.3.6 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.

See [l4sigma0_map_errstr\(\)](#) to get a description of the return value.

16.241.3.7 l4sigma0_new_client()

```
int l4sigma0_new_client (
    l4_cap_idx_t sigma0,
    l4_cap_idx_t gate )
```

Create a new IPC gate for a new Sigma0 client.

Parameters

<i>sigma0</i>	Capability selector for the sigma0 gate.
<i>gate</i>	Capability selector to use for the new gate.

Return values

0	Success.
-L4SIGMA0_IPCERROR	IPC error.
-L4SIGMA0_NOFPAGE	No fpage received.

16.242 sigma0.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  *      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    ~0xFFUL
00041 # define SIGMA0_REQ_MASK    ~0xFFUL
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
00052 #define SIGMA0_REQ_ID_NEW_CLIENT    0xD0
00054 #define SIGMA0_IS_MAGIC_REQ(d1) \
00055     ((d1 & SIGMA0_REQ_MASK) == SIGMA0_REQ_MAGIC)
00057 #define SIGMA0_REQ(x) \
00058     (SIGMA0_REQ_MAGIC + SIGMA0_REQ_ID_ ## x)
00060 /* Use these constants in your code! */
00061 #define SIGMA0_REQ_FPAGE_RAM    (SIGMA0_REQ(FPAGE_RAM))
00062 #define SIGMA0_REQ_FPAGE_IOMEM    (SIGMA0_REQ(FPAGE_IOMEM))
00063 #define SIGMA0_REQ_FPAGE_IOMEM_CACHED    (SIGMA0_REQ(FPAGE_IOMEM_CACHED))
00064 #define SIGMA0_REQ_FPAGE_ANY    (SIGMA0_REQ(FPAGE_ANY))
00065 #define SIGMA0_REQ_KIP    (SIGMA0_REQ(KIP))
00066 #define SIGMA0_REQ_DEBUG_DUMP    (SIGMA0_REQ(DEBUG_DUMP))
00067 #define SIGMA0_REQ_NEW_CLIENT    (SIGMA0_REQ(NEW_CLIENT))
00069
00074
00078 enum l4sigma0_return_flags_t {
00079     L4SIGMA0_OK,
00080     L4SIGMA0_NOTALIGNED,
00081     L4SIGMA0_IPCERROR,
00082     L4SIGMA0_NOFPAGE,
00083     L4SIGMA0_4,
00084     L4SIGMA0_5,
00085     L4SIGMA0_SMALLERFPAGE,
00086 };
00087
00088 EXTERN_C_BEGIN
00089
00099 L4_CV l4_kernel_info_t *
00100 l4sigma0_map_kip(l4_cap_idx_t sigma0, void *addr, unsigned log2_size);
00101
00121 L4_CV int l4sigma0_map_mem(l4_cap_idx_t sigma0,
00122     l4_addr_t phys, l4_addr_t virt, l4_addr_t size);
00123
00146 L4_CV int l4sigma0_map_iomem(l4_cap_idx_t sigma0, l4_addr_t phys,
00147     l4_addr_t virt, l4_addr_t size, int cached);
00172 L4_CV int l4sigma0_map_anypage(l4_cap_idx_t sigma0, l4_addr_t map_area,
00173     unsigned log2_map_size, l4_addr_t *base,
00174     unsigned sz);
00175
00184 L4_CV void l4sigma0_debug_dump(l4_cap_idx_t sigma0);
00185
00196 L4_CV int l4sigma0_new_client(l4_cap_idx_t sigma0, l4_cap_idx_t gate);
00197
00205 L4_INLINE char const *l4sigma0_map_errstr(int err);
00206
00210 /* Implementations */
00211
00212 L4_INLINE char const *l4sigma0_map_errstr(int err)
00213 {
00214     switch (err)
00215     {
00216     case 0: return "No error";
00217     case -1: return "Phys, virt or size not aligned";
00218     case -2: return "IPC error";
00219     case -3: return "No fpage received";
00220 #ifndef SIGMA0_REQ_MAGIC
00221     case -4: return "Bad physical address (old protocol only)";
00222 #endif
00223     case -6: return "Superpage requested but smaller flexpage received";
00224     case -7: return "Cannot map I/O memory cacheable (old protocol only)";
00225     default: return "Unknown error";
00226     }
00227 }
00228
00229
00230 EXTERN_C_END
00231
00232 #endif /* ! __L4_SIGMA0_SIGMA0_H */

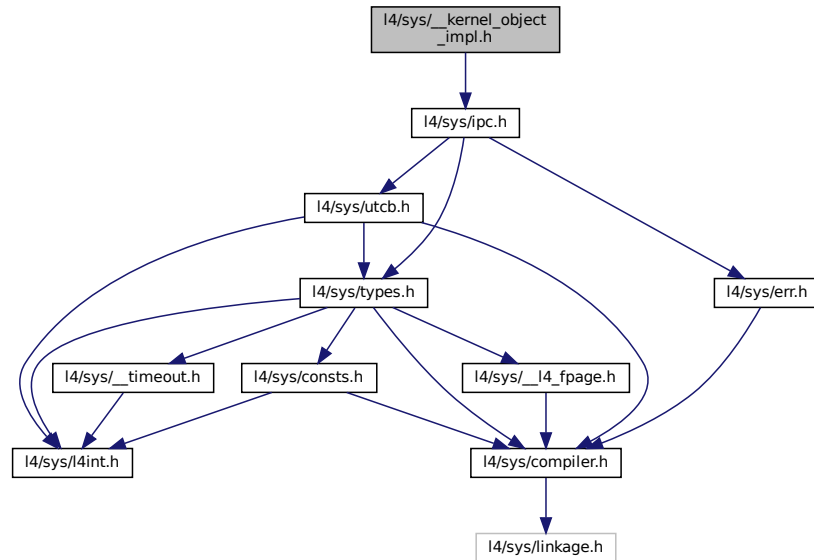
```


16.243 l4/sys/__kernel_object_impl.h File Reference

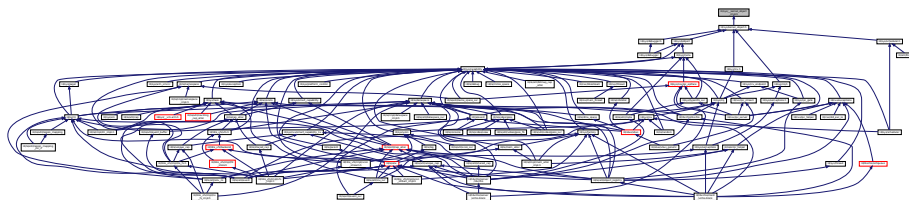
Low-level kernel debugger functions.

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

Include dependency graph for __kernel_object_impl.h:



This graph shows which files directly or indirectly include this file:



16.243.1 Detailed Description

Low-level kernel debugger functions.

Definition in file [__kernel_object_impl.h](#).

16.244 __kernel_object_impl.h

```

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_MSGTOOLONG, 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

```

16.245 l4/sys/__ktrace-impl.h File Reference

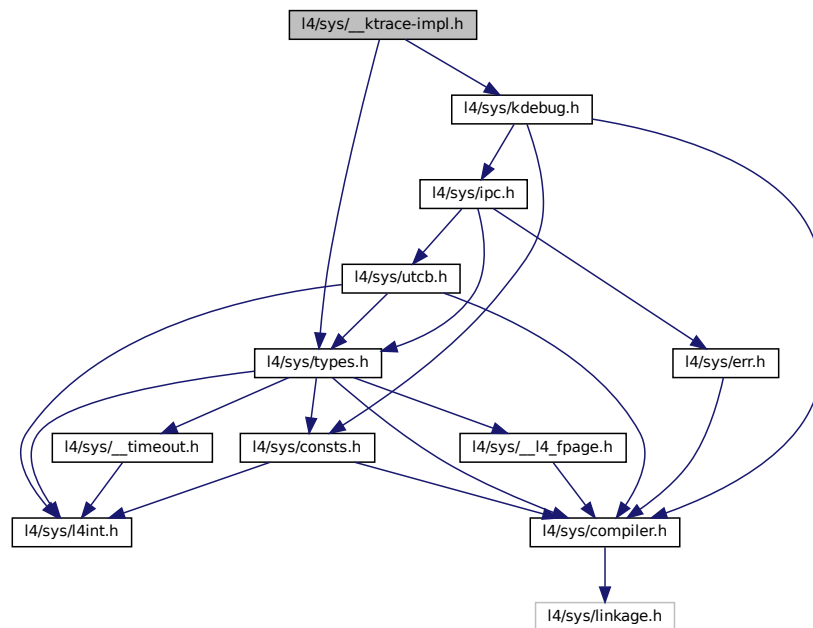
L4 kernel event tracing.

```

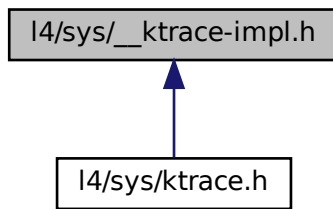
#include <l4/sys/types.h>
#include <l4/sys/kdebug.h>

```

Include dependency graph for __ktrace-impl.h:



This graph shows which files directly or indirectly include this file:



Functions

- [l4_umword_t fiasco_tbuf_log](#) (const char *text)
Create new trace-buffer entry with describing <text>.
- [l4_umword_t fiasco_tbuf_log_3val](#) (const char *text, [l4_umword_t](#) v1, [l4_umword_t](#) v2, [l4_umword_t](#) v3)
Create new trace-buffer entry with describing <text> and three additional values.
- void [fiasco_tbuf_clear](#) (void)
Clear trace-buffer.
- void [fiasco_tbuf_dump](#) (void)
Dump trace-buffer to kernel console.
- [l4_umword_t fiasco_tbuf_log_binary](#) (const unsigned char *data)
Create new trace-buffer entry with binary data.

16.245.1 Detailed Description

[L4](#) kernel event tracing.

Definition in file [__ktrace-impl.h](#).

16.246 __ktrace-impl.h

```

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

```

16.247 l4/sys/__typeinfo.h File Reference

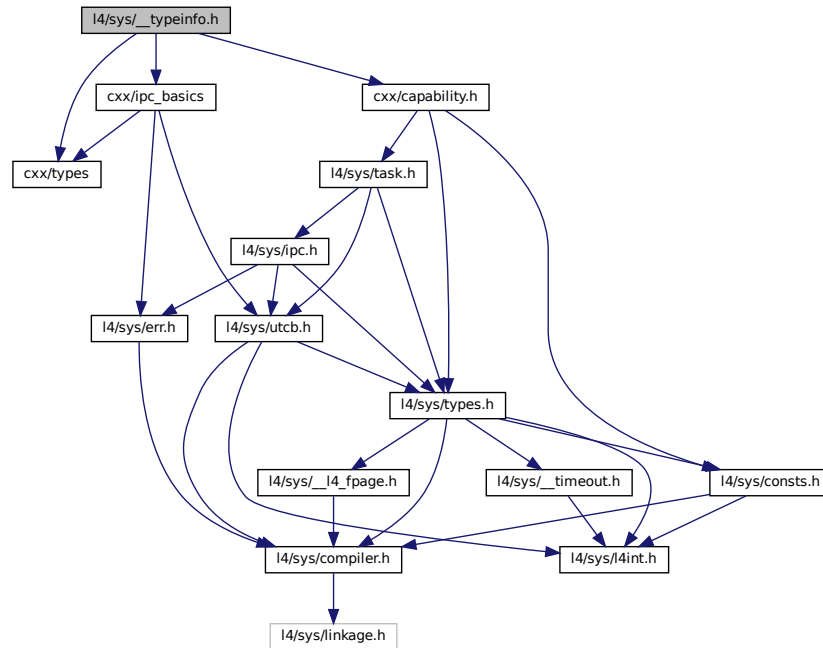
Type information handling.

```

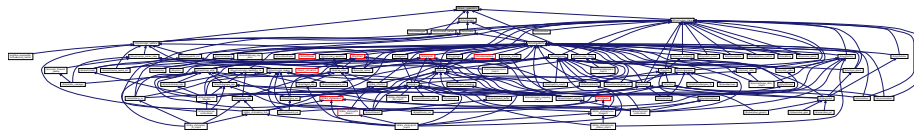
#include "cxx/types"
#include "cxx/ipc_basics"
#include "cxx/capability.h"

```

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

- [L4](#)
- [L4](#) low-level kernel interface.*
- [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.*

16.247.1 Detailed Description

Type information handling.

Definition in file [__typeinfo.h](#).

16.248 __typeinfo.h

```

00001
00005 /*
00006  * Copyright (C) 2015 Kernkonzept GmbH.
00007  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00008  */
00009 /*
00010  * (c) 2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00011  *      economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU General Public License 2.
00015  * Please see the COPYING-GPL-2 file for details.
00016  *
00017  * As a special exception, you may use this file as part of a free software
00018  * library without restriction. Specifically, if other files instantiate
00019  * templates or use macros or inline functions from this file, or you compile
00020  * this file and link it with other files to produce an executable, this
00021  * file does not by itself cause the resulting executable to be covered by
00022  * the GNU General Public License. This exception does not however
00023  * invalidate any other reasons why the executable file might be covered by
00024  * the GNU General Public License.
00025  */
00026 #pragma once
00027 #pragma GCC system_header
00028
00029 #include "cxx/types"
00030 #include "cxx/ipc_basics"
00031 #include "cxx/capability.h"
00032
00033 #if defined(__GXX_RTTI) && !defined(L4_NO_RTTI)
00034 #   include <typeinfo>
00035   typedef std::type_info const *L4_std_type_info_ptr;
00036 #   define L4_KOBJECT_META_RTTI(type) (&typeid(type))
00037   inline char const *L4_kobject_type_name(L4_std_type_info_ptr n) noexcept
00038   { return n ? n->name() : 0; }
00039 #else
00040   typedef void const *L4_std_type_info_ptr;
00041 #   define L4_KOBJECT_META_RTTI(type) (0)
00042   inline char const *L4_kobject_type_name(L4_std_type_info_ptr) noexcept
00043   { return 0; }
00044 #endif
00045
00046 namespace L4 {
00047   typedef int Opcode;
00048   // internal max helpers
00049   namespace __I {
00050     // internal max of A and B helper
00051     template< unsigned char A, unsigned char B>
00052     struct Max { enum { Res = A > B ? A : B }; };
00053   } // namespace __I
00054
00055   enum
00056   {
00058     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 template< typename L1, typename L2 >
00260 struct Conflict : __Conflict<typename L1::type, typename L2::type> {};
00261
00262 // to be removed -----
00263 // p_dispatch code -- for legacy dispatch -----
00264 /*****/
00265 /*
00266  * \internal
00267  * helper: Dispatch helper for calling server-side p_dispatch() functions.
00268  */
00269 template<typename LIST>
00270 struct __P_dispatch;
00271
00272 // No matching dispatcher found
00273 template<>
00274 struct __P_dispatch<Iface_list_end>
00275 {
00276     template< typename THIS, typename A1, typename A2 >
00277     static int f(THIS *, long, A1, A2 &) noexcept
00278     { return -L4_EBADPROTO; }
00279 };
00280
00281 // call matching p_dispatch() function
00282 template< typename I, typename LIST >
00283 struct __P_dispatch<Iface_list<I, LIST> >
00284 {
00285     // special handling for the meta protocol, to avoid 'using' murx
00286     template< typename THIS, typename A1, typename A2 >
00287     static int _f(THIS self, A1, A2 &a2, True::type)
00288     {

```

```

00294         return self->dispatch_meta_request(a2);
00295     }
00296
00297     // normal p_dispatch() dispatching
00298     template< typename THIS, typename A1, typename A2 >
00299     static int _f(THIS self, A1 a1, A2 &a2, False::type)
00300     {
00301         return self->p_dispatch(reinterpret_cast<typename I::iface_type *>(0),
00302                                 a1, a2);
00303     }
00304
00305     // dispatch function with switch for meta protocol
00306     template< typename THIS, typename A1, typename A2 >
00307     static int f(THIS *self, long proto, A1 a1, A2 &a2)
00308     {
00309         if (I::Proto == proto)
00310             return _f(self, a1, a2, 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 Rpcs_end
00326     {
00327         typedef void opcode_type;
00328         typedef Rpcs_end rpc;
00329         typedef Rpcs_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, Rpcs_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     {
00350         template<bool, typename> struct Invalid_opcode {};
00351         template<typename X> struct Invalid_opcode<true, X>;
00352     };
00353 private:
00354     template<typename U, U> struct _chk;
00355     template<typename U> static long _opc(_chk<int, U::Opcode> *);
00356     template<typename U> static char _opc(...);
00357
00358     template<unsigned SZ, typename U>
00359     struct _Opc { enum { value = CODE }; };
00360
00361     template<typename U>
00362     struct _Opc<sizeof(long), U> { enum { value = U::Opcode }; };
00363
00364 public:
00365     enum { value = _Opc<sizeof(_opc<T>(0)), T>::value };
00366     Invalid_opcode<(value < CODE), T> invalid_opcode;
00367 };
00368
00369 template<typename OPCODE, unsigned O, typename ...X>
00370 struct _Rpcs : Rpcs_end {};
00371
00372 template<typename OPCODE, unsigned O, typename R, typename ...X>
00373 struct _Rpcs<OPCODE, O, R, X...>
00374 {
00375     typedef _Rpcs type;
00376     typedef OPCODE opcode_type;
00377     typedef R rpc;
00378     typedef typename _Rpcs<OPCODE, _Get_opcode<R, O>::value + 1, X...>::type next;
00379     enum { Opcode = _Get_opcode<R, O>::value };
00380     template<typename Y> struct Rpc : Typeid::Detail::Rpc<Y, _Rpcs> {};

```

```

00393     };
00394
00395     template<typename OPCODE, unsigned O, typename R>
00396     struct _Rpc<OPCODE, O, Default_op<R> >
00397     {
00398         typedef _Rpc type;
00401         typedef void opcode_type;
00403         typedef R rpc;
00405         typedef Rpc_end next;
00407         enum { Opcode = -99 };
00409         template<typename Y> struct Rpc : Typeid::Detail::Rpc<Y, _Rpc> {};
00410     };
00411
00412     } // namespace Detail
00413
00421     template<typename CLASS>
00422     struct Raw_ipc
00423     {
00424         typedef Raw_ipc type;
00425         typedef Detail::Rpc_end next;
00426         typedef void opcode_type;
00427     };
00428
00437     template<typename ...RPCS>
00438     struct Rpc< : Detail::_Rpc<L4::Opcode, 0, RPCS...> {}> {};
00439
00448     template<typename OPCODE_TYPE>
00449     struct Rpc_code
00450     {
00454         template<typename ...RPCS>
00455         struct F : Detail::_Rpc<OPCODE_TYPE, 0, RPCS...> {};
00456     };
00457
00463     template<typename OPERATION>
00464     struct Rpc_nocode : Detail::_Rpc<void, 0, OPERATION> {};
00465
00474     template<typename ...ARG>
00475     struct Rpc_sys : Detail::_Rpc<l4_umword_t, 0, ARG...> {};
00476
00477     template<typename CLASS>
00478     struct Rights
00479     {
00480         unsigned rights;
00481         Rights(unsigned rights) noexcept : rights(rights) {}
00482         unsigned operator & (unsigned rhs) const noexcept { return rights & rhs; }
00483     };
00484
00485     } // namespace Typeid
00486
00509     struct L4_EXPORT Type_info
00510     {
00516         class L4_EXPORT Demand
00517         {
00518         private:
00520             static unsigned char max(unsigned char a, unsigned char b) noexcept
00521             { return a > b ? a : b; }
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_RTTI(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 template< typename Derived, typename Base, long PROTO, typename S_DEMAND>
00786 Type_info const *const
00787 Kobject_t<Derived, Base, PROTO, S_DEMAND>::
00788     __Kobject_typeid::_b[] = { &Base::__Kobject_typeid::_m };
00789
00795 template< typename Derived, typename Base, long PROTO, typename S_DEMAND>
00796 L4__GEN_TI(Kobject_t<Derived, Base, PROTO, S_DEMAND>);
00797
00798 template<
00799     typename Derived,
00800     typename Base1,
00801     typename Base2,
00802     long PROTO = PROTO_ANY,
00803     typename S_DEMAND = Type_info::Demand_t<>
00804 >
00835 class Kobject_2t : public Base1, public Base2
00836 {
00837 protected:
00838     typedef Derived Class;
00839     typedef Typeid::Iface<PROTO, Derived> __Iface;
00840     typedef Typeid::Merge_list<
00841         Typeid::Iface_list<__Iface>,
00842         Typeid::Merge_list<
00843             typename Base1::__Iface_list,
00844             typename Base2::__Iface_list
00845         >
00846     > __Iface_list;
00847
00852     static void __check_protocols__() noexcept
00853     {
00854         typedef typename Base1::__Iface_list Base1_proto_list;
00855         typedef typename Base2::__Iface_list Base2_proto_list;
00856
00857         typedef Typeid::Iface_conflict<__Iface, Base1_proto_list> Base1_conflict;
00858         typedef Typeid::Iface_conflict<__Iface, Base2_proto_list> Base2_conflict;
00859         static_assert(!Base1_conflict::value, "ambiguous protocol ID, also in Base1");
00860         static_assert(!Base2_conflict::value, "ambiguous protocol ID, also in Base2");
00861
00862         typedef Typeid::Conflict<Base1_proto_list, Base2_proto_list> Bases_conflict;
00863         static_assert(!Bases_conflict::value, "ambiguous protocol IDs in base classes");
00864     }
00865
00866     // disambiguate cap()
00867     l4_cap_idx_t cap() const noexcept
00868     { return Base1::cap(); }
00869
00871     L4::Cap<Class> c() const noexcept { return L4::Cap<Class>(this->cap()); }
00872
00873     L4__GEN_TI_MEMBERS(Type_info::Demand_union_t<
00874         typename Base1::__Kobject_typeid::Demand,
00875         typename Base2::__Kobject_typeid::Demand>
00876     )
00877
00878 public:
00879     // Provide non-ambiguous conversion to Kobject

```

```

00880 operator Kobject const & () const noexcept
00881 { return *static_cast<Base1 const *>(this); }
00882
00883 // Provide non-ambiguous access of dec_refcnt()
00884 l4_msgtag_t dec_refcnt(l4_mword_t diff, l4_utcb_t *utcb = l4_utcb())
00885 noexcept(noexcept(((Base1*)0)->dec_refcnt(diff, utcb)))
00886 { return Base1::dec_refcnt(diff, utcb); }
00887 };
00888
00889
00890 template< typename Derived, typename Base1, typename Base2,
00891           long PROTO, typename S_DEMAND >
00892 Type_info const *const
00893 Kobject_2t<Derived, Base1, Base2, PROTO, S_DEMAND>::__Kobject_typeid::_b[] =
00894 {
00895     &Base1::__Kobject_typeid::_m,
00896     &Base2::__Kobject_typeid::_m
00897 };
00898
00903 template< typename Derived, typename Base1, typename Base2,
00904           long PROTO, typename S_DEMAND >
00905 L4___GEN_TI(Kobject_2t<Derived, Base1, Base2, PROTO, S_DEMAND>);
00906
00907
00908
00928 template<
00929     typename Derived,
00930     typename Base1,
00931     typename Base2,
00932     typename Base3,
00933     long PROTO = PROTO_ANY,
00934     typename S_DEMAND = Type_info::Demand_t<>
00935 >
00936 struct Kobject_3t : Base1, Base2, Base3
00937 {
00938 protected:
00940     typedef Derived Class;
00942     typedef Typeid::Iface<PROTO, Derived> __Iface;
00944     typedef Typeid::Merge_list<
00945         Typeid::Iface_list<__Iface>,
00946         Typeid::Merge_list<
00947             typename Base1::__Iface_list,
00948             Typeid::Merge_list<
00949                 typename Base2::__Iface_list,
00950                 typename Base3::__Iface_list
00951             >
00952         >
00953     > __Iface_list;
00954
00956     static void __check_protocols__() noexcept
00957     {
00958         typedef typename Base1::__Iface_list Base1_proto_list;
00959         typedef typename Base2::__Iface_list Base2_proto_list;
00960         typedef typename Base3::__Iface_list Base3_proto_list;
00961
00962         typedef Typeid::Iface_conflict<__Iface, Base1_proto_list> Base1_conflict;
00963         typedef Typeid::Iface_conflict<__Iface, Base2_proto_list> Base2_conflict;
00964         typedef Typeid::Iface_conflict<__Iface, Base3_proto_list> Base3_conflict;
00965
00966         static_assert(!Base1_conflict::value, "ambiguous protocol ID, also in Base1");
00967         static_assert(!Base2_conflict::value, "ambiguous protocol ID, also in Base2");
00968         static_assert(!Base3_conflict::value, "ambiguous protocol ID, also in Base3");
00969
00970         typedef Typeid::Conflict<Base1_proto_list, Base2_proto_list> Conflict_bases12;
00971         typedef Typeid::Conflict<Base1_proto_list, Base3_proto_list> Conflict_bases13;
00972         typedef Typeid::Conflict<Base2_proto_list, Base3_proto_list> Conflict_bases23;
00973
00974         static_assert(!Conflict_bases12::value, "ambiguous protocol IDs in base classes: Base1 and
00975 Base2");
00975         static_assert(!Conflict_bases13::value, "ambiguous protocol IDs in base classes: Base1 and
00976 Base3");
00976         static_assert(!Conflict_bases23::value, "ambiguous protocol IDs in base classes: Base2 and
00977 Base3");
00977     }
00978
00979 // disambiguate cap()
00980 l4_cap_idx_t cap() const noexcept
00981 { return Base1::cap(); }
00982
00984 L4::Cap<Class> c() const noexcept { return L4::Cap<Class>(this->cap()); }
00985
00986 L4___GEN_TI_MEMBERS(Type_info::Demand_union_t<Type_info::Demand_union_t<
00987     typename Base1::__Kobject_typeid::Demand,
00988     typename Base2::__Kobject_typeid::Demand>,
00989     typename Base3::__Kobject_typeid::Demand>
00990 )
00991

```

```

00992 public:
00993     // Provide non-ambiguous conversion to Kobject
00994     operator Kobject const & () const noexcept
00995     { return *static_cast<Base1 const *>(this); }
00996
00997     // Provide non-ambiguous access of dec_refcnt()
00998     l4_msgtag_t dec_refcnt(l4_mword_t diff, l4_utcb_t *utcb = l4_utcb())
00999     noexcept(noexcept(((Base1*)0)->dec_refcnt(diff, utcb)))
01000     { return Base1::dec_refcnt(diff, utcb); }
01001 };
01002
01003
01004 template< typename Derived, typename Base1, typename Base2, typename Base3,
01005           long PROTO, typename S_DEMAND >
01006 Type_info const *const
01007 Kobject_3t<Derived, Base1, Base2, Base3, PROTO, S_DEMAND>::__Kobject_typeid::_b[] =
01008 {
01009     &Base1::__Kobject_typeid::_m,
01010     &Base2::__Kobject_typeid::_m,
01011     &Base3::__Kobject_typeid::_m
01012 };
01013
01014 template< typename Derived, typename Base1, typename Base2, typename Base3,
01015           long PROTO, typename S_DEMAND >
01016 L4___GEN_TI(Kobject_3t<Derived, Base1, Base2, Base3, PROTO, S_DEMAND>);
01017
01018 }
01019
01020 #if __cplusplus >= 201103L
01021
01022 namespace L4 {
01023
01024     template< typename ...T >
01025     struct Kobject_demand;
01026
01027     template<>
01028     struct Kobject_demand<> : Type_info::Demand_t<> {};
01029
01030     template<typename T>
01031     struct Kobject_demand<T> : Kobject_typeid<T>::Demand {};
01032
01033     template<typename T1, typename ...T2>
01034     struct Kobject_demand<T1, T2...> :
01035         Type_info::Demand_union_t<typename Kobject_typeid<T1>::Demand,
01036                                   Kobject_demand<T2...> >
01037     {};
01038
01039 namespace Typeid_xx {
01040
01041     template<typename ...LISTS>
01042     struct Merge_list;
01043
01044     template<typename L>
01045     struct Merge_list<L> : L {};
01046
01047     template<typename L1, typename L2>
01048     struct Merge_list<L1, L2> : Typeid::Merge_list<L1, L2> {};
01049
01050     template<typename L1, typename L2, typename ...LISTS>
01051     struct Merge_list<L1, L2, LISTS...> :
01052         Merge_list<typename Typeid::Merge_list<L1, L2>::type, LISTS...> {};
01053
01054     template< typename I, typename ...LIST >
01055     struct Iface_conflict;
01056
01057     template< typename I >
01058     struct Iface_conflict<I> : Typeid::False {};
01059
01060     template< typename I, typename L, typename ...LIST >
01061     struct Iface_conflict<I, L, LIST...> :
01062         Typeid::Bool<Typeid::Iface_conflict<typename I::type, typename L::type>::value
01063                     || Iface_conflict<I, LIST...>::value>
01064     {};
01065
01066     template< typename ...LIST >
01067     struct Conflict;
01068
01069     template< typename L >
01070     struct Conflict<L> : Typeid::False {};
01071
01072     template< typename L1, typename L2, typename ...LIST >
01073     struct Conflict<L1, L2, LIST...> :
01074         Typeid::Bool<Typeid::Conflict<typename L1::type, typename L2::type>::value
01075                     || Conflict<L1, LIST...>::value
01076                     || Conflict<L2, LIST...>::value>
01077     {};
01078
01079
01080
01081
01082
01083
01084
01085
01086
01087
01088

```

```

01089     template< typename T >
01090     struct Is_demand
01091     {
01092         static long test(Type_info::Demand const *);
01093         static char test(...);
01094         enum { value = sizeof(test((T*)0)) == sizeof(long) };
01095     };
01096
01097     template< typename T, typename ... >
01098     struct First : T { typedef T type; };
01099 } // Typeid
01100
01106 template< typename Derived, long PROTO, typename S_DEMAND, typename ...BASES>
01107 struct __Kobject_base : BASES...
01108 {
01109 protected:
01110     typedef Derived Class;
01111     typedef Typeid::Iface<PROTO, Derived> __Iface;
01112     typedef Typeid_xx::Merge_list<
01113         Typeid::Iface_list<__Iface>,
01114         typename BASES::__Iface_list...
01115     > __Iface_list;
01116
01117     static void __check_protocols__() noexcept
01118     {
01119         typedef Typeid_xx::Iface_conflict<__Iface, typename BASES::__Iface_list...> Conflict;
01120         static_assert(!Conflict::value, "ambiguous protocol ID, protocol also used in base class");
01121
01122         typedef Typeid_xx::Conflict<typename BASES::__Iface_list...> Base_conflict;
01123         static_assert(!Base_conflict::value, "ambiguous protocol IDs in base classes");
01124     }
01125
01126     // disambiguate cap()
01127     l4_cap_idx_t cap() const noexcept
01128     { return Typeid_xx::First<BASES...>::type::cap(); }
01129
01130     L4::Cap<Class> c() const noexcept { return L4::Cap<Class>(this->cap()); }
01131
01132     L4__GEN_TI_MEMBERS(Kobject_demand<BASES...>)
01133
01134 private:
01135     // This function returns the first base class (used below)
01136     template<typename B1, typename ...> struct Basel { typedef B1 type; };
01137
01138 public:
01139     // Provide non-ambiguous conversion to Kobject
01140     operator Kobject const & () const noexcept
01141     { return *static_cast<typename Basel<BASES...>::type const *>(this); }
01142
01143     // Provide non-ambiguous access of dec_refcnt()
01144     l4_msgtag_t dec_refcnt(l4_mword_t diff, l4_utcb_t *utcb = l4_utcb())
01145     noexcept(noexcept(((typename Basel<BASES...>::type *)0)->dec_refcnt(diff, utcb)))
01146     { return Basel<BASES...>::type::dec_refcnt(diff, utcb); }
01147 };
01148
01149 template< typename Derived, long PROTO, typename S_DEMAND, typename ...BASES>
01150 Type_info const *const
01151 __Kobject_base<Derived, PROTO, S_DEMAND, BASES...>::__Kobject_typeid::b[] =
01152 {
01153     (&BASES::__Kobject_typeid::m)...
01154 };
01155
01156 template< typename Derived, long PROTO, typename S_DEMAND, typename ...BASES>
01157 L4__GEN_TI(__Kobject_base<Derived, PROTO, S_DEMAND, BASES...>);
01158
01159 // Test if the there is a Demand argument to Kobject_x
01160 template< typename Derived, long PROTO, bool HAS_DEMAND, typename DEMAND, typename ...ARGS >
01161 struct __Kobject_x_proto;
01162
01163 // YES: pass it to __Kobject_base
01164 template< typename Derived, long PROTO, typename DEMAND, typename ...BASES>
01165 struct __Kobject_x_proto<Derived, PROTO, true, DEMAND, BASES...> :
01166     __Kobject_base<Derived, PROTO, DEMAND, BASES...> {};
01167
01168 // NO: pass it empty Type_info::Demand_t
01169 template< typename Derived, long PROTO, typename B1, typename ...BASES>
01170 struct __Kobject_x_proto<Derived, PROTO, false, B1, BASES...> :
01171     __Kobject_base<Derived, PROTO, Type_info::Demand_t<>, B1, BASES...> {};
01172
01173 template< long P = PROTO_EMPTY >
01174 struct Proto_t {};
01175
01176 template< typename Derived, typename ...ARGS >
01177 struct Kobject_x;
01178
01179 template< typename Derived, typename A, typename ...ARGS >

```



```

01201 struct Kobject_x<Derived, A, ARGS...> :
01202   __Kobject_x_proto<Derived, PROTO_ANY, Typeid_xx::Is_demand<A>::value, A, ARGS...>
01203 {};
01204
01205 template< typename Derived, long PROTO, typename A, typename ...ARGS >
01206 struct Kobject_x<Derived, Proto_t<PROTO>, A, ARGS...> :
01207   __Kobject_x_proto<Derived, PROTO, Typeid_xx::Is_demand<A>::value, A, ARGS...>
01208 {};
01209
01210 }
01211 #endif
01212
01213 #undef L4___GEN_TI
01214 #undef L4___GEN_TI_MEMBERS
01215

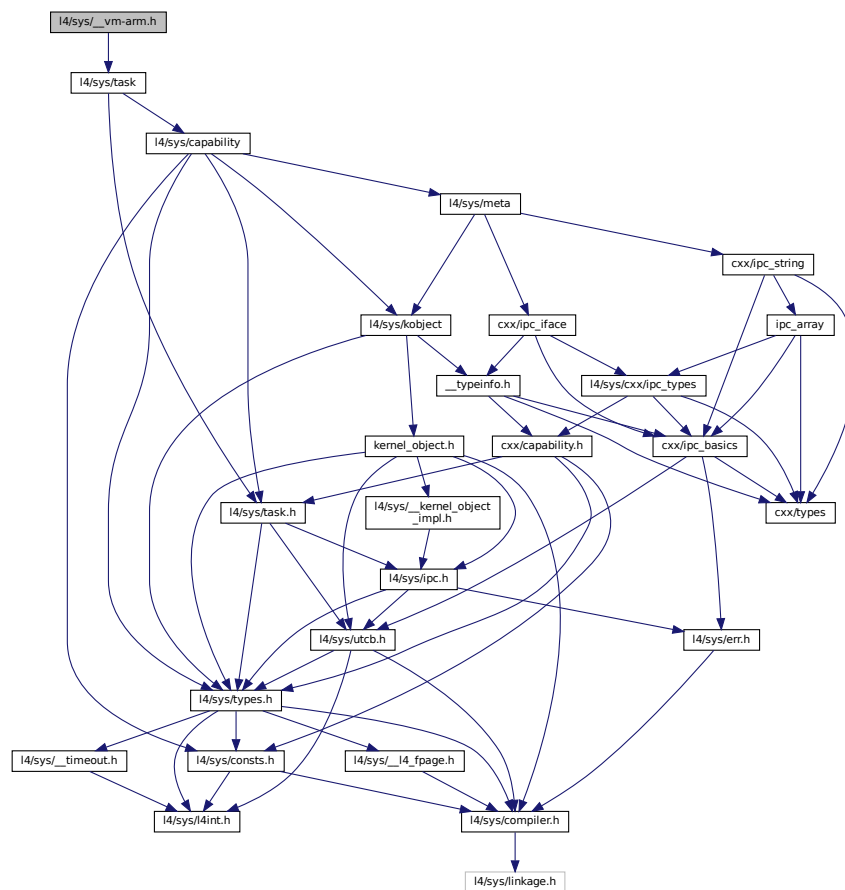
```

16.249 l4/sys/_vm-arm.h File Reference

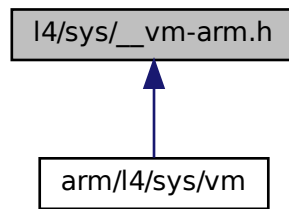
Virtualization interface.

```
#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

- [L4](#)
L4 low-level kernel interface.

16.249.1 Detailed Description

Virtualization interface.

Definition in file [__vm-arm.h](#).

16.250 [__vm-arm.h](#)

```

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
00036 class Vm : public Kobject_t<Vm, Task, L4_PROTO_VM>
00037 {

```

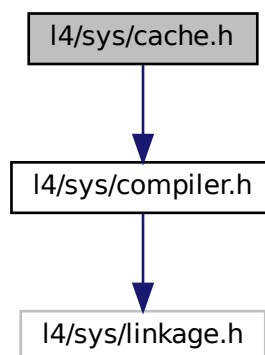
```

00038 public:
00039     /*
00040      * Map the GIC's virtual GICC page to the task.
00041      *
00042      * \param vgicc_fpage    Flexpage that describes an area in the address space
00043      *                       of the destination task to map the vGICC page to.
00044      * \utcb{utcb}
00045      *
00046      * \return Syscall return tag.
00047      */
00048     l4_msgtag_t vgicc_map(l4_fpage_t const vgicc_fpage,
00049                          l4_utcb_t *utcb = l4_utcb()) noexcept
00050     { return l4_task_vgicc_map_u(cap(), vgicc_fpage, utcb); }
00051
00052 protected:
00053     Vm();
00054
00055 private:
00056     Vm(Vm const &);
00057     void operator = (Vm const &);
00058 };
00059
00060 }
```

16.251 l4/sys/cache.h File Reference

Cache-consistency functions.

```
#include <l4/sys/compiler.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.
- int [l4_cache_flush_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache flush a range.
- int [l4_cache_inv_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache invalidate a range.
- int [l4_cache_coherent](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Make memory coherent between I-cache and D-cache.

- int [l4_cache_dma_coherent](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Make memory coherent for use with external memory.
- int [l4_cache_dma_coherent_full](#) (void) [L4_NOTHROW](#)
Make memory coherent for use with external memory.

16.251.1 Detailed Description

Cache-consistency functions.

Date

2007-11

Author

Adam Lackorzynski adam@os.inf.tu-dresden.de

Definition in file [cache.h](#).

16.252 cache.h

```

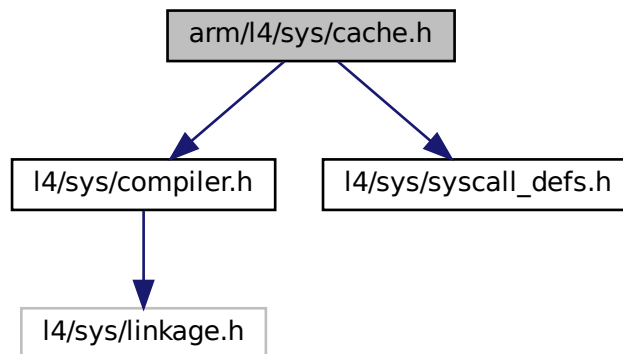
00001
00011 /*
00012  * (c) 2007-2009 Author(s)
00013  *     economic rights: Technische Universität Dresden (Germany)
00014  *
00015  * This file is part of TUD:OS and distributed under the terms of the
00016  * GNU General Public License 2.
00017  * Please see the COPYING-GPL-2 file for details.
00018  *
00019  * As a special exception, you may use this file as part of a free software
00020  * library without restriction. Specifically, if other files instantiate
00021  * templates or use macros or inline functions from this file, or you compile
00022  * this file and link it with other files to produce an executable, this
00023  * file does not by itself cause the resulting executable to be covered by
00024  * the GNU General Public License. This exception does not however
00025  * invalidate any other reasons why the executable file might be covered by
00026  * the GNU General Public License.
00027  */
00028
00029 #ifndef __L4SYS__INCLUDE__CACHE_H__
00030 #define __L4SYS__INCLUDE__CACHE_H__
00031
00032 #include <l4/sys/compiler.h>
00033
00042 EXTERN_C_BEGIN
00043
00058 L4_INLINE int
00059 l4_cache_clean_data(unsigned long start,
00060                    unsigned long end) L4_NOTHROW;
00061
00076 L4_INLINE int
00077 l4_cache_flush_data(unsigned long start,
00078                    unsigned long end) L4_NOTHROW;
00079
00098 L4_INLINE int
00099 l4_cache_inv_data(unsigned long start,
00100                  unsigned long end) L4_NOTHROW;
00101
00113 L4_INLINE int
00114 l4_cache_coherent(unsigned long start,
00115                  unsigned long end) L4_NOTHROW;
00116
00128 L4_INLINE int
00129 l4_cache_dma_coherent(unsigned long start,
00130                      unsigned long end) L4_NOTHROW;
00131
00136 L4_INLINE int
00137 l4_cache_dma_coherent_full(void) L4_NOTHROW;
00138
00139 EXTERN_C_END
00140
00141 #endif /* ! __L4SYS__INCLUDE__CACHE_H__ */

```

16.253 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.
- int [l4_cache_flush_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache flush a range.
- int [l4_cache_inv_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache invalidate a range.
- int [l4_cache_coherent](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Make memory coherent between I-cache and D-cache.
- int [l4_cache_dma_coherent](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Make memory coherent for use with external memory.
- int [l4_cache_dma_coherent_full](#) (void) [L4_NOTHROW](#)
Make memory coherent for use with external memory.

16.253.1 Detailed Description

Cache functions.

Date

2007-11

Author

Adam Lackorzynski adam@os.inf.tu-dresden.de

Definition in file [cache.h](#).

16.254 cache.h

```

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      "mov    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__ */

```

16.255 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.
- int [l4_cache_flush_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache flush a range.
- int [l4_cache_inv_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache invalidate a range.
- int [l4_cache_coherent](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Make memory coherent between I-cache and D-cache.
- int [l4_cache_dma_coherent](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Make memory coherent for use with external memory.
- int [l4_cache_dma_coherent_full](#) (void) [L4_NOTHROW](#)
Make memory coherent for use with external memory.

16.255.1 Detailed Description

Cache functions.

Definition in file [cache.h](#).

16.256 cache.h

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
00013  * As a special exception, you may use this file as part of a free software
00014  * library without restriction. Specifically, if other files instantiate
00015  * templates or use macros or inline functions from this file, or you compile
00016  * this file and link it with other files to produce an executable, this
00017  * file does not by itself cause the resulting executable to be covered by
00018  * the GNU General Public License. This exception does not however
00019  * invalidate any other reasons why the executable file might be covered by
00020  * the GNU General Public License.
00021  */
00022 #ifndef __L4SYS__INCLUDE__ARCH_AMD64__CACHE_H__
00023 #define __L4SYS__INCLUDE__ARCH_AMD64__CACHE_H__
00024
00025 #include_next <l4/sys/cache.h>
00026
00027 L4_INLINE int
00028 l4_cache_clean_data(unsigned long start,
00029                    unsigned long end) L4_NOTHROW
00030 {
00031     (void)start; (void)end;
00032     return 0;
00033 }
00034
00035 L4_INLINE int
00036 l4_cache_flush_data(unsigned long start,
00037                    unsigned long end) L4_NOTHROW
00038 {
00039     (void)start; (void)end;
00040     return 0;
00041 }
00042
00043 L4_INLINE int
00044 l4_cache_inv_data(unsigned long start,
00045                  unsigned long end) L4_NOTHROW
00046 {
00047     (void)start; (void)end;
00048     return 0;
00049 }
00050
00051 L4_INLINE int
00052 l4_cache_coherent(unsigned long start,
00053                  unsigned long end) L4_NOTHROW
00054 {
00055     (void)start; (void)end;
00056     return 0;
00057 }
00058
00059 L4_INLINE int
00060 l4_cache_dma_coherent(unsigned long start,
00061                      unsigned long end) L4_NOTHROW
00062 {
00063     (void)start; (void)end;
00064     return 0;
00065 }
00066
00067 L4_INLINE int
00068 l4_cache_dma_coherent_full(void) L4_NOTHROW
00069 {
00070     return 0;
00071 }
00072
00073 #endif /* ! __L4SYS__INCLUDE__ARCH_AMD64__CACHE_H__ */

```

16.257 x86/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.

- int [l4_cache_flush_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)

Cache flush a range.

- int [l4_cache_inv_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)

Cache invalidate a range.

- int [l4_cache_coherent](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)

Make memory coherent between I-cache and D-cache.

- int [l4_cache_dma_coherent](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)

Make memory coherent for use with external memory.

- int [l4_cache_dma_coherent_full](#) (void) [L4_NOTHROW](#)

Make memory coherent for use with external memory.

16.257.1 Detailed Description

Cache functions.

Definition in file [cache.h](#).

16.258 cache.h

```

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__ */

```

16.259 I4/sys/capability File Reference

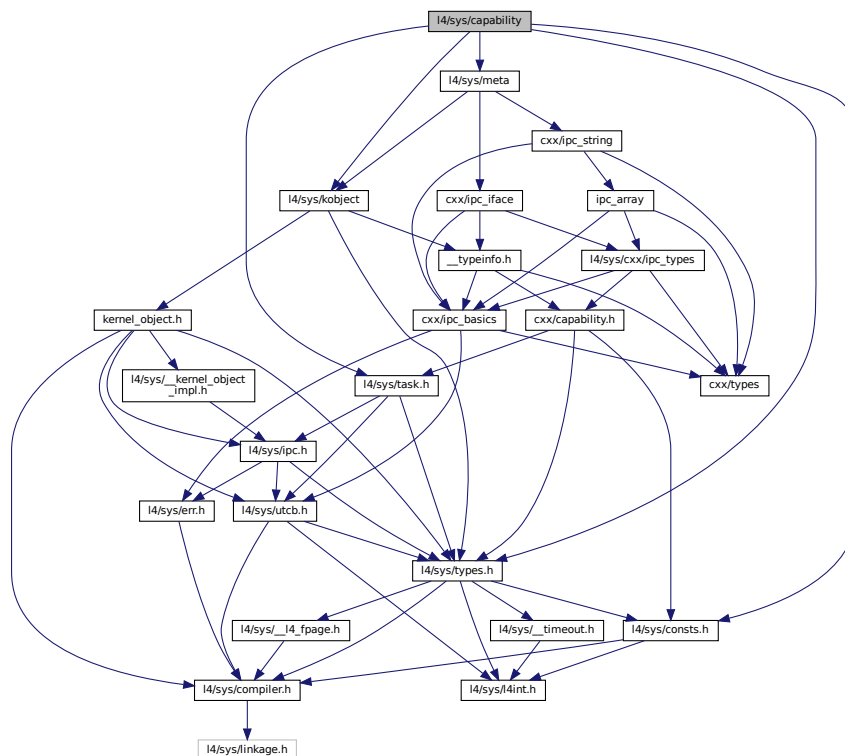
[L4::Cap](#) related definitions.

```

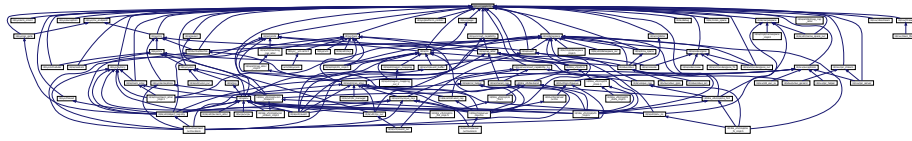
#include <l4/sys/consts.h>
#include <l4/sys/types.h>
#include <l4/sys/kobject>
#include <l4/sys/task.h>
#include <l4/sys/meta>

```

Include dependency graph for capability:



This graph shows which files directly or indirectly include this file:



Namespaces

- [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.

16.259.1 Detailed Description

[L4::Cap](#) related definitions.

Author

Alexander Warg alexander.warg@os.inf.tu-dresden.de

Definition in file [capability](#).

16.259.2 Macro Definition Documentation

16.259.2.1 L4_DISABLE_COPY

```
#define L4_DISABLE_COPY(  
    _class )
```

Value:

```
public:  
    _class(_class const &) = delete; \br/>    _class operator = (_class const &) = delete; \br/>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](#).

16.260 capability

```
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 #define L4_KOBJECT_DISABLE_COPY(_class) \
00074     protected: \
00075         _class(); \
00076         L4_DISABLE_COPY(_class)
00077
00078 #define L4_KOBJECT(_class) L4_KOBJECT_DISABLE_COPY(_class)
00079
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
00102 template< typename T, typename F >
00103 inline
00104 Cap<T> cap_dynamic_cast(Cap<F> const &c) noexcept
00105 {
00106     if (!c.is_valid())
00107         return Cap<T>::Invalid;
00108
00109     Cap<Meta> mc = cap_reinterpret_cast<Meta>(c);
00110     Type_info const *m = kobject_typeid<T>();
00111     if (m->proto() && l4_error(mc->supports(m->proto())) > 0)
00112         return Cap<T>(c.cap());
00113
00114     // FIXME: use generic checker
00115     #if 0
00116     if (l4_error(mc->supports(T::kobject_proto())) > 0)
00117         return Cap<T>(c.cap());
00118     #endif
00119
00120     return Cap<T>::Invalid;
00121 }
00122
00123 }
00124
00125 }

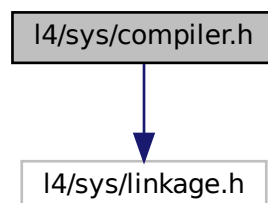
```

16.261 l4/sys/compiler.h File Reference

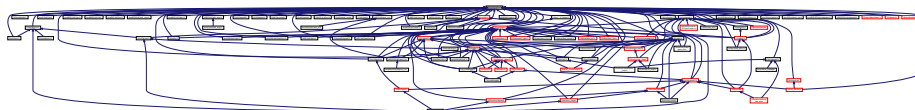
L4 compiler related defines.

```
#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_ALWAYS_INLINE`
L4 Inline function attribute.
- `#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 __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_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

- `void l4_barrier (void)`
Memory barrier.
- `void l4_mb (void)`
Memory barrier.
- `void l4_wmb (void)`
Write memory barrier.

16.261.1 Detailed Description

L4 compiler related defines.

Definition in file [compiler.h](#).

16.261.2 Macro Definition Documentation

16.261.2.1 L4_ALWAYS_INLINE

```
#define L4_ALWAYS_INLINE
```

[L4](#) Inline function attribute.

Always inline a function

Definition at line 67 of file [compiler.h](#).

16.261.2.2 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 216 of file [compiler.h](#).

16.261.2.3 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 186 of file [compiler.h](#).

16.262 compiler.h

```

00001 /*****
00007 */
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00010 *      Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00011 *      Jork Löser <jork@os.inf.tu-dresden.de>,
00012 *      Ronald Aigner <ra3@os.inf.tu-dresden.de>
00013 *      economic rights: Technische Universität Dresden (Germany)
00014 *
00015 * This file is part of TUD:OS and distributed under the terms of the
00016 * GNU General Public License 2.
00017 * Please see the COPYING-GPL-2 file for details.
00018 *
00019 * As a special exception, you may use this file as part of a free software
00020 * library without restriction. Specifically, if other files instantiate
00021 * templates or use macros or inline functions from this file, or you compile
00022 * this file and link it with other files to produce an executable, this
00023 * file does not by itself cause the resulting executable to be covered by
00024 * the GNU General Public License. This exception does not however
00025 * invalidate any other reasons why the executable file might be covered by
00026 * the GNU General Public License.
00027 */
00028 /*****
00029 #ifndef __L4_COMPILER_H__
00030 #define __L4_COMPILER_H__
00031
00032 #if !defined(__ASSEMBLY__) && !defined(__ASSEMBLER__)
00033
00040
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 #endif /* L4_INLINE */
00062
00067 #define L4_ALWAYS_INLINE L4_INLINE __attribute__((__always_inline__))
00068
00069
00070 #define L4_DECLARE_CONSTRUCTOR(func, prio) \
00071     static inline __attribute__((constructor(prio))) void func ## _ctor_func(void) { func(); }
00072
00073
00171 #ifndef __cplusplus
00172 #   define L4_NOTHROW_A      __attribute__((nothrow))
00173 #   define L4_NOTHROW
00174 #   define EXTERN_C_BEGIN
00175 #   define EXTERN_C_END
00176 #   define EXTERN_C
00177 #   ifdef __BEGIN_DECLS
00178 #       define __BEGIN_DECLS
00179 #   endif
00180 #   ifdef __END_DECLS
00181 #       define __END_DECLS
00182 #   endif
00183 #   define L4_DEFAULT_PARAM(x)
00184 #else /* __cplusplus */
00185 #   if __cplusplus >= 201103L
00186 #       define L4_NOTHROW noexcept
00187 #   else /* C++ < 11 */
00188 #       define L4_NOTHROW throw()
00189 #   endif
00190 #   define EXTERN_C_BEGIN extern "C" {
00191 #   define EXTERN_C_END }
00192 #   define EXTERN_C extern "C"
00193 #   ifdef __BEGIN_DECLS
00194 #       define __BEGIN_DECLS extern "C" {
00195 #   endif
00196 #   ifdef __END_DECLS
00197 #       define __END_DECLS }
00198 #   endif
00199 #   define L4_DEFAULT_PARAM(x) = x
00200 #endif /* __cplusplus */
00201

```



```

00206 #define L4_NORETURN __attribute__((noreturn))
00207
00208 #define L4_PURE __attribute__((pure))
00209
00214 #define L4_NOINSTRUMENT __attribute__((no_instrument_function))
00215 #ifndef L4_HIDDEN
00216 #   define L4_HIDDEN __attribute__((visibility("hidden")))
00217 #endif
00218 #ifndef L4_EXPORT
00219 #   define L4_EXPORT __attribute__((visibility("default")))
00220 #endif
00221 #ifndef L4_EXPORT_TYPE
00222 #   ifdef __cplusplus
00223 #       define L4_EXPORT_TYPE __attribute__((visibility("default")))
00224 #   else
00225 #       define L4_EXPORT_TYPE
00226 #   endif
00227 #endif
00228 #define L4_STRONG_ALIAS(name, aliasname) L4_STRONG_ALIAS(name, aliasname)
00229 #define L4__STRONG_ALIAS(name, aliasname) \
00230     extern __typeof (name) aliasname __attribute__((alias (#name)));
00231
00232
00233 #endif /* !__ASSEMBLY__ */
00234
00235 #include <l4/sys/linkage.h>
00236
00237 #define L4_LIKELY(x) __builtin_expect((x),1)
00238 #define L4_UNLIKELY(x) __builtin_expect((x),0)
00239
00240 /* Make sure that the function is not removed by optimization. Without the
00241  * "used" attribute, unreferenced static functions are removed. */
00242 #define L4_STICKY(x) __attribute__((used)) x
00243 #define L4_DEPRECATED(s) __attribute__((deprecated(s)))
00244
00245 #ifndef __GXX_EXPERIMENTAL_CXX0X__
00246 #ifndef static_assert
00247 #define static_assert(x, y) \
00248     do { (void)sizeof(char[!(x)]); } while (0)
00249 #endif
00250 #endif
00251
00252 #define L4_stringify_helper(x) #x
00253 #define L4_stringify(x) L4_stringify_helper(x)
00254
00255 #ifndef __ASSEMBLER__
00259 L4_INLINE void l4_barrier(void);
00260
00264 L4_INLINE void l4_mb(void);
00265
00269 L4_INLINE void l4_wmb(void);
00270
00271
00272 /* Implementations */
00273 L4_INLINE void l4_barrier(void)
00274 {
00275     __asm__ __volatile__ ("": : : "memory");
00276 }
00277
00278 L4_INLINE void l4_mb(void)
00279 {
00280     __asm__ __volatile__ ("": : : "memory");
00281 }
00282
00283 L4_INLINE void l4_wmb(void)
00284 {
00285     __asm__ __volatile__ ("": : : "memory");
00286 }
00287 #endif
00288
00291 #endif /* !__L4_COMPILER_H__ */

```

16.263 l4/sys/consts.h File Reference

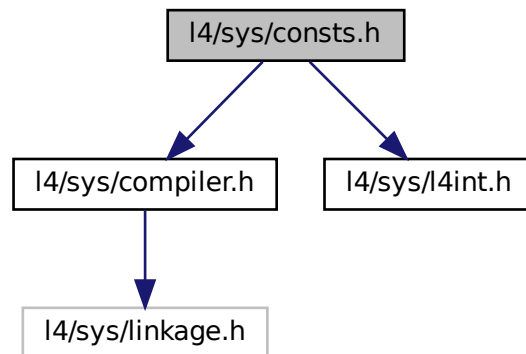
Common constants.

```

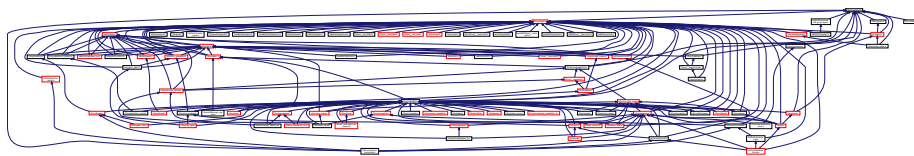
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>

```

Include dependency graph for consts.h:



This graph shows which files directly or indirectly include this file:



Macros

- `#define L4_PAGESIZE`
Minimal page size (in bytes).
- `#define L4_PAGEMASK`
Mask for the page number.
- `#define L4_LOG2_PAGESIZE`
Number of bits used for page offset.
- `#define L4_SUPERPAGESIZE`
Size of a large page.
- `#define L4_SUPERPAGEMASK`
Mask for the number of a large page.
- `#define L4_LOG2_SUPERPAGESIZE`
Number of bits used as offset for a large page.
- `#define L4_INVALID_PTR ((void *)L4_INVALID_ADDR)`
Invalid address as pointer type.

Enumerations

- enum [l4_syscall_flags_t](#) {
[L4_SYSF_NONE](#), [L4_SYSF_SEND](#), [L4_SYSF_RECV](#), [L4_SYSF_OPEN_WAIT](#),
[L4_SYSF_REPLY](#), [L4_SYSF_CALL](#), [L4_SYSF_WAIT](#), [L4_SYSF_SEND_AND_WAIT](#),
[L4_SYSF_REPLY_AND_WAIT](#) }
Capability selector flags.
- enum [l4_cap_consts_t](#) { [L4_CAP_SHIFT](#), [L4_CAP_SIZE](#), [L4_CAP_OFFSET](#) = 1UL << [L4_CAP_SHIFT](#),
[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_unword_t](#)) are required to store a buffer of 'size' bytes.

16.263.1 Detailed Description

Common constants.

Definition in file [consts.h](#).

16.264 consts.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 #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
00039 enum l4_syscall_flags_t
00040 {
00048     L4_SYSF_NONE          = 0x00,
00049
00058     L4_SYSF_SEND          = 0x01,
00059
00069     L4_SYSF_RECV          = 0x02,
00070
00080     L4_SYSF_OPEN_WAIT     = 0x04,
00081
00089     L4_SYSF_REPLY         = 0x08,
00090
00097     L4_SYSF_CALL          = L4_SYSF_SEND | L4_SYSF_RECV,
00098
00105     L4_SYSF_WAIT          = L4_SYSF_OPEN_WAIT | L4_SYSF_RECV,
00106
00113     L4_SYSF_SEND_AND_WAIT = L4_SYSF_OPEN_WAIT | L4_SYSF_CALL,
00114
00121     L4_SYSF_REPLY_AND_WAIT = L4_SYSF_WAIT | L4_SYSF_SEND | L4_SYSF_REPLY
00122 };
00123
00128 enum l4_cap_consts_t
00129 {
00131     L4_CAP_SHIFT          = 12UL,
00133     L4_CAP_SIZE           = 1UL << L4_CAP_SHIFT,
00134     L4_CAP_OFFSET         = 1UL << L4_CAP_SHIFT,
00139     L4_CAP_MASK           = ~0UL << (L4_CAP_SHIFT - 1),
00141     L4_INVALID_CAP        = ~0UL << (L4_CAP_SHIFT - 1),
00142
00143     L4_INVALID_CAP_BIT    = 1UL << (L4_CAP_SHIFT - 1),
00144 };
00145
00146 enum l4_sched_consts_t
00147 {
00148     L4_SCHED_MIN_PRIO     = 0,
00149     L4_SCHED_MAX_PRIO     = 255,
00150 };
00151
00157 enum l4_unmap_flags_t
00158 {
00165     L4_FP_ALL_SPACES      = 0x80000000UL,
00166
00173     L4_FP_DELETE_OBJ      = 0xc0000000UL,
00174
00180     L4_FP_OTHER_SPACES    = 0x00UL
00181 };
00182
00187 enum l4_msg_item_consts_t
00188 {
00189     L4_ITEM_MAP            = 8,
00190
00195     L4_ITEM_CONT          = 1,
00196
00197     // send
00198     L4_MAP_ITEM_GRANT      = 2,
00199     L4_MAP_ITEM_MAP        = 0,
00200
00201     // receive

```

```

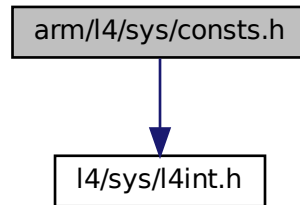
00206 L4_RCV_ITEM_SINGLE_CAP = L4_ITEM_MAP | 2,
00207
00212 L4_RCV_ITEM_LOCAL_ID = 4,
00213 };
00214
00219 enum l4_buffer_desc_consts_t
00220 {
00221 L4_BDR_MEM_SHIFT = 0,
00222 L4_BDR_IO_SHIFT = 5,
00223 L4_BDR_OBJ_SHIFT = 10,
00224 L4_BDR_OFFSET_MASK = (1UL < 20) - 1,
00225 };
00226
00240 enum l4_default_caps_t
00241 {
00243 L4_BASE_TASK_CAP = 1UL < L4_CAP_SHIFT,
00245 L4_BASE_FACTORY_CAP = 2UL < L4_CAP_SHIFT,
00247 L4_BASE_THREAD_CAP = 3UL < L4_CAP_SHIFT,
00255 L4_BASE_PAGER_CAP = 4UL < L4_CAP_SHIFT,
00263 L4_BASE_LOG_CAP = 5UL < L4_CAP_SHIFT,
00265 L4_BASE_ICU_CAP = 6UL < L4_CAP_SHIFT,
00267 L4_BASE_SCHEDULER_CAP = 7UL < L4_CAP_SHIFT,
00274 L4_BASE_IOMMU_CAP = 8UL < L4_CAP_SHIFT,
00282 L4_BASE_DEBUGGER_CAP = 10UL < L4_CAP_SHIFT,
00289 L4_BASE_ARM_SMCCC_CAP = 11UL < L4_CAP_SHIFT,
00290
00292 L4_BASE_CAPS_LAST_P1,
00294 L4_BASE_CAPS_LAST = L4_BASE_CAPS_LAST_P1 - 1
00295 };
00296
00307 #define L4_PAGESIZE (1UL < L4_PAGESHIFT)
00308
00316 #define L4_PAGEMASK (~(L4_PAGESIZE - 1))
00317
00325 #define L4_LOG2_PAGESIZE L4_PAGESHIFT
00326
00334 #define L4_SUPERPAGESIZE (1UL < L4_SUPERPAGESHIFT)
00335
00343 #define L4_SUPERPAGEMASK (~(L4_SUPERPAGESIZE - 1))
00344
00351 #define L4_LOG2_SUPERPAGESIZE L4_SUPERPAGESHIFT
00352
00363 L4_INLINE l4_addr_t l4_trunc_page(l4_addr_t address) L4_NOTHROW;
00364 L4_INLINE l4_addr_t l4_trunc_page(l4_addr_t address) L4_NOTHROW
00365 { return address & L4_PAGEMASK; }
00366
00374 L4_INLINE l4_addr_t l4_trunc_size(l4_addr_t address, unsigned char bits) L4_NOTHROW;
00375 L4_INLINE l4_addr_t l4_trunc_size(l4_addr_t address, unsigned char bits) L4_NOTHROW
00376 { return address & (~0UL < bits); }
00377
00388 L4_INLINE l4_addr_t l4_round_page(l4_addr_t address) L4_NOTHROW;
00389 L4_INLINE l4_addr_t l4_round_page(l4_addr_t address) L4_NOTHROW
00390 { return (address + L4_PAGESIZE - 1) & L4_PAGEMASK; }
00391
00399 L4_INLINE l4_addr_t l4_round_size(l4_addr_t value, unsigned char bits) L4_NOTHROW;
00400 L4_INLINE l4_addr_t l4_round_size(l4_addr_t value, unsigned char bits) L4_NOTHROW
00401 { return (value + (1UL < bits) - 1) & (~0UL < bits); }
00402
00411 L4_INLINE unsigned l4_bytes_to_mwords(unsigned size) L4_NOTHROW;
00412 L4_INLINE unsigned l4_bytes_to_mwords(unsigned size) L4_NOTHROW
00413 { return (size + sizeof(l4_umword_t) - 1) / sizeof(l4_umword_t); }
00414
00419 enum l4_addr_consts_t {
00421 L4_INVALID_ADDR = ~0UL
00422 };
00423
00428 #define L4_INVALID_PTR ((void *)L4_INVALID_ADDR)
00429
00430 #ifndef NULL
00431 #ifndef __cplusplus
00432 # define NULL ((void *)0)
00436 #else
00437 # define NULL 0
00438 #endif
00439 #endif
00440
00441 #endif /* ! __L4_SYS__INCLUDE__CONSTS_H__ */

```

16.265 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.

16.265.1 Detailed Description

Common [L4](#) constants, arm version.

Definition in file [consts.h](#).

16.266 consts.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  *      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 */

```

16.267 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.

16.267.1 Detailed Description

Common [L4](#) constants, amd64 version.

Definition in file [consts.h](#).

16.268 consts.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  * 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__ */

```

16.269 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.

16.269.1 Detailed Description

Common [L4](#) constants, x86 version.

Definition in file [consts.h](#).

16.270 consts.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 *           Lars Reuther <reuther@os.inf.tu-dresden.de>
00012 *           economic rights: Technische Universität Dresden (Germany)
00013 *
00014 * This file is part of TUD:OS and distributed under the terms of the
00015 * GNU General Public License 2.
00016 * Please see the COPYING-GPL-2 file for details.
00017 *
00018 * As a special exception, you may use this file as part of a free software
00019 * library without restriction. Specifically, if other files instantiate
00020 * templates or use macros or inline functions from this file, or you compile
00021 * this file and link it with other files to produce an executable, this
00022 * file does not by itself cause the resulting executable to be covered by
00023 * the GNU General Public License. This exception does not however
00024 * invalidate any other reasons why the executable file might be covered by
00025 * the GNU General Public License.
00026 */
00027 /*****
00028 #ifndef __L4SYS__INCLUDE__ARCH_X86__CONSTS_H__
00029 #define __L4SYS__INCLUDE__ARCH_X86__CONSTS_H__
00030
00035 #define L4_PAGESHIFT      12
00036
00041 #define L4_SUPERPAGESHIFT 22
00042
00043 #include_next <l4/sys/consts.h>
00044
00045 #endif /* ! __L4SYS__INCLUDE__ARCH_X86__CONSTS_H__ */

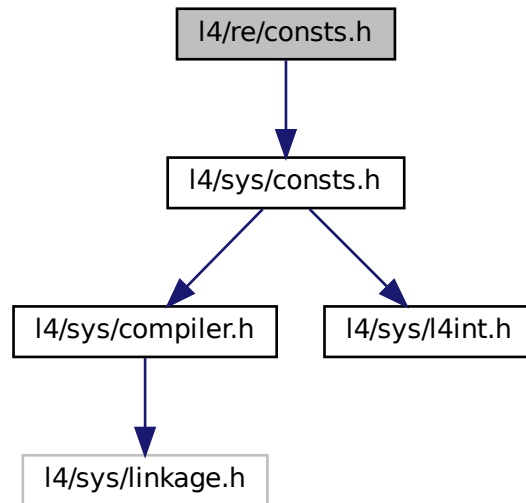
```

16.271 l4/re/consts.h File Reference

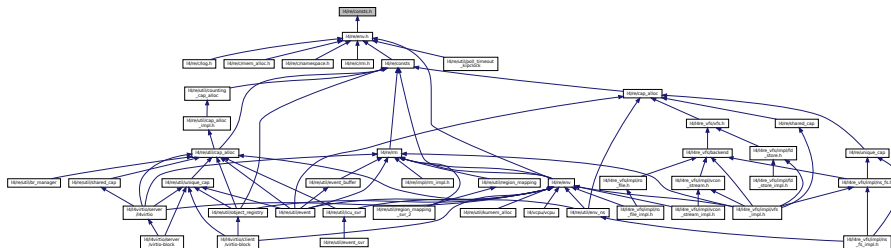
Constants.


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

Include dependency graph for consts.h:



This graph shows which files directly or indirectly include this file:



Enumerations

- enum `l4_sys_thread_prio_defaults`
Defaults for local thread priorities.

16.271.1 Detailed Description

Constants.

Definition in file [consts.h](#).

16.271.2 Enumeration Type Documentation

16.271.2.1 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](#).

16.272 consts.h

```
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/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
```

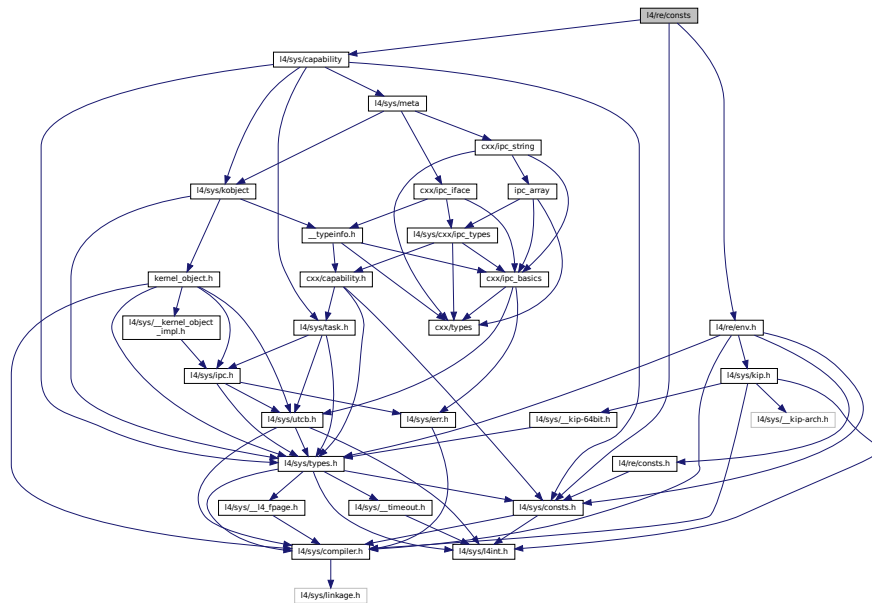
16.273 l4/re/consts File Reference

Constants.

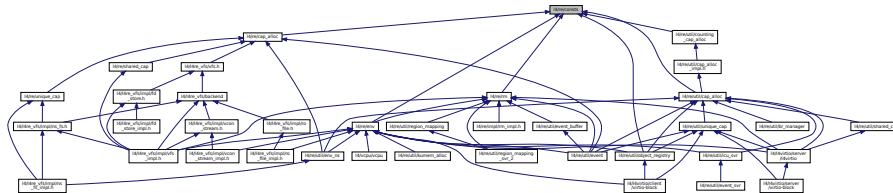
```
#include <l4/sys/capability>
#include <l4/sys/consts.h>
```

```
#include <l4/re/env.h>
```

Include dependency graph for consts:



This graph shows which files directly or indirectly include this file:



Namespaces

- [L4Re](#)
L4Re C++ Interfaces.

16.273.1 Detailed Description

Constants.

Definition in file [consts](#).

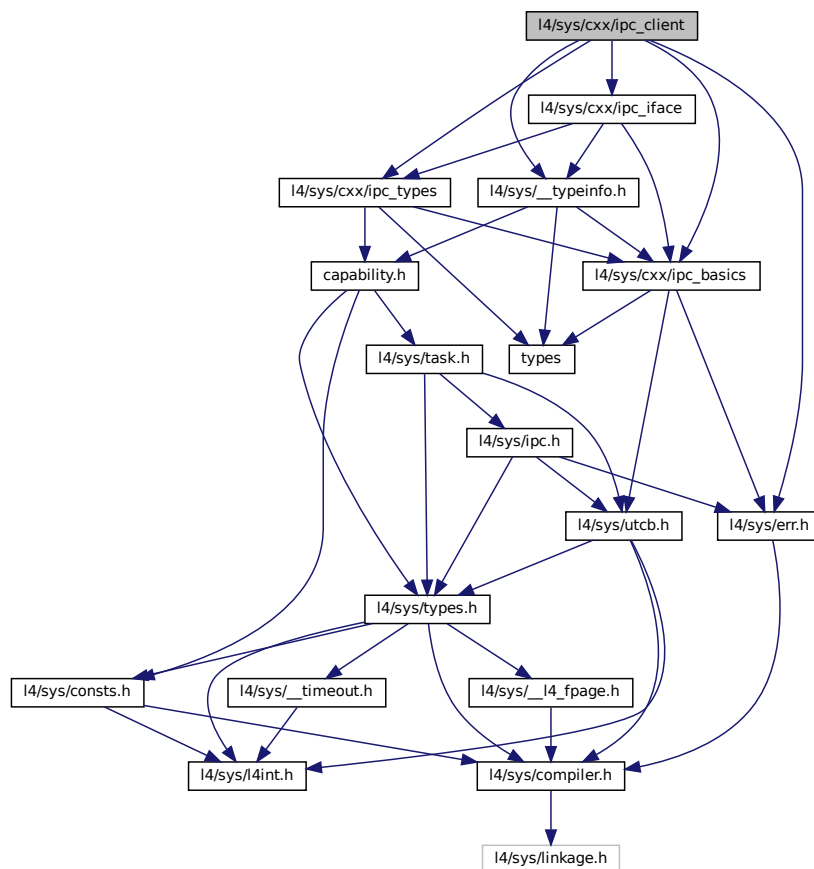
16.274 consts

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024 #pragma once
00025
00026 #include <l4/sys/capability>
00027 #include <l4/sys/consts.h>
00028 #include <l4/re/env.h>
00029
00030 namespace L4Re {
00031     static L4::Cap<L4::Task>::Cap_type const This_task
00032         = (L4::Cap<L4::Task>::Cap_type) (L4RE_THIS_TASK_CAP);
00033 }
```

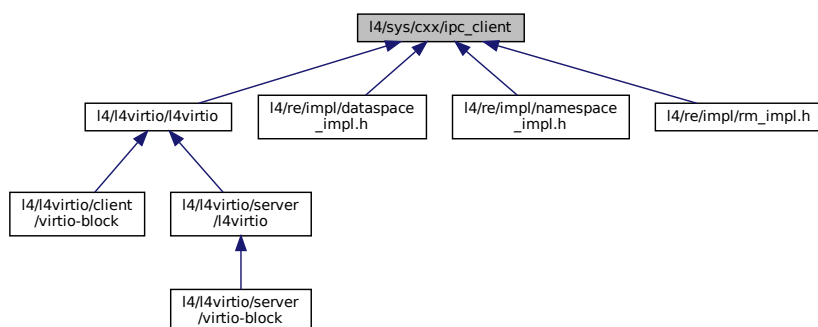
16.275 l4/sys/cxx/ipc_client File Reference

```
#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

- [L4](#)

- [L4](#) *low-level kernel interface.*
- [L4::lpc](#)
IPC related functionality.
- [L4::lpc::Msg](#)
IPC Message related functionality.

Macros

- `#define L4_RPC_DEF(name)`
Generate the definition of an RPC stub.

16.275.1 Macro Definition Documentation

16.275.1.1 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](#).

16.276 ipc_client

```
00001 // vi:set ft=c++: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU General Public License 2.
00007  * Please see the COPYING-GPL-2 file for details.
00008  *
00009  * As a special exception, you may use this file as part of a free software
00010  * library without restriction. Specifically, if other files instantiate
00011  * templates or use macros or inline functions from this file, or you compile
00012  * this file and link it with other files to produce an executable, this
00013  * file does not by itself cause the resulting executable to be covered by
00014  * the GNU General Public License. This exception does not however
00015  * invalidate any other reasons why the executable file might be covered by
00016  * the GNU General Public License.
00017  */
00018 #pragma once
00019 #pragma GCC system_header
00020
```

```

00021 #include <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 }
00059
00060 } // namespace Msg
00061 } // namespace Ipc
00062 } // namespace L4
00063

```

16.277 l4/sys/cxx/ipc_iface File Reference

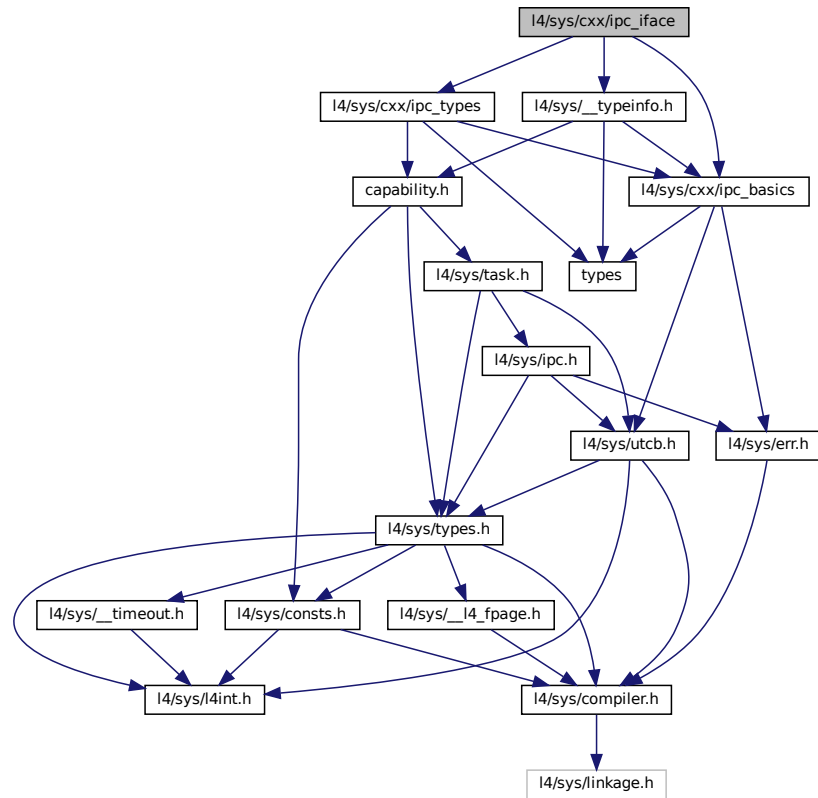
Interface Definition Language.

```

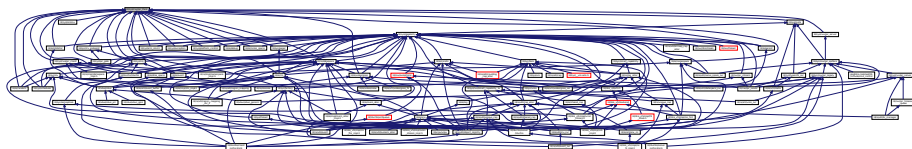
#include <l4/sys/cxx/ipc_basics>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/__typeinfo.h>

```

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

- [L4](#)
L4 low-level kernel interface.
- [L4::ipc](#)
IPC related functionality.
- [L4::ipc::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).

16.277.1 Detailed Description

Interface Definition Language.

See also

[L4_RPC](#), [L4_INLINE_RPC](#), [L4::ipc::Call](#) [L4::ipc::Send_only](#), [L4::ipc::Msg::Rpc_call](#), [L4::ipc::Msg::Rpc_↔
inline_call](#)

Definition in file [ipc_iface](#).

16.277.2 Macro Definition Documentation

16.277.2.1 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::lpc::Call , L4::lpc::Call_t etc.).

Definition at line 469 of file [ipc_iface](#).

16.277.2.2 L4_INLINE_RPC_NF

```
#define L4_INLINE_RPC_NF(
    res,
    name,
    args... )
```

Value:

```
struct name##_t : L4::Ipc::Msg::Rpc_inline_call<name##_t, Class, res args> \
{ \
    typedef L4::Ipc::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::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 440 of file [ipc_iface](#).

16.277.2.3 L4_INLINE_RPC_NF_OP

```
#define L4_INLINE_RPC_NF_OP(
    op,
    res,
    name,
    args... )
```

Value:

```
struct name##_t : L4::Ipc::Msg::Rpc_inline_call<name##_t, Class, res args> \
{ \
    typedef L4::Ipc::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](#).

16.277.2.4 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](#).

16.277.2.5 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::Ipc::Call , L4::Ipc::Call_t etc.).

Definition at line 528 of file [ipc_iface](#).

16.277.2.6 L4_RPC_NF

```
#define L4_RPC_NF(
    res,
    name,
    args... )
```

Value:

```
struct name##_t : L4::Ipc::Msg::Rpc_call<name##_t, Class, res args>
{
    typedef L4::Ipc::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](#).

16.277.2.7 L4_RPC_NF_OP

```
#define L4_RPC_NF_OP(
    op,
    res,
    name,
    args... )
```

Value:

```
struct name##_t : L4::Ipc::Msg::Rpc_call<name##_t, Class, res args>
{
    typedef L4::Ipc::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](#).

16.277.2.8 L4_RPC_OP

```
#define L4_RPC_OP(
    op,
    res,
    name,
    args,
    attr... ) res name args
```

Define an RPC call with specific opcode (type and callable).

Parameters

<i>op</i>	The opcode number for this function
<i>res</i>	The result type of the RPC call
<i>name</i>	The name of the function (<i>name_t</i> is used for the type.)
<i>args</i>	The argument list of the RPC function.
<i>attr</i>	Optional RPC attributes (L4::ipc::Call , L4::ipc::Call_t etc.).

Definition at line 543 of file [ipc_iface](#).

16.278 ipc_iface

```
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 };
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 private:
00380     L4::Cap<class_type> cap() const noexcept
00381     {
00382         return L4::Cap<class_type>(reinterpret_cast<l4_cap_idx_t>(this)
00383                                     & L4_CAP_MASK);
00384     }
00385 public:
00386     result_type operator () (ARGS ...a, l4_utcb_t *utcb = l4_utcb()) const noexcept
00387     { return IPC::call(cap(), a..., utcb); }
00388 };
00389
00390 template<typename IPC> struct Call : _Call<IPC, typename IPC::func_type> {};
00391
00392 template<typename OP,
00393         typename CLASS,
00394         typename FLAGS,
00395         typename R,
00396         typename ...ARGS>
00397 struct L4_EXPORT Rpc_call<OP, CLASS, R (ARGS...), FLAGS> :
00398     Rpc_inline_call<OP, CLASS, R (ARGS...), FLAGS>
00399 {
00400     static R call(L4::Cap<CLASS> cap,
00401                  typename _Elem<ARGS>::arg_type ...a,
00402                  l4_utcb_t *utcb = l4_utcb()) noexcept;
00403 };
00404
00405 #define L4_INLINE_RPC_SRV_FORWARD(name)
00406     template<typename OBJ> struct fwd
00407     {
00408         OBJ *o;
00409         fwd(OBJ *o) noexcept : o(o) {}
00410         template<typename ...ARGS> long call(ARGS ...a) noexcept (noexcept(o->op_##name(a...))) \
00411         { return o->op_##name(a...); }
00412     }
00413
00414 #define L4_INLINE_RPC_NF(res, name, args...)
00415     struct name##_t : L4::Ip::Msg::Rpc_inline_call<name##_t, Class, res args>
00416     {
00417         typedef L4::Ip::Msg::Rpc_inline_call<name##_t, Class, res args> type;
00418         L4_INLINE_RPC_SRV_FORWARD(name);
00419     }
00420
00421 #define L4_INLINE_RPC_NF_OP(op, res, name, args...)
00422     struct name##_t : L4::Ip::Msg::Rpc_inline_call<name##_t, Class, res args>
00423     {
00424         typedef L4::Ip::Msg::Rpc_inline_call<name##_t, Class, res args> type;
00425         enum { Opcode = (op) };
00426         L4_INLINE_RPC_SRV_FORWARD(name);
00427     }
00428
00429 #ifndef DOXYGEN
00430 #define L4_INLINE_RPC(res, name, args, attr...) res name args
00431 #else
00432 #define L4_INLINE_RPC(res, name, args...)
00433     L4_INLINE_RPC_NF(res, name, args); L4::Ip::Msg::Call<name##_t> name
00434 #endif
00435
00436 #ifndef DOXYGEN
00437 #define L4_INLINE_RPC_OP(op, res, name, args, attr...) res name args
00438 #else
00439 #define L4_INLINE_RPC_OP(op, res, name, args...)
00440     L4_INLINE_RPC_NF_OP(op, res, name, args); L4::Ip::Msg::Call<name##_t> name
00441 #endif
00442
00443 #define L4_RPC_NF(res, name, args...)
00444     struct name##_t : L4::Ip::Msg::Rpc_call<name##_t, Class, res args>
00445     {
00446         typedef L4::Ip::Msg::Rpc_call<name##_t, Class, res args> type;
00447         L4_INLINE_RPC_SRV_FORWARD(name);
00448     }
00449
00450 #define L4_RPC_NF_OP(op, res, name, args...)
00451     struct name##_t : L4::Ip::Msg::Rpc_call<name##_t, Class, res args>
00452     {
00453         typedef L4::Ip::Msg::Rpc_call<name##_t, Class, res args> type;
00454         enum { Opcode = (op) };
00455         L4_INLINE_RPC_SRV_FORWARD(name);
00456     }

```

```

00519
00520 #ifdef 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::Rpc::Msg::Call<name##_t> name
00532 #endif
00533
00534 #ifdef 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::Rpc::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

```



```

00629     int r = Data::template read<DIR>((char *)b, offset, limit, ret, m...);
00630     if (L4_LIKELY(r >= offset))
00631         return r - offset;
00632     return r;
00633 }
00634 };
00635
00642 template<typename IPC_TYPE, typename OPCODE = void>
00643 struct Part;
00644
00645 // The version without an op-code
00646 template<typename R, typename ...ARGS>
00647 struct Part<R (ARGS...), void> : _Part<ARGS...>
00648 {
00650     typedef Buf<ARGS...> Data;
00651
00652     // write arguments, skipping the dummy opcode
00653     template<typename DIR>
00654     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

```

```

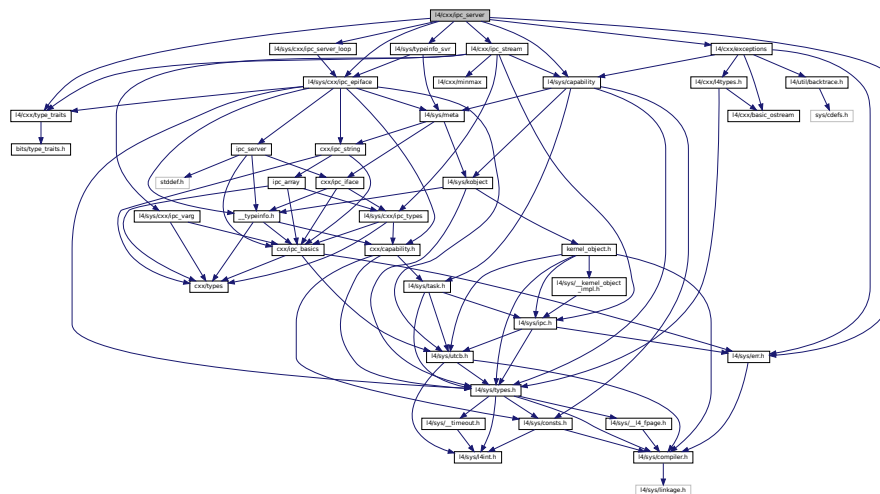
00724     if (L4_UNLIKELY(item_bytes < 0))
00725         return return_err<R>(item_bytes);
00726
00727     int send_items = item_bytes / Item_bytes;
00728
00729     {
00730         // setup the receive buffers for the RPC call
00731         l4_buf_regs_t *brs = l4_utcb_br_u(utcb);
00732         // XXX: 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
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
00760         return return_code<R>(l4_msgtag(0, 0, 0, t.flags()));
00761     }
00762
00763     // unmarshalling starts here -----
00764
00765     // bail out early in the case of an IPC error
00766     if (L4_UNLIKELY(t.has_error()))
00767         return return_ipc_err<R>(t, utcb);
00768
00769     // take the label as return value
00770     long r = t.label();
00771
00772     // bail out on negative error codes too
00773     if (L4_UNLIKELY(r < 0))
00774         return return_err<R>(r);
00775
00776     int const rcv_bytes = t.words() * Word_bytes;
00777
00778     // read the static out-data values to the arguments
00779     int err = Args::template read<Do_out_data>(mrs->mr, 0, rcv_bytes, r, a...);
00780
00781     int const item_limit = t.items() * Item_bytes;
00782
00783     if (L4_UNLIKELY(err < 0 || item_limit > Mr_bytes))
00784         return return_err<R>(-L4_MSGTOOSHORT);
00785
00786     // read the static out-items to the arguments
00787     err = Args::template read<Do_out_items>(&mrs->mr[t.words()], 0, item_limit,
00788                                             r, a...);
00789
00790     if (L4_UNLIKELY(err < 0))
00791         return return_err<R>(-L4_MSGTOOSHORT);
00792
00793     return return_code<R>(t);
00794 }
00795
00796 } // namespace Msg
00797 } // namespace Ipc
00798 } // namespace L4
00799
00800

```

16.279 I4/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>
Include dependency graph for ipc_server:
```



Data Structures

- class `L4::Server_object`
Abstract server object to be used with `L4::Server` and `L4::Basic_registry`.
- struct `L4::Server_object_t < IFACE, BASE >`
Base class (template) for server implementing server objects.
- struct `L4::Server_object_x< Derived, IFACE, BASE >`
Helper class to implement p_dispatch based server objects.
- struct `L4::Irq_handler_object`
Server object base class for handling IRQ messages.

Namespaces

- L4
L4 low-level kernel interface.

16.279.1 Detailed Description

```
IPC server loop.
```

Definition in file [ipc_server](#).

16.280 ipc_server

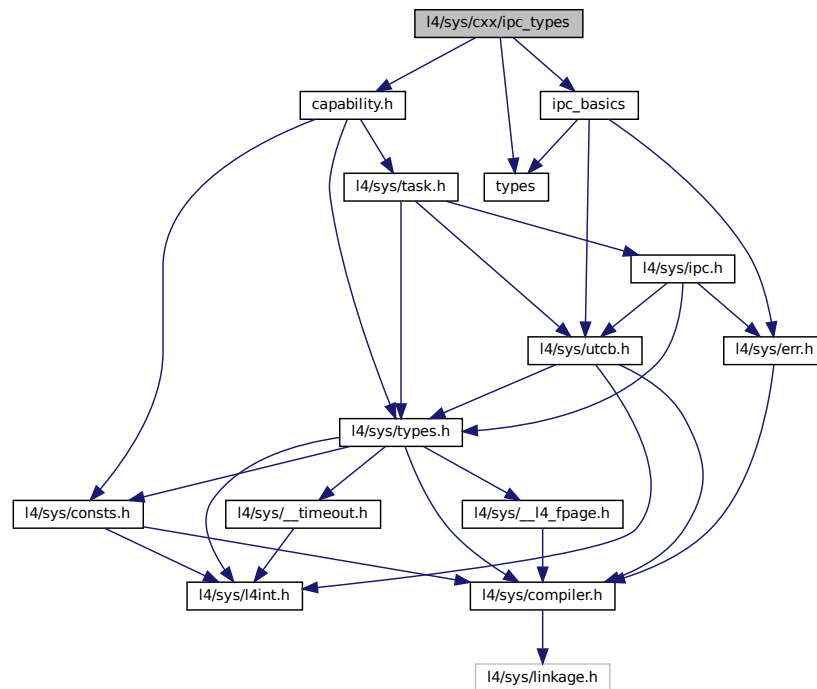
```

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)
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
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 }

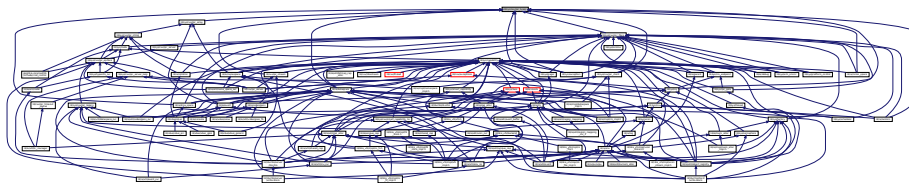
```

16.281 I4/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 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

- [L4](#)
L4 low-level kernel interface.
- [L4::lpc](#)
IPC related functionality.
- [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.

16.282 ipc_types

```

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
00060             template<typename A>
00061             struct Is_valid_rpc_type< In_out<A const*> > : L4::Types::False {};
00062
00063             #ifndef CONFIG_ALLOW_REFS
00064             template<typename A> struct Elem< In_out<A&> > : Elem<A&> {};
00065
00066             template<typename A>
00067             struct Svr_xmit< In_out<A&> > : Svr_xmit<A&>, Svr_xmit<A const&>
00068             {
00069                 using Svr_xmit<A&>::from_svr;
00070                 using Svr_xmit<A const&>::to_svr;
00071             };
00072
00073             template<typename A>
00074             struct Clnt_xmit< In_out<A&> > : Clnt_xmit<A&>, Clnt_xmit<A const&>
00075             {
00076                 using Clnt_xmit<A&>::from_msg;
00077                 using Clnt_xmit<A const&>::to_msg;
00078             };
00079
00080             template<typename A>
00081             struct Is_valid_rpc_type< In_out<A&> > : Is_valid_rpc_type<A&> {};
00082
00083             template<typename A>
00084             struct Is_valid_rpc_type< In_out<A const&> > : L4::Types::False {};
00085
00086             #else
00087
00088         }
00089     }
00090 }

```

```

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 {
00297 public:
00298     enum Type
00299     {

```

```

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; }
00438     bool cap_received() const noexcept { return (T::_base & 0x3e) == 0x38; }
00450     bool id_received() const noexcept { return (T::_base & 0x3e) == 0x3c; }
00460     bool local_id_received() const noexcept { return (T::_base & 0x3e) == 0x3e; }
00461
00468     bool is_compound() const noexcept { return T::_base & 1; }
00470     l4_umword_t data() const noexcept { return T::_data; }
00472     l4_umword_t base_x() const noexcept { return T::_base; }
00473 };
00474
00475
00477 typedef Gen_fpage<Snd_item> Snd_fpage;
00479 typedef Gen_fpage<Buf_item> Rcv_fpage;
00480
00481 #ifdef L4_CXX_IPC_SUPPORT_STRINGS
00482 template <typename T, typename B>
00483 class Gen_string : public T
00484 {
00485 public:
00486     Gen_string() noexcept : T(0, 0) {}
00487     Gen_string(B buf, unsigned long size) noexcept
00488         : T(size « 10, l4_umword_t(buf))
00489     {}
00490
00491     unsigned long len() const noexcept { return T::_base » 10; }
00492 };
00493
00494 typedef Gen_string<Snd_item, void const *> Snd_string;
00495 typedef Gen_string<Buf_item, void *> Rcv_string;
00496 #endif
00497
00498
00499 namespace Msg {
00500
00501     // Snd_fpage are out items
00502     template<> struct Class<L4::Ipc::Snd_fpage> : Cls_item {};
00503
00504     // Rcv_fpage are buffer items
00505     template<> struct Class<L4::Ipc::Rcv_fpage> : Cls_buffer {};
00506
00507     // Remove receive buffers from server-side arguments
00508     template<> struct Elem<L4::Ipc::Rcv_fpage>
00509     {
00510         typedef L4::Ipc::Rcv_fpage arg_type;
00511         typedef void svr_type;
00512         typedef void svr_arg_type;
00513         enum { Is_optional = false };
00514     };
00515
00516     // Rcv_fpage are buffer items
00517     template<> struct Class<L4::Ipc::Small_buf> : Cls_buffer {};
00518
00519     // Remove receive buffers from server-side arguments
00520     template<> struct Elem<L4::Ipc::Small_buf>
00521     {
00522         typedef L4::Ipc::Small_buf arg_type;
00523         typedef void svr_type;
00524         typedef void svr_arg_type;
00525         enum { Is_optional = false };
00526     };
00527 } // namespace Msg
00528
00529 // L4::Cap<> handling
00530
00541 template<typename T> class Cap
00542 {
00543     template<typename O> friend class Cap;
00544     l4_umword_t _cap_n_rights;
00545
00546 public:
00547     enum
00548     {
00554         Rights_mask = 0xff,

```

```

00555
00560     Cap_mask      = L4_CAP_MASK
00561 };
00562
00563 template<typename O>
00564 Cap(Cap<O> const &o) noexcept : _cap_n_rights(o._cap_n_rights)
00565 { T *x = (O*)1; (void)x; }
00566
00567 Cap(L4::Cap<T> cap) noexcept
00570 : _cap_n_rights((cap.cap() & Cap_mask) | (cap ? L4_CAP_FPAGE_R : 0))
00571 {}
00572
00573 template<typename O>
00574 Cap(L4::Cap<O> cap) noexcept
00575 : _cap_n_rights((cap.cap() & Cap_mask) | (cap ? L4_CAP_FPAGE_R : 0))
00576 { T *x = (O*)1; (void)x; }
00577
00578 Cap() noexcept : _cap_n_rights(L4_INVALID_CAP) {}
00579
00580 Cap(L4::Cap<T> cap, unsigned char rights) noexcept
00581 : _cap_n_rights((cap.cap() & Cap_mask) | (rights & Rights_mask)) {}
00582
00583 static Cap from_ci(l4_cap_idx_t c) noexcept
00584 { return Cap(L4::Cap<T>(c & Cap_mask), c & Rights_mask); }
00585
00586 L4::Cap<T> cap() const noexcept
00587 { return L4::Cap<T>(_cap_n_rights & Cap_mask); }
00588
00589 unsigned rights() const noexcept
00590 { return _cap_n_rights & Rights_mask; }
00591
00592 L4::Ipc::Snd_fpage fpage() const noexcept
00593 { return L4::Ipc::Snd_fpage(cap(), rights()); }
00594
00595 bool is_valid() const noexcept
00596 { return !(_cap_n_rights & L4_INVALID_CAP_BIT); }
00597 };
00598
00599 template<typename T>
00600 Cap<T> make_cap(L4::Cap<T> cap, unsigned rights) noexcept
00601 { return Cap<T>(cap, rights); }
00602
00603 template<typename T>
00604 Cap<T> make_cap_rw(L4::Cap<T> cap) noexcept
00605 { return Cap<T>(cap, L4_CAP_FPAGE_RW); }
00606
00607 template<typename T>
00608 Cap<T> make_cap_rws(L4::Cap<T> cap) noexcept
00609 { return Cap<T>(cap, L4_CAP_FPAGE_RWS); }
00610
00611 template<typename T>
00612 Cap<T> make_cap_full(L4::Cap<T> cap) noexcept
00613 { return Cap<T>(cap, L4_CAP_FPAGE_RWSD | L4_FPAGE_C_OBJ_RIGHTS); }
00614
00615 // caps are special the have an invalid representation
00616 template<typename T> struct L4_EXPORT Opt< Cap<T> >
00617 {
00618     Cap<T> _value;
00619     Opt() noexcept {}
00620     Opt(Cap<T> value) noexcept : _value(value) {}
00621     Opt(L4::Cap<T> value) noexcept : _value(value) {}
00622     Opt &operator = (Cap<T> value) noexcept
00623     { this->_value = value; }
00624     Opt &operator = (L4::Cap<T> value) noexcept
00625     { this->_value = value; }
00626
00627     Cap<T> value() const noexcept { return this->_value; }
00628     bool is_valid() const noexcept { return this->_value.is_valid(); }
00629 };
00630
00631 namespace Msg {
00632 // prohibit L4::Cap as argument
00633 template<typename A>
00634 struct Is_valid_rpc_type< L4::Cap<A> > : L4::Types::False {};
00635
00636 template<typename A> struct Class< Cap<A> > : Cls_item {};
00637 template<typename A> struct Elem< Cap<A> >
00638 {
00639     enum { Is_optional = false };
00640     typedef Cap<A> arg_type;
00641     typedef L4::Ipc::Snd_fpage svr_type;
00642     typedef L4::Ipc::Snd_fpage svr_arg_type;
00643 };
00644
00645 template<typename A, typename CLASS>

```

```

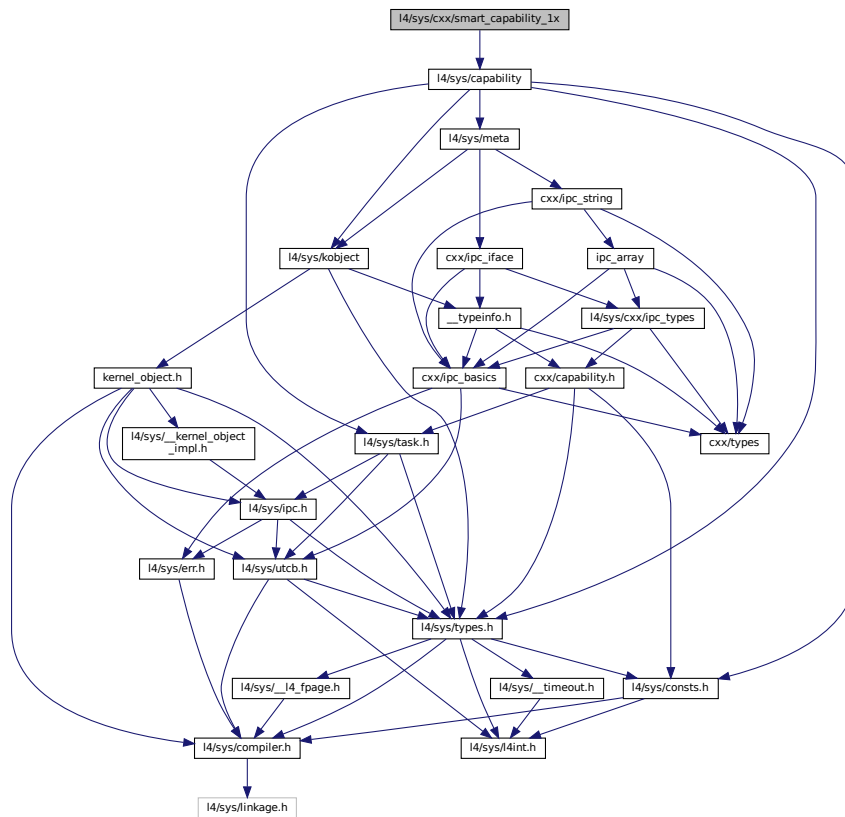
00698 struct Svr_val_ops<Cap<A>, Dir_in, CLASS> :
00699     Svr_val_ops<L4::Ipc::Snd_fpage, Dir_in, CLASS>
00700 {};
00701
00702 template<typename A, typename CLASS>
00703 struct Clnt_val_ops<Cap<A>, Dir_in, CLASS> :
00704     Clnt_noops< Cap<A> >
00705 {
00706     using Clnt_noops< Cap<A> >::to_msg;
00707
00708     static int to_msg(char *msg, unsigned offset, unsigned limit,
00709                     Cap<A> arg, Dir_in, Cls_item) noexcept
00710     {
00711         // passing an invalid cap as mandatory argument is an error
00712         // XXX: This checks for a client calling error, we could
00713         //      also just ignore this for performance reasons and
00714         //      let the client fail badly (Alex: I'd prefer this)
00715         if (L4_UNLIKELY(!arg.is_valid()))
00716             return -L4_MSGMISSARG;
00717
00718         return msg_add(msg, offset, limit, arg.fpage());
00719     }
00720 };
00721
00722 template<typename A>
00723 struct Elem<Out<L4::Cap<A> > > >
00724 {
00725     enum { Is_optional = false };
00726     typedef L4::Cap<A> arg_type;
00727     typedef Ipc::Cap<A> svr_type;
00728     typedef svr_type &svr_arg_type;
00729 };
00730
00731 template<typename A> struct Direction< Out< L4::Cap<A> > > : Dir_out {};
00732 template<typename A> struct Class< Out< L4::Cap<A> > > : Cls_item {};
00733
00734 template<typename A>
00735 struct Clnt_val_ops< L4::Cap<A>, Dir_out, Cls_item > :
00736     Clnt_noops< L4::Cap<A> >
00737 {
00738     using Clnt_noops< L4::Cap<A> >::to_msg;
00739     static int to_msg(char *msg, unsigned offset, unsigned limit,
00740                     L4::Cap<A> arg, Dir_in, Cls_buffer) noexcept
00741     {
00742         if (L4_UNLIKELY(!arg.is_valid()))
00743             return -L4_MSGMISSARG; // no buffer inserted
00744         return msg_add(msg, offset, limit, Small_buf(arg));
00745     }
00746 };
00747
00748 template<typename A>
00749 struct Svr_val_ops< L4::Ipc::Cap<A>, Dir_out, Cls_item > :
00750     Svr_noops<Cap<A> &>
00751 {
00752     using Svr_noops<Cap<A> &>::from_svr;
00753     static int from_svr(char *msg, unsigned offset, unsigned limit, long,
00754                       Cap<A> arg, Dir_out, Cls_item) noexcept
00755     {
00756         if (L4_UNLIKELY(!arg.is_valid()))
00757             // do not map anything
00758             return msg_add(msg, offset, limit, L4::Ipc::Snd_fpage(arg.cap(), 0));
00759
00760         return msg_add(msg, offset, limit, arg.fpage());
00761     }
00762 };
00763
00764 // prohibit a UTcb pointer as normal RPC argument
00765 template<> struct Is_valid_rpc_type<l4_utcb_t *> : L4::Types::False {};
00766
00767 } // namespace Msg
00768 } // namespace Ipc
00769 } // namespace L4
00770

```

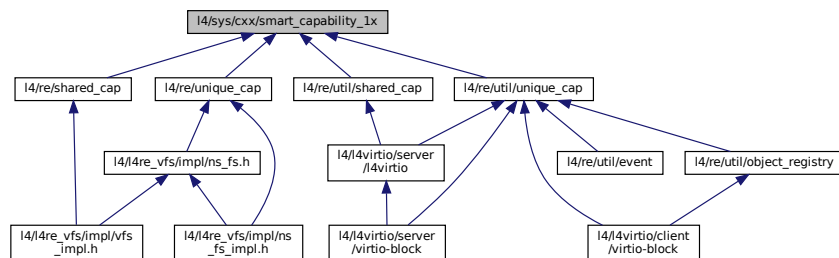
16.283 l4/sys/cxx/smart_capability_1x File Reference

```
#include <l4/sys/capability>
```

Include dependency graph for smart_capability_1x:



This graph shows which files directly or indirectly include this file:



Namespaces

- L4
L4 low-level kernel interface.

16.284 smart_capability_1x

```
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     }

```



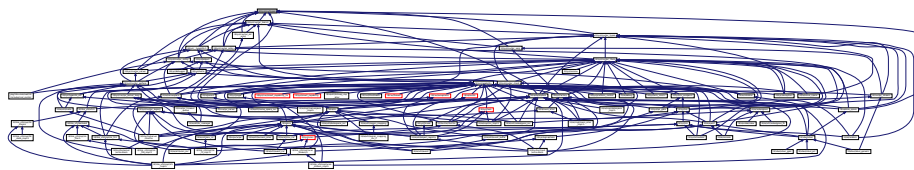
```

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

```

16.285 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

- [L4](#)
[L4](#) low-level kernel interface.
- [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.

16.285.1 Macro Definition Documentation

16.285.1.1 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](#).

16.286 types

```
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
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:
00066         typedef UNDERLYING value_type;
00067         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 };
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 {
00234     friend constexpr DT operator | (DT l, DT r)
00235     { return DT(l.raw | r.raw); }
00236
00237     friend constexpr DT operator & (DT l, DT r)
00238     { return DT(l.raw & r.raw); }
00239
00240     friend constexpr bool operator == (DT l, DT r)
00241     { return l.raw == r.raw; }
00242
00243     friend constexpr bool operator != (DT l, DT r)
00244     { return l.raw != r.raw; }
00245
00246     DT operator |= (DT r)
00247     {
00248         static_cast<DT *>(this)->raw |= r.raw;
00249         return *static_cast<DT *>(this);
00250     }
00251
00252     DT operator &= (DT r)
00253     {
00254         static_cast<DT *>(this)->raw &= r.raw;
00255         return *static_cast<DT *>(this);
00256     }
00257
00258     explicit constexpr operator bool () const
00259     {
00260         return static_cast<DT const *>(this)->raw != 0;
00261     }
00262
00263     constexpr DT operator ~ () const
00264     { return DT(~static_cast<DT const *>(this)->raw); }
00265 };
00266
00267 template<typename DT, typename T>
00268 struct Flags_t : Flags_ops_t<Flags_t<DT, T>
00269 {
00270     T raw;
00271     Flags_t() = default;
00272     explicit constexpr Flags_t(T f) : raw(f) {}
00273 };
00274
00275 template<bool V > struct Bool
00276 {
00277     typedef Bool<V> type;
00278     enum { value = V };
00279 };
00280
00281 struct False : Bool<false> {};
00282
00283 struct True : Bool<true> {};
00284
00285 /*****/
00286 template<typename A, typename B>
00287 struct Same : False {};
00288
00289 template<typename A>
00290 struct Same<A, A> : True {};
00291
00292 template<bool EXP, typename T = void> struct Enable_if {};
00293 template<typename T> struct Enable_if<true, T> { typedef T type; };
00294
00295 template<typename T1, typename T2, typename T = void>
00296 struct Enable_if_same : Enable_if<Same<T1, T2>::value, T> {};
00297
00298 template<typename T> struct Remove_const { typedef T type; };
00299 template<typename T> struct Remove_const<T const> { typedef T type; };
00300 template<typename T> struct Remove_volatile { typedef T type; };
00301 template<typename T> struct Remove_volatile<T volatile> { typedef T type; };
00302 template<typename T> struct Remove_cv
00303 { typedef typename Remove_const<typename Remove_volatile<T>::type>::type type; };
00304
00305 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

```

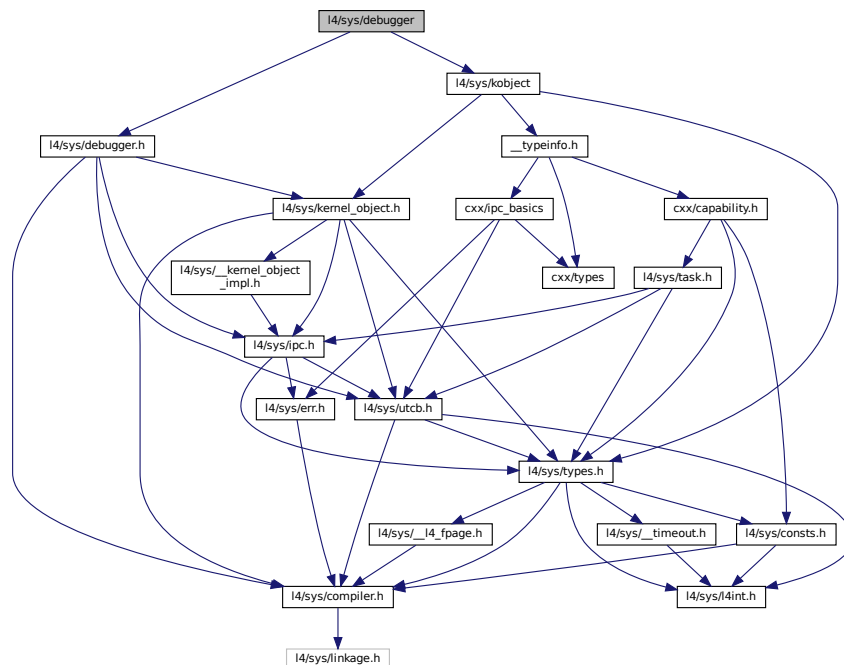
16.287 l4/sys/debugger File Reference

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

- [L4](#)
L4 low-level kernel interface.

16.287.1 Detailed Description

The debugger interface specifies common debugging related definitions.

Definition in file [debugger](#).

16.288 debugger

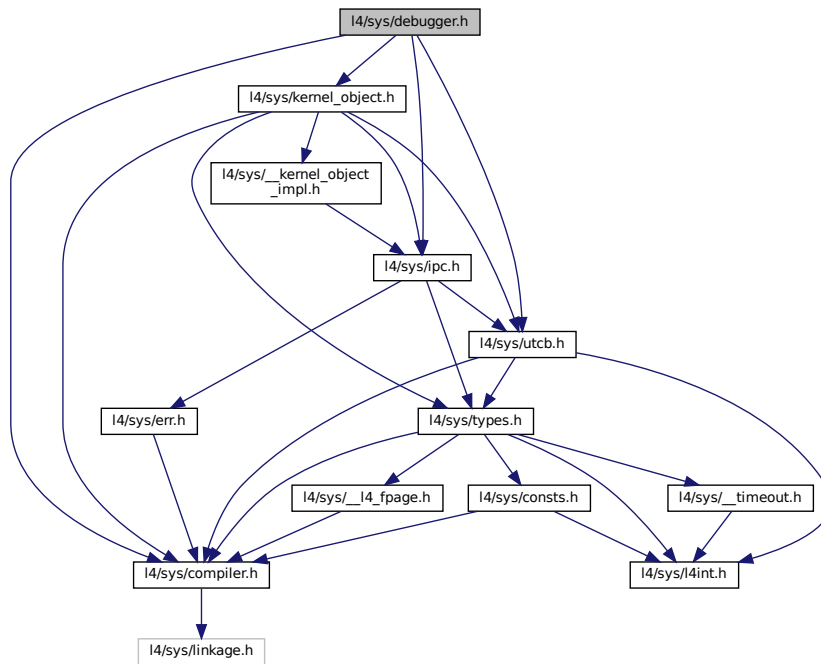
```

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
00031 class Debugger : public Kobject_t<Debugger, Kobject, L4_PROTO_DEBUGGER>
00032 {
00033 public:
00034     enum
00035     {
00036         Switch_log_on = L4_DEBUGGER_SWITCH_LOG_ON,
00037         Switch_log_off = L4_DEBUGGER_SWITCH_LOG_OFF,
00038     };
00039
00040     l4_msgtag_t set_object_name(const char *name,
00041                                l4_utcb_t *utcb = l4_utcb()) noexcept
00042     { return l4_debugger_set_object_name_u(cap(), name, utcb); }
00043
00044     unsigned long global_id(l4_utcb_t *utcb = l4_utcb()) noexcept
00045     { return l4_debugger_global_id_u(cap(), utcb); }
00046
00047     unsigned long kobj_to_id(l4_addr_t kobjp,
00048                             l4_utcb_t *utcb = l4_utcb()) noexcept
00049     { return l4_debugger_kobj_to_id_u(cap(), kobjp, utcb); }
00050
00051     long query_log_typeid(const char *name, unsigned idx,
00052                          l4_utcb_t *utcb = l4_utcb()) noexcept
00053     { return l4_debugger_query_log_typeid_u(cap(), name, idx, utcb); }
00054
00055     long query_log_name(unsigned idx,
00056                        char *name, unsigned namelen,
00057                        char *shortname, unsigned shortnamelen,
00058                        l4_utcb_t *utcb = l4_utcb()) noexcept
00059     { return l4_debugger_query_log_name_u(cap(), idx, name, namelen,
00060                                           shortname, shortnamelen, utcb); }
00061
00062     l4_msgtag_t switch_log(const char *name, unsigned on_off,
00063                           l4_utcb_t *utcb = l4_utcb()) noexcept
00064     { return l4_debugger_switch_log_u(cap(), name, on_off, utcb); }
00065
00066     l4_msgtag_t get_object_name(unsigned id, char *name, unsigned size,
00067                                 l4_utcb_t *utcb = l4_utcb()) noexcept
00068     { return l4_debugger_get_object_name_u(cap(), id, name, size, utcb); }
00069 };
00070
00071 }
```

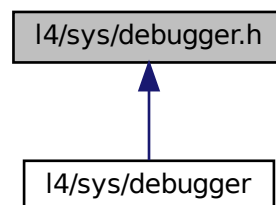
16.289 l4/sys/debugger.h File Reference

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.

- unsigned [__strcpy_maxlen](#) (char *dst, char const *src, unsigned maxlen)

*Copy a number of characters from the C string **src** to the C string **dst**.*

16.289.1 Detailed Description

Debugger related definitions.

Definition in file [debugger.h](#).

16.289.2 Function Documentation

16.289.2.1 [__strcpy_maxlen\(\)](#)

```
unsigned __strcpy_maxlen (
    char * dst,
    char const * src,
    unsigned maxlen ) [inline]
```

Copy a number of characters from the C string **src** to the C string **dst**.

The resulting string **dst** is always '\0'-terminated unless **maxlen** is 0. If the C string in **src** is shorter than the buffer **dst** then the remaining bytes in **dst** are NOT initialized.

Parameters

<i>dst</i>	Target buffer.
<i>src</i>	Source buffer.
<i>maxlen</i>	Maximum number of bytes written to the target buffer.

Returns

The number of bytes written (including the terminating '\0').

Definition at line 231 of file [debugger.h](#).

16.289.2.2 l4_debugger_get_object_name()

```
l4_msgtag_t l4_debugger_get_object_name (
    l4_cap_idx_t cap,
    unsigned id,
    char * name,
    unsigned size ) [inline]
```

Get name of the kernel object with Id *id*.

Parameters

	<i>cap</i>	Capability of the debugger object.
	<i>id</i>	Global id of the object whose name is asked.
out	<i>name</i>	Buffer to copy the name into. The buffer must be allocated by the caller.
	<i>size</i>	Length of the <i>name</i> buffer.

Returns

Syscall return tag

Definition at line 381 of file [debugger.h](#).

16.289.2.3 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 365 of file [debugger.h](#).

16.289.2.4 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 358 of file [debugger.h](#).

16.289.2.5 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 374 of file [debugger.h](#).

16.290 debugger.h

```

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
00073 L4_INLINE l4_msgtag_t
00074 l4_debugger_get_object_name(l4_cap_idx_t cap, unsigned id,
00075                             char *name, unsigned size) L4_NOTHROW;
00076
00080 L4_INLINE l4_msgtag_t
00081 l4_debugger_get_object_name_u(l4_cap_idx_t cap, unsigned id,
00082                               char *name, unsigned size,
00083                               l4_utcb_t *utcb) L4_NOTHROW;
00084
00096 L4_INLINE unsigned long
00097 l4_debugger_global_id(l4_cap_idx_t cap) L4_NOTHROW;
00098
00102 L4_INLINE unsigned long
00103 l4_debugger_global_id_u(l4_cap_idx_t cap, l4_utcb_t *utcb) L4_NOTHROW;
00104
00117 L4_INLINE unsigned long
00118 l4_debugger_kobj_to_id(l4_cap_idx_t cap, l4_addr_t kobjp) L4_NOTHROW;
00119
00123 L4_INLINE unsigned long
00124 l4_debugger_kobj_to_id_u(l4_cap_idx_t cap, l4_addr_t kobjp, l4_utcb_t *utcb) L4_NOTHROW;
00125
00137 L4_INLINE long
00138 l4_debugger_query_log_typeid(l4_cap_idx_t cap, const char *name,
00139                              unsigned idx) L4_NOTHROW;
00140
00144 L4_INLINE long
00145 l4_debugger_query_log_typeid_u(l4_cap_idx_t cap, const char *name,
00146                                unsigned idx, l4_utcb_t *utcb) L4_NOTHROW;
00147
00163 L4_INLINE long
00164 l4_debugger_query_log_name(l4_cap_idx_t cap, unsigned idx,
00165                             char *name, unsigned namelen,
00166                             char *shortname, unsigned shortnamelen) L4_NOTHROW;
00167
00171 L4_INLINE long
00172 l4_debugger_query_log_name_u(l4_cap_idx_t cap, unsigned idx,
00173                               char *name, unsigned namelen,
00174                               char *shortname, unsigned shortnamelen,
00175                               l4_utcb_t *utcb) L4_NOTHROW;
00176
00186 L4_INLINE l4_msgtag_t
00187 l4_debugger_switch_log(l4_cap_idx_t cap, const char *name,
00188                        int on_off) L4_NOTHROW;

```

```

00189
00193 L4_INLINE l4_msgtag_t
00194 l4_debugger_switch_log_u(l4_cap_idx_t cap, const char *name, int on_off,
00195                          l4_utcb_t *utcb) L4_NOTHROW;
00196
00197 enum
00198 {
00199     L4_DEBUGGER_NAME_SET_OP      = 0UL,
00200     L4_DEBUGGER_GLOBAL_ID_OP    = 1UL,
00201     L4_DEBUGGER_KOBJ_TO_ID_OP   = 2UL,
00202     L4_DEBUGGER_QUERY_LOG_TYPEID_OP = 3UL,
00203     L4_DEBUGGER_SWITCH_LOG_OP   = 4UL,
00204     L4_DEBUGGER_NAME_GET_OP     = 5UL,
00205     L4_DEBUGGER_QUERY_LOG_NAME_OP = 6UL,
00206 };
00207
00208 enum
00209 {
00210     L4_DEBUGGER_SWITCH_LOG_ON = 1,
00211     L4_DEBUGGER_SWITCH_LOG_OFF = 0,
00212 };
00213
00214 /* IMPLEMENTATION -----*/
00215
00216 #include <l4/sys/kernel_object.h>
00217
00230 L4_INLINE unsigned
00231 __strcpy_maxlen(char *dst, char const *src, unsigned maxlen)
00232 {
00233     unsigned i;
00234     if (!maxlen)
00235         return 0;
00236
00237     for (i = 0; i < maxlen - 1 && src[i]; ++i)
00238         dst[i] = src[i];
00239     dst[i] = '\0';
00240
00241     return i + 1;
00242 }
00243
00244 L4_INLINE l4_msgtag_t
00245 l4_debugger_set_object_name_u(l4_cap_idx_t cap,
00246                              const char *name, l4_utcb_t *utcb) L4_NOTHROW
00247 {
00248     unsigned i;
00249     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_NAME_SET_OP;
00250     i = __strcpy_maxlen((char *)&l4_utcb_mr_u(utcb)->mr[1], name,
00251                       (L4_UTCB_GENERIC_DATA_SIZE - 2) * sizeof(l4_umword_t));
00252     i = l4_bytes_to_mwords(i);
00253     return l4_invoke_debugger(cap, l4_msgtag(0, 1 + i, 0, 0), utcb);
00254 }
00255
00256 L4_INLINE unsigned long
00257 l4_debugger_global_id_u(l4_cap_idx_t cap, l4_utcb_t *utcb) L4_NOTHROW
00258 {
00259     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_GLOBAL_ID_OP;
00260     if (l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 1, 0, 0), utcb), utcb))
00261         return ~0UL;
00262     return l4_utcb_mr_u(utcb)->mr[0];
00263 }
00264
00265 L4_INLINE unsigned long
00266 l4_debugger_kobj_to_id_u(l4_cap_idx_t cap, l4_addr_t kobjp, l4_utcb_t *utcb) L4_NOTHROW
00267 {
00268     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_KOBJ_TO_ID_OP;
00269     l4_utcb_mr_u(utcb)->mr[1] = kobjp;
00270     if (l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 2, 0, 0), utcb), utcb))
00271         return ~0UL;
00272     return l4_utcb_mr_u(utcb)->mr[0];
00273 }
00274
00275 L4_INLINE long
00276 l4_debugger_query_log_typeid_u(l4_cap_idx_t cap, const char *name,
00277                               unsigned idx,
00278                               l4_utcb_t *utcb) L4_NOTHROW
00279 {
00280     unsigned i;
00281     long e;
00282     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_QUERY_LOG_TYPEID_OP;
00283     l4_utcb_mr_u(utcb)->mr[1] = idx;
00284     i = __strcpy_maxlen((char *)&l4_utcb_mr_u(utcb)->mr[2], name, 32);
00285     i = l4_bytes_to_mwords(i);
00286     e = l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 2 + i, 0, 0), utcb), utcb);
00287     if (e < 0)
00288         return e;
00289     return l4_utcb_mr_u(utcb)->mr[0];
00290 }

```

```

00291
00292 L4_INLINE long
00293 l4_debugger_query_log_name_u(l4_cap_idx_t cap, unsigned idx,
00294                             char *name, unsigned namelen,
00295                             char *shortname, unsigned shortnamelen,
00296                             l4_utcb_t *utcb) L4_NOTHROW
00297 {
00298     long e;
00299     char const *n;
00300     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_QUERY_LOG_NAME_OP;
00301     l4_utcb_mr_u(utcb)->mr[1] = idx;
00302     e = l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 2, 0, 0), utcb), utcb);
00303     if (e < 0)
00304         return e;
00305     n = (char const *)&l4_utcb_mr_u(utcb)->mr[0];
00306     __strcpy_maxlen(name, n, namelen);
00307     __strcpy_maxlen(shortname, n + __builtin_strlen(n) + 1, shortnamelen);
00308     return 0;
00309 }
00310
00311
00312 L4_INLINE l4_msgtag_t
00313 l4_debugger_switch_log_u(l4_cap_idx_t cap, const char *name, int on_off,
00314                         l4_utcb_t *utcb) L4_NOTHROW
00315 {
00316     unsigned i;
00317     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_SWITCH_LOG_OP;
00318     l4_utcb_mr_u(utcb)->mr[1] = on_off;
00319     i = __strcpy_maxlen((char *)&l4_utcb_mr_u(utcb)->mr[2], name, 32);
00320     i = l4_bytes_to_mwords(i);
00321     return l4_invoke_debugger(cap, l4_msgtag(0, 2 + i, 0, 0), utcb);
00322 }
00323
00324 L4_INLINE l4_msgtag_t
00325 l4_debugger_get_object_name_u(l4_cap_idx_t cap, unsigned id,
00326                              char *name, unsigned size,
00327                              l4_utcb_t *utcb) L4_NOTHROW
00328 {
00329     l4_msgtag_t t;
00330     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_NAME_GET_OP;
00331     l4_utcb_mr_u(utcb)->mr[1] = id;
00332     t = l4_invoke_debugger(cap, l4_msgtag(0, 2, 0, 0), utcb);
00333     __strcpy_maxlen(name, (char const *)&l4_utcb_mr_u(utcb)->mr[0], size);
00334     return t;
00335 }
00336
00337
00338 L4_INLINE l4_msgtag_t
00339 l4_debugger_set_object_name(l4_cap_idx_t cap,
00340                             const char *name) L4_NOTHROW
00341 {
00342     return l4_debugger_set_object_name_u(cap, name, l4_utcb());
00343 }
00344
00345 L4_INLINE unsigned long
00346 l4_debugger_global_id(l4_cap_idx_t cap) L4_NOTHROW
00347 {
00348     return l4_debugger_global_id_u(cap, l4_utcb());
00349 }
00350
00351 L4_INLINE unsigned long
00352 l4_debugger_kobj_to_id(l4_cap_idx_t cap, l4_addr_t kobjp) L4_NOTHROW
00353 {
00354     return l4_debugger_kobj_to_id_u(cap, kobjp, l4_utcb());
00355 }
00356
00357 L4_INLINE long
00358 l4_debugger_query_log_typeid(l4_cap_idx_t cap, const char *name,
00359                             unsigned idx) L4_NOTHROW
00360 {
00361     return l4_debugger_query_log_typeid_u(cap, name, idx, l4_utcb());
00362 }
00363
00364 L4_INLINE long
00365 l4_debugger_query_log_name(l4_cap_idx_t cap, unsigned idx,
00366                             char *name, unsigned namelen,
00367                             char *shortname, unsigned shortnamelen) L4_NOTHROW
00368 {
00369     return l4_debugger_query_log_name_u(cap, idx, name, namelen,
00370                                         shortname, shortnamelen, l4_utcb());
00371 }
00372
00373 L4_INLINE l4_msgtag_t
00374 l4_debugger_switch_log(l4_cap_idx_t cap, const char *name,
00375                         int on_off) L4_NOTHROW
00376 {
00377     return l4_debugger_switch_log_u(cap, name, on_off, l4_utcb());

```

```

00378 }
00379
00380 L4_INLINE l4_msgtag_t
00381 l4_debugger_get_object_name(l4_cap_idx_t cap, unsigned id,
00382                             char *name, unsigned size) L4_NOTHROW
00383 {
00384     return l4_debugger_get_object_name_u(cap, id, name, size, l4_utcb());
00385 }

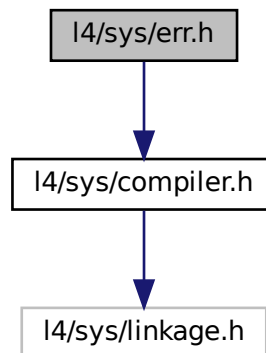
```

16.291 l4/sys/err.h File Reference

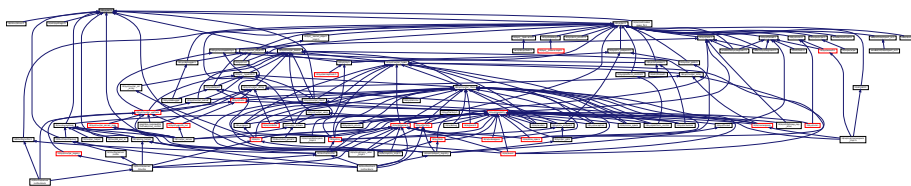
Error codes.

```
#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.*

16.291.1 Detailed Description

Error codes.

Definition in file [err.h](#).

16.292 err.h

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
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_MSGTOOSHORT       = 1001,
00067     L4_MSGTOOLONG        = 1002,
00068     L4_MSGMISSARG        = 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

```

16.293 l4/sys/exception File Reference

Exception C++ interface.

```

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

```


16.294 exception

```

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 }
```

16.295 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>
```

```

graph TD
    l4_sys_factory[l4/sys/factory] --> l4_sys_capability[l4/sys/capability]
    l4_sys_factory --> l4_sys_meta[l4/sys/meta]
    l4_sys_factory --> cxx_pc_string[cxx/pc_string]
    l4_sys_factory --> l4_sys_cxx_pc_varg[l4/sys/cxx/pc_varg]
    l4_sys_factory --> l4_sys_err_h[l4/sys/err.h]
    l4_sys_factory --> l4_sys_compiler_h[l4/sys/compiler.h]
    l4_sys_factory --> l4_sys_4int_h[l4/sys/4int.h]
    l4_sys_factory --> l4_sys_types_h[l4/sys/types.h]
    l4_sys_factory --> l4_sys_ipc_h[l4/sys/ipc.h]
    l4_sys_factory --> l4_sys_kernel_object_impl_h[l4/sys/_kernel_object_impl.h]
    l4_sys_factory --> l4_sys_task_h[l4/sys/task.h]
    l4_sys_factory --> l4_sys_fpage_h[l4/sys/_l4_fpage.h]
    l4_sys_factory --> l4_sys_consts_h[l4/sys/consts.h]
    l4_sys_factory --> l4_sys_timeout_h[l4/sys/_timeout.h]
    l4_sys_factory --> kernel_object_h[kernel/object.h]
    l4_sys_factory --> cxx_capability_h[cxx/capability.h]
    l4_sys_factory --> cxx_pc_basics[cxx/pc_basics]
    l4_sys_factory --> cxx_types[cxx/types]
    l4_sys_factory --> ipc_array[ipc_array]
    l4_sys_factory --> l4_sys_cxx_pc_types[l4/sys/cxx/pc_types]
    l4_sys_factory --> _TypeInfo_h[_TypeInfo.h]
    l4_sys_factory --> cxx_pc_iface[cxx/pc_iface]
    l4_sys_capability --> l4_sys_object[l4/sys/object]
    l4_sys_capability --> l4_sys_types_h
    l4_sys_capability --> l4_sys_compiler_h
    l4_sys_capability --> l4_sys_4int_h
    l4_sys_capability --> l4_sys_err_h
    l4_sys_capability --> l4_sys_ipc_h
    l4_sys_capability --> l4_sys_kernel_object_impl_h
    l4_sys_capability --> l4_sys_task_h
    l4_sys_capability --> l4_sys_fpage_h
    l4_sys_capability --> l4_sys_consts_h
    l4_sys_capability --> l4_sys_timeout_h
    l4_sys_capability --> kernel_object_h
    l4_sys_capability --> cxx_capability_h
    l4_sys_capability --> cxx_pc_basics
    l4_sys_capability --> cxx_types
    l4_sys_capability --> ipc_array
    l4_sys_capability --> l4_sys_cxx_pc_types
    l4_sys_capability --> _TypeInfo_h
    l4_sys_capability --> cxx_pc_iface
    l4_sys_meta --> l4_sys_object
    l4_sys_meta --> l4_sys_types_h
    l4_sys_meta --> l4_sys_compiler_h
    l4_sys_meta --> l4_sys_4int_h
    l4_sys_meta --> l4_sys_err_h
    l4_sys_meta --> l4_sys_ipc_h
    l4_sys_meta --> l4_sys_kernel_object_impl_h
    l4_sys_meta --> l4_sys_task_h
    l4_sys_meta --> l4_sys_fpage_h
    l4_sys_meta --> l4_sys_consts_h
    l4_sys_meta --> l4_sys_timeout_h
    l4_sys_meta --> kernel_object_h
    l4_sys_meta --> cxx_capability_h
    l4_sys_meta --> cxx_pc_basics
    l4_sys_meta --> cxx_types
    l4_sys_meta --> ipc_array
    l4_sys_meta --> l4_sys_cxx_pc_types
    l4_sys_meta --> _TypeInfo_h
    l4_sys_meta --> cxx_pc_iface
    cxx_pc_string --> ipc_array
    ipc_array --> cxx_types
    ipc_array --> l4_sys_err_h
    ipc_array --> l4_sys_ipc_h
    ipc_array --> l4_sys_kernel_object_impl_h
    ipc_array --> l4_sys_task_h
    ipc_array --> l4_sys_fpage_h
    ipc_array --> l4_sys_consts_h
    ipc_array --> l4_sys_timeout_h
    ipc_array --> kernel_object_h
    ipc_array --> cxx_capability_h
    ipc_array --> cxx_pc_basics
    ipc_array --> cxx_types
    l4_sys_cxx_pc_varg --> cxx_types
    l4_sys_err_h --> l4_sys_compiler_h
    l4_sys_err_h --> l4_sys_4int_h
    l4_sys_err_h --> l4_sys_err_h
    l4_sys_compiler_h --> l4_sys_linkage_h[l4/sys/linkage.h]
    l4_sys_4int_h --> l4_sys_linkage_h
    l4_sys_types_h --> l4_sys_compiler_h
    l4_sys_types_h --> l4_sys_4int_h
    l4_sys_types_h --> l4_sys_err_h
    l4_sys_types_h --> l4_sys_ipc_h
    l4_sys_types_h --> l4_sys_kernel_object_impl_h
    l4_sys_types_h --> l4_sys_task_h
    l4_sys_types_h --> l4_sys_fpage_h
    l4_sys_types_h --> l4_sys_consts_h
    l4_sys_types_h --> l4_sys_timeout_h
    l4_sys_types_h --> kernel_object_h
    l4_sys_types_h --> cxx_capability_h
    l4_sys_types_h --> cxx_pc_basics
    l4_sys_types_h --> cxx_types
    l4_sys_types_h --> ipc_array
    l4_sys_types_h --> l4_sys_cxx_pc_types
    l4_sys_types_h --> _TypeInfo_h
    l4_sys_types_h --> cxx_pc_iface
    l4_sys_ipc_h --> l4_sys_err_h
    l4_sys_ipc_h --> l4_sys_ipc_h
    l4_sys_ipc_h --> l4_sys_kernel_object_impl_h
    l4_sys_ipc_h --> l4_sys_task_h
    l4_sys_ipc_h --> l4_sys_fpage_h
    l4_sys_ipc_h --> l4_sys_consts_h
    l4_sys_ipc_h --> l4_sys_timeout_h
    l4_sys_ipc_h --> kernel_object_h
    l4_sys_ipc_h --> cxx_capability_h
    l4_sys_ipc_h --> cxx_pc_basics
    l4_sys_ipc_h --> cxx_types
    l4_sys_ipc_h --> ipc_array
    l4_sys_ipc_h --> l4_sys_cxx_pc_types
    l4_sys_ipc_h --> _TypeInfo_h
    l4_sys_ipc_h --> cxx_pc_iface
    l4_sys_kernel_object_impl_h --> l4_sys_err_h
    l4_sys_kernel_object_impl_h --> l4_sys_ipc_h
    l4_sys_kernel_object_impl_h --> l4_sys_kernel_object_impl_h
    l4_sys_kernel_object_impl_h --> l4_sys_task_h
    l4_sys_kernel_object_impl_h --> l4_sys_fpage_h
    l4_sys_kernel_object_impl_h --> l4_sys_consts_h
    l4_sys_kernel_object_impl_h --> l4_sys_timeout_h
    l4_sys_kernel_object_impl_h --> kernel_object_h
    l4_sys_kernel_object_impl_h --> cxx_capability_h
    l4_sys_kernel_object_impl_h --> cxx_pc_basics
    l4_sys_kernel_object_impl_h --> cxx_types
    l4_sys_kernel_object_impl_h --> ipc_array
    l4_sys_kernel_object_impl_h --> l4_sys_cxx_pc_types
    l4_sys_kernel_object_impl_h --> _TypeInfo_h
    l4_sys_kernel_object_impl_h --> cxx_pc_iface
    l4_sys_task_h --> l4_sys_err_h
    l4_sys_task_h --> l4_sys_ipc_h
    l4_sys_task_h --> l4_sys_kernel_object_impl_h
    l4_sys_task_h --> l4_sys_task_h
    l4_sys_task_h --> l4_sys_fpage_h
    l4_sys_task_h --> l4_sys_consts_h
    l4_sys_task_h --> l4_sys_timeout_h
    l4_sys_task_h --> kernel_object_h
    l4_sys_task_h --> cxx_capability_h
    l4_sys_task_h --> cxx_pc_basics
    l4_sys_task_h --> cxx_types
    l4_sys_task_h --> ipc_array
    l4_sys_task_h --> l4_sys_cxx_pc_types
    l4_sys_task_h --> _TypeInfo_h
    l4_sys_task_h --> cxx_pc_iface
    l4_sys_fpage_h --> l4_sys_err_h
    l4_sys_fpage_h --> l4_sys_ipc_h
    l4_sys_fpage_h --> l4_sys_kernel_object_impl_h
    l4_sys_fpage_h --> l4_sys_task_h
    l4_sys_fpage_h --> l4_sys_fpage_h
    l4_sys_fpage_h --> l4_sys_consts_h
    l4_sys_fpage_h --> l4_sys_timeout_h
    l4_sys_fpage_h --> kernel_object_h
    l4_sys_fpage_h --> cxx_capability_h
    l4_sys_fpage_h --> cxx_pc_basics
    l4_sys_fpage_h --> cxx_types
    l4_sys_fpage_h --> ipc_array
    l4_sys_fpage_h --> l4_sys_cxx_pc_types
    l4_sys_fpage_h --> _TypeInfo_h
    l4_sys_fpage_h --> cxx_pc_iface
    l4_sys_consts_h --> l4_sys_err_h
    l4_sys_consts_h --> l4_sys_ipc_h
    l4_sys_consts_h --> l4_sys_kernel_object_impl_h
    l4_sys_consts_h --> l4_sys_task_h
    l4_sys_consts_h --> l4_sys_fpage_h
    l4_sys_consts_h --> l4_sys_consts_h
    l4_sys_consts_h --> l4_sys_timeout_h
    l4_sys_consts_h --> kernel_object_h
    l4_sys_consts_h --> cxx_capability_h
    l4_sys_consts_h --> cxx_pc_basics
    l4_sys_consts_h --> cxx_types
    l4_sys_consts_h --> ipc_array
    l4_sys_consts_h --> l4_sys_cxx_pc_types
    l4_sys_consts_h --> _TypeInfo_h
    l4_sys_consts_h --> cxx_pc_iface
    l4_sys_timeout_h --> l4_sys_err_h
    l4_sys_timeout_h --> l4_sys_ipc_h
    l4_sys_timeout_h --> l4_sys_kernel_object_impl_h
    l4_sys_timeout_h --> l4_sys_task_h
    l4_sys_timeout_h --> l4_sys_fpage_h
    l4_sys_timeout_h --> l4_sys_consts_h
    l4_sys_timeout_h --> l4_sys_timeout_h
    l4_sys_timeout_h --> kernel_object_h
    l4_sys_timeout_h --> cxx_capability_h
    l4_sys_timeout_h --> cxx_pc_basics
    l4_sys_timeout_h --> cxx_types
    l4_sys_timeout_h --> ipc_array
    l4_sys_timeout_h --> l4_sys_cxx_pc_types
    l4_sys_timeout_h --> _TypeInfo_h
    l4_sys_timeout_h --> cxx_pc_iface
    kernel_object_h --> l4_sys_err_h
    kernel_object_h --> l4_sys_ipc_h
    kernel_object_h --> l4_sys_kernel_object_impl_h
    kernel_object_h --> l4_sys_task_h
    kernel_object_h --> l4_sys_fpage_h
    kernel_object_h --> l4_sys_consts_h
    kernel_object_h --> l4_sys_timeout_h
    kernel_object_h --> kernel_object_h
    kernel_object_h --> cxx_capability_h
    kernel_object_h --> cxx_pc_basics
    kernel_object_h --> cxx_types
    kernel_object_h --> ipc_array
    kernel_object_h --> l4_sys_cxx_pc_types
    kernel_object_h --> _TypeInfo_h
    kernel_object_h --> cxx_pc_iface
    cxx_capability_h --> l4_sys_err_h
    cxx_capability_h --> l4_sys_ipc_h
    cxx_capability_h --> l4_sys_kernel_object_impl_h
    cxx_capability_h --> l4_sys_task_h
    cxx_capability_h --> l4_sys_fpage_h
    cxx_capability_h --> l4_sys_consts_h
    cxx_capability_h --> l4_sys_timeout_h
    cxx_capability_h --> kernel_object_h
    cxx_capability_h --> cxx_capability_h
    cxx_capability_h --> cxx_pc_basics
    cxx_capability_h --> cxx_types
    cxx_capability_h --> ipc_array
    cxx_capability_h --> l4_sys_cxx_pc_types
    cxx_capability_h --> _TypeInfo_h
    cxx_capability_h --> cxx_pc_iface
    cxx_pc_basics --> l4_sys_err_h
    cxx_pc_basics --> l4_sys_ipc_h
    cxx_pc_basics --> l4_sys_kernel_object_impl_h
    cxx_pc_basics --> l4_sys_task_h
    cxx_pc_basics --> l4_sys_fpage_h
    cxx_pc_basics --> l4_sys_consts_h
    cxx_pc_basics --> l4_sys_timeout_h
    cxx_pc_basics --> kernel_object_h
    cxx_pc_basics --> cxx_capability_h
    cxx_pc_basics --> cxx_pc_basics
    cxx_pc_basics --> cxx_types
    cxx_pc_basics --> ipc_array
    cxx_pc_basics --> l4_sys_cxx_pc_types
    cxx_pc_basics --> _TypeInfo_h
    cxx_pc_basics --> cxx_pc_iface
    cxx_types --> l4_sys_err_h
    cxx_types --> l4_sys_ipc_h
    cxx_types --> l4_sys_kernel_object_impl_h
    cxx_types --> l4_sys_task_h
    cxx_types --> l4_sys_fpage_h
    cxx_types --> l4_sys_consts_h
    cxx_types --> l4_sys_timeout_h
    cxx_types --> kernel_object_h
    cxx_types --> cxx_capability_h
    cxx_types --> cxx_pc_basics
    cxx_types --> cxx_types
    cxx_types --> ipc_array
    cxx_types --> l4_sys_cxx_pc_types
    cxx_types --> _TypeInfo_h
    cxx_types --> cxx_pc_iface
    ipc_array --> l4_sys_err_h
    ipc_array --> l4_sys_ipc_h
    ipc_array --> l4_sys_kernel_object_impl_h
    ipc_array --> l4_sys_task_h
    ipc_array --> l4_sys_fpage_h
    ipc_array --> l4
```

[illegible]

- class `L4::Factory`
C++ Factory 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.

- L4
L4 low-level kernel interface.

16.295.1 Detailed Description

Common factory related definitions.

Definition in file [factory](#).

16.296 factory

```

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
00152 operator l4_msgtag_t () noexcept
00153 {
00154     l4_msgtag_t r = l4_factory_create_commit_u(f, t, u);
00155     t.raw = 0;
00156     return r;
00157 }
00158
00166 void put(l4_mword_t i) noexcept
00167 {
00168     l4_factory_create_add_int_u(i, &t, u);
00169 }
00170
00178 void put(l4_umword_t i) noexcept
00179 {
00180     l4_factory_create_add_uint_u(i, &t, u);
00181 }
00182
00192 void put(char const *s) & noexcept
00193 {
00194     l4_factory_create_add_str_u(s, &t, u);
00195 }
00196
00208 void put(Lstr const &s) & noexcept
00209 {
00210     l4_factory_create_add_lstr_u(s.s, s.len, &t, u);
00211 }
00212
00218 void put(Nil) & noexcept
00219 {
00220     l4_factory_create_add_nil_u(&t, u);
00221 }
00222
00230 void put(l4_fpage_t d) & noexcept
00231 {
00232     l4_factory_create_add_fpage_u(d, &t, u);
00233 }
00234
00235 template<typename T>
00236 S &operator « (T const &d) & noexcept
00237 {
00238     put(d);
00239     return *this;
00240 }
00241
00242 template<typename T>
00243 S &&operator « (T const &d) && noexcept
00244 {
00245     put(d);
00246     return _move(*this);
00247 }
00248 };
00249
00250
00251 public:
00252
00277 S create(Cap<void> target, long obj, l4_utcb_t *utcb = l4_utcb()) noexcept
00278 {
00279     return S(cap(), obj, target, utcb);
00280 }
00281
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
00341 l4_msgtag_t create_task(Cap<Task> const & target_cap,
00342                        l4_fpage_t const &utcb_area,
00343                        l4_utcb_t *utcb = l4_utcb()) noexcept
00344 { return l4_factory_create_task_u(cap(), target_cap.cap(), utcb_area, utcb); }

```

```

00345
00359 l4_msgtag_t create_thread(Cap<Thread> const &target_cap,
00360                          l4_utcb_t *utcb = l4_utcb()) noexcept
00361     L4_DEPRECATED("Call create with Cap<Thread> as argument instead.")
00362     { return l4_factory_create_thread_u(cap(), target_cap.cap(), utcb); }
00363
00383 l4_msgtag_t create_factory(Cap<Factory> const &target_cap,
00384                           unsigned long limit,
00385                           l4_utcb_t *utcb = l4_utcb()) noexcept
00386     { return l4_factory_create_factory_u(cap(), target_cap.cap(), limit, utcb); }
00387
00416 l4_msgtag_t create_gate(Cap<void> const &target_cap,
00417                        Cap<Thread> const &thread_cap, l4_umword_t label,
00418                        l4_utcb_t *utcb = l4_utcb()) noexcept
00419     { return l4_factory_create_gate_u(cap(), target_cap.cap(), thread_cap.cap(), label, utcb); }
00420
00434 l4_msgtag_t create_irq(Cap<Irq> const &target_cap,
00435                       l4_utcb_t *utcb = l4_utcb()) throw()
00436     L4_DEPRECATED("Call create with Cap<Irq> as argument instead.")
00437     { return l4_factory_create_irq_u(cap(), target_cap.cap(), utcb); }
00438
00452 l4_msgtag_t create_vm(Cap<Vm> const &target_cap,
00453                     l4_utcb_t *utcb = l4_utcb()) noexcept
00454     L4_DEPRECATED("Call create with Cap<Vm> as argument instead.")
00455     { return l4_factory_create_vm_u(cap(), target_cap.cap(), utcb); }
00456
00457 typedef L4::TypeId::Rpc_nocode<create_t> Rpcs;
00458 };
00459
00460 }

```

16.297 l4/sys/factory.h File Reference

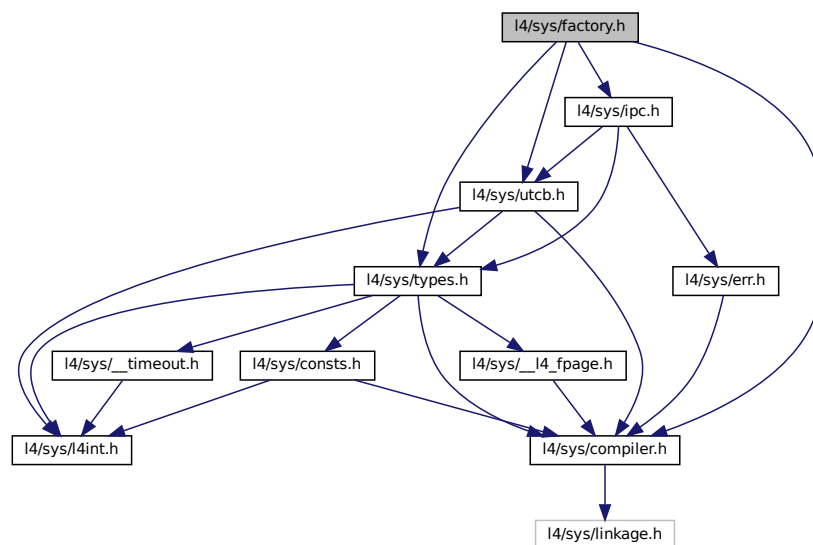
Common factory related definitions.

```

#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:



16.298 factory.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  *      Henning Schild <hschild@os.inf.tu-dresden.de>
00012  *      economic rights: Technische Universität Dresden (Germany)
00013  *
00014  * This file is part of TUD:OS and distributed under the terms of the
00015  * GNU General Public License 2.
00016  * Please see the COPYING-GPL-2 file for details.
00017  *
00018  * As a special exception, you may use this file as part of a free software
00019  * library without restriction. Specifically, if other files instantiate
00020  * templates or use macros or inline functions from this file, or you compile
00021  * this file and link it with other files to produce an executable, this
00022  * file does not by itself cause the resulting executable to be covered by
00023  * the GNU General Public License. This exception does not however
00024  * invalidate any other reasons why the executable file might be covered by
00025  * the GNU General Public License.
00026  */
00027 #pragma once
00028
00029 #include <l4/sys/compiler.h>
00030 #include <l4/sys/types.h>
00031 #include <l4/sys/utcb.h>
00032
00086 L4_INLINE l4_msgtag_t
00087 l4_factory_create_task(l4_cap_idx_t factory,
00088                       l4_cap_idx_t target_cap, l4_fpage_t utcb_area) L4_NOTHROW;
00089
00096 L4_INLINE l4_msgtag_t
00097 l4_factory_create_task_u(l4_cap_idx_t factory, l4_cap_idx_t target_cap,
00098                        l4_fpage_t utcb_area, l4_utcb_t *utcb) L4_NOTHROW;
00099
00111 L4_INLINE l4_msgtag_t
00112 l4_factory_create_thread(l4_cap_idx_t factory,
00113                        l4_cap_idx_t target_cap) L4_NOTHROW;
00114
00121 L4_INLINE l4_msgtag_t
00122 l4_factory_create_thread_u(l4_cap_idx_t factory,
00123                          l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW;
00124
00138 L4_INLINE l4_msgtag_t
00139 l4_factory_create_factory(l4_cap_idx_t factory, l4_cap_idx_t target_cap,
00140                          unsigned long limit) L4_NOTHROW;
00141
00148 L4_INLINE l4_msgtag_t
00149 l4_factory_create_factory_u(l4_cap_idx_t factory, l4_cap_idx_t target_cap,
00150                           unsigned long limit, l4_utcb_t *utcb) L4_NOTHROW;
00151
00176 L4_INLINE l4_msgtag_t
00177 l4_factory_create_gate(l4_cap_idx_t factory,
00178                       l4_cap_idx_t target_cap,
00179                       l4_cap_idx_t thread_cap, l4_umword_t label) L4_NOTHROW;
00180
00187 L4_INLINE l4_msgtag_t
00188 l4_factory_create_gate_u(l4_cap_idx_t factory,
00189                        l4_cap_idx_t target_cap,
00190                        l4_cap_idx_t thread_cap, l4_umword_t label,
00191                        l4_utcb_t *utcb) L4_NOTHROW;
00192
00205 L4_INLINE l4_msgtag_t
00206 l4_factory_create_irq(l4_cap_idx_t factory,
00207                      l4_cap_idx_t target_cap) L4_NOTHROW;
00208
00215 L4_INLINE l4_msgtag_t
00216 l4_factory_create_irq_u(l4_cap_idx_t factory,
00217                       l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW;
00218
00230 L4_INLINE l4_msgtag_t
00231 l4_factory_create_vm(l4_cap_idx_t factory,
00232                    l4_cap_idx_t target_cap) L4_NOTHROW;
00233
00240 L4_INLINE l4_msgtag_t
00241 l4_factory_create_vm_u(l4_cap_idx_t factory,
00242                      l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW;
00243
00244 L4_INLINE l4_msgtag_t
00245 l4_factory_create_start_u(long obj, l4_cap_idx_t target,
00246                          l4_utcb_t *utcb) L4_NOTHROW;
00247
00248 L4_INLINE int
00249 l4_factory_create_add_fpage_u(l4_fpage_t d, l4_msgtag_t *tag,

```

```

00250             l4_utcb_t *utcb) L4_NOTHROW;
00251
00252 L4_INLINE int
00253 l4_factory_create_add_int_u(l4_mword_t d, l4_msgtag_t *tag,
00254             l4_utcb_t *utcb) L4_NOTHROW;
00255
00256 L4_INLINE int
00257 l4_factory_create_add_uint_u(l4_umword_t d, l4_msgtag_t *tag,
00258             l4_utcb_t *utcb) L4_NOTHROW;
00259
00260 L4_INLINE int
00261 l4_factory_create_add_str_u(char const *s, l4_msgtag_t *tag,
00262             l4_utcb_t *utcb) L4_NOTHROW;
00263
00264 L4_INLINE int
00265 l4_factory_create_add_lstr_u(char const *s, unsigned len, l4_msgtag_t *tag,
00266             l4_utcb_t *utcb) L4_NOTHROW;
00267
00268 L4_INLINE int
00269 l4_factory_create_add_nil_u(l4_msgtag_t *tag, l4_utcb_t *utcb) L4_NOTHROW;
00270
00271 L4_INLINE l4_msgtag_t
00272 l4_factory_create_commit_u(l4_cap_idx_t factory, l4_msgtag_t tag,
00273             l4_utcb_t *utcb) L4_NOTHROW;
00274
00275 L4_INLINE l4_msgtag_t
00276 l4_factory_create_u(l4_cap_idx_t factory, long obj, l4_cap_idx_t target,
00277             l4_utcb_t *utcb) L4_NOTHROW;
00278
00279
00280 L4_INLINE l4_msgtag_t
00281 l4_factory_create(l4_cap_idx_t factory, long obj,
00282             l4_cap_idx_t target) L4_NOTHROW;
00283
00284 /* IMPLEMENTATION ----- */
00285
00286 #include <l4/sys/ipc.h>
00287
00288 L4_INLINE l4_msgtag_t
00289 l4_factory_create_task_u(l4_cap_idx_t factory,
00290             l4_cap_idx_t target_cap, l4_fpage_t utcb_area,
00291             l4_utcb_t *u) L4_NOTHROW
00292 {
00293     l4_msgtag_t t;
00294     t = l4_factory_create_start_u(L4_PROTO_TASK, target_cap, u);
00295     l4_factory_create_add_fpage_u(utcb_area, &t, u);
00296     return l4_factory_create_commit_u(factory, t, u);
00297 }
00298
00299 L4_INLINE l4_msgtag_t
00300 l4_factory_create_thread_u(l4_cap_idx_t factory,
00301             l4_cap_idx_t target_cap, l4_utcb_t *u) L4_NOTHROW
00302 {
00303     return l4_factory_create_u(factory, L4_PROTO_THREAD, target_cap, u);
00304 }
00305
00306 L4_INLINE l4_msgtag_t
00307 l4_factory_create_factory_u(l4_cap_idx_t factory,
00308             l4_cap_idx_t target_cap, unsigned long limit,
00309             l4_utcb_t *u) L4_NOTHROW
00310 {
00311     l4_msgtag_t t;
00312     t = l4_factory_create_start_u(L4_PROTO_FACTORY, target_cap, u);
00313     l4_factory_create_add_uint_u(limit, &t, u);
00314     return l4_factory_create_commit_u(factory, t, u);
00315 }
00316
00317 L4_INLINE l4_msgtag_t
00318 l4_factory_create_gate_u(l4_cap_idx_t factory,
00319             l4_cap_idx_t target_cap,
00320             l4_cap_idx_t thread_cap, l4_umword_t label,
00321             l4_utcb_t *u) L4_NOTHROW
00322 {
00323     l4_msgtag_t t;
00324     l4_msg_regs_t *v;
00325     int items = 0;
00326     t = l4_factory_create_start_u(0, target_cap, u);
00327     l4_factory_create_add_uint_u(label, &t, u);
00328     v = l4_utcb_mr_u(u);
00329     if (!(thread_cap & L4_INVALID_CAP_BIT))
00330     {
00331         items = 1;
00332         v->mr[3] = l4_map_obj_control(0,0);
00333         v->mr[4] = l4_obj_fpage(thread_cap, 0, L4_CAP_FPAGE_RWS).raw;
00334     }
00335     t = l4_msgtag(l4_msgtag_label(t), l4_msgtag_words(t), items, l4_msgtag_flags(t));
00336     return l4_factory_create_commit_u(factory, t, u);

```



```

00337 }
00338
00339 L4_INLINE l4_msgtag_t
00340 l4_factory_create_irq_u(l4_cap_idx_t factory,
00341                        l4_cap_idx_t target_cap, l4_utcb_t *u) L4_NOTHROW
00342 {
00343     return l4_factory_create_u(factory, L4_PROTO_IRQ_SENDER, target_cap, u);
00344 }
00345
00346 L4_INLINE l4_msgtag_t
00347 l4_factory_create_vm_u(l4_cap_idx_t factory,
00348                       l4_cap_idx_t target_cap,
00349                       l4_utcb_t *u) L4_NOTHROW
00350 {
00351     return l4_factory_create_u(factory, L4_PROTO_VM, target_cap, u);
00352 }
00353
00354
00355
00356
00357
00358 L4_INLINE l4_msgtag_t
00359 l4_factory_create_task(l4_cap_idx_t factory,
00360                       l4_cap_idx_t target_cap, l4_fpage_t utcb_area) L4_NOTHROW
00361 {
00362     return l4_factory_create_task_u(factory, target_cap, utcb_area, l4_utcb());
00363 }
00364
00365 L4_INLINE l4_msgtag_t
00366 l4_factory_create_thread(l4_cap_idx_t factory,
00367                          l4_cap_idx_t target_cap) L4_NOTHROW
00368 {
00369     return l4_factory_create_thread_u(factory, target_cap, l4_utcb());
00370 }
00371
00372 L4_INLINE l4_msgtag_t
00373 l4_factory_create_factory(l4_cap_idx_t factory,
00374                          l4_cap_idx_t target_cap, unsigned long limit) L4_NOTHROW
00375 {
00376     return l4_factory_create_factory_u(factory, target_cap, limit, l4_utcb());
00377 }
00378
00379
00380 L4_INLINE l4_msgtag_t
00381 l4_factory_create_gate(l4_cap_idx_t factory,
00382                       l4_cap_idx_t target_cap,
00383                       l4_cap_idx_t thread_cap, l4_umword_t label) L4_NOTHROW
00384 {
00385     return l4_factory_create_gate_u(factory, target_cap, thread_cap, label, l4_utcb());
00386 }
00387
00388 L4_INLINE l4_msgtag_t
00389 l4_factory_create_irq(l4_cap_idx_t factory,
00390                      l4_cap_idx_t target_cap) L4_NOTHROW
00391 {
00392     return l4_factory_create_irq_u(factory, target_cap, l4_utcb());
00393 }
00394
00395 L4_INLINE l4_msgtag_t
00396 l4_factory_create_vm(l4_cap_idx_t factory,
00397                     l4_cap_idx_t target_cap) L4_NOTHROW
00398 {
00399     return l4_factory_create_vm_u(factory, target_cap, l4_utcb());
00400 }
00401
00402 L4_INLINE l4_msgtag_t
00403 l4_factory_create_start_u(long obj, l4_cap_idx_t target_cap,
00404                          l4_utcb_t *u) L4_NOTHROW
00405 {
00406     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00407     l4_buf_regs_t *b = l4_utcb_br_u(u);
00408     v->mr[0] = obj;
00409     b->bdr = 0;
00410     b->br[0] = target_cap | L4_RCV_ITEM_SINGLE_CAP;
00411     return l4_msgtag(L4_PROTO_FACTORY, 1, 0, 0);
00412 }
00413
00414 L4_INLINE int
00415 l4_factory_create_add_fpage_u(l4_fpage_t d, l4_msgtag_t *tag,
00416                              l4_utcb_t *u) L4_NOTHROW
00417 {
00418     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00419     int w = l4_msgtag_words(*tag);
00420     if (w + 2 > L4_UTCB_GENERIC_DATA_SIZE)
00421         return 0;
00422     v->mr[w] = L4_VARG_TYPE_FPAGE | (sizeof(l4_fpage_t) << 16);
00423     v->mr[w + 1] = d.raw;

```

```

00424     w += 2;
00425     tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00426     return 1;
00427 }
00428
00429 L4_INLINE int
00430 l4_factory_create_add_int_u(l4_mword_t d, l4_msgtag_t *tag,
00431                            l4_utcb_t *u) L4_NOTHROW
00432 {
00433     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00434     int w = l4_msgtag_words(*tag);
00435     if (w + 2 > L4_UTCB_GENERIC_DATA_SIZE)
00436         return 0;
00437     v->mr[w] = L4_VARG_TYPE_MWORD | (sizeof(l4_mword_t) << 16);
00438     v->mr[w + 1] = d;
00439     w += 2;
00440     tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00441     return 1;
00442 }
00443
00444 L4_INLINE int
00445 l4_factory_create_add_uint_u(l4_umword_t d, l4_msgtag_t *tag,
00446                             l4_utcb_t *u) L4_NOTHROW
00447 {
00448     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00449     int w = l4_msgtag_words(*tag);
00450     if (w + 2 > L4_UTCB_GENERIC_DATA_SIZE)
00451         return 0;
00452     v->mr[w] = L4_VARG_TYPE_UMWORD | (sizeof(l4_umword_t) << 16);
00453     v->mr[w + 1] = d;
00454     w += 2;
00455     tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00456     return 1;
00457 }
00458
00459 L4_INLINE int
00460 l4_factory_create_add_str_u(char const *s, l4_msgtag_t *tag,
00461                            l4_utcb_t *u) L4_NOTHROW
00462 {
00463     return l4_factory_create_add_lstr_u(s, __builtin_strlen(s) + 1, tag, u);
00464 }
00465
00466 L4_INLINE int
00467 l4_factory_create_add_lstr_u(char const *s, unsigned len, l4_msgtag_t *tag,
00468                             l4_utcb_t *u) L4_NOTHROW
00469 {
00470     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00471     unsigned w = l4_msgtag_words(*tag);
00472     char *c;
00473     unsigned i;
00474     if (w + 1 + l4_bytes_to_mwords(len) > L4_UTCB_GENERIC_DATA_SIZE)
00475         return 0;
00476     v->mr[w] = L4_VARG_TYPE_STRING | (len << 16);
00477     c = (char*)&v->mr[w + 1];
00478     for (i = 0; i < len; ++i)
00479         *c++ = *s++;
00480     w = w + 1 + l4_bytes_to_mwords(len);
00481     tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00482     return 1;
00483 }
00484
00485 L4_INLINE int
00486 l4_factory_create_add_nil_u(l4_msgtag_t *tag, l4_utcb_t *utcb) L4_NOTHROW
00487 {
00488     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00489     int w = l4_msgtag_words(*tag);
00490     v->mr[w] = L4_VARG_TYPE_NIL;
00491     ++w;
00492     tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00493     return 1;
00494 }
00495
00496 L4_INLINE l4_msgtag_t
00497 l4_factory_create_commit_u(l4_cap_idx_t factory, l4_msgtag_t tag,
00498                           l4_utcb_t *u) L4_NOTHROW
00499 {
00500     return l4_ipc_call(factory, u, tag, L4_IPC_NEVER);
00501 }
00502
00503 L4_INLINE l4_msgtag_t
00504 l4_factory_create_u(l4_cap_idx_t factory, long obj, l4_cap_idx_t target,

```

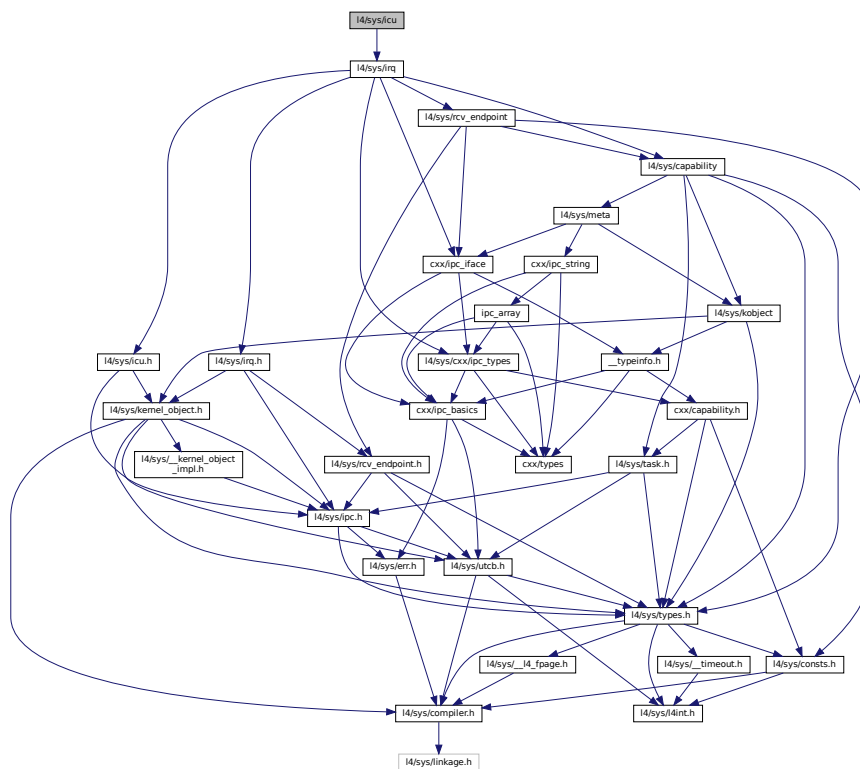
```
00511                                     l4_utcb_t *utcb) L4_NOTHROW
00512 {
00513     l4_msgtag_t t = l4_factory_create_start_u(obj, target, utcb);
00514     return l4_factory_create_commit_u(factory, t, utcb);
00515 }
00516
00517
00518 L4_INLINE l4_msgtag_t
00519 l4_factory_create(l4_cap_idx_t factory, long obj,
00520                  l4_cap_idx_t target) L4_NOTHROW
00521 {
00522     return l4_factory_create_u(factory, obj, target, l4_utcb());
00523 }
```

16.299 I4/sys/icu File Reference

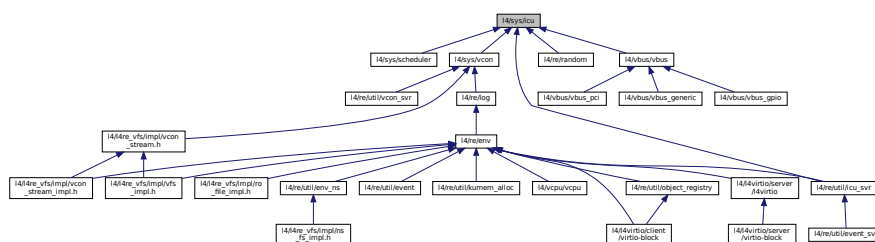
Interrupt controller.

```
#include <linux/sys/irq>
```

Include dependency graph for icu:



This graph shows which files directly or indirectly include this file:



16.299.1 Detailed Description

Interrupt controller.

Definition in file [icu](#).

16.300 icu

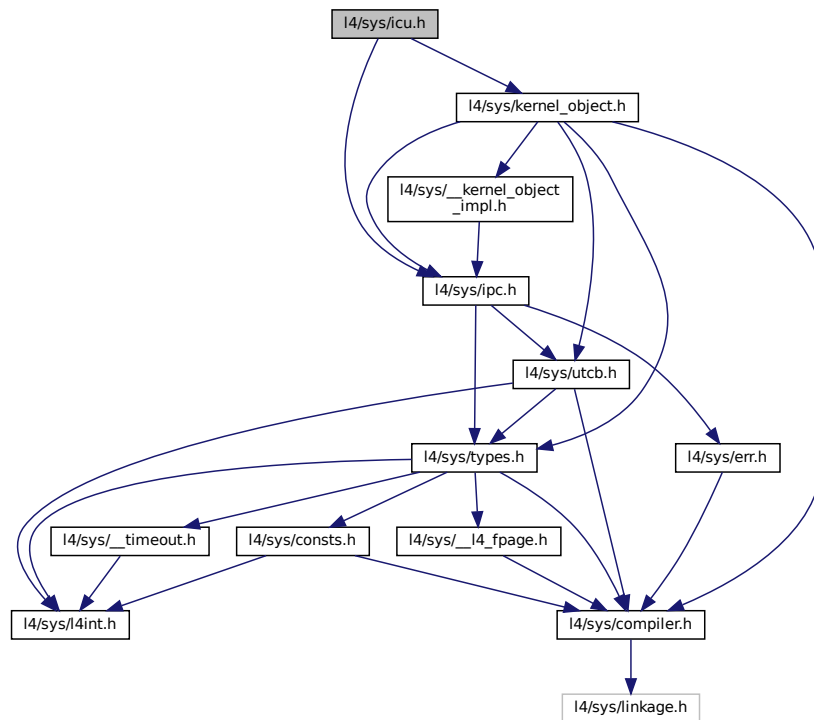
```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00007 /*
00008  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public 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>
```

16.301 l4/sys/icu.h File Reference

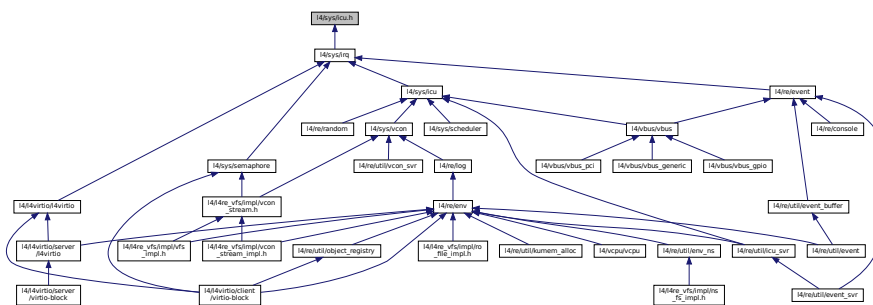
Interrupt controller.

```
#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 capabilities of the ICU.
- [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 capabilities of the ICU.
- [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`
Acknowledge 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.

16.301.1 Detailed Description

Interrupt controller.

Definition in file [icu.h](#).

16.302 icu.h

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #pragma once
00026
00027 #include <l4/sys/kernel_object.h>
00028 #include <l4/sys/ipc.h>
00029
00050 enum l4_icu_flags
00051 {
00059     L4_ICU_FLAG_MSI = 0x80000000,
00060 };
00061
00062
00067 enum l4_irq_mode
00068 {
00070     L4_IRQ_F_NONE           = 0,
00071     L4_IRQ_F_LEVEL         = 0x2,
00072     L4_IRQ_F_EDGE          = 0x0,
00073     L4_IRQ_F_POS           = 0x0,
00074     L4_IRQ_F_NEG           = 0x4,
00075     L4_IRQ_F_BOTH          = 0x8,
00076     L4_IRQ_F_LEVEL_HIGH    = 0x3,
00077     L4_IRQ_F_LEVEL_LOW     = 0x7,
00078     L4_IRQ_F_POS_EDGE      = 0x1,
00079     L4_IRQ_F_NEG_EDGE      = 0x5,
00080     L4_IRQ_F_BOTH_EDGE     = 0x9,
00081     L4_IRQ_F_MASK          = 0xf,
00084     L4_IRQ_F_SET_WAKEUP    = 0x10,
00085     L4_IRQ_F_CLEAR_WAKEUP  = 0x20,
00086 };
00087

```

```

00088
00093 enum L4_icu_opcode
00094 {
00100     L4_ICU_OP_BIND = 0,
00101
00107     L4_ICU_OP_UNBIND = 1,
00108
00114     L4_ICU_OP_INFO = 2,
00115
00121     L4_ICU_OP_MSI_INFO = 3,
00122
00128     L4_ICU_OP_UNMASK = 4,
00129
00135     L4_ICU_OP_MASK = 5,
00136
00142     L4_ICU_OP_SET_MODE = 6,
00143 };
00144
00145 enum L4_icu_ctl_op
00146 {
00147     L4_ICU_CTL_UNMASK = 0,
00148     L4_ICU_CTL_MASK = 1
00149 };
00150
00151
00159 typedef struct l4_icu_info_t
00160 {
00166     unsigned features;
00167
00171     unsigned nr_irqs;
00172
00176     unsigned nr_msis;
00177 } l4_icu_info_t;
00178
00180 typedef struct l4_icu_msi_info_t
00181 {
00183     l4_uint64_t msi_addr;
00185     l4_uint32_t msi_data;
00186 } l4_icu_msi_info_t;
00187
00204 L4_INLINE l4_msgtag_t
00205 l4_icu_bind(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq) L4_NOTHROW;
00206
00213 L4_INLINE l4_msgtag_t
00214 l4_icu_bind_u(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq,
00215               l4_utcb_t *utcb) L4_NOTHROW;
00216
00227 L4_INLINE l4_msgtag_t
00228 l4_icu_unbind(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq) L4_NOTHROW;
00229
00236 L4_INLINE l4_msgtag_t
00237 l4_icu_unbind_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_set_mode(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t mode) L4_NOTHROW;
00252
00259 L4_INLINE l4_msgtag_t
00260 l4_icu_set_mode_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t mode,
00261                  l4_utcb_t *utcb) L4_NOTHROW;
00262
00274 L4_INLINE l4_msgtag_t
00275 l4_icu_info(l4_cap_idx_t icu, l4_icu_info_t *info) L4_NOTHROW;
00276
00283 L4_INLINE l4_msgtag_t
00284 l4_icu_info_u(l4_cap_idx_t icu, l4_icu_info_t *info,
00285               l4_utcb_t *utcb) L4_NOTHROW;
00286
00293 L4_INLINE l4_msgtag_t
00294 l4_icu_msi_info(l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source,
00295                 l4_icu_msi_info_t *msi_info) L4_NOTHROW;
00296
00303 L4_INLINE l4_msgtag_t
00304 l4_icu_msi_info_u(l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source,
00305                  l4_icu_msi_info_t *msi_info, l4_utcb_t *utcb) L4_NOTHROW;
00306
00307
00320 L4_INLINE l4_msgtag_t
00321 l4_icu_unmask(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00322               l4_timeout_t to) L4_NOTHROW;
00323
00330 L4_INLINE l4_msgtag_t
00331 l4_icu_unmask_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00332                 l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW;
00333
00345 L4_INLINE l4_msgtag_t
00346 l4_icu_mask(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_mask_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00357               l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW;
00358
00362 L4_INLINE l4_msgtag_t
00363 l4_icu_control_u(l4_cap_idx_t icu, unsigned irqnum, unsigned op,
00364                 l4_umword_t *label, l4_timeout_t to,
00365                 l4_utcb_t *utcb) L4_NOTHROW;
00366
00367
00368 /*****
00369  * Implementations
00370  */
00371
00372 L4_INLINE l4_msgtag_t
00373 l4_icu_bind_u(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq,
00374              l4_utcb_t *utcb) L4_NOTHROW
00375 {
00376     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00377     m->mr[0] = L4_ICU_OP_BIND;
00378     m->mr[1] = irqnum;
00379     m->mr[2] = l4_map_obj_control(0, 0);
00380     m->mr[3] = l4_obj_fpage(irq, 0, L4_CAP_FPAGE_RWS).raw;
00381     return l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 2, 1, 0), L4_IPC_NEVER);
00382 }
00383
00384 L4_INLINE l4_msgtag_t
00385 l4_icu_unbind_u(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq,
00386                l4_utcb_t *utcb) L4_NOTHROW
00387 {
00388     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00389     m->mr[0] = L4_ICU_OP_UNBIND;
00390     m->mr[1] = irqnum;
00391     m->mr[2] = l4_map_obj_control(0, 0);
00392     m->mr[3] = l4_obj_fpage(irq, 0, L4_CAP_FPAGE_RWS).raw;
00393     return l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 2, 1, 0), L4_IPC_NEVER);
00394 }
00395
00396 L4_INLINE l4_msgtag_t
00397 l4_icu_info_u(l4_cap_idx_t icu, l4_icu_info_t *info,
00398              l4_utcb_t *utcb) L4_NOTHROW
00399 {
00400     l4_msgtag_t res;
00401     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00402     m->mr[0] = L4_ICU_OP_INFO;
00403     res = l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 1, 0, 0), L4_IPC_NEVER);
00404     info->features = m->mr[0];
00405     info->nr_irqs = m->mr[1];
00406     info->nr_msis = m->mr[2];
00407     return res;
00408 }
00409
00410 L4_INLINE l4_msgtag_t
00411 l4_icu_msi_info_u(l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source,
00412                  l4_icu_msi_info_t *msi_info, l4_utcb_t *utcb) L4_NOTHROW
00413 {
00414     l4_msgtag_t res;
00415     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00416     m->mr[0] = L4_ICU_OP_MSI_INFO;
00417     m->mr[1] = irqnum;
00418     m->mr64[l4_utcb_mr64_idx(2)] = source;
00419     res = l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ,
00420                                           2 + 1 * sizeof(l4_uint64_t)
00421                                           / sizeof(l4_umword_t),
00422                                           0, 0), L4_IPC_NEVER);
00423     if (L4_UNLIKELY(l4_msgtag_has_error(res)))
00424         return res;
00425
00426     if (L4_UNLIKELY(l4_msgtag_words(res) * sizeof(l4_umword_t) < sizeof(*msi_info)))
00427         return res;
00428
00429     __builtin_memcpy(msi_info, &m->mr[0], sizeof(*msi_info));
00430     return res;
00431 }
00432
00433 L4_INLINE l4_msgtag_t
00434 l4_icu_set_mode_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t mode,
00435                  l4_utcb_t *utcb) L4_NOTHROW
00436 {
00437     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00438     m->mr[0] = L4_ICU_OP_SET_MODE;
00439     m->mr[1] = irqnum;
00440     m->mr[2] = mode;
00441     return l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 3, 0, 0), L4_IPC_NEVER);
00442 }

```

```

00443
00444 L4_INLINE l4_msgtag_t
00445 l4_icu_control_u(l4_cap_idx_t icu, unsigned irqnum, unsigned op,
00446                 l4_umword_t *label, l4_timeout_t to,
00447                 l4_utcb_t *utcb) L4_NOTHROW
00448 {
00449     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00450     m->mr[0] = L4_ICU_OP_UNMASK + op;
00451     m->mr[1] = irqnum;
00452     if (label)
00453         return l4_ipc_send_and_wait(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 2, 0, 0),
00454                                     label, to);
00455     else
00456         return l4_ipc_send(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 2, 0, 0), to);
00457 }
00458
00459 L4_INLINE l4_msgtag_t
00460 l4_icu_mask_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00461              l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW
00462 { return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_MASK, label, to, utcb); }
00463
00464 L4_INLINE l4_msgtag_t
00465 l4_icu_unmask_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00466                l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW
00467 { return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_UNMASK, label, to, utcb); }
00468
00469
00470
00471
00472 L4_INLINE l4_msgtag_t
00473 l4_icu_bind(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq) L4_NOTHROW
00474 { return l4_icu_bind_u(icu, irqnum, irq, l4_utcb()); }
00475
00476 L4_INLINE l4_msgtag_t
00477 l4_icu_unbind(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq) L4_NOTHROW
00478 { return l4_icu_unbind_u(icu, irqnum, irq, l4_utcb()); }
00479
00480 L4_INLINE l4_msgtag_t
00481 l4_icu_info(l4_cap_idx_t icu, l4_icu_info_t *info) L4_NOTHROW
00482 { return l4_icu_info_u(icu, info, l4_utcb()); }
00483
00484 L4_INLINE l4_msgtag_t
00485 l4_icu_msi_info(l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source,
00486                l4_icu_msi_info_t *msi_info) L4_NOTHROW
00487 { return l4_icu_msi_info_u(icu, irqnum, source, msi_info, l4_utcb()); }
00488
00489 L4_INLINE l4_msgtag_t
00490 l4_icu_unmask(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00491              l4_timeout_t to) L4_NOTHROW
00492 { return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_UNMASK, label, to, l4_utcb()); }
00493
00494 L4_INLINE l4_msgtag_t
00495 l4_icu_mask(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00496            l4_timeout_t to) L4_NOTHROW
00497 { return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_MASK, label, to, l4_utcb()); }
00498
00499 L4_INLINE l4_msgtag_t
00500 l4_icu_set_mode(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t mode) L4_NOTHROW
00501 {
00502     return l4_icu_set_mode_u(icu, irqnum, mode, l4_utcb());
00503 }

```

16.303 l4/sys/ipc.h File Reference

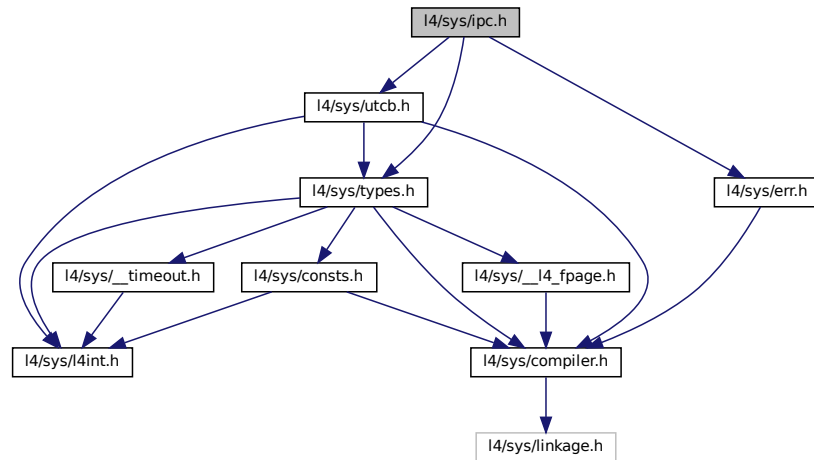
Common IPC interface.

```

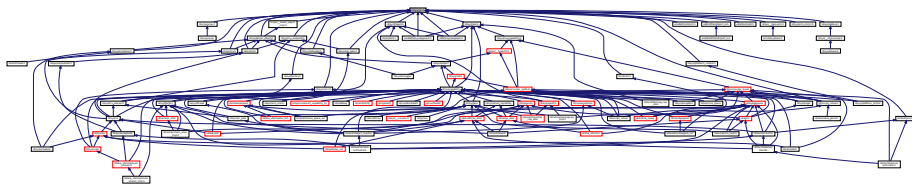
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/err.h>

```

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 error code for an object invocation.
- `long l4_error (l4_msgtag_t tag) L4_NOTHROW`
Return error code of a system call return message tag or the tag label.
- `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_ALWAYS_INLINE l4_msgtag_t l4_ipc (l4_cap_idx_t dest, l4_utcb_t *utcb, l4_umword_t flags, l4_umword_t label, 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.*
- `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.*

16.303.1 Detailed Description

Common IPC interface.

Definition in file [ipc.h](#).

16.303.2 Function Documentation

16.303.2.1 l4_ipc_to_errno()

```
long l4_ipc_to_errno (
    unsigned long ipc_error_code )    [inline]
```

Get a negative error code for the given IPC error code.

Parameters

<code>ipc_error_code</code>	IPC error code as delivered by the kernel. (or returned by the l4_ipc_error_code() function).
-----------------------------	---

Returns

negative error code in the range of L4_EIPC_LO to L4_EIPC_HI.

Definition at line 459 of file [ipc.h](#).

References [L4_EIPC_LO](#).

16.304 ipc.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 #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,
00124     L4_IPC_REMSGCUT            = 0x09,
00128     L4_IPC_SEMSGCUT            = 0x08,
00132 };
00133
00134
00145 L4_INLINE l4_umword_t
00146 l4_ipc_error(l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW;
00147
00148
00158 L4_INLINE long
00159 l4_error(l4_msgtag_t tag) L4_NOTHROW;
00160

```

```

00161 L4_INLINE long
00162 l4_error_u(l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW;
00163
00164 /*****
00165 *** IPC results
00166 *****/
00167
00177 L4_INLINE int l4_ipc_is_snd_error(l4_utcb_t *utcb) L4_NOTHROW;
00178
00188 L4_INLINE int l4_ipc_is_rcv_error(l4_utcb_t *utcb) L4_NOTHROW;
00189
00199 L4_INLINE int l4_ipc_error_code(l4_utcb_t *utcb) L4_NOTHROW;
00200
00207 L4_INLINE long l4_ipc_to_errno(unsigned long ipc_error_code) L4_NOTHROW;
00208
00209
00210 /*****
00211 *** IPC calls
00212 *****/
00213
00234 L4_INLINE l4_msgtag_t
00235 l4_ipc_send(l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag,
00236             l4_timeout_t timeout) L4_NOTHROW;
00237
00238
00259 L4_INLINE l4_msgtag_t
00260 l4_ipc_wait(l4_utcb_t *utcb, l4_umword_t *label,
00261             l4_timeout_t timeout) L4_NOTHROW;
00262
00263
00286 L4_INLINE l4_msgtag_t
00287 l4_ipc_receive(l4_cap_idx_t object, l4_utcb_t *utcb,
00288                l4_timeout_t timeout) L4_NOTHROW;
00289
00309 L4_INLINE l4_msgtag_t
00310 l4_ipc_call(l4_cap_idx_t object, l4_utcb_t *utcb, l4_msgtag_t tag,
00311             l4_timeout_t timeout) L4_NOTHROW;
00312
00313
00333 L4_INLINE l4_msgtag_t
00334 l4_ipc_reply_and_wait(l4_utcb_t *utcb, l4_msgtag_t tag,
00335                       l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW;
00336
00359 L4_INLINE l4_msgtag_t
00360 l4_ipc_send_and_wait(l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag,
00361                      l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW;
00362
00369 #if 0
00380 L4_INLINE l4_msgtag_t
00381 l4_ipc_wait_next_period(l4_utcb_t *utcb,
00382                         l4_umword_t *label,
00383                         l4_timeout_t timeout);
00384
00385 #endif
00386
00404 L4_ALWAYS_INLINE l4_msgtag_t
00405 l4_ipc(l4_cap_idx_t dest,
00406        l4_utcb_t *utcb,
00407        l4_umword_t flags,
00408        l4_umword_t slabel,
00409        l4_msgtag_t tag,
00410        l4_umword_t *rlabel,
00411        l4_timeout_t timeout) L4_NOTHROW;
00412
00427 L4_INLINE l4_msgtag_t
00428 l4_ipc_sleep(l4_timeout_t timeout) L4_NOTHROW;
00429
00442 L4_INLINE int
00443 l4_sndfpage_add(l4_fpage_t const snd_fpage, unsigned long snd_base,
00444                 l4_msgtag_t *tag) L4_NOTHROW;
00445
00446 /*
00447 * \internal
00448 * \ingroup l4_ipc_api
00449 */
00450 L4_INLINE int
00451 l4_sndfpage_add_u(l4_fpage_t const snd_fpage, unsigned long snd_base,
00452                  l4_msgtag_t *tag, l4_utcb_t *utcb) L4_NOTHROW;
00453
00454
00455 /*****
00456 * Implementations
00457 *****/
00458
00459 L4_INLINE long l4_ipc_to_errno(unsigned long ipc_error_code) L4_NOTHROW
00460 { return -(L4_EIPC_LO + ipc_error_code); }
00461

```

```

00462 L4_INLINE l4_msgtag_t
00463 l4_ipc_call(l4_cap_idx_t dest, l4_utcb_t *utcb,
00464             l4_msgtag_t tag,
00465             l4_timeout_t timeout) L4_NOTHROW
00466 {
00467     return l4_ipc(dest, utcb, L4_SYSF_CALL, 0, tag, 0, timeout);
00468 }
00469
00470 L4_INLINE l4_msgtag_t
00471 l4_ipc_reply_and_wait(l4_utcb_t *utcb, l4_msgtag_t tag,
00472                      l4_umword_t *label,
00473                      l4_timeout_t timeout) L4_NOTHROW
00474 {
00475     return l4_ipc(L4_INVALID_CAP, utcb, L4_SYSF_REPLY_AND_WAIT, 0, tag, label, timeout);
00476 }
00477
00478 L4_INLINE l4_msgtag_t
00479 l4_ipc_send_and_wait(l4_cap_idx_t dest, l4_utcb_t *utcb,
00480                     l4_msgtag_t tag,
00481                     l4_umword_t *src,
00482                     l4_timeout_t timeout) L4_NOTHROW
00483 {
00484     return l4_ipc(dest, utcb, L4_SYSF_SEND_AND_WAIT, 0, tag, src, timeout);
00485 }
00486
00487 L4_INLINE l4_msgtag_t
00488 l4_ipc_send(l4_cap_idx_t dest, l4_utcb_t *utcb,
00489            l4_msgtag_t tag,
00490            l4_timeout_t timeout) L4_NOTHROW
00491 {
00492     return l4_ipc(dest, utcb, L4_SYSF_SEND, 0, tag, 0, timeout);
00493 }
00494
00495 L4_INLINE l4_msgtag_t
00496 l4_ipc_wait(l4_utcb_t *utcb, l4_umword_t *src,
00497            l4_timeout_t timeout) L4_NOTHROW
00498 {
00499     l4_msgtag_t t;
00500     t.raw = 0;
00501     return l4_ipc(L4_INVALID_CAP, utcb, L4_SYSF_WAIT, 0, t, src, timeout);
00502 }
00503
00504 L4_INLINE l4_msgtag_t
00505 l4_ipc_receive(l4_cap_idx_t src, l4_utcb_t *utcb,
00506              l4_timeout_t timeout) L4_NOTHROW
00507 {
00508     l4_msgtag_t t;
00509     t.raw = 0;
00510     return l4_ipc(src, utcb, L4_SYSF_RECV, 0, t, 0, timeout);
00511 }
00512
00513 L4_INLINE l4_msgtag_t
00514 l4_ipc_sleep(l4_timeout_t timeout) L4_NOTHROW
00515 { return l4_ipc_receive(L4_INVALID_CAP, NULL, timeout); }
00516
00517 L4_INLINE l4_umword_t
00518 l4_ipc_error(l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW
00519 {
00520     if (!l4_msgtag_has_error(tag))
00521         return 0;
00522     return l4_utcb_tcr_u(utcb)->error & L4_IPC_ERROR_MASK;
00523 }
00524
00525 L4_INLINE long
00526 l4_error_u(l4_msgtag_t tag, l4_utcb_t *u) L4_NOTHROW
00527 {
00528     if (l4_msgtag_has_error(tag))
00529         return l4_ipc_to_errno(l4_utcb_tcr_u(u)->error & L4_IPC_ERROR_MASK);
00530
00531     return l4_msgtag_label(tag);
00532 }
00533
00534 L4_INLINE long
00535 l4_error(l4_msgtag_t tag) L4_NOTHROW
00536 {
00537     return l4_error_u(tag, l4_utcb());
00538 }
00539
00540
00541 L4_INLINE int l4_ipc_is_snd_error(l4_utcb_t *u) L4_NOTHROW
00542 { return (l4_utcb_tcr_u(u)->error & 1) == 0; }
00543
00544 L4_INLINE int l4_ipc_is_rcv_error(l4_utcb_t *u) L4_NOTHROW
00545 { return l4_utcb_tcr_u(u)->error & 1; }
00546
00547 L4_INLINE int l4_ipc_error_code(l4_utcb_t *u) L4_NOTHROW
00548 { return l4_utcb_tcr_u(u)->error & L4_IPC_ERROR_MASK; }

```

```

00549
00550
00551 /*
00552  * \internal
00553  * \ingroup l4_ipc_api
00554  */
00555 L4_INLINE int
00556 l4_sndfpage_add_u(l4_fpage_t const snd_fpage, unsigned long snd_base,
00557                  l4_msgtag_t *tag, l4_utcb_t *utcb) L4_NOTHROW
00558 {
00559     l4_msgregs_t *v = l4_utcb_mr_u(utcb);
00560     int i = l4_msgtag_words(*tag) + 2 * l4_msgtag_items(*tag);
00561
00562     if (i >= L4_UTCB_GENERIC_DATA_SIZE - 1)
00563         return -L4_ENOMEM;
00564
00565     v->mr[i] = snd_base | L4_ITEM_MAP | L4_ITEM_CONT;
00566     v->mr[i + 1] = snd_fpage.raw;
00567
00568     *tag = l4_msgtag(l4_msgtag_label(*tag), l4_msgtag_words(*tag),
00569                     l4_msgtag_items(*tag) + 1, l4_msgtag_flags(*tag));
00570     return 0;
00571 }
00572
00573 L4_INLINE int
00574 l4_sndfpage_add(l4_fpage_t const snd_fpage, unsigned long snd_base,
00575                 l4_msgtag_t *tag) L4_NOTHROW
00576 {
00577     return l4_sndfpage_add_u(snd_fpage, snd_base, tag, l4_utcb());
00578 }
00579
00580
00581 #endif /* ! __L4SYS__INCLUDE__L4API_FIASCO__IPC_H__ */

```

16.305 arm/l4f/l4/sys/ipc.h File Reference

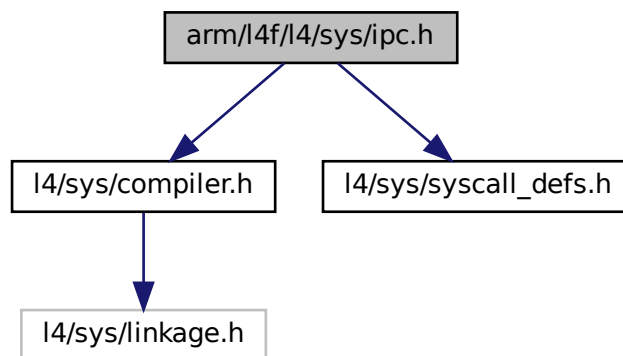
L4 IPC System Calls, ARM.

```

#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.

16.305.1 Detailed Description

[L4 IPC System Calls, ARM.](#)

Definition in file [ipc.h](#).

16.306 ipc.h

```

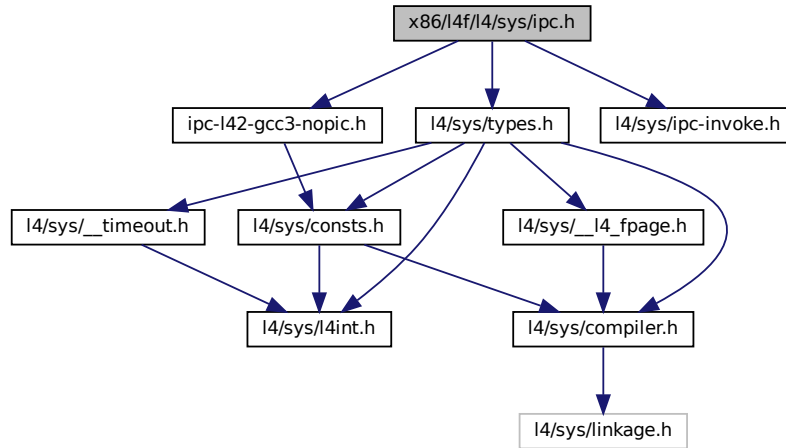
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * 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_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_mword_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      "mov 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__

```

16.307 x86/l4/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:
```



16.307.1 Detailed Description

[L4 IPC System Calls](#), x86.

Definition in file [ipc.h](#).

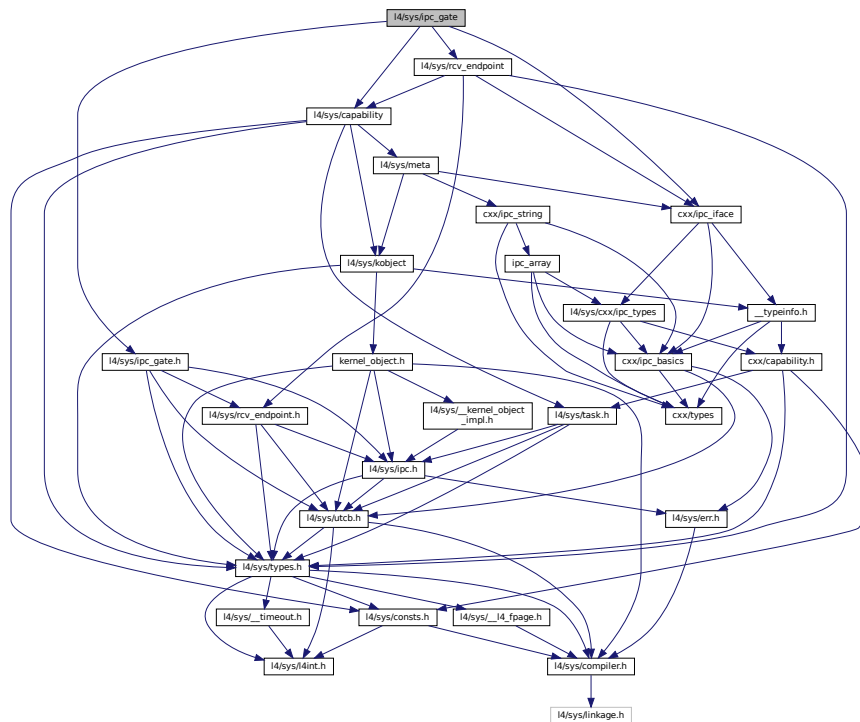
16.308 ipc.h

```
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  * Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  * 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__ */
```

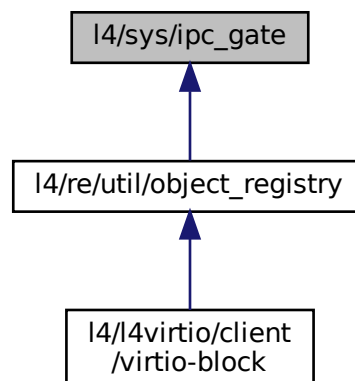
16.309 l4/sys/ipc_gate File Reference

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.

Namespaces

- [L4](#)
L4 low-level kernel interface.

16.309.1 Detailed Description

The C++ IPC gate interface.

Definition in file [ipc_gate](#).

16.310 ipc_gate

```

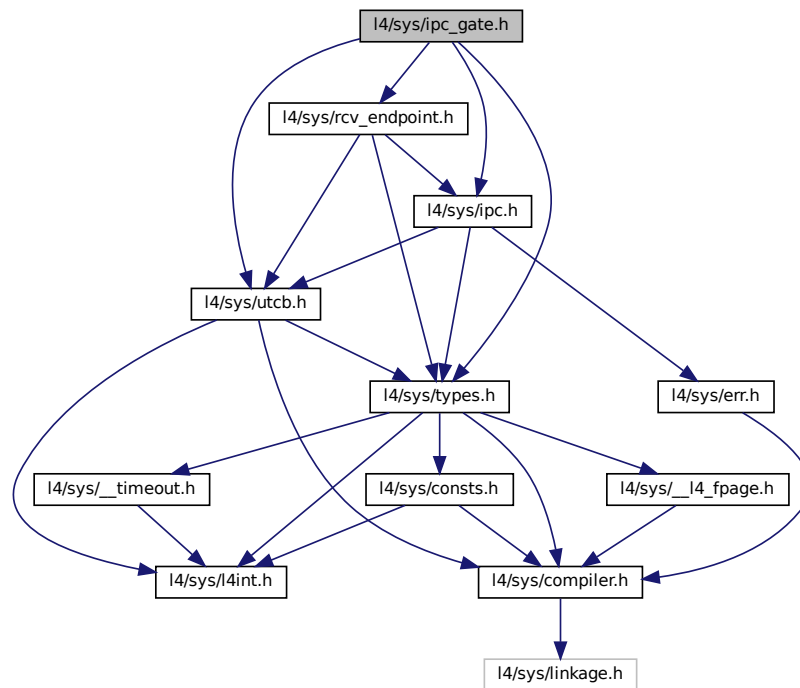
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2009-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed 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
00061 class L4_EXPORT Ipc_gate :
00062     public Kobject_t<Ipc_gate, Rcv_endpoint, L4_PROTO_KOBJECT,
00063         Type_info::Demand_t<1> >
00064 {
00065 public:
00073     L4_INLINE_RPC_OP(L4_IPC_GATE_GET_INFO_OP,
00074         l4_msgtag_t, get_infos, (l4_umword_t *label));
00075
00076     typedef L4::Typeid::Rpcsys<bind_thread_t, get_infos_t> Rpcsys;
00077 };
00078
00079 }
```

16.311 l4/sys/ipc_gate.h File Reference

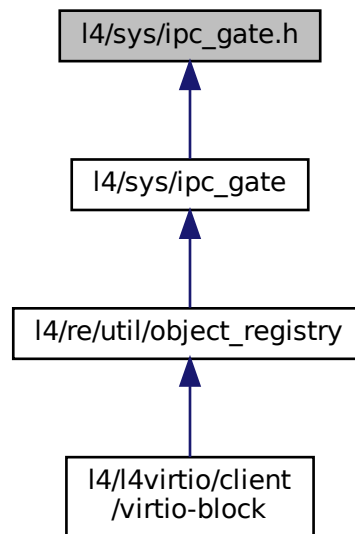
The C IPC gate interface.

```
#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_bind_thread](#) ([l4_cap_idx_t](#) gate, [l4_cap_idx_t](#) thread, [l4_umword_t](#) label)
Bind the IPC gate to a thread.
- [l4_msgtag_t l4_ipc_gate_get_infos](#) ([l4_cap_idx_t](#) gate, [l4_umword_t](#) *label)
Get information about the IPC-gate.

16.311.1 Detailed Description

The C IPC gate interface.

IPC gates are used to create secure communication channels between threads. An IPC gate object can be created using the [Factory](#) interface. With [l4_ipc_gate_bind_thread\(\)](#) a thread is bound to an IPC gate which then receives all messages sent to that IPC gate.

The [l4_ipc_gate_bind_thread\(\)](#) call allows to assign each IPC gate a kernel protected, machine-word sized payload called a *label*. It securely identifies the gate. The lower two bits of the *label* can be used to encode rights bits. The kernel combines these bits with the capability rights, so a programmer usually should not pick the lower two bits for the *label*. The *label* is only visible in the task which is running the thread the IPC gate was bound to and cannot be altered by the sender.

Definition in file [ipc_gate.h](#).

16.312 ipc_gate.h

```

00001
00018 /*
00019  * (c) 2009-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00020  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00021  *      economic rights: Technische Universität Dresden (Germany)
00022  *
00023  * This file is part of TUD:OS and distributed under the terms of the
00024  * GNU General Public License 2.
00025  * Please see the COPYING-GPL-2 file for details.
00026  *
00027  * As a special exception, you may use this file as part of a free software
00028  * library without restriction. Specifically, if other files instantiate
00029  * templates or use macros or inline functions from this file, or you compile
00030  * this file and link it with other files to produce an executable, this
00031  * file does not by itself cause the resulting executable to be covered by
00032  * the GNU General Public License. This exception does not however
00033  * invalidate any other reasons why the executable file might be covered by
00034  * the GNU General Public License.
00035  */
00036 #pragma once
00037
00038 #include <l4/sys/utcb.h>
00039 #include <l4/sys/types.h>
00040 #include <l4/sys/rcv_endpoint.h>
00041
00058 L4_INLINE l4_msgtag_t
00059 l4_ipc_gate_bind_thread(l4_cap_idx_t gate, l4_cap_idx_t thread,
00060                        l4_umword_t label)
00061 {
00062     L4_DEPRECATED("Use l4_rcv_ep_bind_thread().");
00063 }
00064
00067 L4_INLINE l4_msgtag_t
00068 l4_ipc_gate_bind_thread_u(l4_cap_idx_t gate, l4_cap_idx_t thread,
00069                          l4_umword_t label, l4_utcb_t *utcb)
00070 {
00071     L4_DEPRECATED("Use l4_rcv_ep_bind_thread_u().");
00072 }
00073
00078 L4_INLINE l4_msgtag_t
00079 l4_ipc_gate_get_infos(l4_cap_idx_t gate, l4_umword_t *label);
00080
00085 L4_INLINE l4_msgtag_t
00086 l4_ipc_gate_get_infos_u(l4_cap_idx_t gate, l4_umword_t *label, l4_utcb_t *utcb);
00087
00094 enum l4_ipc_gate_ops
00095 {
00096     L4_IPC_GATE_BIND_OP      = 0x10,
00097     L4_IPC_GATE_GET_INFO_OP  = 0x11,
00098 };
00099
00100
00101 /* IMPLEMENTATION -----*/
00102
00103 #include <l4/sys/ipc.h>
00104
00105 L4_INLINE l4_msgtag_t
00106 l4_ipc_gate_bind_thread_u(l4_cap_idx_t gate,
00107                          l4_cap_idx_t thread, l4_umword_t label,
00108                          l4_utcb_t *utcb)
00109 {
00110     return l4_rcv_ep_bind_thread_u(gate, thread, label, utcb);
00111 }
00112
00113 L4_INLINE l4_msgtag_t
00114 l4_ipc_gate_get_infos_u(l4_cap_idx_t gate, l4_umword_t *label, l4_utcb_t *utcb)
00115 {
00116     l4_msgtag_t tag;
00117     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00118     m->mr[0] = L4_IPC_GATE_GET_INFO_OP;
00119     tag = l4_ipc_call(gate, utcb, l4_msgtag(L4_PROTO_KOBJECT, 1, 0, 0),
00120                     L4_IPC_NEVER);
00121     if (!l4_msgtag_has_error(tag) && l4_msgtag_label(tag) >= 0)
00122         *label = m->mr[0];
00123     return tag;
00124 }
00125
00126
00127
00128
00129 L4_INLINE l4_msgtag_t
00130 l4_ipc_gate_bind_thread(l4_cap_idx_t gate, l4_cap_idx_t thread,
00131                        l4_umword_t label)
00132 {
00133     return l4_rcv_ep_bind_thread_u(gate, thread, label, l4_utcb());
00134 }
00135
00136 L4_INLINE l4_msgtag_t
00137 l4_ipc_gate_get_infos(l4_cap_idx_t gate, l4_umword_t *label)

```

16.313 I4/sys/irq File Reference

```
#include <l4/sys/icu.h>
#include <l4/sys/irq.h>
#include <l4/sys/capability>
#include <l4/sys/rcv_endpoint>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/ipc_types>
```

Include dependency graph for irq:



Data Structures

- class [L4::Irq_eoi](#)
Interface for sending an acknowledge message to an object.
- struct [L4::Triggerable](#)
Interface that allows an object to be triggered by some source.
- class [L4::Irq](#)
C++ [Irq](#) interface.
- struct [L4::Irq_mux](#)
IRQ multiplexer for shared IRQs.
- class [L4::Icu](#)
C++ [Icu](#) interface.
- class [L4::Icu::Info](#)
This class encapsulates information about an ICU.

Namespaces

- [L4](#)
[L4](#) low-level kernel interface.

16.313.1 Detailed Description

C++ Irq interface.

Definition in file [irq](#).

16.314 irq

```

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/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
00043 class Irq_eoi : public Kobject_0t<Irq_eoi, L4::PROTO_EMPTY>
00044 {
00045 public:
00065     l4_msgtag_t unmask(unsigned irqnum, l4_umword_t *label = 0,
```

```

00066         l4_timeout_t to = L4_IPC_NEVER,
00067         l4_utcb_t *utcb = l4_utcb()) noexcept
00068     {
00069         return l4_icu_control_u(cap(), irqnum, L4_ICU_CTL_UNMASK, label, to, utcb);
00070     }
00071 };
00072
00073 struct Triggerable : Kobject_t<Triggerable, Irq_eoi, L4_PROTO_IRQ>
00074 {
00075     l4_msgtag_t trigger(l4_utcb_t *utcb = l4_utcb()) noexcept
00076     { return l4_irq_trigger_u(cap(), utcb); }
00077 };
00078
00079 class Irq : public Kobject_2t<Irq, Triggerable, Rcv_endpoint, L4_PROTO_IRQ_SENDER>
00080 {
00081 public:
00082     using Triggerable::unmask;
00083
00084     l4_msgtag_t detach(l4_utcb_t *utcb = l4_utcb()) noexcept
00085     { return l4_irq_detach_u(cap(), utcb); }
00086
00087     l4_msgtag_t receive(l4_timeout_t timeout = L4_IPC_NEVER,
00088                        l4_utcb_t *utcb = l4_utcb()) noexcept
00089     { return l4_irq_receive_u(cap(), timeout, utcb); }
00090
00091     l4_msgtag_t wait(l4_umword_t *label, l4_timeout_t timeout = L4_IPC_NEVER,
00092                    l4_utcb_t *utcb = l4_utcb()) noexcept
00093     { return unmask(-1, label, timeout, utcb); }
00094
00095     l4_msgtag_t unmask(l4_utcb_t *utcb = l4_utcb()) noexcept
00096     { return unmask(-1, 0, L4_IPC_NEVER, utcb); }
00097 };
00098
00099 struct Irq_mux : Kobject_t<Irq_mux, Triggerable, L4_PROTO_IRQ_MUX>
00100 {
00101     l4_msgtag_t chain(Cap<Triggerable> const &slave,
00102                     l4_utcb_t *utcb = l4_utcb()) noexcept
00103     { return l4_irq_mux_chain_u(cap(), slave.cap(), utcb); }
00104 };
00105
00106 class Icu :
00107     public Kobject_t<Icu, Irq_eoi, L4_PROTO_IRQ,
00108                     Type_info::Demand_t<1> >
00109 {
00110 public:
00111     enum Mode
00112     {
00113         F_none           = L4_IRQ_F_NONE,
00114         F_level_high     = L4_IRQ_F_LEVEL_HIGH,
00115         F_level_low      = L4_IRQ_F_LEVEL_LOW,
00116         F_pos_edge       = L4_IRQ_F_POS_EDGE,
00117         F_neg_edge       = L4_IRQ_F_NEG_EDGE,
00118         F_both_edge      = L4_IRQ_F_BOTH_EDGE,
00119         F_mask           = L4_IRQ_F_MASK,
00120
00121         F_set_wakeup     = L4_IRQ_F_SET_WAKEUP,
00122         F_clear_wakeup  = L4_IRQ_F_CLEAR_WAKEUP,
00123     };
00124
00125     enum Flags
00126     {
00127         F_msi = L4_ICU_FLAG_MSI
00128     };
00129
00130     class Info : public l4_icu_info_t
00131     {
00132     public:
00133         bool supports_msi() const noexcept { return features & F_msi; }
00134     };
00135
00136     l4_msgtag_t bind(unsigned irqnum, L4::Cap<Triggerable> irq,
00137                    l4_utcb_t *utcb = l4_utcb()) noexcept
00138     { return l4_icu_bind_u(cap(), irqnum, irq.cap(), utcb); }
00139
00140     L4_RPC_NF_OP(
00141         L4_ICU_OP_BIND,
00142         l4_msgtag_t, bind, (l4_umword_t irqnum, Ipc::Cap<Irq> irq)
00143     );
00144
00145     l4_msgtag_t unbind(unsigned irqnum, L4::Cap<Triggerable> irq,
00146                      l4_utcb_t *utcb = l4_utcb()) noexcept
00147     { return l4_icu_unbind_u(cap(), irqnum, irq.cap(), utcb); }
00148
00149     L4_RPC_NF_OP(
00150         L4_ICU_OP_UNBIND,

```

```

00309     l4_msgtag_t, unbind, (l4_umword_t irqnum, l4::Cap<Irq> irq)
00310 );
00311
00320 l4_msgtag_t info(l4_icu_info_t *info, l4_utcb_t *utcb = l4_utcb()) noexcept
00321 { return l4_icu_info_u(cap(), info, utcb); }
00322
00323 struct _Info { l4_umword_t features, nr_irqs, nr_msis; };
00324 L4_RPC_NF_OP(L4_ICU_OP_INFO, l4_msgtag_t, info, (_Info *info));
00325
00338 L4_INLINE_RPC_OP(L4_ICU_OP_MSI_INFO,
00339     l4_msgtag_t, msi_info, (l4_umword_t irqnum, l4_uint64_t source,
00340     l4_icu_msi_info_t *msi_info));
00341
00345 l4_msgtag_t control(unsigned irqnum, unsigned op, l4_umword_t *label,
00346     l4_timeout_t to, l4_utcb_t *utcb = l4_utcb()) noexcept
00347 { return l4_icu_control_u(cap(), irqnum, op, label, to, utcb); }
00348
00363 l4_msgtag_t mask(unsigned irqnum,
00364     l4_umword_t *label = 0,
00365     l4_timeout_t to = L4_IPC_NEVER,
00366     l4_utcb_t *utcb = l4_utcb()) noexcept
00367 { return l4_icu_mask_u(cap(), irqnum, label, to, utcb); }
00368
00369 L4_RPC_NF_OP(
00370     L4_ICU_OP_MASK,
00371     l4_msgtag_t, mask, (l4_umword_t irqnum),
00372     L4::Ipc::Send_only
00373 );
00374
00375
00376 L4_RPC_NF_OP(
00377     L4_ICU_OP_UNMASK,
00378     l4_msgtag_t, unmask, (l4_umword_t irqnum),
00379     L4::Ipc::Send_only
00380 );
00381
00391 l4_msgtag_t set_mode(unsigned irqnum, l4_umword_t mode,
00392     l4_utcb_t *utcb = l4_utcb()) noexcept
00393 { return l4_icu_set_mode_u(cap(), irqnum, mode, utcb); }
00394
00395 L4_RPC_NF_OP(
00396     L4_ICU_OP_SET_MODE,
00397     l4_msgtag_t, set_mode, (l4_umword_t irqnum, l4_umword_t mode)
00398 );
00399
00400 typedef L4::Typeid::Rpcsys<
00401     bind_t, unbind_t, info_t, msi_info_t, unmask_t, mask_t, set_mode_t
00402 > Rpcsys;
00403 };
00404
00405 }

```

16.315 l4/sys/kernel_object.h File Reference

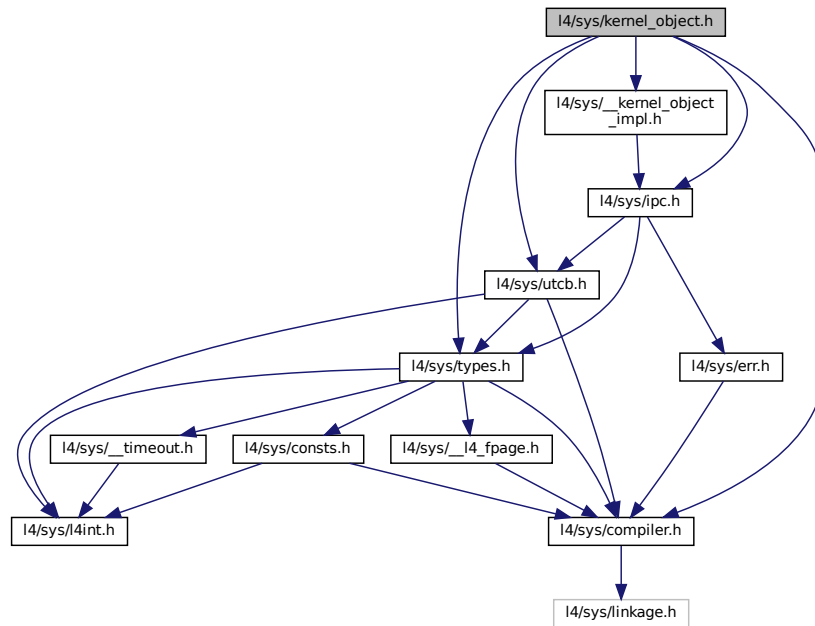
Kernel object system calls.

```

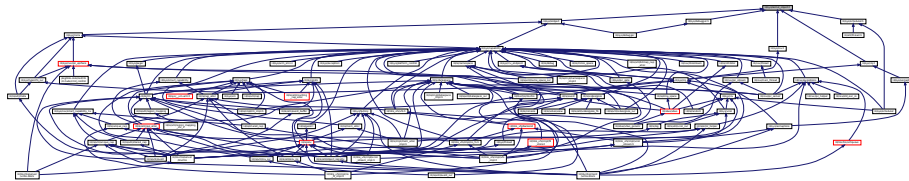
#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
#include <l4/sys/utcb.h>
#include <l4/sys/__kernel_object_impl.h>
#include <l4/sys/ipc.h>

```

Include dependency graph for `kernel_object.h`:



This graph shows which files directly or indirectly include this file:



16.315.1 Detailed Description

Kernel object system calls.

Definition in file [kernel_object.h](#).

16.316 kernel_object.h

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *           Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *           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
  
```

```

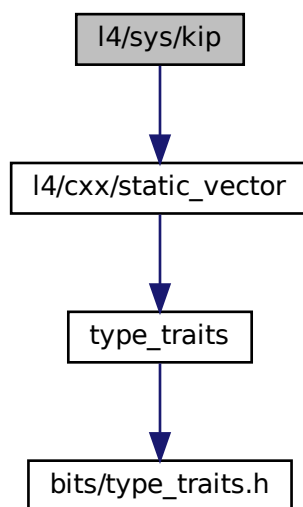
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does 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__ */

```

16.317 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

- [L4](#)

[L4](#) low-level kernel interface.

16.317.1 Detailed Description

L4::Kip class, memory descriptors.

Author

Alexander Warg alexander.warg@os.inf.tu-dresden.de

Definition in file [kip](#).

16.318 kip

```

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
00080         private:
00081             unsigned long _l, _h;
00082
00083             static unsigned long &memory_info(void *kip) noexcept
00084             { return *((unsigned long *)kip + 21); }
00085
00086             static unsigned long memory_info(void const *kip) noexcept
00087             { return *((unsigned long const *)kip + 21); }
00088
00089         public:
00097             static Mem_desc *first(void *kip) noexcept
00098             {
00099                 return (Mem_desc *)((char *)kip
00100                     + (memory_info(kip) » ((sizeof(unsigned long) / 2) * 8)));
00101             }
00102
00103             static Mem_desc const *first(void const *kip) noexcept
00104             {
00105                 return (Mem_desc const *)((char const *)kip
00106                     + (memory_info(kip) » ((sizeof(unsigned long) / 2) * 8)));
00107             }
00108
00116             static unsigned long count(void const *kip) noexcept
00117             {
00118                 return memory_info(kip)
00119                     & ((1UL « ((sizeof(unsigned long) / 2) * 8)) - 1);
00120             }
00121
00128             static void count(void *kip, unsigned count) noexcept
00129             {
00130                 unsigned long &mi = memory_info(kip);
00131                 mi = (mi & ~((1UL « ((sizeof(unsigned long) / 2) * 8)) - 1)) | count;
00132             }
00133
00139             static inline cxx::static_vector<Mem_desc const> all(void const *kip)

```

```

00140     {
00141         return cxx::static_vector<Mem_desc const>(Mem_desc::first(kip),
00142                                                    Mem_desc::count(kip));
00143     }
00144
00150     static inline cxx::static_vector<Mem_desc> all(void *kip)
00151     {
00152         return cxx::static_vector<Mem_desc>(Mem_desc::first(kip),
00153                                              Mem_desc::count(kip));
00154     }
00155
00166     Mem_desc(unsigned long start, unsigned long end,
00167              Mem_type t, unsigned char st = 0, bool virt = false) noexcept
00168     : _l((start & ~0x3ffUL) | (t & 0x0f) | ((st << 4) & 0x0f0)
00169         | (virt ? 0x0200 : 0x0)), _h(end | 0x3ffUL)
00170     {}
00171
00177     unsigned long start() const noexcept { return _l & ~0x3ffUL; }
00178
00184     unsigned long end() const noexcept { return _h | 0x3ffUL; }
00185
00191     unsigned long size() const noexcept { return end() + 1 - start(); }
00192
00198     Mem_type type() const noexcept { return (Mem_type)(_l & 0x0f); }
00199
00205     unsigned char sub_type() const noexcept { return (_l >> 4) & 0x0f; }
00206
00213     unsigned is_virtual() const noexcept { return _l & 0x200; }
00214
00224     void set(unsigned long start, unsigned long end,
00225             Mem_type t, unsigned char st = 0, bool virt = false) noexcept
00226     {
00227         _l = (start & ~0x3ffUL) | (t & 0x0f) | ((st << 4) & 0x0f0)
00228             | (virt?0x0200:0x0);
00229
00230         _h = end | 0x3ffUL;
00231     }
00232
00233 };
00234 };
00235 };
00236
00237 #endif

```

16.319 l4/sys/ktrace.h File Reference

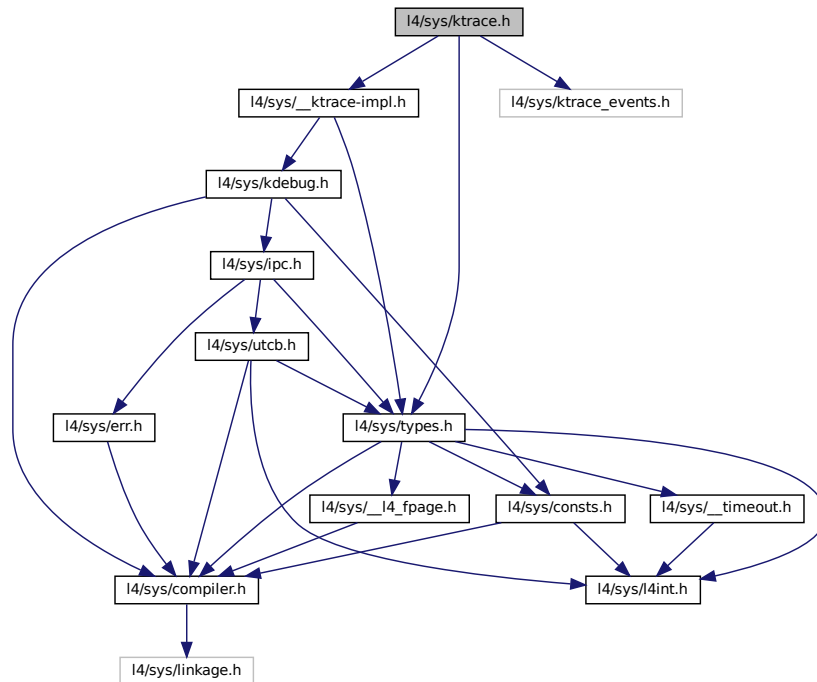
[L4](#) kernel event tracing.

```

#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.

16.319.1 Detailed Description

[L4](#) kernel event tracing.

Definition in file [ktrace.h](#).

16.320 ktrace.h

```

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
00040 L4_INLINE l4_umword_t
00041 fiasco_tbuf_log(const char *text);
00042
00054 L4_INLINE l4_umword_t
00055 fiasco_tbuf_log_3val(const char *text, l4_umword_t v1, l4_umword_t v2, l4_umword_t v3);
00056
00064 L4_INLINE l4_umword_t
00065 fiasco_tbuf_log_binary(const unsigned char *data);
00066
00071 L4_INLINE void
00072 fiasco_tbuf_clear(void);
00073
00078 L4_INLINE void
00079 fiasco_tbuf_dump(void);
00080
00081 #include <l4/sys/__ktrace-impl.h>
00082
00083 #endif

```

16.321 l4/sys/l4int.h File Reference

Fixed sized integer types, generic version.

This graph shows which files directly or indirectly include this file:



Typedefs

- typedef signed char [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.

16.321.1 Detailed Description

Fixed sized integer types, generic version.

Definition in file [l4int.h](#).

16.322 l4int.h

```

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__ */

```

16.323 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.

16.323.1 Detailed Description

Fixed sized integer types, arm version.

Definition in file [l4int.h](#).

16.324 l4int.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_next <l4/sys/l4int.h>
00027
00032
00033 #define L4_MWORD_BITS          32
00035 typedef unsigned int           l4_size_t;
00036 typedef signed int             l4_ssize_t;
00038

```

16.325 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.

16.325.1 Detailed Description

Fixed sized integer types, amd64 version.

Definition in file [l4int.h](#).

16.326 l4int.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  *      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
00033
00034  #define L4_MWORD_BITS      64
00036  typedef unsigned long      l4_size_t;
00037  typedef signed long        l4_ssize_t;
00039

```

16.327 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.

16.327.1 Detailed Description

Fixed sized integer types, x86 version.

Definition in file [l4int.h](#).

16.328 l4int.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_next <l4/sys/l4int.h>
00027
00032
00033 #define L4_MWORD_BITS      32
00035 typedef unsigned int      l4_size_t;
00036 typedef signed int        l4_ssize_t;
00038

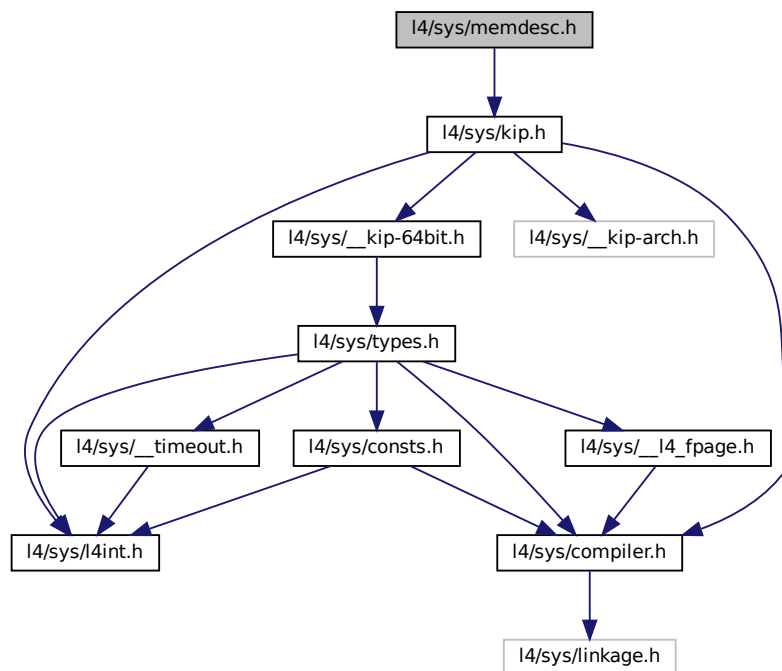
```

16.329 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.

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.

16.329.1 Detailed Description

Memory description functions.

Definition in file [memdesc.h](#).

16.330 memdesc.h

```

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
00066
00074 typedef struct l4_kernel_info_mem_desc_t
00075 {
00077     l4_umword_t l;
00079     l4_umword_t h;
00080 } l4_kernel_info_mem_desc_t;
00081
00082
00087 L4_INLINE
00088 l4_kernel_info_mem_desc_t *
00089 l4_kernel_info_get_mem_descs(l4_kernel_info_t *kip) L4_NOTHROW;
00090
00097 L4_INLINE
00098 unsigned
00099 l4_kernel_info_get_num_mem_descs(l4_kernel_info_t *kip) L4_NOTHROW;
00100
00112 L4_INLINE
00113 void
00114 l4_kernel_info_set_mem_desc(l4_kernel_info_mem_desc_t *md,
00115                             l4_addr_t start,
00116                             l4_addr_t end,
00117                             unsigned type,
00118                             unsigned virt,
00119                             unsigned sub_type) L4_NOTHROW;
00120
00127 L4_INLINE
00128 l4_umword_t
00129 l4_kernel_info_get_mem_desc_start(l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00130
00137 L4_INLINE
00138 l4_umword_t
00139 l4_kernel_info_get_mem_desc_end(l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00140
00147 L4_INLINE
00148 l4_umword_t
00149 l4_kernel_info_get_mem_desc_type(l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00150
00160 L4_INLINE
00161 l4_umword_t
00162 l4_kernel_info_get_mem_desc_subtype(l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00163
00170 L4_INLINE
00171 l4_umword_t
00172 l4_kernel_info_get_mem_desc_is_virtual(l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00173
00174 /*****
00175  * Implementations
00176  *****/
00177
00178 L4_INLINE
00179 l4_kernel_info_mem_desc_t *
00180 l4_kernel_info_get_mem_descs(l4_kernel_info_t *kip) L4_NOTHROW
00181 {
00182     return (l4_kernel_info_mem_desc_t *) (((l4_addr_t)kip)
00183         + (kip->mem_info » (sizeof(l4_umword_t) * 4)));
00184 }
00185
00186 L4_INLINE
00187 unsigned
00188 l4_kernel_info_get_num_mem_descs(l4_kernel_info_t *kip) L4_NOTHROW
00189 {
00190     return kip->mem_info & ((1UL « (sizeof(l4_umword_t)*4)) -1);
00191 }
00192
00193 L4_INLINE
00194 void
00195 l4_kernel_info_set_mem_desc(l4_kernel_info_mem_desc_t *md,
00196                             l4_addr_t start,
00197                             l4_addr_t end,
00198                             unsigned type,
00199                             unsigned virt,
00200                             unsigned sub_type) L4_NOTHROW
00201 {
00202     md->l = (start & ~0x3fffUL) | (type & 0x0f) | ((sub_type < 4) & 0x0f0)
00203         | (virt ? 0x200 : 0x0);
00204     md->h = end;
00205 }
00206
00207
00208 L4_INLINE
00209 l4_umword_t
00210 l4_kernel_info_get_mem_desc_start(l4_kernel_info_mem_desc_t *md) L4_NOTHROW

```

```

00211 {
00212     return md->l & ~0x3ffUL;
00213 }
00214
00215 L4_INLINE
00216 l4_umword_t
00217 l4_kernel_info_get_mem_desc_end(l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00218 {
00219     return md->h | 0x3ffUL;
00220 }
00221
00222 L4_INLINE
00223 l4_umword_t
00224 l4_kernel_info_get_mem_desc_type(l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00225 {
00226     return md->l & 0xf;
00227 }
00228
00229 L4_INLINE
00230 l4_umword_t
00231 l4_kernel_info_get_mem_desc_subtype(l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00232 {
00233     return (md->l & 0xf0) >> 4;
00234 }
00235
00236 L4_INLINE
00237 l4_umword_t
00238 l4_kernel_info_get_mem_desc_is_virtual(l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00239 {
00240     return md->l & 0x200;
00241 }
00242
00243 #endif /* ! __L4SYS__MEMDESC_H__ */

```

16.331 l4/sys/meta File Reference

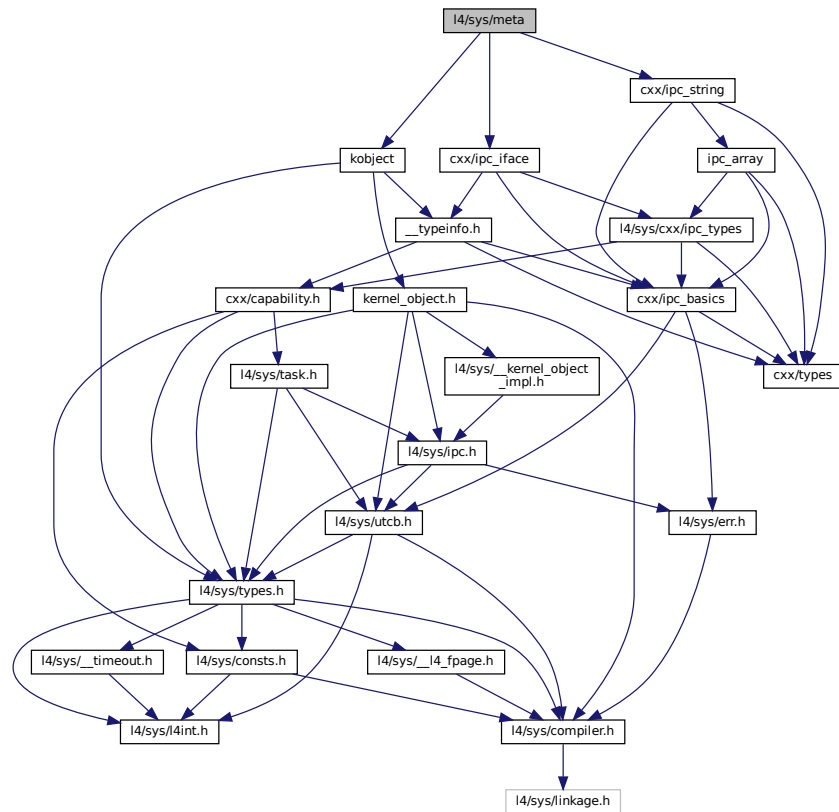
Meta interface for getting dynamic type information about objects behind capabilities.

```

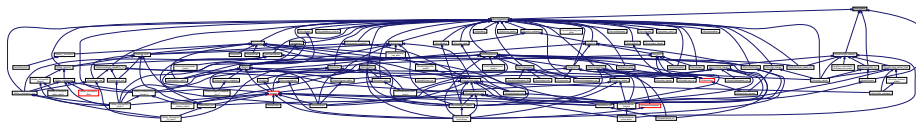
#include "kobject"
#include "cxx/ipc_iface"
#include "cxx/ipc_string"

```

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

- [L4](#)

L4 low-level kernel interface.

16.331.1 Detailed Description

Meta interface for getting dynamic type information about objects behind capabilities.

Definition in file [meta](#).

16.332 meta

```

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 "kobject"
00022 #include "cxx/ipc_iface"
00023 #include "cxx/ipc_string"
00024
00025 namespace L4 {
00026
00027 class Meta : public Kobject_t<Meta, Kobject, L4_PROTO_META>
00028 {
00029 public:
00030     L4_INLINE_RPC(l4_msgtag_t, num_interfaces, ());
00031     L4_INLINE_RPC(l4_msgtag_t, interface, (l4_umword_t idx, long *proto,
00032                                           L4::Ipc::String<char> *name));
00033     L4_INLINE_RPC(l4_msgtag_t, supports, (l4_mword_t protocol));
00034     typedef L4::Typeid::Rpc<num_interfaces_t, interface_t, supports_t> Rpc;
00035 };
00036
00037 }
```

16.333 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>
```

[illegible]

- class `L4::lo_pager`
lo_pager interface.
- class `L4::Pager`
Pager interface including the *lo_pager* interface.

- L4

L4 low-level kernel interface.

16.333.1 Detailed Description

Pager and `Io_pager` C++ interface.

Definition in file [pager](#).

16.334 pager

```

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> Rpccs;
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> Rpccs;
00140 };
00141
00142 }
```

16.335 l4/sys/platform_control File Reference

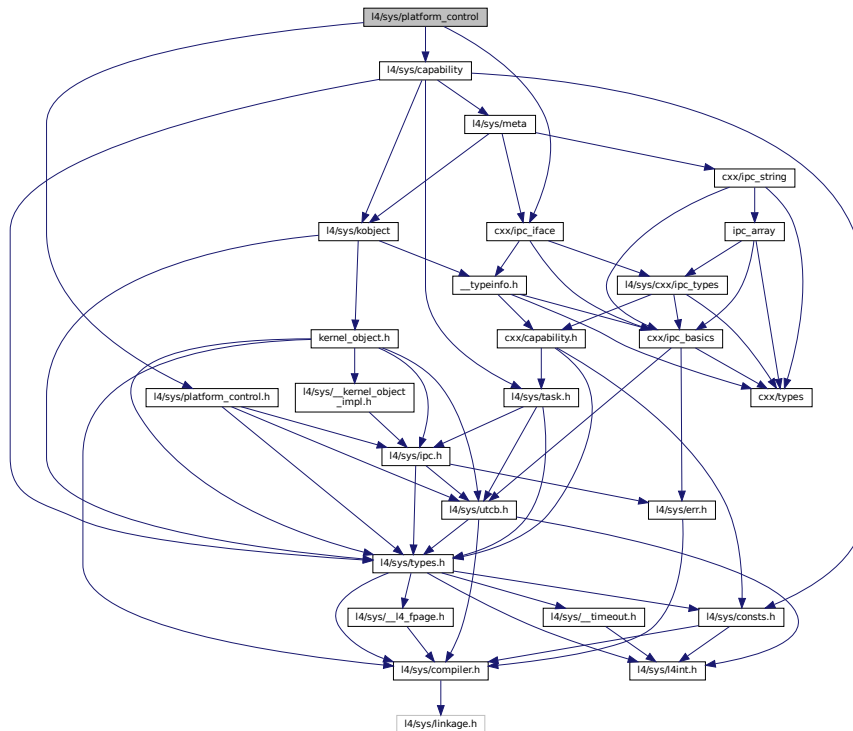
Platform control object.

```

#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.

Namespaces

- [L4](#)
L4 low-level kernel interface.

16.335.1 Detailed Description

Platform control object.

Definition in file [platform_control](#).

16.336 platform_control

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Steffen Liebergeld <steffen.liebergeld@kernkonzept.com>
00004  *      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/sys/capability>
00025 #include <l4/sys/platform_control.h>
00026 #include <l4/sys/cxx/ipc_iface>
00027
00028 namespace L4 {
00029
00030 class L4_EXPORT Platform_control
00031 : public Kobject_t<Platform_control, Kobject, L4_PROTO_PLATFORM_CTL>
00032 {
00033 public:
00034     enum Opcode
00035     {
00036         Suspend = L4_PLATFORM_CTL_SYS_SUSPEND_OP,
00037         Shutdown = L4_PLATFORM_CTL_SYS_SHUTDOWN_OP,
00038         Cpu_enable = L4_PLATFORM_CTL_CPU_ENABLE_OP,
00039         Cpu_disable = L4_PLATFORM_CTL_CPU_DISABLE_OP
00040     };
00041
00042     L4_INLINE_RPC_OP(L4_PLATFORM_CTL_SYS_SUSPEND_OP,
00043                     l4_msgtag_t, system_suspend, (l4_umword_t extras));
00044
00045     L4_INLINE_RPC_OP(L4_PLATFORM_CTL_SYS_SHUTDOWN_OP,
00046                     l4_msgtag_t, system_shutdown, (l4_umword_t reboot));
00047
00048     L4_INLINE_RPC_OP(L4_PLATFORM_CTL_CPU_ENABLE_OP,
00049                     l4_msgtag_t, cpu_enable, (l4_umword_t phys_id));
00050
00051     L4_INLINE_RPC_OP(L4_PLATFORM_CTL_CPU_DISABLE_OP,
00052                     l4_msgtag_t, cpu_disable, (l4_umword_t phys_id));
00053
00054     typedef L4::Typeid::Rpcsys<system_suspend_t, system_shutdown_t,
00055                                cpu_enable_t, cpu_disable_t> Rpcsys;
00056 };
00057
00058 }
00059
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

```

16.337 l4/sys/platform_control.h File Reference

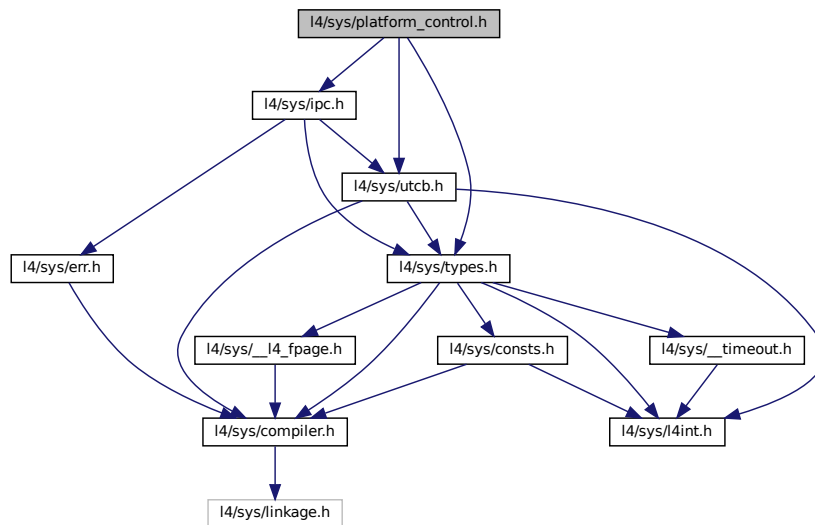
Platform control object.

```

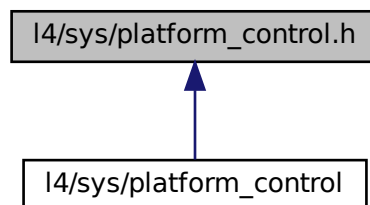
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>

```


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_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_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.

16.337.1 Detailed Description

Platform control object.

Definition in file [platform_control.h](#).

16.338 platform_control.h

```

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
00051 L4_INLINE l4_msgtag_t
00052 l4_platform_ctl_system_suspend(l4_cap_idx_t pfc,
00053                                l4_umword_t extras) L4_NOTHROW;
00054
00058 L4_INLINE l4_msgtag_t
00059 l4_platform_ctl_system_suspend_u(l4_cap_idx_t pfc,
00060                                  l4_umword_t extras,
00061                                  l4_utcb_t *utcb) L4_NOTHROW;
00062
00063
00072 L4_INLINE l4_msgtag_t
00073 l4_platform_ctl_system_shutdown(l4_cap_idx_t pfc,
00074                                 l4_umword_t reboot) L4_NOTHROW;
00075
00079 L4_INLINE l4_msgtag_t
00080 l4_platform_ctl_system_shutdown_u(l4_cap_idx_t pfc,
00081                                   l4_umword_t reboot,
00082                                   l4_utcb_t *utcb) L4_NOTHROW;
00083
00092 L4_INLINE l4_msgtag_t
00093 l4_platform_ctl_cpu_enable(l4_cap_idx_t pfc,
00094                            l4_umword_t phys_id) L4_NOTHROW;
00095
00099 L4_INLINE l4_msgtag_t
00100 l4_platform_ctl_cpu_enable_u(l4_cap_idx_t pfc,
00101                              l4_umword_t phys_id,
00102                              l4_utcb_t *utcb) L4_NOTHROW;
00103
00112 L4_INLINE l4_msgtag_t
00113 l4_platform_ctl_cpu_disable(l4_cap_idx_t pfc,
00114                             l4_umword_t phys_id) L4_NOTHROW;
00115
00119 L4_INLINE l4_msgtag_t
00120 l4_platform_ctl_cpu_disable_u(l4_cap_idx_t pfc,
00121                               l4_umword_t phys_id,

```

```

00122                                     l4_utcb_t *utcb) L4_NOTHROW;
00123 /* ends l4_platform_control_api group */
00125
00126
00135 enum L4_platform_ctl_ops
00136 {
00137     L4_PLATFORM_CTL_SYS_SUSPEND_OP = 0UL,
00138     L4_PLATFORM_CTL_SYS_SHUTDOWN_OP = 1UL,
00139     L4_PLATFORM_CTL_CPU_ENABLE_OP = 3UL,
00140     L4_PLATFORM_CTL_CPU_DISABLE_OP = 4UL,
00141 };
00142
00147 enum L4_platform_ctl_proto
00148 {
00154     L4_PROTO_PLATFORM_CTL = 0
00155 };
00156
00157 /* IMPLEMENTATION -----*/
00158
00159 #include <l4/sys/ipc.h>
00160
00161 L4_INLINE l4_msgtag_t
00162 l4_platform_ctl_system_suspend_u(l4_cap_idx_t pfc,
00163                                 l4_umword_t extras,
00164                                 l4_utcb_t *utcb) L4_NOTHROW
00165 {
00166     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00167     v->mr[0] = L4_PLATFORM_CTL_SYS_SUSPEND_OP;
00168     v->mr[1] = extras;
00169     return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 2, 0, 0),
00170                       L4_IPC_NEVER);
00171 }
00172
00173 L4_INLINE l4_msgtag_t
00174 l4_platform_ctl_system_shutdown_u(l4_cap_idx_t pfc,
00175                                  l4_umword_t reboot,
00176                                  l4_utcb_t *utcb) L4_NOTHROW
00177 {
00178     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00179     v->mr[0] = L4_PLATFORM_CTL_SYS_SHUTDOWN_OP;
00180     v->mr[1] = reboot;
00181     return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 2, 0, 0),
00182                       L4_IPC_NEVER);
00183 }
00184
00185
00186 L4_INLINE l4_msgtag_t
00187 l4_platform_ctl_system_suspend(l4_cap_idx_t pfc,
00188                                l4_umword_t extras) L4_NOTHROW
00189 {
00190     return l4_platform_ctl_system_suspend_u(pfc, extras, l4_utcb());
00191 }
00192
00193 L4_INLINE l4_msgtag_t
00194 l4_platform_ctl_system_shutdown(l4_cap_idx_t pfc,
00195                                 l4_umword_t reboot) L4_NOTHROW
00196 {
00197     return l4_platform_ctl_system_shutdown_u(pfc, reboot, l4_utcb());
00198 }
00199
00200 L4_INLINE l4_msgtag_t
00201 l4_platform_ctl_cpu_enable_u(l4_cap_idx_t pfc,
00202                              l4_umword_t phys_id,
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_CPU_ENABLE_OP;
00207     v->mr[1] = phys_id;
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_cpu_disable_u(l4_cap_idx_t pfc,
00214                               l4_umword_t phys_id,
00215                               l4_utcb_t *utcb) L4_NOTHROW
00216 {
00217     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00218     v->mr[0] = L4_PLATFORM_CTL_CPU_DISABLE_OP;
00219     v->mr[1] = phys_id;
00220     return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 2, 0, 0),
00221                       L4_IPC_NEVER);
00222 }
00223
00224 L4_INLINE l4_msgtag_t
00225 l4_platform_ctl_cpu_enable(l4_cap_idx_t pfc,
00226                             l4_umword_t phys_id) L4_NOTHROW

```


Data Structures

- class [L4::Rcv_endpoint](#)

Interface for kernel objects that allow to receive IPC from them.

Namespaces

- [L4](#)

[L4](#) low-level kernel interface.

16.339.1 Detailed Description

The C++ Receive endpoint interface.

Definition in file [rcv_endpoint](#).

16.340 rcv_endpoint

```

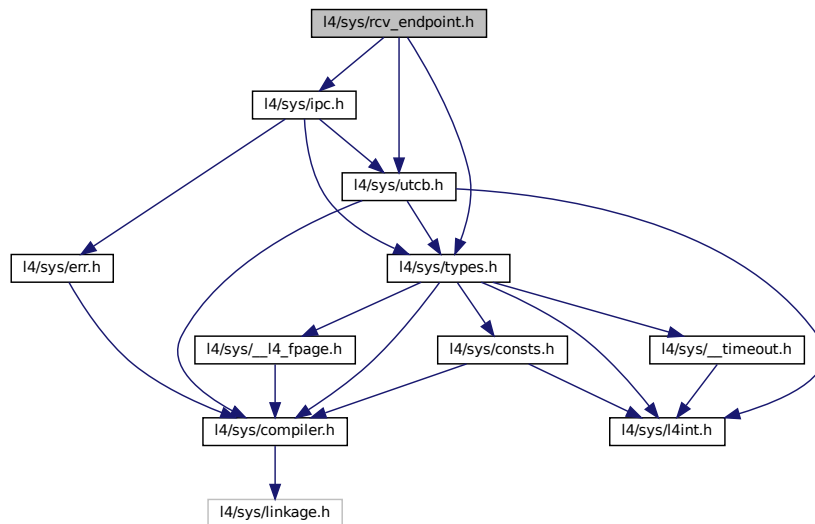
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
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:
00060     L4_INLINE_RPC_OP(L4_RCV_EP_BIND_OP,
00061         l4_msgtag_t, bind_thread, (Ipc::Opt<Ipc::Cap<Thread> > t, l4_umword_t label));
00062
00063     typedef L4::Typeid::Rpcsys_sys<bind_thread_t> Rpcsys;
00064 };
00065
00066 }
```

16.341 l4/sys/rcv_endpoint.h File Reference

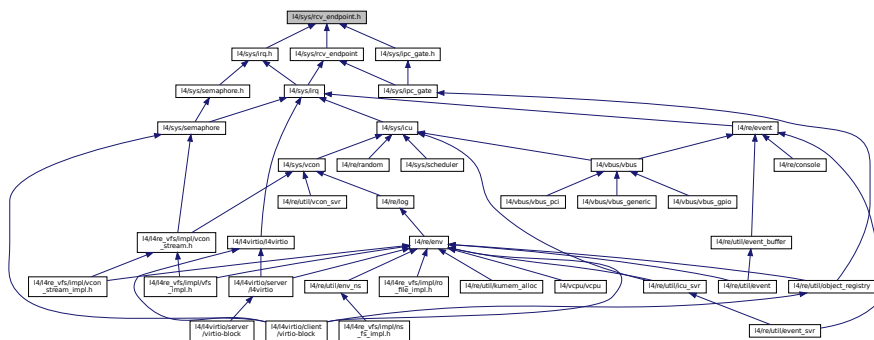
Receive endpoint C interface.

```
#include <linux/sys/utcb.h>
#include <linux/sys/types.h>
#include <linux/sys/ipc.h>
```

Include dependency graph for rcv_endpoint.h:



This graph shows which files directly or indirectly include this file:



Enumerations

- enum `L4_rcv_ep_ops` { `L4_RCV_EP_BIND_OP` = 0x10 }

Receive endpoint operations.

Functions

- `l4_msgtag_t l4_rcv_ep_bind_thread (l4_cap_idx_t ep, l4_cap_idx_t thread, l4_umword_t label)`

Bind the IPC gate to a thread.

16.341.1 Detailed Description

Receive endpoint C interface.

Definition in file [rcv_endpoint.h](#).

16.341.2 Enumeration Type Documentation

16.341.2.1 L4_rcv_ep_ops

enum [L4_rcv_ep_ops](#)

Receive endpoint operations.

Enumerator

L4_RCV_EP_BIND_OP	Bind operation.
-------------------	-----------------

Definition at line 56 of file [rcv_endpoint.h](#).

16.342 rcv_endpoint.h

```

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
00043 L4_INLINE l4_msgtag_t
00044 l4_rcv_ep_bind_thread(l4_cap_idx_t ep, l4_cap_idx_t thread,
00045                      l4_umword_t label);
00046
00051 L4_INLINE l4_msgtag_t
00052 l4_rcv_ep_bind_thread_u(l4_cap_idx_t ep, l4_cap_idx_t thread,
00053                        l4_umword_t label, l4_utcb_t *utcb);
00054
00056 enum L4_rcv_ep_ops
00057 {
00058     L4_RCV_EP_BIND_OP      = 0x10,
00059 };
00060
00061 /* IMPLEMENTATION -----*/
00062
00063 #include <l4/sys/ipc.h>
00064
00065 L4_INLINE l4_msgtag_t

```

```

00066 l4_rcv_ep_bind_thread_u(l4_cap_idx_t ep,
00067                          l4_cap_idx_t thread, l4_umword_t label,
00068                          l4_utcb_t *utcb)
00069 {
00070     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00071     m->mr[0] = L4_RCV_EP_BIND_OP;
00072     m->mr[1] = label;
00073     m->mr[2] = l4_map_obj_control(0, 0);
00074     m->mr[3] = l4_obj_fpage(thread, 0, L4_CAP_FPAGE_RWS).raw;
00075     return l4_ipc_call(ep, utcb, l4_msgtag(L4_PROTO_KOBJECT, 2, 1, 0),
00076                       L4_IPC_NEVER);
00077 }
00078
00079 L4_INLINE l4_msgtag_t
00080 l4_rcv_ep_bind_thread(l4_cap_idx_t ep, l4_cap_idx_t thread,
00081                      l4_umword_t label)
00082 {
00083     return l4_rcv_ep_bind_thread_u(ep, thread, label, l4_utcb());
00084 }
00085
00086

```

16.343 l4/sys/scheduler File Reference

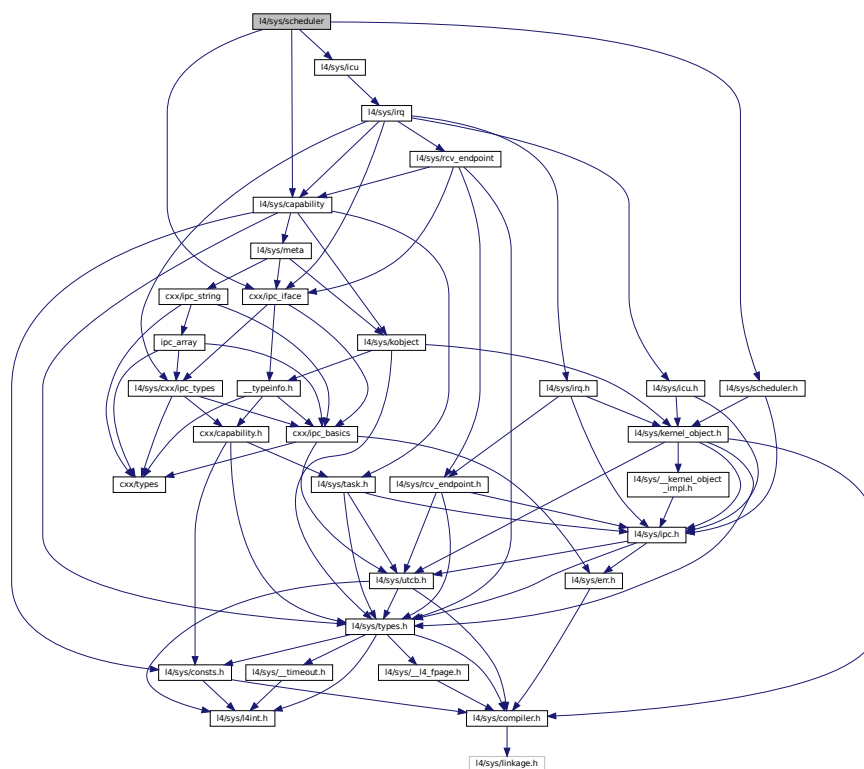
Scheduler object functions.

```

#include <l4/sys/icu>
#include <l4/sys/scheduler.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>

```

Include dependency graph for scheduler:



Data Structures

- class [L4::Scheduler](#)

C++ interface of the [Scheduler](#) kernel object.

Namespaces

- [L4](#)

[L4](#) low-level kernel interface.

16.343.1 Detailed Description

Scheduler object functions.

Definition in file [scheduler](#).

16.344 scheduler

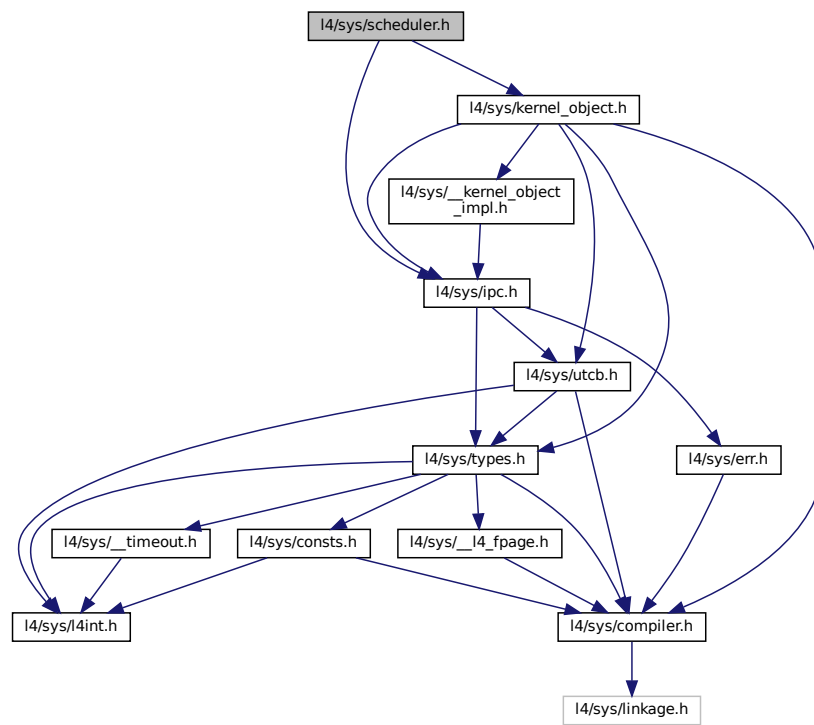
```

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/icu>
00023 #include <l4/sys/scheduler.h>
00024 #include <l4/sys/capability>
00025 #include <l4/sys/cxx/ipc_iface>
00026
00027 namespace L4 {
00028
00029 class L4_EXPORT Scheduler :
00030     public Kobject_t<Scheduler, Icu, L4_PROTO_SCHEDULER,
00031         Type_info::Demand_t<1> >
00032 {
00033 public:
00034     // ABI function for 'info' call
00035     L4_INLINE_RPC_NF_OP(L4_SCHEDULER_INFO_OP,
00036         l4_msgtag_t, info, (l4_umword_t gran_offset, l4_umword_t *map,
00037             l4_umword_t *cpu_max));
00038
00039     l4_msgtag_t info(l4_umword_t *cpu_max, l4_sched_cpu_set_t *cpus,
00040         l4_utcb_t *utcb = l4_utcb()) const noexcept
00041     {
00042         l4_umword_t max = 0;
00043         l4_msgtag_t t =
00044             info_t::call(c(), cpus->gran_offset, &cpus->map, &max, utcb);
00045         if (cpu_max) *cpu_max = max;
00046         return t;
00047     }
00048
00049     L4_INLINE_RPC_OP(L4_SCHEDULER_RUN_THREAD_OP,
00050         l4_msgtag_t, run_thread, (Ipc::Cap<Thread> thread, l4_sched_param_t const &sp));
00051
00052     L4_INLINE_RPC_OP(L4_SCHEDULER_IDLE_TIME_OP,
00053         l4_msgtag_t, idle_time, (l4_sched_cpu_set_t const &cpus,
00054             l4_kernel_clock_t *us));
00055
00056     bool is_online(l4_umword_t cpu, l4_utcb_t *utcb = l4_utcb()) const noexcept
00057     { return l4_scheduler_is_online_u(cap(), cpu, utcb); }
00058
00059     typedef L4::Typeid::Rpcsys<info_t, run_thread_t, idle_time_t> Rpcsys;
00060 };
00061
00062 }
```

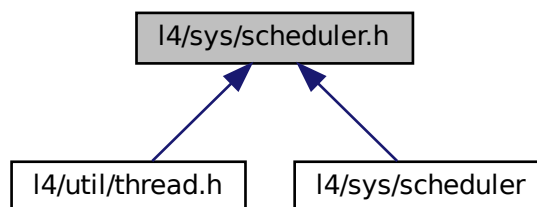
16.345 l4/sys/scheduler.h File Reference

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_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_sched_param_t](#) [l4_sched_param](#) (unsigned prio, [l4_cpu_time_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.

16.345.1 Detailed Description

Scheduler object functions.

Definition in file [scheduler.h](#).

16.346 scheduler.h

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
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
00044 typedef struct l4_sched_cpu_set_t
00045 {
00058     l4_umword_t gran_offset;
00059
00063     l4_umword_t map;
00064
00065 #ifdef __cplusplus
00067     unsigned char granularity() const { return gran_offset » 24; }
00069     unsigned offset() const { return gran_offset & 0x00ffffff; }
00071     void set(unsigned char granularity, unsigned offset)
00072     { gran_offset = ((l4_umword_t)granularity « 24) | (offset & 0x00ffffff); }
00073 #endif
00074 } l4_sched_cpu_set_t;
00075
00086 L4_INLINE l4_sched_cpu_set_t
00087 l4_sched_cpu_set(l4_umword_t offset, unsigned char granularity,
00088                 l4_umword_t map L4_DEFAULT_PARAM(1)) L4_NOTHROW;
00089
00104 L4_INLINE l4_msgtag_t
00105 l4_scheduler_info(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00106                  l4_sched_cpu_set_t *cpus) L4_NOTHROW;
00107
00111 L4_INLINE l4_msgtag_t
00112 l4_scheduler_info_u(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00113                    l4_sched_cpu_set_t *cpus, l4_utcb_t *utcb) L4_NOTHROW;
00114
00115
00120 typedef struct l4_sched_param_t
00121 {
00122     l4_sched_cpu_set_t affinity;
00123     l4_umword_t prio;
00124     l4_umword_t quantum;
00125 } l4_sched_param_t;
00126
00131 L4_INLINE l4_sched_param_t
00132 l4_sched_param(unsigned prio,
00133                l4_cpu_time_t quantum L4_DEFAULT_PARAM(0)) L4_NOTHROW;
00134
00142 L4_INLINE l4_msgtag_t
00143 l4_scheduler_run_thread(l4_cap_idx_t scheduler,
00144                        l4_cap_idx_t thread, l4_sched_param_t const *sp) L4_NOTHROW;
00145
00149 L4_INLINE l4_msgtag_t
00150 l4_scheduler_run_thread_u(l4_cap_idx_t scheduler, l4_cap_idx_t thread,
00151                           l4_sched_param_t const *sp, l4_utcb_t *utcb) L4_NOTHROW;
00152
00160 L4_INLINE l4_msgtag_t
00161 l4_scheduler_idle_time(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus,
00162                       l4_kernel_clock_t *us) L4_NOTHROW;
00163
00167 L4_INLINE l4_msgtag_t
00168 l4_scheduler_idle_time_u(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus,
00169                          l4_kernel_clock_t *us, l4_utcb_t *utcb) L4_NOTHROW;
00170
00171
00172
00183 L4_INLINE int
00184 l4_scheduler_is_online(l4_cap_idx_t scheduler, l4_umword_t cpu) L4_NOTHROW;
00185
00189 L4_INLINE int
00190 l4_scheduler_is_online_u(l4_cap_idx_t scheduler, l4_umword_t cpu,

```

```

00191         l4_utcb_t *utcb) L4_NOTHROW;
00192
00193
00194
00201 enum L4_scheduler_ops
00202 {
00203     L4_SCHEDULER_INFO_OP      = 0UL,
00204     L4_SCHEDULER_RUN_THREAD_OP = 1UL,
00205     L4_SCHEDULER_IDLE_TIME_OP = 2UL,
00206 };
00207
00208 /***** Implementations *****/
00209
00210 L4_INLINE l4_sched_cpu_set_t
00211 l4_sched_cpu_set(l4_umword_t offset, unsigned char granularity,
00212                 l4_umword_t map) L4_NOTHROW
00213 {
00214     l4_sched_cpu_set_t cs;
00215     cs.gran_offset = ((l4_umword_t)granularity << 24) | offset;
00216     cs.map         = map;
00217     return cs;
00218 }
00219
00220 L4_INLINE l4_sched_param_t
00221 l4_sched_param(unsigned prio, l4_cpu_time_t quantum) L4_NOTHROW
00222 {
00223     l4_sched_param_t sp;
00224     sp.prio          = prio;
00225     sp.quantum       = quantum;
00226     sp.affinity      = l4_sched_cpu_set(0, ~0, 1);
00227     return sp;
00228 }
00229
00230
00231 L4_INLINE l4_msgtag_t
00232 l4_scheduler_info_u(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00233                    l4_sched_cpu_set_t *cpus, l4_utcb_t *utcb) L4_NOTHROW
00234 {
00235     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00236     l4_msgtag_t res;
00237
00238     m->mr[0] = L4_SCHEDULER_INFO_OP;
00239     m->mr[1] = cpus->gran_offset;
00240
00241     res = l4_ipc_call(scheduler, utcb, l4_msgtag(L4_PROTO_SCHEDULER, 2, 0, 0), L4_IPC_NEVER);
00242
00243     if (l4_msgtag_has_error(res))
00244         return res;
00245
00246     cpus->map = m->mr[0];
00247
00248     if (cpu_max)
00249         *cpu_max = m->mr[1];
00250
00251     return res;
00252 }
00253
00254 L4_INLINE l4_msgtag_t
00255 l4_scheduler_run_thread_u(l4_cap_idx_t scheduler, l4_cap_idx_t thread,
00256                          l4_sched_param_t const *sp, l4_utcb_t *utcb) L4_NOTHROW
00257 {
00258     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00259     m->mr[0] = L4_SCHEDULER_RUN_THREAD_OP;
00260     m->mr[1] = sp->affinity.gran_offset;
00261     m->mr[2] = sp->affinity.map;
00262     m->mr[3] = sp->prio;
00263     m->mr[4] = sp->quantum;
00264     m->mr[5] = l4_map_obj_control(0, 0);
00265     m->mr[6] = l4_obj_fpage(thread, 0, L4_CAP_FPAGE_RWS).raw;
00266
00267     return l4_ipc_call(scheduler, utcb, l4_msgtag(L4_PROTO_SCHEDULER, 5, 1, 0), L4_IPC_NEVER);
00268 }
00269
00270 L4_INLINE l4_msgtag_t
00271 l4_scheduler_idle_time_u(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus,
00272                         l4_kernel_clock_t *us, l4_utcb_t *utcb) L4_NOTHROW
00273 {
00274     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00275     l4_msgtag_t res;
00276
00277     v->mr[0] = L4_SCHEDULER_IDLE_TIME_OP;
00278     v->mr[1] = cpus->gran_offset;
00279     v->mr[2] = cpus->map;
00280
00281     res = l4_ipc_call(scheduler, utcb,
00282                      l4_msgtag(L4_PROTO_SCHEDULER, 3, 0, 0), L4_IPC_NEVER);
00283

```

```

00284     if (l4_msgtag_has_error(res))
00285         return res;
00286
00287     *us = v->mr64[l4_utcb_mr64_idx(0)];
00288
00289     return res;
00290 }
00291
00292
00293 L4_INLINE int
00294 l4_scheduler_is_online_u(l4_cap_idx_t scheduler, l4_umword_t cpu,
00295                        l4_utcb_t *utcb) L4_NOTHROW
00296 {
00297     l4_sched_cpu_set_t s;
00298     l4_msgtag_t r;
00299     s.gran_offset = cpu;
00300     r = l4_scheduler_info_u(scheduler, NULL, &s, utcb);
00301     if (l4_msgtag_has_error(r) || l4_msgtag_label(r) < 0)
00302         return 0;
00303
00304     return s.map & 1;
00305 }
00306
00307
00308 L4_INLINE l4_msgtag_t
00309 l4_scheduler_info(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00310                 l4_sched_cpu_set_t *cpus) L4_NOTHROW
00311 {
00312     return l4_scheduler_info_u(scheduler, cpu_max, cpus, l4_utcb());
00313 }
00314
00315 L4_INLINE l4_msgtag_t
00316 l4_scheduler_run_thread(l4_cap_idx_t scheduler,
00317                       l4_cap_idx_t thread, l4_sched_param_t const *sp) L4_NOTHROW
00318 {
00319     return l4_scheduler_run_thread_u(scheduler, thread, sp, l4_utcb());
00320 }
00321
00322 L4_INLINE l4_msgtag_t
00323 l4_scheduler_idle_time(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus,
00324                      l4_kernel_clock_t *us) L4_NOTHROW
00325 {
00326     return l4_scheduler_idle_time_u(scheduler, cpus, us, l4_utcb());
00327 }
00328
00329 L4_INLINE int
00330 l4_scheduler_is_online(l4_cap_idx_t scheduler, l4_umword_t cpu) L4_NOTHROW
00331 {
00332     return l4_scheduler_is_online_u(scheduler, cpu, l4_utcb());
00333 }

```

16.347 l4/sys/semaphore File Reference

Semaphore class definition.

```

#include <l4/sys/irq>
#include <l4/sys/semaphore.h>

```

[illegible]

```
graph BT; A[l4/sys/semaphore] --> B[l4/l4virtio/client/virtio-block]; A --> C[l4/l4re_vfs/impl/vcon_stream.h]; C --> D[l4/l4re_vfs/impl/vcon_stream_impl.h]; C --> E[l4/l4re_vfs/impl/vfs_impl.h]; style A fill:#ccc
```

- struct L4::Semaphore
Kernel-provided semaphore object.

- **L4**
L4 low-level kernel interface.

16.347.1 Detailed Description

Semaphore class definition.

Definition in file [semaphore](#).

16.348 semaphore

```

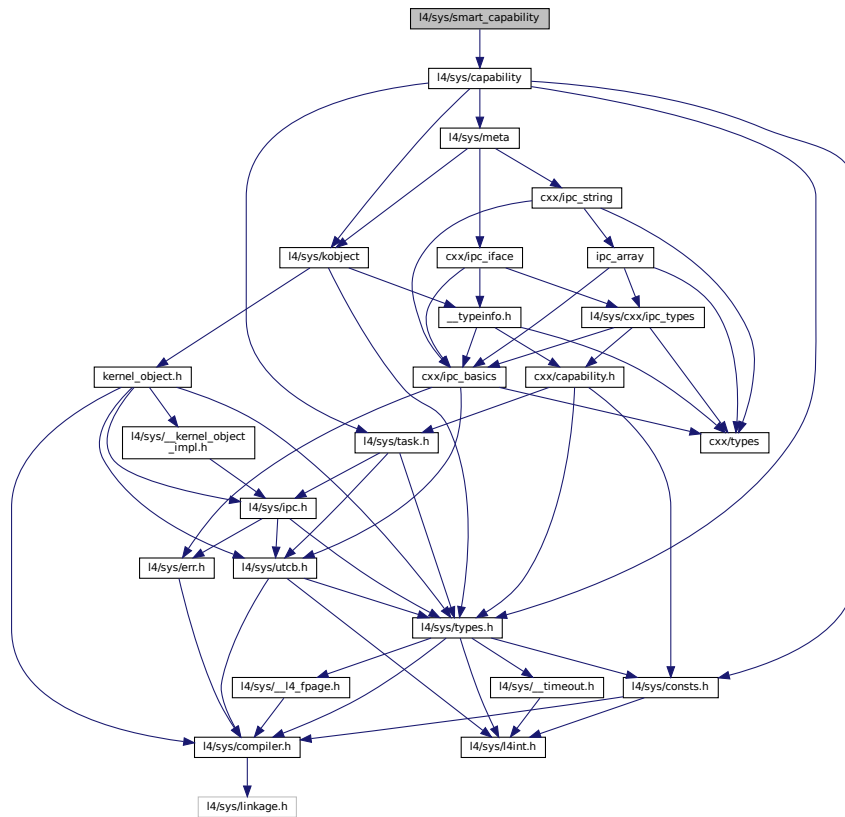
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2015 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU General Public License 2.
00011  * Please see the COPYING-GPL-2 file for details.
00012  *
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
00051 struct Semaphore : Kobject_t<Semaphore, Triggerable, L4_PROTO_SEMAPHORE>
00052 {
00065     l4_msgtag_t up(l4_utcb_t *utcb = l4_utcb()) noexcept
00066     { return trigger(utcb); }
00067
00084     l4_msgtag_t down(l4_timeout_t timeout = L4_IPC_NEVER,
00085                     l4_utcb_t *utcb = l4_utcb()) noexcept
00086     { return l4_semaphore_down_u(cap(), timeout, utcb); }
00087 };
00088
00089 }
```

16.349 l4/sys/smart_capability File Reference

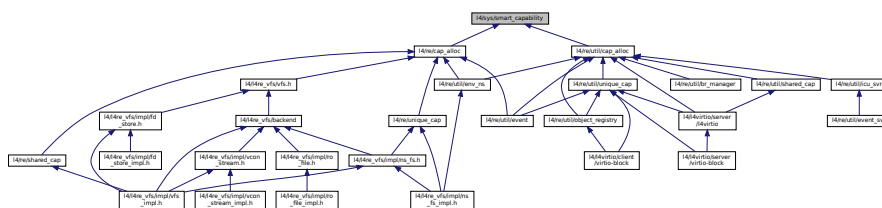
L4::Capability class.


```
#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

- [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.

16.349.1 Detailed Description

L4::Capability class.

Author

Alexander Warg alexander.warg@os.inf.tu-dresden.de

Definition in file [smart_capability](#).

16.350 smart_capability

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00009 /*
00010  * (c) 2008-2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * This file is part of TUD:OS and distributed under the terms of the
00014  * GNU General Public License 2.
00015  * Please see the COPYING-GPL-2 file for details.
00016  *
00017  * As a special exception, you may use this file as part of a free software
00018  * library without restriction. Specifically, if other files instantiate
00019  * templates or use macros or inline functions from this file, or you compile
00020  * this file and link it with other files to produce an executable, this
00021  * file does not by itself cause the resulting executable to be covered by
00022  * the GNU General Public License. This exception does not however
00023  * invalidate any other reasons why the executable file might be covered by
00024  * the GNU General Public License.
00025  */
00026 #pragma once
00027
00028 #include <l4/sys/capability>
00029
00030 namespace L4 {
00031
00032 template< typename T, typename SMART >
00033 class Smart_cap : public Cap_base, private SMART
00034 {
00035 public:
00036     SMART const &smart() const noexcept { return *this; }
00037
00038     void _delete() noexcept
00039     {
00040         SMART::free(const_cast<Smart_cap<T, SMART>&>(*this));
00041     }
00042
00043     Cap<T> release() const noexcept
00044     {
00045         l4_cap_idx_t r = cap();
00046         SMART::invalidate(const_cast<Smart_cap<T, SMART>&>(*this));
00047
00048         return Cap<T>(r);
00049     }
00050
00051     void reset() noexcept
00052     {
00053         _c = L4_INVALID_CAP;
00054     }
00055 }
```

```

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

```

16.351 l4/sys/task File Reference

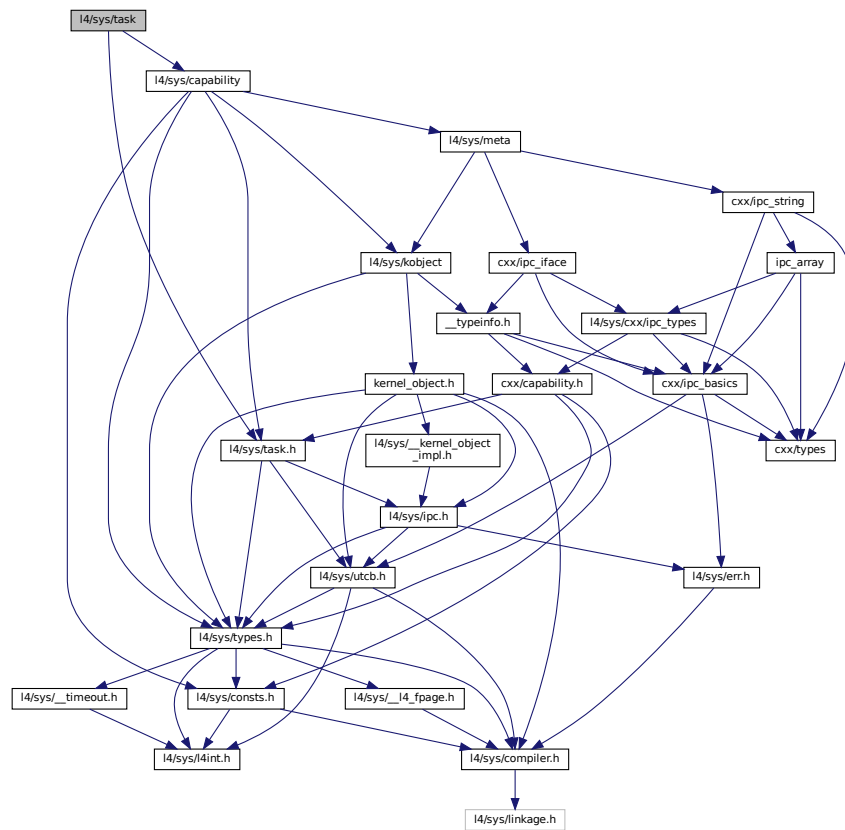
Common task related definitions.

```

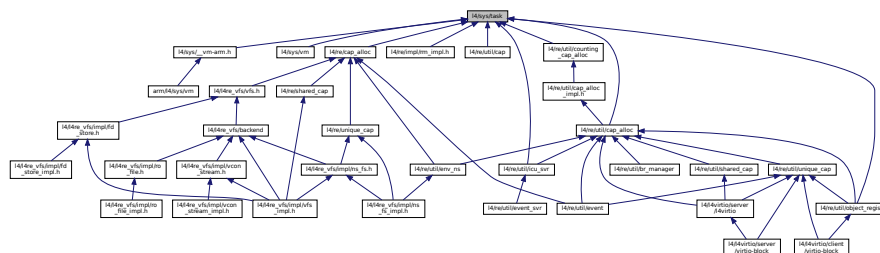
#include <l4/sys/task.h>
#include <l4/sys/capability>

```

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.

Namespaces

- [L4](#)
[L4](#) low-level kernel interface.

16.351.1 Detailed Description

Common task related definitions.

Definition in file [task](#).

16.352 task

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU General Public License 2.
00013  * Please see the COPYING-GPL-2 file for details.
00014  *
00015  * As a special exception, you may use this file as part of a free software
00016  * library without restriction. Specifically, if other files instantiate
00017  * templates or use macros or inline functions from this file, or you compile
00018  * this file and link it with other files to produce an executable, this
00019  * file does not by itself cause the resulting executable to be covered by
00020  * the GNU General Public License. This exception does not however
00021  * invalidate any other reasons why the executable file might be covered by
00022  * the GNU General Public License.
00023  */
00024
00025 #pragma once
00026
00027 #include <l4/sys/task.h>
00028 #include <l4/sys/capability>
00029
00030 namespace L4 {
00031
00043 class Task :
00044     public Kobject_t<Task, Kobject, L4_PROTO_TASK,
00045         Type_info::Demand_t<2> >
00046 {
00047 public:
00067     l4_msgtag_t map(Cap<Task> const &src_task,
00068         l4_fpage_t const &snd_fpage, l4_umword_t snd_base,
00069         l4_utcb_t *utcb = l4_utcb()) noexcept
00070     { return l4_task_map_u(cap(), src_task.cap(), snd_fpage, snd_base, utcb); }
00071
00090     l4_msgtag_t unmap(l4_fpage_t const &fpage,
00091         l4_umword_t map_mask,
00092         l4_utcb_t *utcb = l4_utcb()) noexcept
00093     { return l4_task_unmap_u(cap(), fpage, map_mask, utcb); }
00094
00115     l4_msgtag_t unmap_batch(l4_fpage_t const *fpages,
00116         unsigned num_fpages,
00117         l4_umword_t map_mask,
00118         l4_utcb_t *utcb = l4_utcb()) noexcept
00119     { return l4_task_unmap_batch_u(cap(), fpages, num_fpages, map_mask, utcb); }
00120
00133     l4_msgtag_t delete_obj(L4::Cap<void> obj,
00134         l4_utcb_t *utcb = l4_utcb()) noexcept
00135     { return l4_task_delete_obj_u(cap(), obj.cap(), utcb); }
00136
00147     l4_msgtag_t release_cap(L4::Cap<void> cap,
00148         l4_utcb_t *utcb = l4_utcb()) noexcept
00149     { return l4_task_release_cap_u(this->cap(), cap.cap(), utcb); }
00150
00167     l4_msgtag_t cap_valid(Cap<void> const &cap,
00168         l4_utcb_t *utcb = l4_utcb()) noexcept
00169     { return l4_task_cap_valid_u(this->cap(), cap.cap(), utcb); }
00170
00183     l4_msgtag_t cap_equal(Cap<void> const &cap_a,
00184         Cap<void> const &cap_b,
00185         l4_utcb_t *utcb = l4_utcb()) noexcept
00186     { return l4_task_cap_equal_u(cap(), cap_a.cap(), cap_b.cap(), utcb); }
00187
00202     l4_msgtag_t add_ku_mem(l4_fpage_t const &fpage,
00203         l4_utcb_t *utcb = l4_utcb()) noexcept
00204     { return l4_task_add_ku_mem_u(cap(), fpage, utcb); }
00205
00206 };
00207 }
00208
00209

```

16.353 l4/sys/task.h File Reference

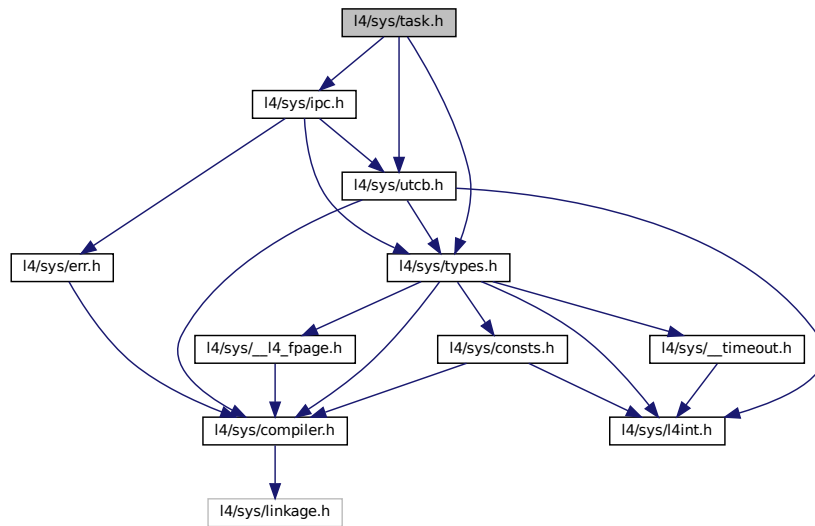
Common task related definitions.

```
#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 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.

16.353.1 Detailed Description

Common task related definitions.

Definition in file [task.h](#).

16.354 task.h

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *      Björn Döbel <doebel@os.inf.tu-dresden.de>,
00009  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU General Public License 2.
00014  * Please see the COPYING-GPL-2 file for details.
00015  *
00016  * As a special exception, you may use this file as part of a free software
00017  * library without restriction. Specifically, if other files instantiate
00018  * templates or use macros or inline functions from this file, or you compile
00019  * this file and link it with other files to produce an executable, this
00020  * file does not by itself cause the resulting executable to be covered by
00021  * the GNU General Public License. This exception does not however
00022  * invalidate any other reasons why the executable file might be covered by
00023  * the GNU General Public License.
00024  */
00025 #pragma once
00026 #include <l4/sys/types.h>
00027 #include <l4/sys/utcb.h>
00028
00060 L4_INLINE l4_msgtag_t
00061 l4_task_map(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00062            l4_fpage_t snd_fpage, l4_umword_t snd_base) L4_NOTHROW;
00063
00067 L4_INLINE l4_msgtag_t
00068 l4_task_map_u(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00069              l4_fpage_t snd_fpage, l4_umword_t snd_base, l4_utcb_t *utcb) L4_NOTHROW;
00070
00089 L4_INLINE l4_msgtag_t
00090 l4_task_unmap(l4_cap_idx_t task, l4_fpage_t fpage,
00091              l4_umword_t map_mask) L4_NOTHROW;
00092
00096 L4_INLINE l4_msgtag_t
00097 l4_task_unmap_u(l4_cap_idx_t task, l4_fpage_t fpage,
00098                l4_umword_t map_mask, l4_utcb_t *utcb) L4_NOTHROW;
00099
00123 L4_INLINE l4_msgtag_t
00124 l4_task_unmap_batch(l4_cap_idx_t task, l4_fpage_t const *fpages,
```



```

00125         unsigned num_fpages, l4_umword_t map_mask) L4_NOTHROW;
00126
00130 L4_INLINE l4_msgtag_t
00131 l4_task_unmap_batch_u(l4_cap_idx_t task, l4_fpage_t const *fpages,
00132         unsigned num_fpages, l4_umword_t map_mask,
00133         l4_utcb_t *u) L4_NOTHROW;
00134
00150 L4_INLINE l4_msgtag_t
00151 l4_task_delete_obj(l4_cap_idx_t task, l4_cap_idx_t obj) L4_NOTHROW;
00152
00156 L4_INLINE l4_msgtag_t
00157 l4_task_delete_obj_u(l4_cap_idx_t task, l4_cap_idx_t obj,
00158         l4_utcb_t *u) L4_NOTHROW;
00159
00171 L4_INLINE l4_msgtag_t
00172 l4_task_release_cap(l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW;
00173
00177 L4_INLINE l4_msgtag_t
00178 l4_task_release_cap_u(l4_cap_idx_t task, l4_cap_idx_t cap,
00179         l4_utcb_t *u) L4_NOTHROW;
00180
00181
00198 L4_INLINE l4_msgtag_t
00199 l4_task_cap_valid(l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW;
00200
00204 L4_INLINE l4_msgtag_t
00205 l4_task_cap_valid_u(l4_cap_idx_t task, l4_cap_idx_t cap, l4_utcb_t *utcb) L4_NOTHROW;
00206
00219 L4_INLINE l4_msgtag_t
00220 l4_task_cap_equal(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00221         l4_cap_idx_t cap_b) L4_NOTHROW;
00222
00226 L4_INLINE l4_msgtag_t
00227 l4_task_add_ku_mem_u(l4_cap_idx_t task, l4_fpage_t ku_mem,
00228         l4_utcb_t *u) L4_NOTHROW;
00229
00239 L4_INLINE l4_msgtag_t
00240 l4_task_add_ku_mem(l4_cap_idx_t task, l4_fpage_t ku_mem) L4_NOTHROW;
00241
00242
00246 L4_INLINE l4_msgtag_t
00247 l4_task_cap_equal_u(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00248         l4_cap_idx_t cap_b, l4_utcb_t *utcb) L4_NOTHROW;
00249
00254 enum l4_task_ops
00255 {
00256     L4_TASK_MAP_OP           = 0UL,
00257     L4_TASK_UNMAP_OP        = 1UL,
00258     L4_TASK_CAP_INFO_OP     = 2UL,
00259     L4_TASK_ADD_KU_MEM_OP   = 3UL,
00260     L4_TASK_LDT_SET_X86_OP  = 0x11UL,
00261     L4_TASK_MAP_VGICC_ARM_OP = 0x12UL,
00262 };
00263
00264
00265 /* IMPLEMENTATION -----*/
00266
00267 #include <l4/sys/ipc.h>
00268
00269
00270 L4_INLINE l4_msgtag_t
00271 l4_task_map_u(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00272         l4_fpage_t snd_fpage, l4_umword_t snd_base, l4_utcb_t *u) L4_NOTHROW
00273 {
00274     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00275     v->mr[0] = L4_TASK_MAP_OP;
00276     v->mr[3] = l4_map_obj_control(0,0);
00277     v->mr[4] = l4_obj_fpage(src_task, 0, L4_CAP_FPAGE_RWS).raw;
00278     v->mr[1] = snd_base;
00279     v->mr[2] = snd_fpage.raw;
00280     return l4_ipc_call(dst_task, u, l4_msgtag(L4_PROTO_TASK, 3, 1, 0), L4_IPC_NEVER);
00281 }
00282
00283 L4_INLINE l4_msgtag_t
00284 l4_task_unmap_u(l4_cap_idx_t task, l4_fpage_t fpage,
00285         l4_umword_t map_mask, l4_utcb_t *u) L4_NOTHROW
00286 {
00287     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00288     v->mr[0] = L4_TASK_UNMAP_OP;
00289     v->mr[1] = map_mask;
00290     v->mr[2] = fpage.raw;
00291     return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 3, 0, 0), L4_IPC_NEVER);
00292 }
00293
00294 L4_INLINE l4_msgtag_t
00295 l4_task_unmap_batch_u(l4_cap_idx_t task, l4_fpage_t const *fpages,
00296         unsigned num_fpages, l4_umword_t map_mask,

```

```

00297         l4_utcb_t *u) L4_NOTHROW
00298 {
00299     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00300     v->mr[0] = L4_TASK_UNMAP_OP;
00301     v->mr[1] = map_mask;
00302     __builtin_memcpy(&v->mr[2], fpages, num_fpages * sizeof(l4_fpage_t));
00303     return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2 + num_fpages, 0, 0), L4_IPC_NEVER);
00304 }
00305
00306 L4_INLINE l4_msgtag_t
00307 l4_task_cap_valid_u(l4_cap_idx_t task, l4_cap_idx_t cap, l4_utcb_t *u) L4_NOTHROW
00308 {
00309     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00310     v->mr[0] = L4_TASK_CAP_INFO_OP;
00311     v->mr[1] = cap & ~1UL;
00312     return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2, 0, 0), L4_IPC_NEVER);
00313 }
00314
00315 L4_INLINE l4_msgtag_t
00316 l4_task_cap_equal_u(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00317                    l4_cap_idx_t cap_b, l4_utcb_t *u) L4_NOTHROW
00318 {
00319     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00320     v->mr[0] = L4_TASK_CAP_INFO_OP;
00321     v->mr[1] = cap_a;
00322     v->mr[2] = cap_b;
00323     return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 3, 0, 0), L4_IPC_NEVER);
00324 }
00325
00326 L4_INLINE l4_msgtag_t
00327 l4_task_add_ku_mem_u(l4_cap_idx_t task, l4_fpage_t ku_mem,
00328                    l4_utcb_t *u) L4_NOTHROW
00329 {
00330     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00331     v->mr[0] = L4_TASK_ADD_KU_MEM_OP;
00332     v->mr[1] = ku_mem.raw;
00333     return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2, 0, 0), L4_IPC_NEVER);
00334 }
00335
00336
00337
00338 L4_INLINE l4_msgtag_t
00339 l4_task_map(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00340            l4_fpage_t snd_fpage, l4_umword_t snd_base) L4_NOTHROW
00341 {
00342     return l4_task_map_u(dst_task, src_task, snd_fpage, snd_base, l4_utcb());
00343 }
00344
00345 L4_INLINE l4_msgtag_t
00346 l4_task_unmap(l4_cap_idx_t task, l4_fpage_t fpage,
00347              l4_umword_t map_mask) L4_NOTHROW
00348 {
00349     return l4_task_unmap_u(task, fpage, map_mask, l4_utcb());
00350 }
00351
00352 L4_INLINE l4_msgtag_t
00353 l4_task_unmap_batch(l4_cap_idx_t task, l4_fpage_t const *fpages,
00354                   unsigned num_fpages, l4_umword_t map_mask) L4_NOTHROW
00355 {
00356     return l4_task_unmap_batch_u(task, fpages, num_fpages, map_mask,
00357                                l4_utcb());
00358 }
00359
00360 L4_INLINE l4_msgtag_t
00361 l4_task_delete_obj_u(l4_cap_idx_t task, l4_cap_idx_t obj,
00362                    l4_utcb_t *u) L4_NOTHROW
00363 {
00364     return l4_task_unmap_u(task, l4_obj_fpage(obj, 0, L4_CAP_FPAGE_RWSD),
00365                           L4_FP_DELETE_OBJ, u);
00366 }
00367
00368 L4_INLINE l4_msgtag_t
00369 l4_task_delete_obj(l4_cap_idx_t task, l4_cap_idx_t obj) L4_NOTHROW
00370 {
00371     return l4_task_delete_obj_u(task, obj, l4_utcb());
00372 }
00373
00374
00375 L4_INLINE l4_msgtag_t
00376 l4_task_release_cap_u(l4_cap_idx_t task, l4_cap_idx_t cap,
00377                    l4_utcb_t *u) L4_NOTHROW
00378 {
00379     return l4_task_unmap_u(task, l4_obj_fpage(cap, 0, L4_CAP_FPAGE_RWSD),
00380                           L4_FP_ALL_SPACES, u);
00381 }
00382
00383 L4_INLINE l4_msgtag_t

```

```

00384 l4_task_release_cap(l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW
00385 {
00386     return l4_task_release_cap_u(task, cap, l4_utcb());
00387 }
00388
00389 L4_INLINE l4_msgtag_t
00390 l4_task_cap_valid(l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW
00391 {
00392     return l4_task_cap_valid_u(task, cap, l4_utcb());
00393 }
00394
00395 L4_INLINE l4_msgtag_t
00396 l4_task_cap_equal(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00397                  l4_cap_idx_t cap_b) L4_NOTHROW
00398 {
00399     return l4_task_cap_equal_u(task, cap_a, cap_b, l4_utcb());
00400 }
00401
00402 L4_INLINE l4_msgtag_t
00403 l4_task_add_ku_mem(l4_cap_idx_t task, l4_fpage_t ku_mem) L4_NOTHROW
00404 {
00405     return l4_task_add_ku_mem_u(task, ku_mem, l4_utcb());
00406 }

```

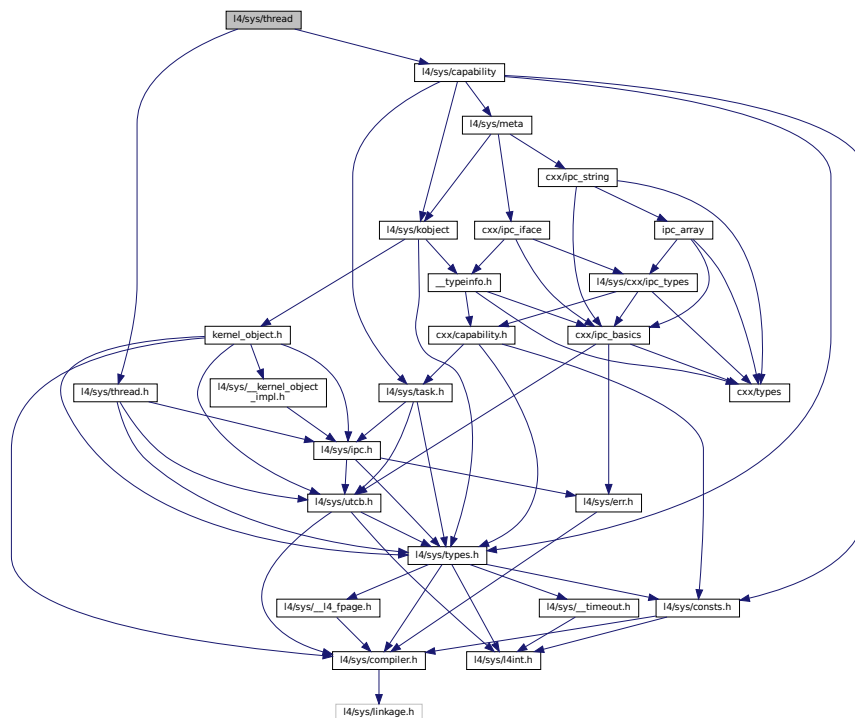
16.355 l4/sys/thread File Reference

Common thread related definitions.

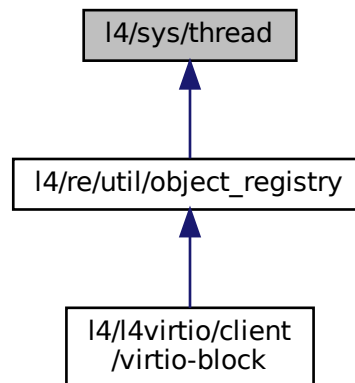
```
#include <l4/sys/capability>
```

```
#include <l4/sys/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.
- class [L4::Thread::Attr](#)
Thread attributes used for control_commit().
- class [L4::Thread::Modify_senders](#)
Wrapper class for modifying senders.

Namespaces

- [L4](#)
L4 low-level kernel interface.

16.355.1 Detailed Description

Common thread related definitions.

Definition in file [thread](#).

16.356 thread

```

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
00058 class Thread :
00059     public Kobject_t<Thread, Kobject, L4_PROTO_THREAD,
00060                     Type_info::Demand_t<1> >
00061 {
00062 public:
00085     l4_msgtag_t ex_regs(l4_addr_t ip, l4_addr_t sp,
00086                        l4_umword_t flags,
00087                        l4_utcb_t *utcb = l4_utcb()) noexcept
00088     { return l4_thread_ex_regs_u(cap(), ip, sp, flags, utcb); }
00089
00111     l4_msgtag_t ex_regs(l4_addr_t *ip, l4_addr_t *sp,
00112                        l4_umword_t *flags,
00113                        l4_utcb_t *utcb = l4_utcb()) noexcept
00114     { return l4_thread_ex_regs_ret_u(cap(), ip, sp, flags, utcb); }
00115
00116
00129     class Attr
00130     {
00131     private:
00132         friend class L4::Thread;
00133         l4_utcb_t *_u;
00134
00135     public:
00142         explicit Attr(l4_utcb_t *utcb = l4_utcb()) noexcept : _u(utcb)
00143         { l4_thread_control_start_u(utcb); }
00144
00152         void pager(Cap<void> const &pager) noexcept
00153         { l4_thread_control_pager_u(pager.cap(), _u); }
00154
00161         Cap<void> pager() noexcept
00162         { return Cap<void>(l4_utcb_mr_u(_u)->mr[1]); }
00163
00171         void exc_handler(Cap<void> const &exc_handler) noexcept
00172         { l4_thread_control_exc_handler_u(exc_handler.cap(), _u); }
00173
00180         Cap<void> exc_handler() noexcept
00181         { return Cap<void>(l4_utcb_mr_u(_u)->mr[2]); }
00182
00202         void bind(l4_utcb_t *thread_utcb, Cap<Task> const &task) noexcept
00203         { l4_thread_control_bind_u(thread_utcb, task.cap(), _u); }
00204
00208         void alien(int on) noexcept
00209         { l4_thread_control_alien_u(_u, on); }
00210
00216         void ux_host_syscall(int on) noexcept
00217         { l4_thread_control_ux_host_syscall_u(_u, on); }
00218
00219     };
00220
00226     l4_msgtag_t control(Attr const &attr) noexcept
00227     { return l4_thread_control_commit_u(cap(), attr._u); }
00228
00236     l4_msgtag_t switch_to(l4_utcb_t *utcb = l4_utcb()) noexcept
00237     { return l4_thread_switch_u(cap(), utcb); }
00238
00247     l4_msgtag_t stats_time(l4_kernel_clock_t *us,
00248                           l4_utcb_t *utcb = l4_utcb()) noexcept
00249     { return l4_thread_stats_time_u(cap(), us, utcb); }

```

```

00250
00256  l4_msgtag_t vcpu_resume_start(l4_utcb_t *utcb = l4_utcb()) noexcept
00257  { return l4_thread_vcpu_resume_start_u(utcb); }
00258
00264  l4_msgtag_t vcpu_resume_commit(l4_msgtag_t tag,
00265                                l4_utcb_t *utcb = l4_utcb()) noexcept
00266  { return l4_thread_vcpu_resume_commit_u(cap(), tag, utcb); }
00267
00283  l4_msgtag_t vcpu_control(l4_addr_t vcpu_state, l4_utcb_t *utcb = l4_utcb())
00284  noexcept
00285  { return l4_thread_vcpu_control_u(cap(), vcpu_state, utcb); }
00286
00307  l4_msgtag_t vcpu_control_ext(l4_addr_t ext_vcpu_state,
00308                               l4_utcb_t *utcb = l4_utcb()) noexcept
00309  { return l4_thread_vcpu_control_ext_u(cap(), ext_vcpu_state, utcb); }
00310
00324  l4_msgtag_t register_del_irq(Cap<Irq> irq, l4_utcb_t *u = l4_utcb()) noexcept
00325  { return l4_thread_register_del_irq_u(cap(), irq.cap(), u); }
00326
00334  class Modify_senders
00335  {
00336  private:
00337      friend class Thread;
00338      l4_utcb_t *utcb;
00339      unsigned cnt;
00340
00341  public:
00342      explicit Modify_senders(l4_utcb_t *u = l4_utcb()) noexcept
00343      : utcb(u), cnt(1)
00344      {
00345          l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_MODIFY_SENDER_OP;
00346      }
00347
00367  int add(l4_umword_t match_mask, l4_umword_t match,
00368          l4_umword_t del_bits, l4_umword_t add_bits) noexcept
00369  {
00370      l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00371      if (cnt >= L4_UTCB_GENERIC_DATA_SIZE - 4)
00372          return -L4_ENOMEM;
00373      m->mr[cnt++] = match_mask;
00374      m->mr[cnt++] = match;
00375      m->mr[cnt++] = del_bits;
00376      m->mr[cnt++] = add_bits;
00377      return 0;
00378  }
00379  };
00380
00394  l4_msgtag_t modify_senders(Modify_senders const &todo) noexcept
00395  {
00396      return l4_ipc_call(cap(), todo.utcb, l4_msgtag(L4_PROTO_THREAD, todo.cnt, 0, 0), L4_IPC_NEVER);
00397  }
00398  };
00399  }

```

16.357 l4/cxx/thread File Reference

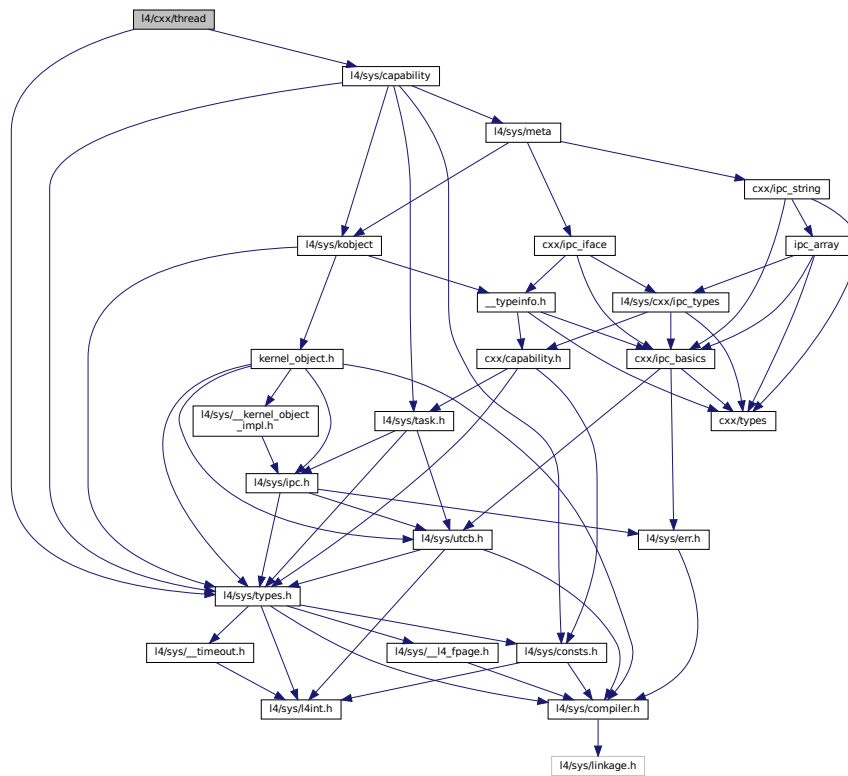
Thread implementation.

```

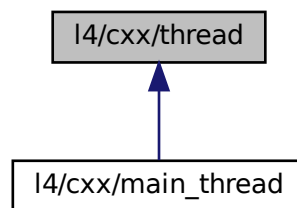
#include <l4/sys/capability>
#include <l4/sys/types.h>

```

Include dependency graph for thread:



This graph shows which files directly or indirectly include this file:



Namespaces

- [cxx](#)

Our C++ library.

16.357.1 Detailed Description

Thread implementation.

Definition in file [thread](#).

16.358 thread

```

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

```

16.359 l4/sys/typeinfo_svr File Reference

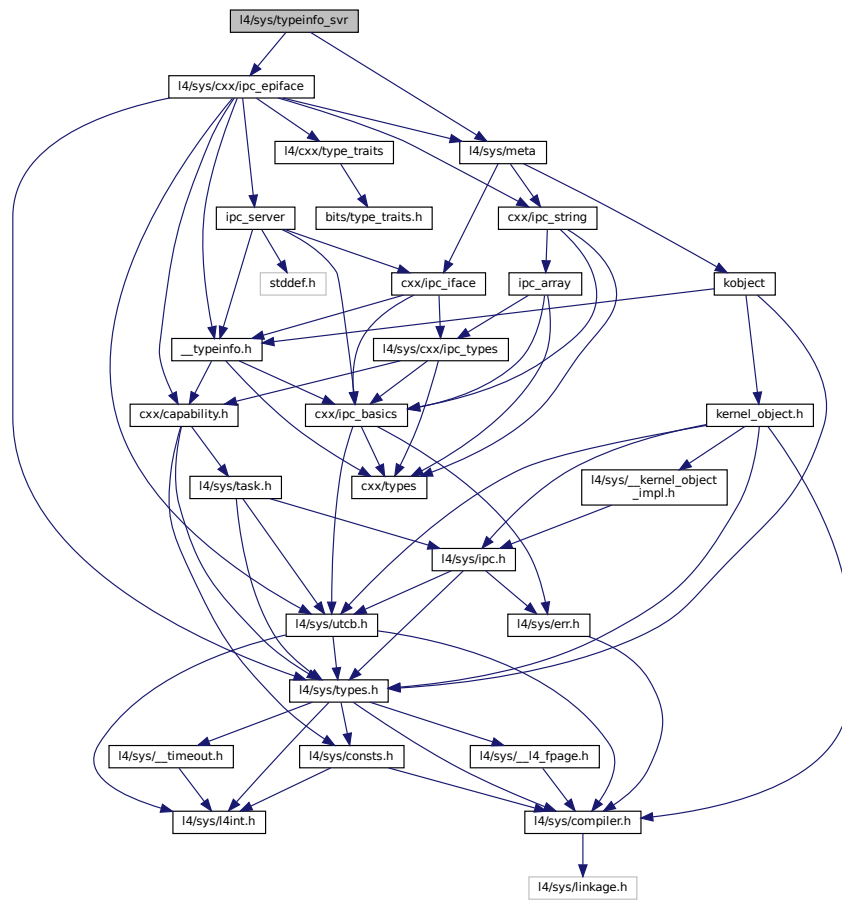
Type information server template.

```

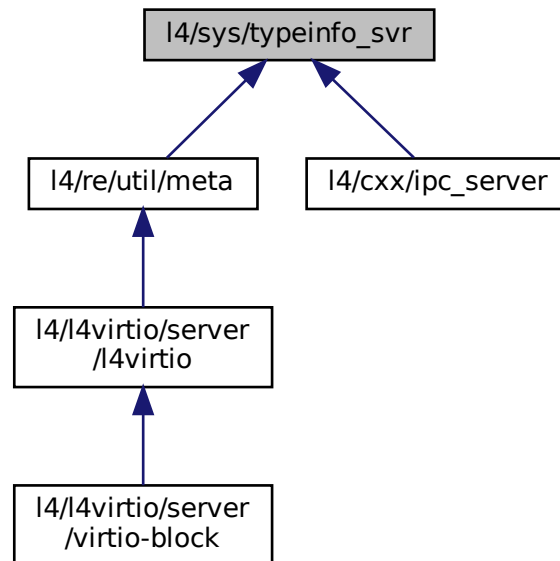
#include <l4/sys/meta>
#include <l4/sys/cxx/ipc_epiface>

```


Include dependency graph for typeinfo_svr:



This graph shows which files directly or indirectly include this file:



Namespaces

- [L4](#)

[L4](#) low-level kernel interface.

16.359.1 Detailed Description

Type information server template.

Definition in file [typeinfo_svr](#).

16.360 typeinfo_svr

```

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::Rpc>((Msvr *)0, ios.utcb(),
00037                                                    ios.tag(), 0);
00038     ios.set_ipc_params(tag);
00039     return tag.label();
00040 }
00041
00042 }}

```

16.361 l4/sys/types.h File Reference

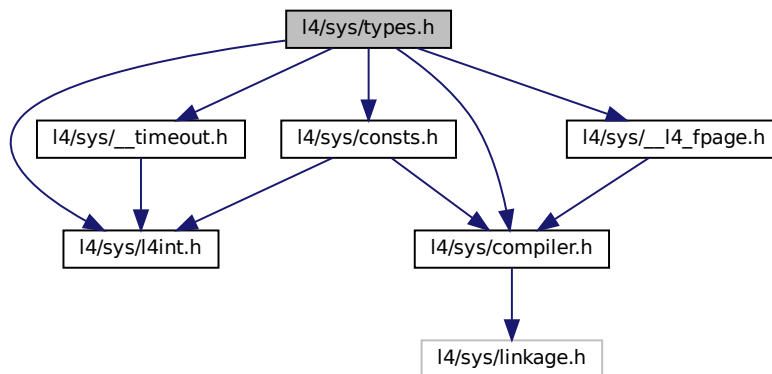
Common [L4](#) ABI Data Types.

```

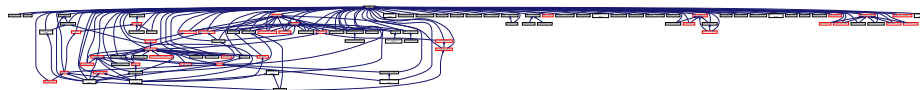
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
#include <l4/sys/consts.h>
#include <l4/sys/__l4_fpage.h>
#include <l4/sys/__timeout.h>

```

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](#)
L4 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](#)
, [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 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.

16.361.1 Detailed Description

Common [L4](#) ABI Data Types.

Definition in file [types.h](#).

16.361.2 Function Documentation

16.361.2.1 [l4_capability_next\(\)](#)

```
l4\_cap\_idx\_t l4\_capability\_next (
    l4\_cap\_idx\_t c ) [inline]
```

Get the next capability selector after c.

Parameters

c	The capability selector for which the next selector shall be computed.
-------------------	--

Returns

The next capability selector after c.

Definition at line [460](#) of file [types.h](#).

16.362 types.h

```
00001  /*****
00007  */
00008  * (c) 2008-2013 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/l4int.h>
00031 #include <l4/sys/compiler.h>
00032 #include <l4/sys/consts.h>
00033
00034
00049 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,
00133     L4_MSGTAG_PROPAGATE   = 0x4000,
00134
00139     L4_MSGTAG_FLAGS       = 0xf000,
00140 };
00141
00142
00159 typedef struct l4_msgtag_t
00160 {
00161     l4_mword_t raw;
00162 #ifdef __cplusplus
00164     long label() const L4_NOTHROW { return raw >> 16; }
00166     void label(long v) L4_NOTHROW { raw = (raw & 0x0ffff) | ((l4_umword_t)v << 16); }
00168     unsigned words() const L4_NOTHROW { return raw & 0x3f; }
00170     unsigned items() const L4_NOTHROW { return (raw >> 6) & 0x3f; }
00177     unsigned flags() const L4_NOTHROW { return raw & 0xf000; }

```

```

00179 bool is_page_fault() const L4_NOTHROW { return label() == L4_PROTO_PAGE_FAULT; }
00181 bool is_preemption() const L4_NOTHROW { return label() == L4_PROTO_PREEMPTION; }
00183 bool is_sys_exception() const L4_NOTHROW { return label() == L4_PROTO_SYS_EXCEPTION; }
00185 bool is_exception() const L4_NOTHROW { return label() == L4_PROTO_EXCEPTION; }
00187 bool is_sigma0() const L4_NOTHROW { return label() == L4_PROTO_SIGMA0; }
00189 bool is_io_page_fault() const L4_NOTHROW { return label() == L4_PROTO_IO_PAGE_FAULT; }
00191 unsigned has_error() const L4_NOTHROW { return raw & L4_MSGTAG_ERROR; }
00192 #endif
00193 } l4_msgtag_t;
00194
00195
00196
00208 L4_INLINE l4_msgtag_t l4_msgtag(long label, unsigned words, unsigned items,
00209                                unsigned flags) L4_NOTHROW;
00210
00219 L4_INLINE long l4_msgtag_label(l4_msgtag_t t) L4_NOTHROW;
00220
00229 L4_INLINE unsigned l4_msgtag_words(l4_msgtag_t t) L4_NOTHROW;
00230
00239 L4_INLINE unsigned l4_msgtag_items(l4_msgtag_t t) L4_NOTHROW;
00240
00251 L4_INLINE unsigned l4_msgtag_flags(l4_msgtag_t t) L4_NOTHROW;
00252
00265 L4_INLINE unsigned l4_msgtag_has_error(l4_msgtag_t t) L4_NOTHROW;
00266
00275 L4_INLINE unsigned l4_msgtag_is_page_fault(l4_msgtag_t t) L4_NOTHROW;
00276
00284 L4_INLINE unsigned l4_msgtag_is_preemption(l4_msgtag_t t) L4_NOTHROW;
00285
00294 L4_INLINE unsigned l4_msgtag_is_sys_exception(l4_msgtag_t t) L4_NOTHROW;
00295
00304 L4_INLINE unsigned l4_msgtag_is_exception(l4_msgtag_t t) L4_NOTHROW;
00305
00314 L4_INLINE unsigned l4_msgtag_is_sigma0(l4_msgtag_t t) L4_NOTHROW;
00315
00324 L4_INLINE unsigned l4_msgtag_is_io_page_fault(l4_msgtag_t t) L4_NOTHROW;
00325
00342 typedef unsigned long l4_cap_idx_t;
00343
00353 L4_INLINE unsigned l4_is_invalid_cap(l4_cap_idx_t c) L4_NOTHROW;
00354
00364 L4_INLINE unsigned l4_is_valid_cap(l4_cap_idx_t c) L4_NOTHROW;
00365
00376 L4_INLINE unsigned l4_capability_equal(l4_cap_idx_t c1, l4_cap_idx_t c2) L4_NOTHROW;
00377
00386 L4_INLINE l4_cap_idx_t l4_capability_next(l4_cap_idx_t c) L4_NOTHROW;
00387
00388 /* ***** */
00389 /* Implementation */
00390
00391 L4_INLINE unsigned
00392 l4_is_invalid_cap(l4_cap_idx_t c) L4_NOTHROW
00393 { return c & L4_INVALID_CAP_BIT; }
00394
00395 L4_INLINE unsigned
00396 l4_is_valid_cap(l4_cap_idx_t c) L4_NOTHROW
00397 { return !(c & L4_INVALID_CAP_BIT); }
00398
00399 L4_INLINE unsigned
00400 l4_capability_equal(l4_cap_idx_t c1, l4_cap_idx_t c2) L4_NOTHROW
00401 { return (c1 >> L4_CAP_SHIFT) == (c2 >> L4_CAP_SHIFT); }
00402
00403
00407 L4_INLINE
00408 l4_msgtag_t l4_msgtag(long label, unsigned words, unsigned items,
00409                       unsigned flags) L4_NOTHROW
00410 {
00411     return (l4_msgtag_t){ (l4_mword_t)((l4_umword_t)label << 16)
00412                          | (l4_mword_t)(words & 0x3f)
00413                          | (l4_mword_t)((items & 0x3f) << 6)
00414                          | (l4_mword_t)(flags & 0xf000)};
00415 }
00416
00417
00418
00419 L4_INLINE
00420 long l4_msgtag_label(l4_msgtag_t t) L4_NOTHROW
00421 { return t.raw >> 16; }
00422
00423 L4_INLINE
00424 unsigned l4_msgtag_words(l4_msgtag_t t) L4_NOTHROW
00425 { return t.raw & 0x3f; }
00426
00427 L4_INLINE
00428 unsigned l4_msgtag_items(l4_msgtag_t t) L4_NOTHROW
00429 { return (t.raw >> 6) & 0x3f; }
00430

```

```

00431 L4_INLINE
00432 unsigned l4_msgtag_flags(l4_msgtag_t t) L4_NOTHROW
00433 { return t.raw & 0xf000; }
00434
00435
00436 L4_INLINE
00437 unsigned l4_msgtag_has_error(l4_msgtag_t t) L4_NOTHROW
00438 { return t.raw & L4_MSGTAG_ERROR; }
00439
00440
00441
00442 L4_INLINE unsigned l4_msgtag_is_page_fault(l4_msgtag_t t) L4_NOTHROW
00443 { return l4_msgtag_label(t) == L4_PROTO_PAGE_FAULT; }
00444
00445 L4_INLINE unsigned l4_msgtag_is_preemption(l4_msgtag_t t) L4_NOTHROW
00446 { return l4_msgtag_label(t) == L4_PROTO_PREEMPTION; }
00447
00448 L4_INLINE unsigned l4_msgtag_is_sys_exception(l4_msgtag_t t) L4_NOTHROW
00449 { return l4_msgtag_label(t) == L4_PROTO_SYS_EXCEPTION; }
00450
00451 L4_INLINE unsigned l4_msgtag_is_exception(l4_msgtag_t t) L4_NOTHROW
00452 { return l4_msgtag_label(t) == L4_PROTO_EXCEPTION; }
00453
00454 L4_INLINE unsigned l4_msgtag_is_sigma0(l4_msgtag_t t) L4_NOTHROW
00455 { return l4_msgtag_label(t) == L4_PROTO_SIGMA0; }
00456
00457 L4_INLINE unsigned l4_msgtag_is_io_page_fault(l4_msgtag_t t) L4_NOTHROW
00458 { return l4_msgtag_label(t) == L4_PROTO_IO_PAGE_FAULT; }
00459
00460 L4_INLINE l4_cap_idx_t l4_capability_next(l4_cap_idx_t c) L4_NOTHROW
00461 { return c + L4_CAP_OFFSET; }
00462
00463 #include <l4/sys/__l4_fpage.h>
00464 #include <l4/sys/__timeout.h>

```

16.363 l4/sys/utcb.h File Reference

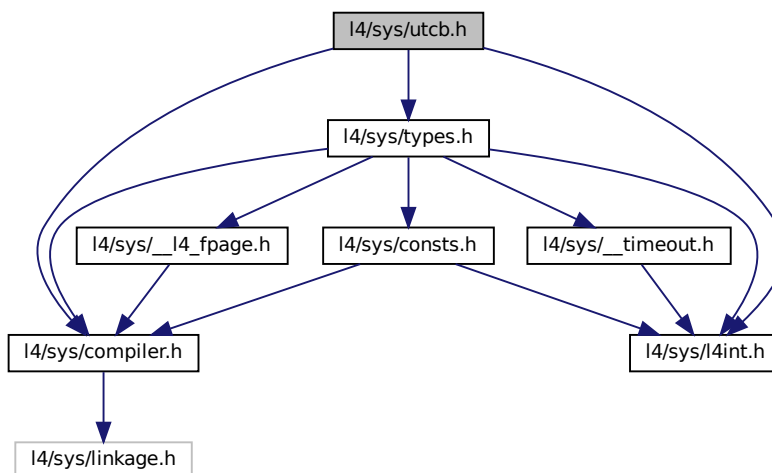
UTCB definitions.

```

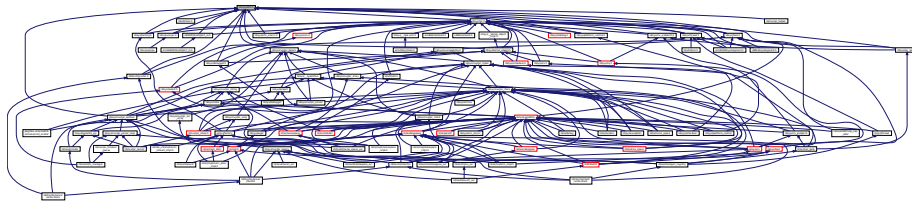
#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>

```

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.

16.363.1 Detailed Description

UTCB definitions.

Definition in file `utcb.h`.

16.364 utcb.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 *      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_timeout_t xfer;
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 */

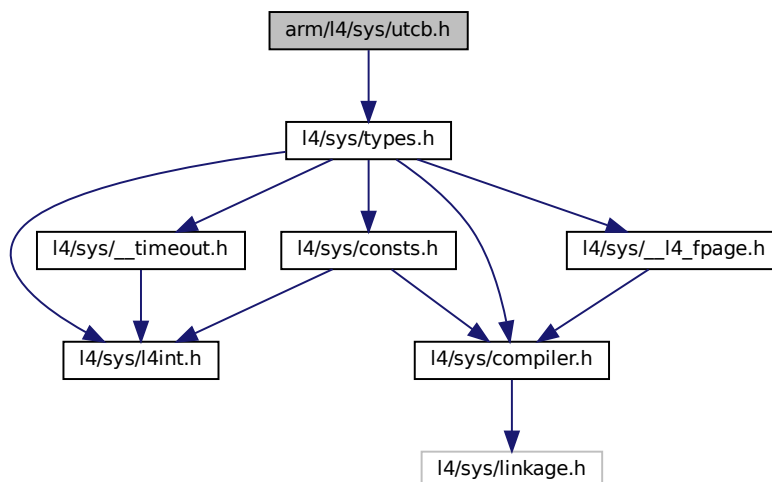
```

16.365 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\(\)](#).

16.365.1 Detailed Description

UTCB definitions for ARM.

Definition in file [utcb.h](#).

16.366 utcb.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 #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_exc_regs_t;
00051
00057 enum L4_utcb_consts_arm
00058 {
00059     L4_UTCB_EXCEPTION_REGS_SIZE    = sizeof(l4_exc_regs_t) / sizeof(l4_umword_t),
00060     L4_UTCB_GENERIC_DATA_SIZE      = 63,
00061     L4_UTCB_GENERIC_BUFFERS_SIZE   = 58,
00062
00063     L4_UTCB_MSG_REGS_OFFSET        = 0,
00064     L4_UTCB_BUF_REGS_OFFSET        = 64 * sizeof(l4_umword_t),
00065     L4_UTCB_THREAD_REGS_OFFSET     = 123 * sizeof(l4_umword_t),
00066
00067     L4_UTCB_INHERIT_FPU            = 1UL < 24,
00068
00069     L4_UTCB_OFFSET                 = 512,
00070 };
00071
00072 #include_next <l4/sys/utcb.h>
00073
00074 /*
00075  * =====
00076  * Implementations.
00077  */
00078
00079 #ifdef __GNUC__
00080 L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW
00081 {
00082     register l4_utcb_t *utcb __asm__ ("r0");
00083     __asm__ ("mov lr, pc\n"
00084             "mov pc, #0xffffffff0\n"
00085             : "=r"(utcb) : : "lr");
00086     return utcb;
00087 }
00088 #endif
00089
00090 L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u) L4_NOTHROW
00091 {
00092     return u->pc;
00093 }
00094
00095 L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW
00096 {
00097     u->pc = pc;
00098 }
00099
00100 L4_INLINE l4_umword_t l4_utcb_exc_typeval(l4_exc_regs_t const *u) L4_NOTHROW
00101 {
00102     return u->err >> 26;
00103 }
00104

```

```

00105 L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u) L4_NOTHROW
00106 {
00107     return ((u->err >> 26) & 0x30) == 0x20;
00108 }
00109
00110 L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u) L4_NOTHROW
00111 {
00112     return (u->pfa & ~7UL) | ((u->err >> 5) & 2);
00113 }
00114
00115 L4_INLINE int l4_utcb_exc_is_ex_regs_exception(l4_exc_regs_t const *u) L4_NOTHROW
00116 {
00117     return l4_utcb_exc_typeval(u) == 0x3e;
00118 }
00119
00120 #endif /* ! __L4_SYS__INCLUDE__ARCH_ARM__UTCB_H__ */

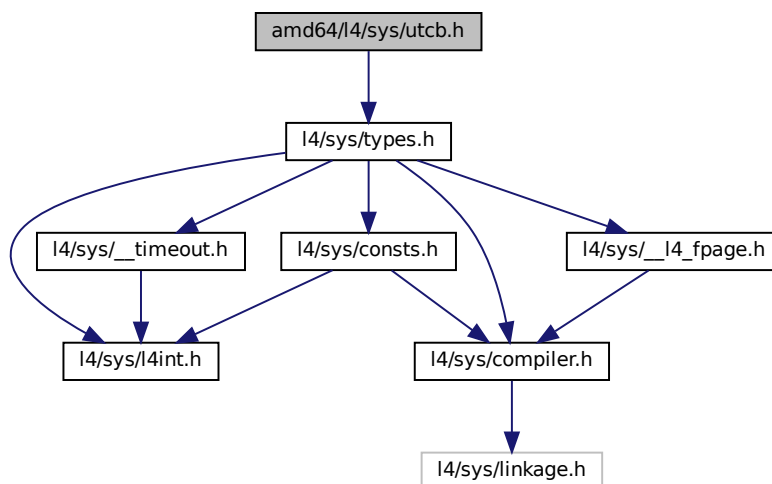
```

16.367 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\(\)](#).

16.367.1 Detailed Description

UTCB definitions for amd64.

Definition in file [utcb.h](#).

16.368 utcb.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 /*****
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__ */

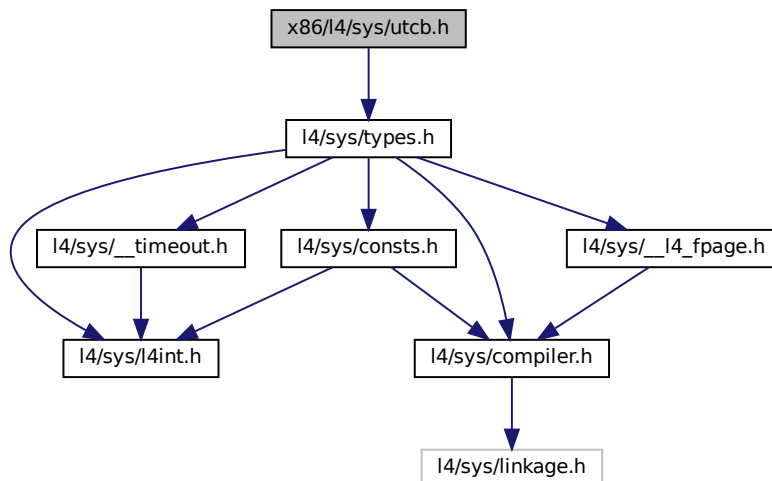
```

16.369 x86/l4/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()`.

16.369.1 Detailed Description

UTCB definitions for X86.

Definition in file `utcb.h`.

16.370 utcb.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 *      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__ */

```

16.371 l4/sys/vcon File Reference

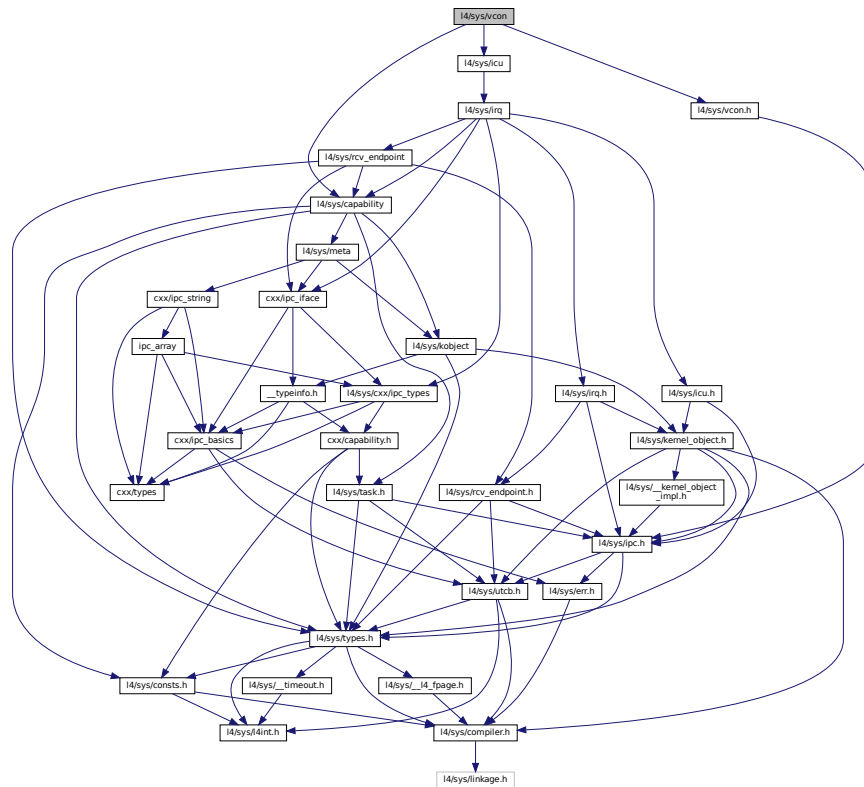
Virtual console interface.

```

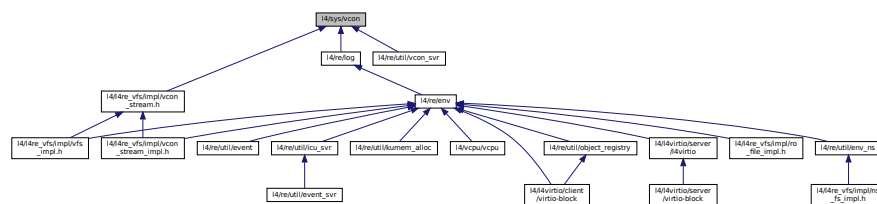
#include <l4/sys/icu>
#include <l4/sys/vcon.h>

```

```
#include <linux/sys/capability>
```



This graph shows which files directly or indirectly include this file:



Data Structures

- class `L4::Vcon`
C++ L4 Vcon interface.

Namespaces

- L4
L4 low-level kernel interface.

16.371.1 Detailed Description

Virtual console interface.

Definition in file [vcon](#).

16.372 vcon

```

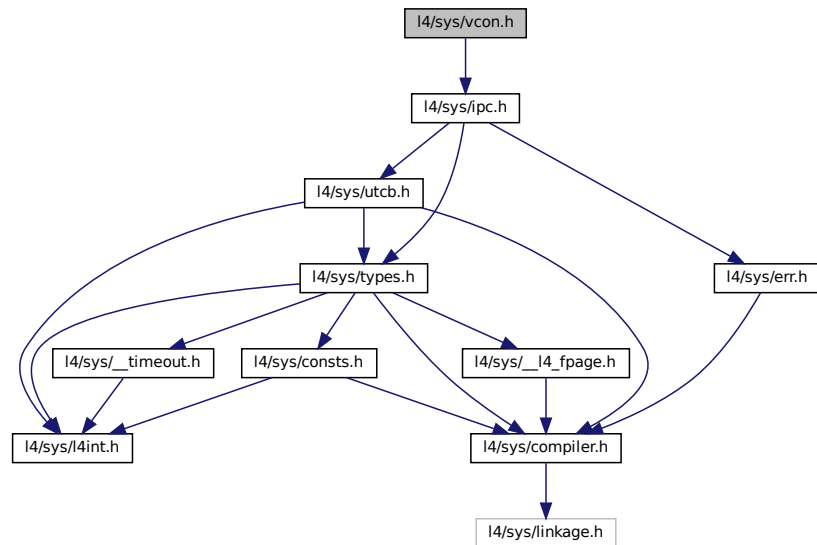
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *                Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *                Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *                economic rights: 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
00043 class Vcon :
00044     public Kobject_t<Vcon, Icu, L4_PROTO_LOG>
00045 {
00046 public:
00062     l4_msgtag_t
00063     send(char const *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const noexcept
00064     { return l4_vcon_send_u(cap(), buf, size, utcb); }
00065
00076     long
00077     write(char const *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const noexcept
00078     { return l4_vcon_write_u(cap(), buf, size, utcb); }
00079
00093     int
00094     read(char *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const noexcept
00095     { return l4_vcon_read_u(cap(), buf, size, utcb); }
00096
00118     int
00119     read_with_flags(char *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const noexcept
00120     { return l4_vcon_read_with_flags_u(cap(), buf, size, utcb); }
00121
00131     l4_msgtag_t
00132     set_attr(l4_vcon_attr_t const *attr, l4_utcb_t *utcb = l4_utcb()) const noexcept
00133     { return l4_vcon_set_attr_u(cap(), attr, utcb); }
00134
00144     l4_msgtag_t
00145     get_attr(l4_vcon_attr_t *attr, l4_utcb_t *utcb = l4_utcb()) const noexcept
00146     { return l4_vcon_get_attr_u(cap(), attr, utcb); }
00147
00148     typedef L4::Typeid::Raw_ipc<Vcon> Rpcs;
00149 };
00150
00151 }

```

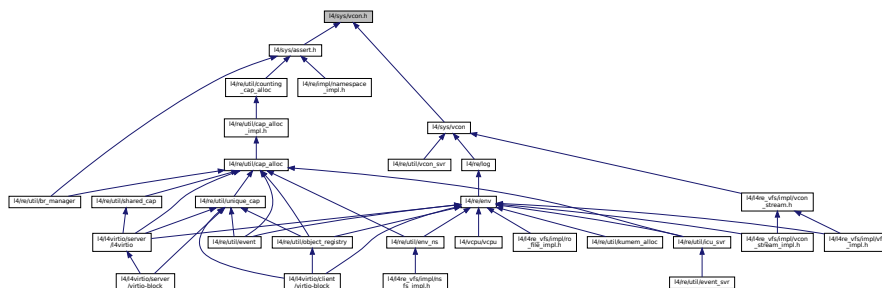
16.373 l4/sys/vcon.h File Reference

Virtual console interface.

```
#include <linux/sys/ipc.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.

16.373.1 Detailed Description

Virtual console interface.

Definition in file [vcon.h](#).

16.373.2 Enumeration Type Documentation

16.373.2.1 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 167 of file [vcon.h](#).

16.374 vcon.h

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *           Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *           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
00056 L4_INLINE l4_msgtag_t
00057 l4_vcon_send(l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW;
00058
00065 L4_INLINE l4_msgtag_t
00066 l4_vcon_send_u(l4_cap_idx_t vcon, char const *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW;
00067
00079 L4_INLINE long
00080 l4_vcon_write(l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW;
00081
00088 L4_INLINE long
00089 l4_vcon_write_u(l4_cap_idx_t vcon, char const *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW;
00090
00095 enum L4_vcon_size_consts
00096 {
00098     L4_VCON_WRITE_SIZE = (L4_UTCB_GENERIC_DATA_SIZE - 2) * sizeof(l4_umword_t),
00100     L4_VCON_READ_SIZE  = (L4_UTCB_GENERIC_DATA_SIZE - 1) * sizeof(l4_umword_t),
00101 };
00102
00118 L4_INLINE int
00119 l4_vcon_read(l4_cap_idx_t vcon, char *buf, unsigned size) L4_NOTHROW;
00120
00127 L4_INLINE int
00128 l4_vcon_read_u(l4_cap_idx_t vcon, char *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW;
00129

```

```

00154 L4_INLINE int
00155 l4_vcon_read_with_flags(l4_cap_idx_t vcon, char *buf, unsigned size) L4_NOTHROW;
00156
00160 L4_INLINE int
00161 l4_vcon_read_with_flags_u(l4_cap_idx_t vcon, char *buf, unsigned size,
00162                          l4_utcb_t *utcb) L4_NOTHROW;
00163
00167 enum L4_vcon_read_flags
00168 {
00169     L4_VCON_READ_SIZE_MASK = 0x3fffffff,
00170     L4_VCON_READ_STAT_BREAK = 1 « 30,
00171     L4_VCON_READ_STAT_DONE = 1 « 31,
00172 };
00173
00178 typedef struct l4_vcon_attr_t
00179 {
00180     l4_umword_t i_flags;
00181     l4_umword_t o_flags;
00182     l4_umword_t l_flags;
00183 } l4_vcon_attr_t;
00184
00189 enum L4_vcon_i_flags
00190 {
00191     L4_VCON_INLCR = 000100,
00192     L4_VCON_IGNCR = 000200,
00193     L4_VCON_ICRNL = 000400,
00194 };
00195
00200 enum L4_vcon_o_flags
00201 {
00202     L4_VCON_ONLCR = 000004,
00203     L4_VCON_OCRNL = 000010,
00204     L4_VCON_ONLRET = 000040,
00205 };
00206
00211 enum L4_vcon_l_flags
00212 {
00213     L4_VCON_ICANON = 000002,
00214     L4_VCON_ECHO = 000010,
00215 };
00216
00225 L4_INLINE l4_msgtag_t
00226 l4_vcon_set_attr(l4_cap_idx_t vcon, l4_vcon_attr_t const *attr) L4_NOTHROW;
00227
00234 L4_INLINE l4_msgtag_t
00235 l4_vcon_set_attr_u(l4_cap_idx_t vcon, l4_vcon_attr_t const *attr,
00236                   l4_utcb_t *utcb) L4_NOTHROW;
00237
00246 L4_INLINE l4_msgtag_t
00247 l4_vcon_get_attr(l4_cap_idx_t vcon, l4_vcon_attr_t *attr) L4_NOTHROW;
00248
00255 L4_INLINE l4_msgtag_t
00256 l4_vcon_get_attr_u(l4_cap_idx_t vcon, l4_vcon_attr_t *attr,
00257                   l4_utcb_t *utcb) L4_NOTHROW;
00258
00259
00264 enum L4_vcon_ops
00265 {
00266     L4_VCON_WRITE_OP = 0UL,
00267     L4_VCON_READ_OP = 1UL,
00268     L4_VCON_SET_ATTR_OP = 2UL,
00269     L4_VCON_GET_ATTR_OP = 3UL,
00270 };
00271
00272 /***** Implementations *****/
00273
00274 L4_INLINE l4_msgtag_t
00275 l4_vcon_send_u(l4_cap_idx_t vcon, char const *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW
00276 {
00277     l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
00278     mr->mr[0] = L4_VCON_WRITE_OP;
00279     mr->mr[1] = size;
00280     __builtin_memcpy(&mr->mr[2], buf, size);
00281     return l4_ipc_send(vcon, utcb,
00282                       l4_msgtag(L4_PROTO_LOG, 2 + l4_bytes_to_mwords(size),
00283                                0, L4_MSGTAG_SCHEDULE),
00284                       L4_IPC_NEVER);
00285 }
00286
00287 L4_INLINE l4_msgtag_t
00288 l4_vcon_send(l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW
00289 {
00290     return l4_vcon_send_u(vcon, buf, size, l4_utcb());
00291 }
00292
00293 L4_INLINE long
00294 l4_vcon_write_u(l4_cap_idx_t vcon, char const *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW

```

```

00295 {
00296     l4_msgtag_t t;
00297
00298     if (size > L4_VCON_WRITE_SIZE)
00299         size = L4_VCON_WRITE_SIZE;
00300
00301     t = l4_vcon_send_u(vcon, buf, size, utcb);
00302     if (l4_msgtag_has_error(t))
00303         return l4_error(t);
00304
00305     return (long) size;
00306 }
00307
00308 L4_INLINE long
00309 l4_vcon_write(l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW
00310 {
00311     return l4_vcon_write_u(vcon, buf, size, l4_utcb());
00312 }
00313
00314 L4_INLINE int
00315 l4_vcon_read_with_flags_u(l4_cap_idx_t vcon, char *buf, unsigned size,
00316                          l4_utcb_t *utcb) L4_NOTHROW
00317 {
00318     int ret;
00319     unsigned r;
00320     l4_msg_regs_t *mr;
00321
00322     mr = l4_utcb_mr_u(utcb);
00323     mr->mr[0] = (size << 16) | L4_VCON_READ_OP;
00324
00325     ret = l4_error_u(l4_ipc_call(vcon, utcb,
00326                                l4_msgtag(L4_PROTO_LOG, 1, 0, 0),
00327                                L4_IPC_NEVER),
00328                    utcb);
00329     if (ret < 0)
00330         return ret;
00331
00332     r = mr->mr[0] & L4_VCON_READ_SIZE_MASK;
00333
00334     if (!(mr->mr[0] & L4_VCON_READ_STAT_DONE)) // !eof
00335         ret = size + 1;
00336     else if (r < size)
00337         ret = r;
00338     else
00339         ret = size;
00340
00341     if (L4_LIKELY(buf != NULL))
00342         __builtin_memcpy(buf, &mr->mr[1], r < size ? r : size);
00343
00344     return ret | (mr->mr[0] & ~(L4_VCON_READ_STAT_DONE | L4_VCON_READ_SIZE_MASK));
00345 }
00346
00347 L4_INLINE int
00348 l4_vcon_read_with_flags(l4_cap_idx_t vcon, char *buf, unsigned size) L4_NOTHROW
00349 {
00350     return l4_vcon_read_with_flags_u(vcon, buf, size, l4_utcb());
00351 }
00352
00353 L4_INLINE int
00354 l4_vcon_read_u(l4_cap_idx_t vcon, char *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW
00355 {
00356     int r = l4_vcon_read_with_flags_u(vcon, buf, size, utcb);
00357     if (r < 0)
00358         return r;
00359
00360     return r & L4_VCON_READ_SIZE_MASK;
00361 }
00362
00363 L4_INLINE int
00364 l4_vcon_read(l4_cap_idx_t vcon, char *buf, unsigned size) L4_NOTHROW
00365 {
00366     return l4_vcon_read_u(vcon, buf, size, l4_utcb());
00367 }
00368
00369 L4_INLINE l4_msgtag_t
00370 l4_vcon_set_attr_u(l4_cap_idx_t vcon, l4_vcon_attr_t const *attr,
00371                  l4_utcb_t *utcb) L4_NOTHROW
00372 {
00373     l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
00374
00375     mr->mr[0] = L4_VCON_SET_ATTR_OP;
00376     __builtin_memcpy(&mr->mr[1], attr, sizeof(*attr));
00377
00378     return l4_ipc_call(vcon, utcb,
00379                      l4_msgtag(L4_PROTO_LOG, 4, 0, 0),
00380                      L4_IPC_NEVER);
00381 }

```

```

00382
00383 L4_INLINE l4_msgtag_t
00384 l4_vcon_set_attr(l4_cap_idx_t vcon, l4_vcon_attr_t const *attr) L4_NOTHROW
00385 {
00386     return l4_vcon_set_attr_u(vcon, attr, l4_utcb());
00387 }
00388
00389 L4_INLINE l4_msgtag_t
00390 l4_vcon_get_attr_u(l4_cap_idx_t vcon, l4_vcon_attr_t *attr,
00391                  l4_utcb_t *utcb) L4_NOTHROW
00392 {
00393     l4_msgtag_t res;
00394     l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
00395
00396     mr->mr[0] = L4_VCON_GET_ATTR_OP;
00397
00398     res = l4_ipc_call(vcon, utcb,
00399                     l4_msgtag(L4_PROTO_LOG, 1, 0, 0),
00400                     L4_IPC_NEVER);
00401     if (l4_error_u(res, utcb) >= 0)
00402         __builtin_memcpy(attr, &mr->mr[1], sizeof(*attr));
00403
00404     return res;
00405 }
00406
00407 L4_INLINE l4_msgtag_t
00408 l4_vcon_get_attr(l4_cap_idx_t vcon, l4_vcon_attr_t *attr) L4_NOTHROW
00409 {
00410     return l4_vcon_get_attr_u(vcon, attr, l4_utcb());
00411 }

```

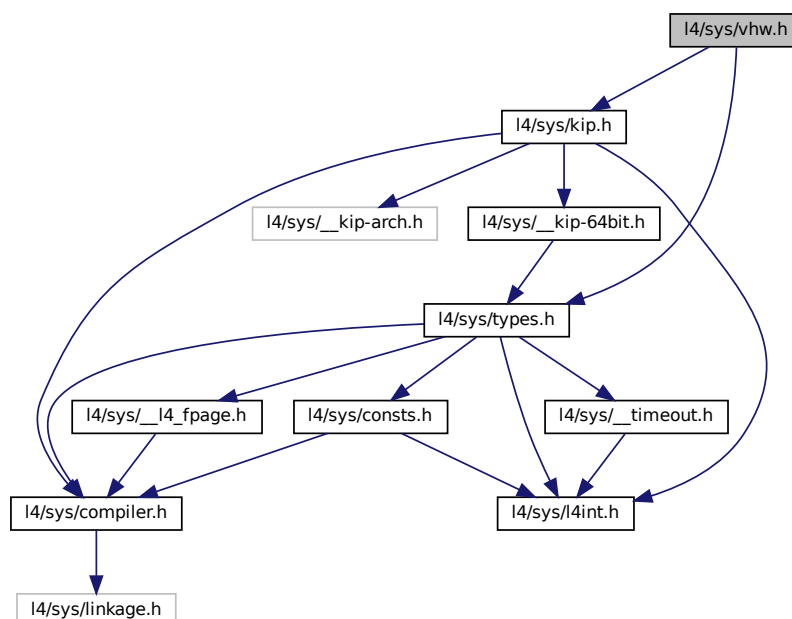
16.375 l4/sys/vhw.h File Reference

Descriptors for virtual hardware (under UX).

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

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

Include dependency graph for vhw.h:



Data Structures

- struct [l4_vhw_entry](#)
Description of a device.
- struct [l4_vhw_descriptor](#)
Virtual hardware devices description.

Enumerations

- enum [l4_vhw_entry_type](#) { [L4_TYPE_VHW_NONE](#), [L4_TYPE_VHW_FRAMEBUFFER](#), [L4_TYPE_VHW_INPUT](#), [L4_TYPE_VHW_NET](#) }
Type of device.

16.375.1 Detailed Description

Descriptors for virtual hardware (under UX).

Definition in file [vhw.h](#).

16.376 vhw.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  *      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  */
00022 #ifndef _L4_SYS_VHW_H
00023 #define _L4_SYS_VHW_H
00024
00025 #include <l4/sys/types.h>
00026 #include <l4/sys/kip.h>
00027
00028 enum l4_vhw_entry_type {
00029     L4_TYPE_VHW_NONE,
00030     L4_TYPE_VHW_FRAMEBUFFER,
00031     L4_TYPE_VHW_INPUT,
00032     L4_TYPE_VHW_NET,
00033 };
00034
00035 struct l4_vhw_entry {
00036     enum l4_vhw_entry_type type;
00037     l4_uint32_t provider_pid;
00038     l4_addr_t mem_start;
00039     l4_addr_t mem_size;
00040     l4_uint32_t irq_no;
00041     l4_uint32_t fd;
00042 };
00043
00044 struct l4_vhw_descriptor {
00045     l4_uint32_t magic;
00046     l4_uint8_t version;
00047     l4_uint8_t count;
00048     l4_uint8_t pad1;
00049     l4_uint8_t pad2;
00050 };

```

```

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 *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 */

```

16.377 l4/sys/vm File Reference

Virtualization interface.

```

#include <l4/sys/vm.h>
#include <l4/sys/task>

```


16.378 vm

```

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
00040 class Vm : public Kobject_t<Vm, Task, L4_PROTO_VM>
00041 {
00042 protected:
00043     Vm();
00044
00045 private:
00046     Vm(Vm const &);
00047     void operator = (Vm const &);
00048 };
00049
00050 };

```

16.379 l4/util/assert.h File Reference

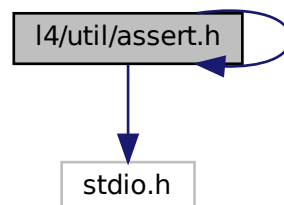
Some useful assert-style macros.

```


#include <stdio.h>
#include <assert.h>

```

Include dependency graph for assert.h:



This graph shows which files directly or indirectly include this file:

l4/util/assert.h 

16.379.1 Detailed Description

Some useful assert-style macros.

Date

09/2009

Author

Bjoern Doebel doebel@tudos.org

Definition in file [assert.h](#).

16.380 assert.h

```

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_VALID(c)     DO_NOTHING
00024 #define ASSERT_EQUAL(a,b)   DO_NOTHING
00025 #define ASSERT_NOT_EQUAL(a,b) DO_NOTHING
00026 #define ASSERT_LOWER_EQ(a,b) DO_NOTHING
00027 #define ASSERT_GREATER_EQ(a,b) DO_NOTHING
00028 #define ASSERT_BETWEEN(a,b,c) DO_NOTHING
00029 #define ASSERT_IPC_OK(i)    DO_NOTHING
00030 #define ASSERT_OK(e)         do { (void)e; } while (0)
00031 #define ASSERT_NOT_NULL(p)   DO_NOTHING
00032 #ifndef assert
00033 #define assert(cond)          DO_NOTHING
00034 #endif
00035
00036 #else // NDEBUG
00037
00038 #ifndef ASSERT_PRINTF
00039 #include <stdio.h>
00040 #define ASSERT_PRINTF printf
00041 #endif
00042 #ifndef ASSERT_ASSERT
00043 #include <assert.h>
00044 #define ASSERT_ASSERT(x) assert(x)
00045 #endif

```

```

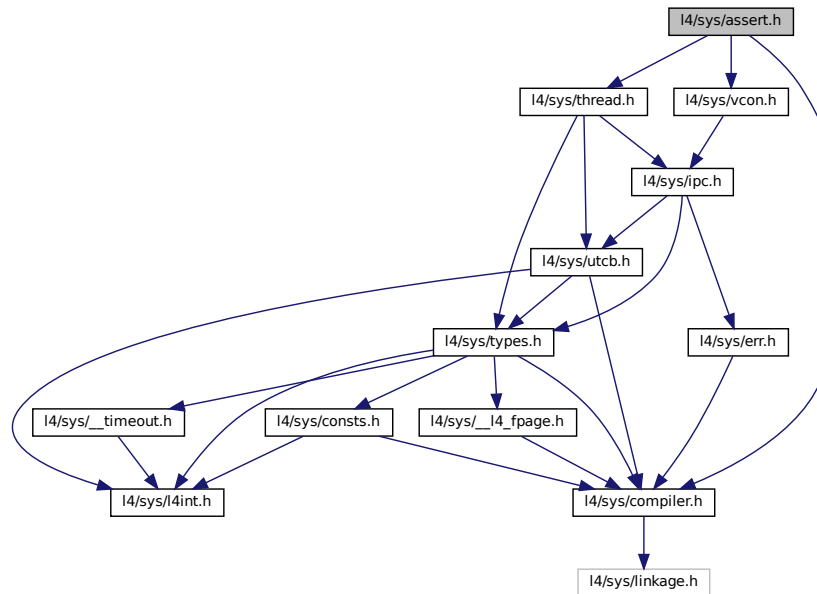
00046
00047 #define ASSERT_VALID(cap) \
00048     do { \
00049         typeof(cap) _cap = cap; \
00050         if (!l4_is_invalid_cap(_cap)) { \
00051             ASSERT_PRINTF("%s: Cap invalid.\n", __func__); \
00052             ASSERT_ASSERT(!l4_is_invalid_cap(_cap)); \
00053         } \
00054     } while (0)
00055
00056
00057 #define ASSERT_EQUAL(a, b) \
00058     do { \
00059         typeof(a) _a = a; \
00060         typeof(b) _b = b; \
00061         if (_a != _b) { \
00062             ASSERT_PRINTF("%s:\n", __func__); \
00063             ASSERT_PRINTF("    "#a" (%lx) != "#b" (%lx)\n", (unsigned long)_a, (unsigned long)_b); \
00064             ASSERT_ASSERT(_a == _b); \
00065         } \
00066     } while (0)
00067
00068
00069 #define ASSERT_NOT_EQUAL(a, b) \
00070     do { \
00071         typeof(a) _a = a; \
00072         typeof(b) _b = b; \
00073         if (_a == _b) { \
00074             ASSERT_PRINTF("%s:\n", __func__); \
00075             ASSERT_PRINTF("    "#a" (%lx) == "#b" (%lx)\n", (unsigned long)_a, (unsigned long)_b); \
00076             ASSERT_ASSERT(_a != _b); \
00077         } \
00078     } while (0)
00079
00080
00081 #define ASSERT_LOWER_EQ(val, max) \
00082     do { \
00083         typeof(val) _val = val; \
00084         typeof(max) _max = max; \
00085         if (_val > _max) { \
00086             ASSERT_PRINTF("%s:\n", __func__); \
00087             ASSERT_PRINTF("    "#val" (%lx) > "#max" (%lx)\n", (unsigned long)_val, (unsigned long)_max); \
00088             ASSERT_ASSERT(_val <= _max); \
00089         } \
00090     } while (0)
00091
00092
00093 #define ASSERT_GREATER_EQ(val, min) \
00094     do { \
00095         typeof(val) _val = val; \
00096         typeof(min) _min = min; \
00097         if (_val < _min) { \
00098             ASSERT_PRINTF("%s:\n", __func__); \
00099             ASSERT_PRINTF("    "#val" (%lx) < "#min" (%lx)\n", (unsigned long)_val, (unsigned long)_min); \
00100             ASSERT_ASSERT(_val >= _min); \
00101         } \
00102     } while (0)
00103
00104
00105 #define ASSERT_BETWEEN(val, min, max) \
00106     ASSERT_LOWER_EQ((val), (max)); \
00107     ASSERT_GREATER_EQ((val), (min));
00108
00109
00110 #define ASSERT_IPC_OK(msgtag) \
00111     do { \
00112         int _r = l4_ipc_error(msgtag, l4_utcb()); \
00113         if (_r) { \
00114             ASSERT_PRINTF("%s: IPC Error: %lx\n", __func__, _r); \
00115             ASSERT_ASSERT(_r == 0); \
00116         } \
00117     } while (0)
00118
00119 #define ASSERT_OK(val)          ASSERT_EQUAL((val), 0)
00120 #define ASSERT_NOT_NULL(ptr)    ASSERT_NOT_EQUAL((ptr), (void *)0)
00121
00122 #endif // NDEBUG

```

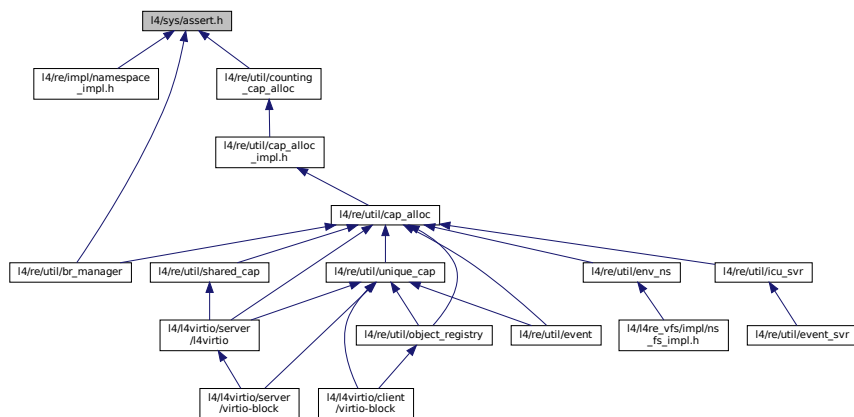
16.381 I4/sys/assert.h File Reference

Low-level assert implementation.

```
#include <l4/sys/compiler.h>
#include <l4/sys/thread.h>
#include <l4/sys/vcon.h>
Include dependency graph for assert.h:
```



This graph shows which files directly or indirectly include this file:



Macros

- `#define l4_assert(expr)`
Low-level assert.

16.381.1 Detailed Description

Low-level assert implementation.

Definition in file [assert.h](#).

16.381.2 Macro Definition Documentation

16.381.2.1 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](#).

16.382 assert.h

```
00001
00005 /*
00006  * (c) 2015 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  * 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 #ifndef NDEBUG
00024
00025 #define l4_assert(x) do { } while (0)
00026 #define l4_check(x) do { (void)(x); } while (0)
00027
00028 #else
00029
00030 #include <l4/sys/compiler.h>
00031 #include <l4/sys/thread.h>
00032 #include <l4/sys/vcon.h>
```

```

00033
00043 #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
00079     l4_assert_abort(text);
00080 }
00081
00082 #endif /* NDEBUG */

```

16.383 l4/util/atomic.h File Reference

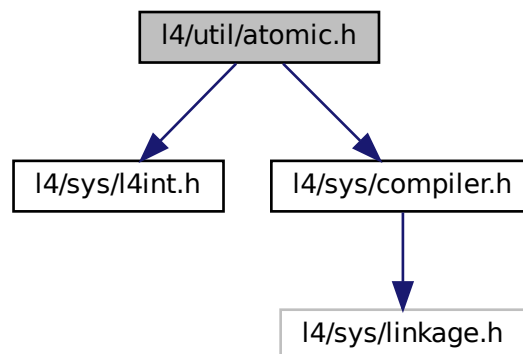
atomic operations header and generic implementations

```

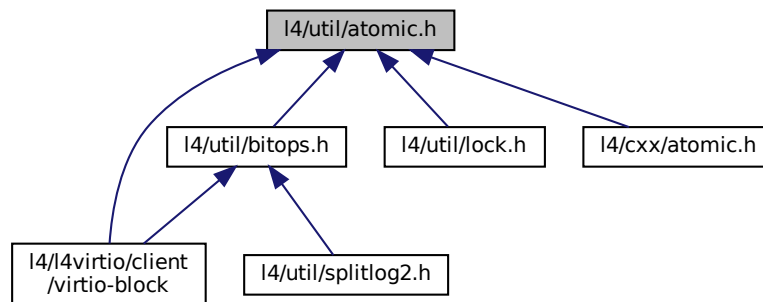
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>

```

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)

16.383.1 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](#).

16.384 atomic.h

```

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
00031 L4_INLINE int
00032 l4util_cmpxchg64(volatile l4_uint64_t * dest,
00033                 l4_uint64_t cmp_val, l4_uint64_t new_val);
00034
00035 L4_INLINE int
00036 l4util_cmpxchg32(volatile l4_uint32_t * dest,
00037                  l4_uint32_t cmp_val, l4_uint32_t new_val);
00038
00039 L4_INLINE int
00040 l4util_cmpxchg16(volatile l4_uint16_t * dest,
00041                  l4_uint16_t cmp_val, l4_uint16_t new_val);
00042
00043 L4_INLINE int
00044 l4util_cmpxchg8(volatile l4_uint8_t * dest,
00045                  l4_uint8_t cmp_val, l4_uint8_t new_val);
00046
00047 L4_INLINE int
00048 l4util_cmpxchg(volatile l4_umword_t * dest,
00049                 l4_umword_t cmp_val, l4_umword_t new_val);
00050
00051 L4_INLINE l4_uint32_t
00052 l4util_xchg32(volatile l4_uint32_t * dest, l4_uint32_t val);
00053
00054 L4_INLINE l4_uint16_t
00055 l4util_xchg16(volatile l4_uint16_t * dest, l4_uint16_t val);
00056
00057 L4_INLINE l4_uint8_t
00058 l4util_xchg8(volatile l4_uint8_t * dest, l4_uint8_t val);
00059
00060 L4_INLINE l4_umword_t
00061 l4util_xchg(volatile l4_umword_t * dest, l4_umword_t val);
00062
00063 L4_INLINE void
00064 l4util_add8(volatile l4_uint8_t *dest, l4_uint8_t val);
00065 L4_INLINE void
00066 l4util_add16(volatile l4_uint16_t *dest, l4_uint16_t val);
00067 L4_INLINE void
00068 l4util_add32(volatile l4_uint32_t *dest, l4_uint32_t val);
00069 L4_INLINE void
00070 l4util_sub8(volatile l4_uint8_t *dest, l4_uint8_t val);
00071 L4_INLINE void
00072 l4util_sub16(volatile l4_uint16_t *dest, l4_uint16_t val);
00073 L4_INLINE void
00074 l4util_sub32(volatile l4_uint32_t *dest, l4_uint32_t val);
00075 L4_INLINE void
00076 l4util_and8(volatile l4_uint8_t *dest, l4_uint8_t val);
00077 L4_INLINE void
00078 l4util_and16(volatile l4_uint16_t *dest, l4_uint16_t val);
00079 L4_INLINE void
00080 l4util_and32(volatile l4_uint32_t *dest, l4_uint32_t val);
00081 L4_INLINE void
00082 l4util_or8(volatile l4_uint8_t *dest, l4_uint8_t val);
00083 L4_INLINE void
00084 l4util_or16(volatile l4_uint16_t *dest, l4_uint16_t val);
00085 L4_INLINE void
00086 l4util_or32(volatile l4_uint32_t *dest, l4_uint32_t val);
00087
00088 L4_INLINE l4_uint8_t
00089 l4util_add8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
00090 L4_INLINE l4_uint16_t
00091 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         14_uint8_t cmp_val, 14_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 14util_cmpxchg(volatile 14_umword_t * dest,
00368     14_umword_t cmp_val, 14_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 14_uint32_t
00375 14util_xchg32(volatile 14_uint32_t * dest, 14_uint32_t val)
00376 {
00377     return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00378 }
00379
00380 L4_INLINE 14_uint16_t
00381 14util_xchgl6(volatile 14_uint16_t * dest, 14_uint16_t val)
00382 {
00383     return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00384 }
00385
00386 L4_INLINE 14_uint8_t
00387 14util_xchg8(volatile 14_uint8_t * dest, 14_uint8_t val)
00388 {
00389     return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00390 }
00391
00392 L4_INLINE 14_umword_t
00393 14util_xchg(volatile 14_umword_t * dest, 14_umword_t val)
00394 {
00395     return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00396 }
00397
00398 L4_INLINE void
00399 14util_inc8(volatile 14_uint8_t *dest)
00400 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00401
00402 L4_INLINE void
00403 14util_inc16(volatile 14_uint16_t *dest)
00404 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00405
00406 L4_INLINE void
00407 14util_inc32(volatile 14_uint32_t *dest)
00408 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00409
00410 L4_INLINE void
00411 14util_atomic_inc(volatile long *dest)
00412 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00413
00414 L4_INLINE void
00415 14util_dec8(volatile 14_uint8_t *dest)
00416 { __atomic_fetch_sub(dest, 1, __ATOMIC_SEQ_CST); }
00417
00418 L4_INLINE void
00419 14util_dec16(volatile 14_uint16_t *dest)
00420 { __atomic_fetch_sub(dest, 1, __ATOMIC_SEQ_CST); }
00421
00422 L4_INLINE void
00423 14util_dec32(volatile 14_uint32_t *dest)
00424 { __atomic_fetch_sub(dest, 1, __ATOMIC_SEQ_CST); }
00425
00426
00427 L4_INLINE 14_uint8_t
00428 14util_inc8_res(volatile 14_uint8_t *dest)
00429 { return __atomic_add_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00430
00431 L4_INLINE 14_uint16_t
00432 14util_inc16_res(volatile 14_uint16_t *dest)
00433 { return __atomic_add_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00434
00435 L4_INLINE 14_uint32_t
00436 14util_inc32_res(volatile 14_uint32_t *dest)
00437 { return __atomic_add_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00438
00439 L4_INLINE 14_uint8_t
00440 14util_dec8_res(volatile 14_uint8_t *dest)
00441 { return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00442
00443 L4_INLINE 14_uint16_t
00444 14util_dec16_res(volatile 14_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__ */

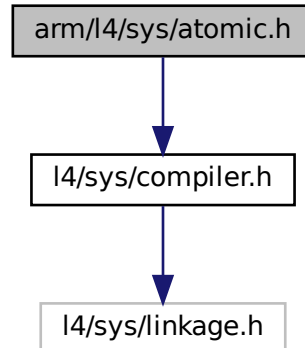
```

16.385 arm/l4/sys/atomic.h File Reference

Atomic memory modifications.

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

Include dependency graph for atomic.h:



16.385.1 Detailed Description

Atomic memory modifications.

Definition in file [atomic.h](#).

16.386 atomic.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/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;

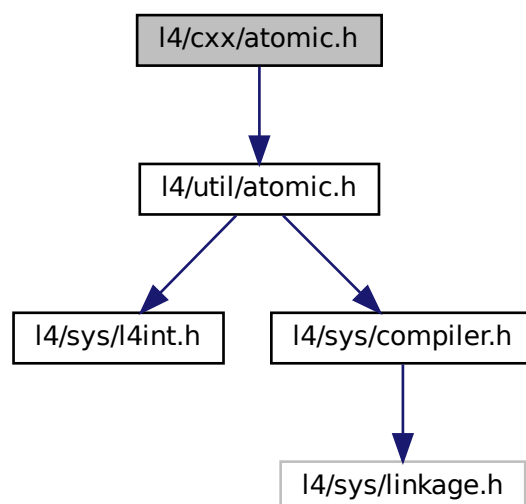
```

16.387 l4/cxx/atomic.h File Reference

Atomic template.

```
#include <l4/util/atomic.h>
```

Include dependency graph for atomic.h:



Namespaces

- [L4](#)

[L4](#) low-level kernel interface.

16.387.1 Detailed Description

Atomic template.

Definition in file [atomic.h](#).

16.388 atomic.h

```

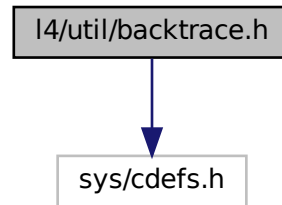
00001
00002 /*
00003  * (c) 2004-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * 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/util/atomic.h>
00023
00024 extern "C" void ____error_compare_and_swap_does_not_support_3_bytes____();
00025 extern "C" void ____error_compare_and_swap_does_not_support_more_than_4_bytes____();
00026
00027 namespace L4
00028 {
00029     template< typename X >
00030     inline int compare_and_swap(X volatile *dst, X old_val, X new_val)
00031     {
00032         switch (sizeof(X))
00033         {
00034             case 1:
00035                 return l4util_cmpxchg8((l4_uint8_t volatile*)dst, old_val, new_val);
00036             case 2:
00037                 return l4util_cmpxchg16((l4_uint16_t volatile *)dst, old_val, new_val);
00038             case 3: ____error_compare_and_swap_does_not_support_3_bytes____();
00039             case 4:
00040                 return l4util_cmpxchg32((l4_uint32_t volatile*)dst, old_val, new_val);
00041             default:
00042                 ____error_compare_and_swap_does_not_support_more_than_4_bytes____();
00043         }
00044         return 0;
00045     }
00046 }
00047
00048 }
```

16.389 l4/util/backtrace.h File Reference

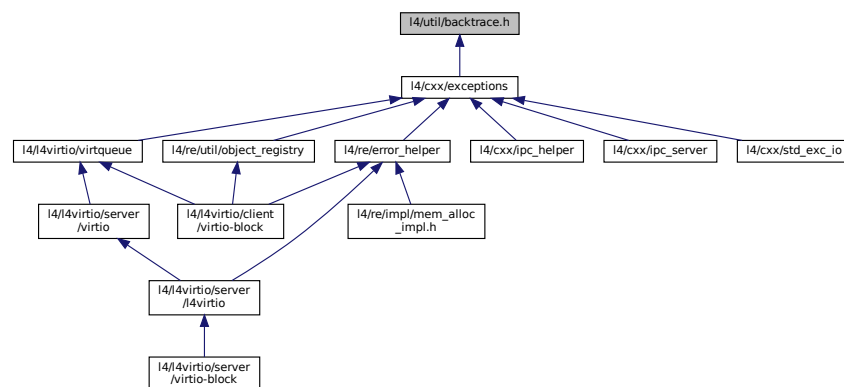
Backtrace.

```
#include <sys/cdefs.h>
```

Include dependency graph for backtrace.h:



This graph shows which files directly or indirectly include this file:



Functions

- int [l4util_backtrace](#) (void **pc_array, int max_len)
Fill backtrace structure.

16.389.1 Detailed Description

Backtrace.

Definition in file [backtrace.h](#).

16.389.2 Function Documentation

16.389.2.1 l4util_backtrace()

```
int l4util_backtrace (
    void ** pc_array,
    int max_len )
```

Fill backtrace structure.

Parameters

<i>pc_array</i>	Array of instruction pointers.
<i>max_len</i>	Length of array.

Returns

Number of entries

16.390 backtrace.h

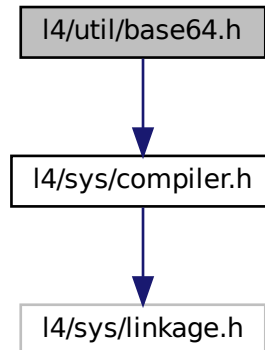
```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU Lesser General Public License 2.1.
00011  * Please see the COPYING-LGPL-2.1 file for details.
00012  */
00013 #pragma once
00014
00015 #include <sys/cdefs.h>
00016
00017 __BEGIN_DECLS
00018
00026 int l4util_backtrace(void **pc_array, int max_len);
00027
00028 __END_DECLS
```

16.391 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

16.391.1 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](#).

16.392 base64.h

```

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
00030
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

```

16.393 l4/util/bitops.h File Reference

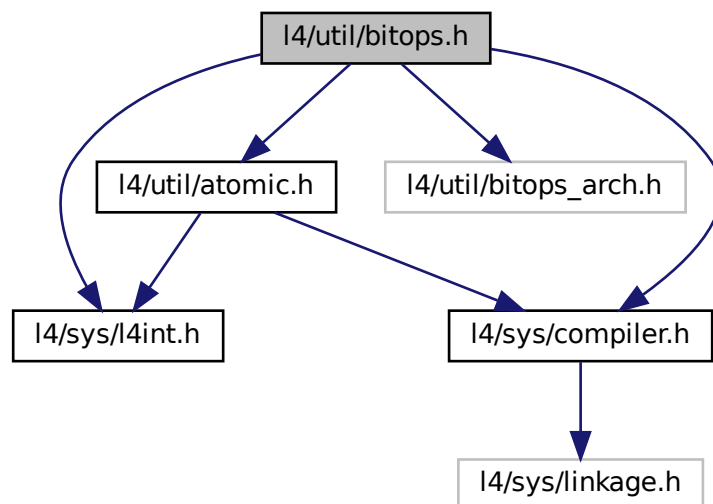
bit manipulation functions

```

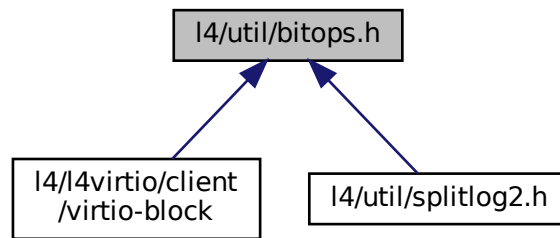
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
#include <l4/util/bitops_arch.h>
#include <l4/util/atomic.h>

```

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.

16.393.1 Detailed Description

bit manipulation functions

Date

07/03/2001

Author

Lars Reuther reuther@os.inf.tu-dresden.de

Definition in file [bitops.h](#).

16.394 bitops.h

```

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
00164 L4_INLINE int
00165 l4util_find_first_set_bit(const void * dest, l4_size_t size);
00166

```

```

00177 L4_INLINE int
00178 l4util_find_first_zero_bit(const void * dest, l4_size_t size);
00179
00180
00189 L4_INLINE int
00190 l4util_next_power2(unsigned long val);
00191
00192 EXTERN_C_END
00193
00194 /***** Implementation of specific version *****/
00195
00196 /*****
00197
00198 #include <l4/util/bitops_arch.h>
00199
00200 /***** Generic implementations *****/
00201
00202 /*****
00203
00204 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_SET_BIT
00205 #include <l4/util/atomic.h>
00206 L4_INLINE void
00207 l4util_set_bit(int b, volatile l4_umword_t * dest)
00208 {
00209     l4_umword_t oldval, newval;
00210
00211     dest += b / (sizeof(*dest) * 8); /* advance dest to the proper element */
00212     b    &= sizeof(*dest) * 8 - 1; /* modulo; cut off all upper bits */
00213
00214     do
00215     {
00216         oldval = *dest;
00217         newval = oldval | (1UL < b);
00218     }
00219     while (!l4util_cmpxchg(dest, oldval, newval));
00220 }
00221 #endif
00222
00223 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_CLEAR_BIT
00224 #include <l4/util/atomic.h>
00225 L4_INLINE void
00226 l4util_clear_bit(int b, volatile l4_umword_t * dest)
00227 {
00228     l4_umword_t oldval, newval;
00229
00230     dest += b / (sizeof(*dest) * 8);
00231     b    &= sizeof(*dest) * 8 - 1;
00232
00233     do
00234     {
00235         oldval = *dest;
00236         newval = oldval & ~(1UL < b);
00237     }
00238     while (!l4util_cmpxchg(dest, oldval, newval));
00239 }
00240 #endif
00241
00242 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_TEST_BIT
00243 L4_INLINE int
00244 l4util_test_bit(int b, const volatile l4_umword_t * dest)
00245 {
00246     dest += b / (sizeof(*dest) * 8);
00247     b    &= sizeof(*dest) * 8 - 1;
00248
00249     return (*dest >> b) & 1;
00250 }
00251 #endif
00252
00253 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_SET
00254 #include <l4/util/atomic.h>
00255 L4_INLINE int
00256 l4util_bts(int b, volatile l4_umword_t * dest)
00257 {
00258     l4_umword_t oldval, newval;
00259
00260     dest += b / (sizeof(*dest) * 8);
00261     b    &= sizeof(*dest) * 8 - 1;
00262
00263     do
00264     {
00265         oldval = *dest;
00266         newval = oldval | (1UL < b);
00267     }
00268     while (!l4util_cmpxchg(dest, oldval, newval));
00269
00270     /* Return old bit */
00271     return (oldval >> b) & 1;

```

```

00272 }
00273 #endif
00274
00275 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_RESET
00276 #include <l4util/atomic.h>
00277 L4_INLINE int
00278 l4util_btr(int b, volatile l4_umword_t * dest)
00279 {
00280     l4_umword_t oldval, newval;
00281
00282     dest += b / (sizeof(*dest) * 8);
00283     b    &= sizeof(*dest) * 8 - 1;
00284
00285     do
00286     {
00287         oldval = *dest;
00288         newval = oldval & ~(1UL << b);
00289     }
00290     while (!l4util_cmpxchg(dest, oldval, newval));
00291
00292     /* Return old bit */
00293     return (oldval >> b) & 1;
00294 }
00295 #endif
00296
00297 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_REVERSE
00298 L4_INLINE int
00299 l4util_bsr(l4_umword_t word)
00300 {
00301     int i;
00302
00303     if (!word)
00304         return -1;
00305
00306     for (i = 8 * sizeof(word) - 1; i >= 0; i--)
00307         if ((1UL << i) & word)
00308             return i;
00309
00310     return -1;
00311 }
00312 #endif
00313
00314 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_FORWARD
00315 L4_INLINE int
00316 l4util_bsf(l4_umword_t word)
00317 {
00318     unsigned int i;
00319
00320     if (!word)
00321         return -1;
00322
00323     for (i = 0; i < sizeof(word) * 8; i++)
00324         if ((1UL << i) & word)
00325             return i;
00326
00327     return -1;
00328 }
00329 #endif
00330
00331 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_ZERO_BIT
00332 L4_INLINE int
00333 l4util_find_first_zero_bit(const void * dest, l4_size_t size)
00334 {
00335     l4_size_t i, j;
00336     unsigned long *v = (unsigned long*)dest;
00337
00338     if (!size)
00339         return 0;
00340
00341     size = (size + 31) & ~0x1f; /* Grmbl: adapt to x86 implementation... */
00342
00343     for (i = j = 0; i < size; i++, j++)
00344     {
00345         if (j >= sizeof(*v) * 8)
00346         {
00347             j = 0;
00348             v++;
00349         }
00350         if (!(1UL << j) & *v)
00351             return i;
00352     }
00353     return size + 1;
00354 }
00355 #endif
00356
00357 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_COMPLEMENT_BIT
00358 L4_INLINE void

```

```

00359 l4util_complement_bit(int b, volatile l4_umword_t * dest)
00360 {
00361     dest += b / (sizeof(*dest) * 8);
00362     b     &= sizeof(*dest) * 8 - 1;
00363
00364     *dest ^= 1UL << b;
00365 }
00366 #endif
00367
00368 /*
00369  * Adapted from:
00370  * http://en.wikipedia.org/wiki/Power\_of\_two#Algorithm\_to\_find\_the\_next-highest\_power\_of\_two
00371  */
00372 L4_INLINE int
00373 l4util_next_power2(unsigned long val)
00374 {
00375     unsigned i;
00376
00377     if (val == 0)
00378         return 1;
00379
00380     val--;
00381     for (i=1; i < sizeof(unsigned long)*8; i<=1)
00382         val = val | val >> i;
00383
00384     return val+1;
00385 }
00386
00387
00388 /* Non-implemented version, catch with a linker warning */
00389
00390 extern int __this_l4util_bitops_function_is_not_implemented_for_this_arch__sorry(void);
00391
00392 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_COMPLEMENT
00393 L4_INLINE int
00394 l4util_btc(int b, volatile l4_umword_t * dest)
00395 { (void)b; (void)dest; __this_l4util_bitops_function_is_not_implemented_for_this_arch__sorry(); return
0; }
00396 #endif
00397
00398 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_SET_BIT
00399 L4_INLINE int
00400 l4util_find_first_set_bit(const void * dest, l4_size_t size)
00401 { (void)dest; (void)size; __this_l4util_bitops_function_is_not_implemented_for_this_arch__sorry();
    return 0; }
00402 #endif
00403
00404 #endif /* ! __L4UTIL__INCLUDE__BITOPS_H__ */

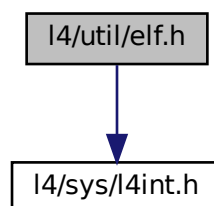
```

16.395 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 - figure 1-9, page 1-9.
- 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_Sym](#)
ELF32 symbol table entry.
- struct [Elf64_Sym](#)
ELF64 symbol table entry.

ELF types

- #define **ElfW**(type) [_ElfW](#)(Elf, 32, type)
- #define [__ElfW](#)(e, w, t) [__ElfW](#)(e, w, [_##t](#))
- #define [__ElfW](#)(e, w, t) [e##w##t](#)
- #define [EI_NIDENT](#) 16
number of characters
- #define [EI_CLASS](#) 4
ELF class byte index.
- #define [ELFCLASSNONE](#) 0
Invalid ELF class.
- #define [ELFCLASS32](#) 1
32-bit objects
- #define [ELFCLASS64](#) 2
64-bit objects
- #define [ELFCLASSNUM](#) 3
Mask for 32-bit or 64-bit class.
- #define [EI_DATA](#) 5
Data encoding byte index.
- #define [ELFDATANONE](#) 0
Invalid data encoding.
- #define [ELFDATA2LSB](#) 1
2's complement, little endian
- #define [ELFDATA2MSB](#) 2
2's complement, big endian
- #define **ELFDATANUM** 3
- #define [EI_VERSION](#) 6

- File version byte index.*
- #define [EI_OSABI](#) 7
 - OS ABI identification.*
- #define [ELFOSABI_NONE](#) 0
 - UNIX System V ABI.*
- #define [ELFOSABI_SYSV](#) 0
 - Alias.*
- #define [ELFOSABI_HPUX](#) 1
 - HP-UX.*
- #define [ELFOSABI_NETBSD](#) 2
 - NetBSD.*
- #define [ELFOSABI_LINUX](#) 3
 - Linux.*
- #define [ELFOSABI_SOLARIS](#) 6
 - Sun Solaris.*
- #define [ELFOSABI_AIX](#) 7
 - IBM AIX.*
- #define [ELFOSABI_IRIX](#) 8
 - SGI Irix.*
- #define [ELFOSABI_FREEBSD](#) 9
 - FreeBSD.*
- #define [ELFOSABI_TRU64](#) 10
 - Compaq TRU64 UNIX.*
- #define [ELFOSABI_MODESTO](#) 11
 - Novell Modesto.*
- #define [ELFOSABI_OPENBSD](#) 12
 - OpenBSD.*
- #define [ELFOSABI_ARM](#) 97
 - ARM.*
- #define [ELFOSABI_STANDALONE](#) 255
 - Standalone (embedded) application.*
- #define [EI_ABIVERSION](#) 8
 - ABI version.*
- #define [EI_PAD](#) 9
 - Byte index of padding bytes.*
- #define [ET_NONE](#) 0
 - no file type*
- #define [ET_REL](#) 1
 - relocatable file*
- #define [ET_EXEC](#) 2
 - executable file*
- #define [ET_DYN](#) 3
 - shared object file*
- #define [ET_CORE](#) 4
 - core file*
- #define [ET_LOPROC](#) 0xff00
 - processor-specific*
- #define [ET_HIPROC](#) 0xffff
 - processor-specific*
- #define [EM_NONE](#) 0
 - no machine*

- #define [EM_M32](#) 1
AT&T WE 32100.
- #define [EM_SPARC](#) 2
SPARC.
- #define [EM_386](#) 3
Intel 80386.
- #define [EM_68K](#) 4
Motorola 68000.
- #define [EM_88K](#) 5
Motorola 88000.
- #define [EM_860](#) 7
Intel 80860.
- #define [EM_MIPS](#) 8
MIPS RS3000 big-endian.
- #define [EM_MIPS_RS4_BE](#) 10
MIPS RS4000 big-endian.
- #define [EM_SPARC64](#) 11
SPARC 64-bit.
- #define [EM_PARISC](#) 15
HP PA-RISC.
- #define [EM_VPP500](#) 17
Fujitsu VPP500.
- #define [EM_SPARC32PLUS](#) 18
Sun's V8plus.
- #define [EM_960](#) 19
Intel 80960.
- #define [EM_PPC](#) 20
PowerPC.
- #define [EM_V800](#) 36
NEC V800.
- #define [EM_FR20](#) 37
Fujitsu FR20.
- #define [EM_RH32](#) 38
TRW RH-32.
- #define [EM_RCE](#) 39
Motorola RCE.
- #define [EM_ARM](#) 40
Advanced RISC Machines ARM.
- #define [EM_ALPHA](#) 41
Digital Alpha.
- #define [EM_SH](#) 42
Hitachi SuperH.
- #define [EM_SPARCV9](#) 43
SPARC v9 64-bit.
- #define [EM_TRICORE](#) 44
Siemens Tricore embedded processor.
- #define [EM_ARC](#) 45
Argonaut RISC Core, Argonaut Techn Inc.
- #define [EM_H8_300](#) 46
Hitachi H8/300.
- #define [EM_H8_300H](#) 47

- Hitachi H8/300H.*
- #define [EM_H8S](#) 48
- Hitachi H8/S.*
- #define [EM_H8_500](#) 49
- Hitachi H8/500.*
- #define [EM_IA_64](#) 50
- HP/Intel IA-64.*
- #define [EM_MIPS_X](#) 51
- Stanford MIPS-X.*
- #define [EM_COLDFIRE](#) 52
- Motorola Coldfire.*
- #define [EM_68HC12](#) 53
- Motorola M68HC12.*
- #define [EM_X86_64](#) 62
- Advanced Micro Devices x86-64.*
- #define [EM_PDSP](#) 63
- Sony DSP Processor.*
- #define [EM_FX66](#) 66
- Siemens FX66 microcontroller.*
- #define [EM_ST9PLUS](#) 67
- STMicroelectronics ST9+ 8/16 mc.*
- #define [EM_ST7](#) 68
- STmicroelectronics ST7 8 bit mc.*
- #define [EM_68HC16](#) 69
- Motorola MC68HC16 microcontroller.*
- #define [EM_68HC11](#) 70
- Motorola MC68HC11 microcontroller.*
- #define [EM_68HC08](#) 71
- Motorola MC68HC08 microcontroller.*
- #define [EM_68HC05](#) 72
- Motorola MC68HC05 microcontroller.*
- #define [EM_SVX](#) 73
- Silicon Graphics SVx.*
- #define [EM_ST19](#) 74
- STMicroelectronics ST19 8 bit mc.*
- #define [EM_VAX](#) 75
- Digital VAX.*
- #define [EM_CRIS](#) 76
- Axis Communications 32-bit embedded processor.*
- #define [EM_JAVELIN](#) 77
- Infineon Technologies 32-bit embedded processor.*
- #define [EM_FIREPATH](#) 78
- Element 14 64-bit DSP Processor.*
- #define [EM_ZSP](#) 79
- LSI Logic 16-bit DSP Processor.*
- #define [EM_MMIX](#) 80
- Donald Knuth's educational 64-bit processor.*
- #define [EM_HUANY](#) 81
- Harvard University machine-independent object files.*
- #define [EM_PRISM](#) 82
- SiTera Prism.*

- #define [EM_AVR](#) 83
Atmel AVR 8-bit microcontroller.
- #define [EM_FR30](#) 84
Fujitsu FR30.
- #define [EM_D10V](#) 85
Mitsubishi D10V.
- #define [EM_D30V](#) 86
Mitsubishi D30V.
- #define [EM_V850](#) 87
NEC v850.
- #define [EM_M32R](#) 88
Mitsubishi M32R.
- #define [EM_MN10300](#) 89
Matsushita MN10300.
- #define [EM_MN10200](#) 90
Matsushita MN10200.
- #define [EM_PJ](#) 91
picoJava
- #define [EM_OPENRISC](#) 92
OpenRISC 32-bit embedded processor.
- #define [EM_ARC_A5](#) 93
ARC Cores Tangent-A5.
- #define [EM_XTENSA](#) 94
Tensilica Xtensa Architecture.
- #define [EM_ALTERA_NIOS2](#) 113
Altera Nios II.
- #define [EM_AARCH64](#) 183
ARM AARCH64.
- #define [EM_TILEPRO](#) 188
Tilera TILEPro.
- #define [EM_MICROBLAZE](#) 189
Xilinx MicroBlaze.
- #define [EM_TILEGX](#) 191
Tilera TILE-Gx.
- #define **EM_NUM** 192
- #define [EV_NONE](#) 0
Invalid version.
- #define [EV_CURRENT](#) 1
Current version.
- #define [EI_MAG0](#) 0
file id
- #define [EI_MAG1](#) 1
file id
- #define [EI_MAG2](#) 2
file id
- #define [EI_MAG3](#) 3
file id
- #define [EI_CLASS](#) 4
ELF class byte index.
- #define [EI_DATA](#) 5
Data encoding byte index.

- #define **EI_VERSION** 6
File version byte index.
- #define **EI_OSABI** 7
OS ABI identification.
- #define **EI_ABIVERSION** 8
ABI version.
- #define **EI_PAD** 9
Byte index of padding bytes.
- #define **ELFMAG0** 0x7f
e_ident[EI_MAG0]
- #define **ELFMAG1** 'E'
e_ident[EI_MAG1]
- #define **ELFMAG2** 'L'
e_ident[EI_MAG2]
- #define **ELFMAG3** 'F'
e_ident[EI_MAG3]
- #define **ELFCLASSNONE** 0
Invalid ELF class.
- #define **ELFCLASS32** 1
32-bit object
- #define **ELFCLASS64** 2
64-bit object
- #define **ELFDATANONE** 0
Invalid data encoding.
- #define **ELFDATA2LSB** 1
2's complement, little endian
- #define **ELFDATA2MSB** 2
2's complement, big endian
- #define **ELFOSABI_SYSV** 0
Alias.
- #define **ELFOSABI_HPUX** 1
HP-UX.
- #define **ELFOSABI_STANDALONE** 255
Standalone (embedded) application.
- #define **SHN_UNDEF** 0
undefined section header entry
- #define **SHN_LORESERVE** 0xff00
lower bound of reserved indexes
- #define **SHN_LOPROC** 0xff00
lower bound of proc spec entr
- #define **SHN_HIPROC** 0xff1f
upper bound of proc spec entr
- #define **SHN_ABS** 0xffff
absolute values for ref
- #define **SHN_COMMON** 0xffff2
common symbols
- #define **SHN_HIRESERVE** 0xfffff
upper bound of reserved indexes
- #define **SHT_NULL** 0
- #define **SHT_PROGBITS** 1
- #define **SHT_SYMTAB** 2

- #define **SHT_STRTAB** 3
- #define **SHT_RELA** 4
- #define **SHT_HASH** 5
- #define **SHT_DYNAMIC** 6
- #define **SHT_NOTE** 7
- #define **SHT_NOBITS** 8
- #define **SHT_REL** 9
- #define **SHT_SHLIB** 10
- #define **SHT_DYNSYM** 11
- #define **SHT_INIT_ARRAY** 14
 - Array of constructors.*
- #define **SHT_FINI_ARRAY** 15
 - Array of destructors.*
- #define **SHT_PREINIT_ARRAY** 16
 - Array of pre-constructors.*
- #define **SHT_GROUP** 17
 - Section group.*
- #define **SHT_SYMTAB_SHNDX** 18
 - Extended section indeces.*
- #define **SHT_NUM** 19
 - Number of defined types.*
- #define **SHT_LOOS** 0x60000000
- #define **SHT_HIOS** 0x6fffffff
- #define **SHT_LOPROC** 0x70000000
- #define **SHT_HIPROC** 0x7fffffff
- #define **SHT_LOUSER** 0x80000000
- #define **SHT_HIUSER** 0xffffffff
- #define **SHF_WRITE** 0x1
 - writable during execution*
- #define **SHF_ALLOC** 0x2
 - section occupies virt memory*
- #define **SHF_EXECINSTR** 0x4
 - code section*
- #define **SHF_MERGE** 0x10
 - Might be merged.*
- #define **SHF_STRINGS** 0x20
 - Contains nul-terminated strings.*
- #define **SHF_INFO_LINK** 0x40
 - 'sh_info' contains SHT index*
- #define **SHF_LINK_ORDER** 0x80
 - Preserve order after combining.*
- #define **SHF_OS_NONCONFORMING** 0x100
 - Non-standard OS specific handling required.*
- #define **SHF_GROUP** 0x200
 - Section is member of a group.*
- #define **SHF_TLS** 0x400
 - Section hold thread-local data.*
- #define **SHF_MASKOS** 0x0ff00000
 - OS-specific.*
- #define **SHF_MASKPROC** 0xf0000000
 - proc spec mask*
- #define **PT_NULL** 0

- array is unused*
- #define [PT_LOAD](#) 1
 - loadable*
- #define [PT_DYNAMIC](#) 2
 - dynamic linking information*
- #define [PT_INTERP](#) 3
 - path to interpreter*
- #define [PT_NOTE](#) 4
 - auxiliary information*
- #define [PT_SHLIB](#) 5
 - reserved*
- #define [PT_PHDR](#) 6
 - location of the pht itself*
- #define [PT_TLS](#) 7
 - Thread-local storage segment.*
- #define [PT_NUM](#) 8
 - Number of defined types.*
- #define [PT_LOOS](#) 0x60000000
 - os spec.*
- #define [PT_HIOS](#) 0x6fffffff
 - os spec.*
- #define [PT_LOPROC](#) 0x70000000
 - processor spec.*
- #define [PT_HIPROC](#) 0x7fffffff
 - processor spec.*
- #define [PT_GNU_EH_FRAME](#) ([PT_LOOS](#) + 0x474e550)
 - EH frame information.*
- #define [PT_GNU_STACK](#) ([PT_LOOS](#) + 0x474e551)
 - Flags for stack.*
- #define [PT_GNU_RELRO](#) ([PT_LOOS](#) + 0x474e552)
 - Read only after reloc.*
- #define [PT_L4_STACK](#) ([PT_LOOS](#) + 0x12)
 - Address of the stack.*
- #define [PT_L4_KIP](#) ([PT_LOOS](#) + 0x13)
 - Address of the KIP.*
- #define [PT_L4_AUX](#) ([PT_LOOS](#) + 0x14)
 - Address of the AUX strcutures.*
- #define [PF_X](#) 0x1
- #define [PF_W](#) 0x2
- #define [PF_R](#) 0x4
- #define [PF_MASKOS](#) 0x0ff00000
- #define [PF_MASKPROC](#) 0x7fffffff
- #define [NT_PRSTATUS](#) 1
 - Contains copy of prstatus struct.*
- #define [NT_FPREGSET](#) 2
 - Contains copy of fpregset struct.*
- #define [NT_PRPSINFO](#) 3
 - Contains copy of prpsinfo struct.*
- #define [NT_PRXREG](#) 4
 - Contains copy of prxregset struct.*
- #define [NT_TASKSTRUCT](#) 4

- Contains copy of task structure.*
- #define [NT_PLATFORM](#) 5
 - String from sysinfo(SI_PLATFORM)*
- #define [NT_AUXV](#) 6
 - Contains copy of auxv array.*
- #define [NT_GWINDOWS](#) 7
 - Contains copy of gwindows struct.*
- #define [NT_ASRS](#) 8
 - Contains copy of asrset struct.*
- #define [NT_PSTATUS](#) 10
 - Contains copy of pstatus struct.*
- #define [NT_PSINFO](#) 13
 - Contains copy of psinfo struct.*
- #define [NT_PRCRED](#) 14
 - Contains copy of prcred struct.*
- #define [NT_UTSNAME](#) 15
 - Contains copy of utsname struct.*
- #define [NT_LWPSTATUS](#) 16
 - Contains copy of lwpstatus struct.*
- #define [NT_LWPSINFO](#) 17
 - Contains copy of lwpinfo struct.*
- #define [NT_PRFPXREG](#) 20
 - Contains copy of fprxregset struct.*
- #define [NT_VERSION](#) 1
 - Contains a version string.*
- #define [DT_NULL](#) 0
 - Dynamic Array Tags, d_tag - figure 2-10, page 2-12.*
- #define [DT_NEEDED](#) 1
 - name of a needed library*
- #define [DT_PLTRELSZ](#) 2
 - total size of relocation entry*
- #define [DT_PLTGOT](#) 3
 - address assoc with prog link table*
- #define [DT_HASH](#) 4
 - address of symbol hash table*
- #define [DT_STRTAB](#) 5
 - address of string table*
- #define [DT_SYMTAB](#) 6
 - address of symbol table*
- #define [DT_RELA](#) 7
 - address of relocation table*
- #define [DT_RELASZ](#) 8
 - total size of relocation table*
- #define [DT_RELAENT](#) 9
 - size of DT_RELA relocation entry*
- #define [DT_STRSZ](#) 10
 - size of the string table*
- #define [DT_SYMENT](#) 11
 - size of a symbol table entry*
- #define [DT_INIT](#) 12
 - address of initialization function*

- #define [DT_FINI](#) 13
address of termination function
- #define [DT_SONAME](#) 14
name of the shared object
- #define [DT_RPATH](#) 15
search library path
- #define [DT_SYMBOLIC](#) 16
alter symbol resolution algorithm
- #define [DT_REL](#) 17
address of relocation table
- #define [DT_RELSZ](#) 18
total size of DT_REL relocation table
- #define [DT_RELENT](#) 19
size of the DT_REL relocation entry
- #define [DT_PTRREL](#) 20
type of relocation entry
- #define [DT_DEBUG](#) 21
for debugging purposes
- #define [DT_TEXTREL](#) 22
at least on entry changes r/o section
- #define [DT_JMPREL](#) 23
address of relocation entries
- #define [DT_BIND_NOW](#) 24
Process relocations of object.
- #define [DT_INIT_ARRAY](#) 25
Array with addresses of init fct.
- #define [DT_FINI_ARRAY](#) 26
Array with addresses of fini fct.
- #define [DT_INIT_ARRAYSZ](#) 27
Size in bytes of DT_INIT_ARRAY.
- #define [DT_FINI_ARRAYSZ](#) 28
Size in bytes of DT_FINI_ARRAY.
- #define [DT_RUNPATH](#) 29
Library search path.
- #define [DT_FLAGS](#) 30
Flags for the object being loaded.
- #define [DT_ENCODING](#) 32
Start of encoded range.
- #define [DT_PREINIT_ARRAY](#) 32
Array with addresses of preinit fct.
- #define [DT_PREINIT_ARRAYSZ](#) 33
size in bytes of DT_PREINIT_ARRAY
- #define [DT_NUM](#) 34
Number used.
- #define [DT_LOOS](#) 0x6000000d
Start of OS-specific.
- #define [DT_HIOS](#) 0x6ffff000
End of OS-specific.
- #define [DT_LOPROC](#) 0x70000000
processor spec.
- #define [DT_HIPROC](#) 0x7fffffff

- processor spec.*
- #define **DF_ORIGIN** 0x00000001
Object may use DF_ORIGIN.
- #define **DF_SYMBOLIC** 0x00000002
Symbol resolutions starts here.
- #define **DF_TEXTREL** 0x00000004
Object contains text relocations.
- #define **DF_BIND_NOW** 0x00000008
No lazy binding for this object.
- #define **DF_STATIC_TLS** 0x00000010
Module uses the static TLS model.
- #define **DF_1_NOW** 0x00000001
Set RTLD_NOW for this object.
- #define **DF_1_GLOBAL** 0x00000002
Set RTLD_GLOBAL for this object.
- #define **DF_1_GROUP** 0x00000004
Set RTLD_GROUP for this object.
- #define **DF_1_NODELETE** 0x00000008
Set RTLD_NODELETE for this object.
- #define **DF_1_LOADFLTR** 0x00000010
Trigger filtee loading at runtime.
- #define **DF_1_INITFIRST** 0x00000020
Set RTLD_INITFIRST for this object.
- #define **DF_1_NOOPEN** 0x00000040
Set RTLD_NOOPEN for this object.
- #define **DF_1_ORIGIN** 0x00000080
\$ORIGIN must be handled.
- #define **DF_1_DIRECT** 0x00000100
Direct binding enabled.
- #define **DF_1_TRANS** 0x00000200
- #define **DF_1_INTERPOSE** 0x00000400
Object is used to interpose.
- #define **DF_1_NODEFLIB** 0x00000800
Ignore default lib search path.
- #define **DF_1_NODUMP** 0x00001000
Object can't be dldump'ed.
- #define **DF_1_CONFALT** 0x00002000
Configuration alternative created.
- #define **DF_1_ENDFILTEE** 0x00004000
Filtee terminates filters search.
- #define **DF_1_DISPRELDNE** 0x00008000
Disp reloc applied at build time.
- #define **DF_1_DISPRELPND** 0x00010000
Disp reloc applied at run-time.
- #define **DTF_1_PARINIT** 0x00000001
- #define **DTF_1_CONFEXP** 0x00000002
- #define **DF_P1_LAZYLOAD** 0x00000001
Lazyload following object.
- #define **DF_P1_GROUPPERM** 0x00000002
Symbols from next object are not generally available.
- #define **ELF32_R_SYM**(i) ((i)>>8)

- `#define ELF32_R_TYPE(i) ((unsigned char)(i))`
- `#define ELF32_R_INFO(s, t) (((s)<<8)+(unsigned char)(t))`
- `#define ELF64_R_SYM(i) ((i)>>32)`
- `#define ELF64_R_TYPE(i) ((i)&0xffffffffL)`
- `#define ELF64_R_INFO(s, t) (((s)<<32)+(t)&0xffffffffL)`
- `#define R_386_NONE 0`
none
- `#define R_386_32 1`
S + A.
- `#define R_386_PC32 2`
S + A - P.
- `#define R_386_GOT32 3`
G + A - P.
- `#define R_386_PLT32 4`
L + A - P.
- `#define R_386_COPY 5`
none
- `#define R_386_GLOB_DAT 6`
S.
- `#define R_386_JMP_SLOT 7`
S.
- `#define R_386_RELATIVE 8`
B + A.
- `#define R_386_GOTOFF 9`
S + A - GOT.
- `#define R_386_GOTPC 10`
GOT + A - P.
- `#define R_386_32PLT 11`
- `#define R_386_TLS_TPOFF 14 /* Offset in static TLS block */`
- `#define R_386_TLS_IE`
- `#define R_386_TLS_GOTIE`
- `#define R_386_TLS_LE`
- `#define R_386_TLS_GD`
- `#define R_386_TLS_LDM`
- `#define R_386_16 20`
- `#define R_386_PC16 21`
- `#define R_386_8 22`
- `#define R_386_PC8 23`
- `#define R_386_TLS_GD_32`
- `#define R_386_TLS_GD_PUSH 25 /* Tag for pushl in GD TLS code */`
- `#define R_386_TLS_GD_CALL`
- `#define R_386_TLS_GD_POP 27 /* Tag for popl in GD TLS code */`
- `#define R_386_TLS_LDM_32`
- `#define R_386_TLS_LDM_PUSH 29 /* Tag for pushl in LDM TLS code */`
- `#define R_386_TLS_LDM_CALL`
- `#define R_386_TLS_LDM_POP 31 /* Tag for popl in LDM TLS code */`
- `#define R_386_TLS_LDO_32 32 /* Offset relative to TLS block */`
- `#define R_386_TLS_IE_32`
- `#define R_386_TLS_LE_32`
- `#define R_386_TLS_DTPMOD32 35 /* ID of module containing symbol */`
- `#define R_386_TLS_DTPOFF32 36 /* Offset in TLS block */`
- `#define R_386_TLS_TPOFF32 37 /* Negated offset in static TLS block */`
- `#define R_386_NUM 38`

- #define **EF_ARM_RELEXEC** 0x01
- #define **EF_ARM_HASENTRY** 0x02
- #define **EF_ARM_INTERWORK** 0x04
- #define **EF_ARM_APCS_26** 0x08
- #define **EF_ARM_APCS_FLOAT** 0x10
- #define **EF_ARM_PIC** 0x20
- #define **EF_ARM_ALIGN8** 0x40 /* 8-bit structure alignment is in use */
- #define **EF_ARM_NEW_ABI** 0x80
- #define **EF_ARM_OLD_ABI** 0x100
- #define **EF_ARM_SYMSARESORTED** 0x04
- #define **EF_ARM_DYNSYMSUSESEGIDX** 0x08
- #define **EF_ARM_MAPSYMSFIRST** 0x10
- #define **EF_ARM_EABIMASK** 0xFF000000
- #define **EF_ARM_EABI_VERSION**(flags) ((flags) & EF_ARM_EABIMASK)
- #define **EF_ARM_EABI_UNKNOWN** 0x00000000
- #define **EF_ARM_EABI_VER1** 0x01000000
- #define **EF_ARM_EABI_VER2** 0x02000000
- #define **STT_ARM_TFUNC** 0xd
- #define **SHF_ARM_ENTRYSECT** 0x10000000 /* Section contains an entry point */
- #define **SHF_ARM_COMDEF**
- #define **PF_ARM_SB**
- #define **R_ARM_NONE** 0 /* No reloc */
- #define **R_ARM_PC24** 1 /* PC relative 26 bit branch */
- #define **R_ARM_ABS32** 2 /* Direct 32 bit */
- #define **R_ARM_REL32** 3 /* PC relative 32 bit */
- #define **R_ARM_PC13** 4
- #define **R_ARM_ABS16** 5 /* Direct 16 bit */
- #define **R_ARM_ABS12** 6 /* Direct 12 bit */
- #define **R_ARM_THM_ABS5** 7
- #define **R_ARM_ABS8** 8 /* Direct 8 bit */
- #define **R_ARM_SBREL32** 9
- #define **R_ARM_THM_PC22** 10
- #define **R_ARM_THM_PC8** 11
- #define **R_ARM AMP_VCALL9** 12
- #define **R_ARM_SWI24** 13
- #define **R_ARM_THM_SWI8** 14
- #define **R_ARM_XPC25** 15
- #define **R_ARM_THM_XPC22** 16
- #define **R_ARM_COPY** 20 /* Copy symbol at runtime */
- #define **R_ARM_GLOB_DAT** 21 /* Create GOT entry */
- #define **R_ARM_JUMP_SLOT** 22 /* Create PLT entry */
- #define **R_ARM_RELATIVE** 23 /* Adjust by program base */
- #define **R_ARM_GOTOFF** 24 /* 32 bit offset to GOT */
- #define **R_ARM_GOTPC** 25 /* 32 bit PC relative offset to GOT */
- #define **R_ARM_GOT32** 26 /* 32 bit GOT entry */
- #define **R_ARM_PLT32** 27 /* 32 bit PLT address */
- #define **R_ARM_ALU_PCREL_7_0** 32
- #define **R_ARM_ALU_PCREL_15_8** 33
- #define **R_ARM_ALU_PCREL_23_15** 34
- #define **R_ARM_LDR_SBREL_11_0** 35
- #define **R_ARM_ALU_SBREL_19_12** 36
- #define **R_ARM_ALU_SBREL_27_20** 37
- #define **R_ARM_GNU_VTENTRY** 100
- #define **R_ARM_GNU_VTINHERIT** 101
- #define **R_ARM_THM_PC11** 102 /* thumb unconditional branch */

- `#define R_ARM_THM_PC9 103` /* thumb conditional branch */
- `#define R_ARM_RXPC25 249`
- `#define R_ARM_RSBREL32 250`
- `#define R_ARM_THM_RPC22 251`
- `#define R_ARM_RREL32 252`
- `#define R_ARM_RABS22 253`
- `#define R_ARM_RPC24 254`
- `#define R_ARM_RBASE 255`
- `#define R_ARM_NUM 256`
- `#define R_X86_64_NONE 0` /* No reloc */
- `#define R_X86_64_64 1` /* Direct 64 bit */
- `#define R_X86_64_PC32 2` /* PC relative 32 bit signed */
- `#define R_X86_64_GOT32 3` /* 32 bit GOT entry */
- `#define R_X86_64_PLT32 4` /* 32 bit PLT address */
- `#define R_X86_64_COPY 5` /* Copy symbol at runtime */
- `#define R_X86_64_GLOB_DAT 6` /* Create GOT entry */
- `#define R_X86_64_JUMP_SLOT 7` /* Create PLT entry */
- `#define R_X86_64_RELATIVE 8` /* Adjust by program base */
- `#define R_X86_64_GOTPCREL`
- `#define R_X86_64_32 10` /* Direct 32 bit zero extended */
- `#define R_X86_64_32S 11` /* Direct 32 bit sign extended */
- `#define R_X86_64_16 12` /* Direct 16 bit zero extended */
- `#define R_X86_64_PC16 13` /* 16 bit sign extended pc relative */
- `#define R_X86_64_8 14` /* Direct 8 bit sign extended */
- `#define R_X86_64_PC8 15` /* 8 bit sign extended pc relative */
- `#define R_X86_64_DTPMOD64 16` /* ID of module containing symbol */
- `#define R_X86_64_DTPOFF64 17` /* Offset in module's TLS block */
- `#define R_X86_64_TPOFF64 18` /* Offset in initial TLS block */
- `#define R_X86_64_TLSD`
- `#define R_X86_64_TLSD`
- `#define R_X86_64_DTPOFF32 21` /* Offset in TLS block */
- `#define R_X86_64_GOTTPOFF`
- `#define R_X86_64_TPOFF32 23` /* Offset in initial TLS block */
- `#define R_X86_64_NUM 24`
- `#define STN_UNDEF 0`
- `#define ELF32_ST_BIND(i) ((i)>>4)`
- `#define ELF32_ST_TYPE(i) ((i)&0xf)`
- `#define ELF32_ST_INFO(b, t) (((b)<<4)+((t)&0xf))`
- `#define ELF64_ST_BIND(i) ((i)>>4)`
- `#define ELF64_ST_TYPE(i) ((i)&0xf)`
- `#define ELF64_ST_INFO(b, t) (((b)<<4)+((t)&0xf))`
- `#define STB_LOCAL 0`
not visible outside object file
- `#define STB_GLOBAL 1`
visible to all objects beeing combined
- `#define STB_WEAK 2`
resemble global symbols
- `#define STB_LOOS 10`
os specific
- `#define STB_HIOS 12`
os specific
- `#define STB_LOPROC 13`
proc specific
- `#define STB_HIPROC 15`

- proc specific*
- #define [STT_NOTYPE](#) 0
 - symbol's type not specified*
- #define [STT_OBJECT](#) 1
 - associated with a data object*
- #define [STT_FUNC](#) 2
 - associated with a function or other code*
- #define [STT_SECTION](#) 3
 - associated with a section*
- #define [STT_FILE](#) 4
 - source file name associated with object*
- #define [STT_LOOS](#) 10
 - os specific*
- #define [STT_HIOS](#) 12
 - os specific*
- #define [STT_LOPROC](#) 13
 - proc specific*
- #define [STT_HIPROC](#) 15
 - proc specific*
- enum **Elf_ATs**
- 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*
- typedef struct [Elf32_Auxv](#) **Elf32_Auxv**
- typedef struct [Elf64_Auxv](#) **Elf64_Auxv**

16.395.1 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](#).

16.395.2 Macro Definition Documentation

16.395.2.1 DF_1_DIRECT

```
#define DF_1_DIRECT 0x00000100
```

Direct binding enabled.

Definition at line [575](#) of file [elf.h](#).

16.395.2.2 DF_1_DISPRELPND

```
#define DF_1_DISPRELPND 0x00010000
```

Disp reloc applied at run-time.

Definition at line [583](#) of file [elf.h](#).

16.395.2.3 DF_1_GLOBAL

```
#define DF_1_GLOBAL 0x00000002
```

Set RTLD_GLOBAL for this object.

Definition at line [568](#) of file [elf.h](#).

16.395.2.4 DF_1_GROUP

```
#define DF_1_GROUP 0x00000004
```

Set RTLD_GROUP for this object.

Definition at line [569](#) of file [elf.h](#).

16.395.2.5 DF_1_INTERPOSE

```
#define DF_1_INTERPOSE 0x00000400
```

Object is used to interpose.

Definition at line [577](#) of file [elf.h](#).

16.395.2.6 DF_1_NODEFLIB

```
#define DF_1_NODEFLIB 0x00000800
```

Ignore default lib search path.

Definition at line [578](#) of file [elf.h](#).

16.395.2.7 DF_1_NODUMP

```
#define DF_1_NODUMP 0x00001000
```

Object can't be dldump'ed.

Definition at line 579 of file [elf.h](#).

16.395.2.8 DF_1_NOOPEN

```
#define DF_1_NOOPEN 0x00000040
```

Set RTLD_NOOPEN for this object.

Definition at line 573 of file [elf.h](#).

16.395.2.9 DF_1_NOW

```
#define DF_1_NOW 0x00000001
```

Set RTLD_NOW for this object.

Definition at line 567 of file [elf.h](#).

16.395.2.10 DF_1_ORIGIN

```
#define DF_1_ORIGIN 0x00000080
```

\$ORIGIN must be handled.

Definition at line 574 of file [elf.h](#).

16.395.2.11 DF_P1_GROUPPERM

```
#define DF_P1_GROUPPERM 0x00000002
```

Symbols from next object are not generally available.

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

16.395.2.12 DF_P1_LAZYLOAD

```
#define DF_P1_LAZYLOAD 0x00000001
```

Lazyload following object.

Definition at line 590 of file [elf.h](#).

16.395.2.13 DT_NULL

```
#define DT_NULL 0
```

Dynamic Array Tags, d_tag - figure 2-10, page 2-12.

end of _DYNAMIC array

Definition at line 518 of file [elf.h](#).

16.395.2.14 EI_CLASS [1/2]

```
#define EI_CLASS 4
```

ELF class byte index.

file class

Definition at line 294 of file [elf.h](#).

16.395.2.15 EI_CLASS [2/2]

```
#define EI_CLASS 4
```

ELF class byte index.

file class

Definition at line 294 of file [elf.h](#).

16.395.2.16 EI_DATA [1/2]

```
#define EI_DATA 5
```

Data encoding byte index.

data encoding

Definition at line 295 of file [elf.h](#).

16.395.2.17 EI_DATA [2/2]

```
#define EI_DATA 5
```

Data encoding byte index.

data encoding

Definition at line 295 of file [elf.h](#).

16.395.2.18 EI_OSABI [1/2]

```
#define EI_OSABI 7
```

OS ABI identification.

Operating system / ABI identification.

Definition at line 297 of file [elf.h](#).

16.395.2.19 EI_OSABI [2/2]

```
#define EI_OSABI 7
```

OS ABI identification.

Operating system / ABI identification.

Definition at line 297 of file [elf.h](#).

16.395.2.20 EI_PAD [1/2]

```
#define EI_PAD 9
```

Byte index of padding bytes.

start of padding bytes

Definition at line 299 of file [elf.h](#).

16.395.2.21 EI_PAD [2/2]

```
#define EI_PAD 9
```

Byte index of padding bytes.

start of padding bytes

Definition at line 299 of file [elf.h](#).

16.395.2.22 EI_VERSION [1/2]

```
#define EI_VERSION 6
```

File version byte index.

file version

Value must be EV_CURRENT

Definition at line 296 of file [elf.h](#).

16.395.2.23 EI_VERSION [2/2]

```
#define EI_VERSION 6
```

File version byte index.

file version

Value must be EV_CURRENT

Definition at line 296 of file [elf.h](#).

16.395.2.24 ELFCLASSNONE [1/2]

```
#define ELFCLASSNONE 0
```

Invalid ELF class.

Invalid class.

Definition at line 310 of file [elf.h](#).

16.395.2.25 ELFCLASSNONE [2/2]

```
#define ELFCLASSNONE 0
```

Invalid ELF class.

Invalid class.

Definition at line 310 of file [elf.h](#).

16.395.2.26 ELFDATA2LSB [1/2]

```
#define ELFDATA2LSB 1
```

2's complement, little endian

0x01020304 => [0x04|0x03|0x02|0x01]

Definition at line 317 of file [elf.h](#).

16.395.2.27 ELFDATA2LSB [2/2]

```
#define ELFDATA2LSB 1
```

2's complement, little endian

0x01020304 => [0x04|0x03|0x02|0x01]

Definition at line 317 of file [elf.h](#).

16.395.2.28 ELFDATA2MSB [1/2]

```
#define ELFDATA2MSB 2
```

2's complement, big endian

0x01020304 => [0x01|0x02|0x03|0x04]

Definition at line 318 of file [elf.h](#).

16.395.2.29 ELFDATA2MSB [2/2]

```
#define ELFDATA2MSB 2
```

2's complement, big endian

0x01020304 => [0x01|0x02|0x03|0x04]

Definition at line 318 of file [elf.h](#).

16.395.2.30 ELFDATANONE [1/2]

```
#define ELFDATANONE 0
```

Invalid data encoding.

invalid data encoding

Definition at line 316 of file [elf.h](#).

16.395.2.31 ELFDATANONE [2/2]

```
#define ELFDATANONE 0
```

Invalid data encoding.

invalid data encoding

Definition at line 316 of file [elf.h](#).

16.395.2.32 ELFOSABI_AIX

```
#define ELFOSABI_AIX 7
```

IBM AIX.

Definition at line 181 of file [elf.h](#).

16.395.2.33 ELFOSABI_FREEBSD

```
#define ELFOSABI_FREEBSD 9
```

FreeBSD.

Definition at line 183 of file [elf.h](#).

16.395.2.34 ELFOSABI_HPUX [1/2]

```
#define ELFOSABI_HPUX 1
```

HP-UX.

HP-UX operating system.

Definition at line 323 of file [elf.h](#).

16.395.2.35 ELFOSABI_HPUX [2/2]

```
#define ELFOSABI_HPUX 1
```

HP-UX.

HP-UX operating system.

Definition at line 323 of file [elf.h](#).

16.395.2.36 ELFOSABI_IRIX

```
#define ELFOSABI_IRIX 8
```

SGI Irix.

Definition at line 182 of file [elf.h](#).

16.395.2.37 ELFOSABI_LINUX

```
#define ELFOSABI_LINUX 3
```

Linux.

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

16.395.2.38 ELFOSABI_MODESTO

```
#define ELFOSABI_MODESTO 11
```

Novell Modesto.

Definition at line 185 of file [elf.h](#).

16.395.2.39 ELFOSABI_NETBSD

```
#define ELFOSABI_NETBSD 2
```

NetBSD.

Definition at line 178 of file [elf.h](#).

16.395.2.40 ELFOSABI_OPENBSD

```
#define ELFOSABI_OPENBSD 12
```

OpenBSD.

Definition at line 186 of file [elf.h](#).

16.395.2.41 ELFOSABI_SOLARIS

```
#define ELFOSABI_SOLARIS 6
```

Sun Solaris.

Definition at line 180 of file [elf.h](#).

16.395.2.42 ELFOSABI_SYSV [1/2]

```
#define ELFOSABI_SYSV 0
```

Alias.

UNIX System V ABI (this specification)

Definition at line 322 of file [elf.h](#).

16.395.2.43 ELFOSABI_SYSV [2/2]

```
#define ELFOSABI_SYSV 0
```

Alias.

UNIX System V ABI (this specification)

Definition at line [322](#) of file [elf.h](#).

16.395.2.44 ELFOSABI_TRU64

```
#define ELFOSABI_TRU64 10
```

Compaq TRU64 UNIX.

Definition at line [184](#) of file [elf.h](#).

16.395.2.45 NT_VERSION

```
#define NT_VERSION 1
```

Contains a version string.

Definition at line [494](#) of file [elf.h](#).

16.395.2.46 SHF_GROUP

```
#define SHF_GROUP 0x200
```

Section is member of a group.

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

16.395.2.47 SHF_MASKOS

```
#define SHF_MASKOS 0xff00000
```

OS-specific.

Definition at line 409 of file [elf.h](#).

16.395.2.48 SHF_TLS

```
#define SHF_TLS 0x400
```

Section hold thread-local data.

Definition at line 408 of file [elf.h](#).

16.395.2.49 SHT_NUM

```
#define SHT_NUM 19
```

Number of defined types.

Definition at line 388 of file [elf.h](#).

16.396 elf.h

```

00001
00019 /*
00020  * (c) 2008-2009 Author(s)
00021  *     economic rights: Technische Universität Dresden (Germany)
00022  * This file is part of the exec package, which is 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 #ifndef _L4_EXEC_ELF_H
00033 #define _L4_EXEC_ELF_H
00034
00035 #include <l4/sys/l4int.h>
00036
00047 typedef l4_uint32_t  Elf32_Addr;
00048 typedef l4_uint32_t  Elf32_Off;
00049 typedef l4_uint16_t  Elf32_Half;
00050 typedef l4_uint32_t  Elf32_Word;
00051 typedef l4_int32_t   Elf32_Sword;
00052 typedef l4_uint64_t  Elf64_Addr;
00053 typedef l4_uint64_t  Elf64_Off;
00054 typedef l4_uint16_t  Elf64_Half;
00055 typedef l4_uint32_t  Elf64_Word;
00056 typedef l4_int32_t   Elf64_Sword;
00057 typedef l4_uint64_t  Elf64_Xword;
00058 typedef l4_int64_t   Elf64_Sxword;
00060
00061 #if L4_MWORD_BITS == 64
00062 #define ElfW(type)      _ElfW(Elf, 64, type)
00063 #else
00064 #define ElfW(type)      _ElfW(Elf, 32, type)
00065 #endif
00066 #define _ElfW(e,w,t)    __ElfW(e, w, _##t)
00067 #define __ElfW(e,w,t)   e##w##t
00068
00069 #if defined(ARCH_x86)
00070 #define L4_ARCH_EI_DATA      ELFDATA2LSB
00071 #define L4_ARCH_E_MACHINE    EM_386
00072 #define L4_ARCH_EI_CLASS     ELFCLASS32
00073 #elif defined(ARCH_amd64)
00074 #define L4_ARCH_EI_DATA      ELFDATA2LSB
00075 #define L4_ARCH_E_MACHINE    EM_X86_64
00076 #define L4_ARCH_EI_CLASS     ELFCLASS64
00077 #elif defined(ARCH_arm)
00078 #define L4_ARCH_EI_DATA      ELFDATA2LSB
00079 #define L4_ARCH_E_MACHINE    EM_ARM
00080 #define L4_ARCH_EI_CLASS     ELFCLASS32
00081 #elif defined(ARCH_arm64)
00082 #define L4_ARCH_EI_DATA      ELFDATA2LSB
00083 #define L4_ARCH_E_MACHINE    EM_ARM
00084 #define L4_ARCH_EI_CLASS     ELFCLASS32
00085 #elif defined(ARCH_arm64)
00086 #define L4_ARCH_EI_DATA      ELFDATA2LSB
00087 #define L4_ARCH_E_MACHINE    EM_AARCH64
00088 #define L4_ARCH_EI_CLASS     ELFCLASS64
00089 #elif defined(ARCH_ppc32)
00090 #define L4_ARCH_EI_DATA      ELFDATA2MSB
00091 #define L4_ARCH_E_MACHINE    EM_PPC
00092 #define L4_ARCH_EI_CLASS     ELFCLASS32
00093 #elif defined(ARCH_sparc)
00094 #define L4_ARCH_EI_DATA      ELFDATA2MSB
00095 #define L4_ARCH_E_MACHINE    EM_SPARC
00096 #define L4_ARCH_EI_CLASS     ELFCLASS32
00097 #elif defined(ARCH_mips)
00098 #define L4_ARCH_EI_DATA      ELFDATA2LSB
00099 #define L4_ARCH_E_MACHINE    EM_MIPS
00100 #ifndef __mips64
00101 #define L4_ARCH_EI_CLASS     ELFCLASS64
00102 #else
00103 #define L4_ARCH_EI_CLASS     ELFCLASS32
00104 #endif
00105 #else
00106 #warning elf.h: Unsupported build architecture!
00107 #endif
00108
00109
00110 /*****
00111  * ELF Header - figure 1-3, page 1-3 */
00112 /*****
00113
00114 #define EI_NIDENT 16
00115 typedef struct {
00116     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     unsigned char e_ident[EI_NIDENT];
00144     Elf64_Half    e_type;
00145     Elf64_Half    e_machine;
00146     Elf64_Word    e_version;
00147     Elf64_Addr    e_entry;
00148     Elf64_Off     e_phoff;
00149     Elf64_Off     e_shoff;
00150     Elf64_Word    e_flags;
00151     Elf64_Half    e_ehsize;
00152     Elf64_Half    e_phentsize;
00153     Elf64_Half    e_phnum;
00154     Elf64_Half    e_shentsize;
00155     Elf64_Half    e_shnum;
00156     Elf64_Half    e_shstrndx;
00157 } Elf64_Ehdr;
00158
00159 #define EI_CLASS 4
00160 #define ELFCLASSNONE 0
00161 #define ELFCLASS32 1
00162 #define ELFCLASS64 2
00163 #define ELFCLASSNUM 3
00165 #define EI_DATA 5
00166 #define ELFDATANONE 0
00167 #define ELFDATA2LSB 1
00168 #define ELFDATA2MSB 2
00169 #define ELFDATANUM 3
00170
00171 #define EI_VERSION 6
00174 #define EI_OSABI 7
00175 #define ELFOSABI_NONE 0
00176 #define ELFOSABI_SYSV 0
00177 #define ELFOSABI_HPUX 1
00178 #define ELFOSABI_NETBSD 2
00179 #define ELFOSABI_LINUX 3
00180 #define ELFOSABI_SOLARIS 6
00181 #define ELFOSABI_AIX 7
00182 #define ELFOSABI_IRIX 8
00183 #define ELFOSABI_FREEBSD 9
00184 #define ELFOSABI_TRU64 10
00185 #define ELFOSABI_MODESTO 11
00186 #define ELFOSABI_OPENBSD 12
00187 #define ELFOSABI_ARM 97
00188 #define ELFOSABI_STANDALONE 255
00190 #define EI_ABIVERSION 8
00192 #define EI_PAD 9
00194 /* object file type - page 1-3 (e_type) */
00195
00196 #define ET_NONE 0
00197 #define ET_REL 1
00198 #define ET_EXEC 2
00199 #define ET_DYN 3
00200 #define ET_CORE 4
00201 #define ET_LOPROC 0xff00
00202 #define ET_HIPROC 0xffff
00204 /* required architecture - page 1-4 (e_machine) */
00205
00206 #define EM_NONE 0
00207 #define EM_M32 1
00208 #define EM_SPARC 2
00209 #define EM_386 3
00210 #define EM_68K 4
00211 #define EM_88K 5
00212 #define EM_860 7
00213 #define EM_MIPS 8
00214 #define EM_MIPS_RS4_BE 10
00215 #define EM_SPARC64 11
00216 #define EM_PARISC 15
00217 #define EM_VPP500 17
00218 #define EM_SPARC32PLUS 18
00219 #define EM_960 19
00220 #define EM_PPC 20
```

```

00221 #define EM_V800      36
00222 #define EM_FR20       37
00223 #define EM_RH32       38
00224 #define EM_RCE        39
00225 #define EM_ARM        40
00226 #define EM_ALPHA      41
00227 #define EM_SH         42
00228 #define EM_SPARCV9    43
00229 #define EM_TRICORE    44
00230 #define EM_ARC         45
00231 #define EM_H8_300     46
00232 #define EM_H8_300H    47
00233 #define EM_H8S        48
00234 #define EM_H8_500     49
00235 #define EM_IA_64      50
00236 #define EM_MIPS_X     51
00237 #define EM_COLDFIRE    52
00238 #define EM_68HC12     53
00239 #define EM_X86_64     62
00240 #define EM_PDSP       63
00241 #define EM_FX66       66
00242 #define EM_ST9PLUS    67
00243 #define EM_ST7        68
00244 #define EM_68HC16     69
00245 #define EM_68HC11     70
00246 #define EM_68HC08     71
00247 #define EM_68HC05     72
00248 #define EM_SVX        73
00249 #define EM_ST19       74
00250 #define EM_VAX        75
00251 #define EM_CRIS       76
00252 #define EM_JAVELIN    77
00253 #define EM_FIREPATH   78
00254 #define EM_ZSP        79
00255 #define EM_MMIX       80
00256 #define EM_HUANY      81
00257 #define EM_PRISM      82
00258 #define EM_AVR        83
00259 #define EM_FR30       84
00260 #define EM_D10V      85
00261 #define EM_D30V      86
00262 #define EM_V850      87
00263 #define EM_M32R       88
00264 #define EM_MN10300    89
00265 #define EM_MN10200    90
00266 #define EM_PJ         91
00267 #define EM_OPENRISC   92
00268 #define EM_ARC_A5     93
00269 #define EM_XTENSA     94
00270 #define EM_ALTERA_NIOS2 113
00271 #define EM_AARCH64    183
00272 #define EM_TILEPRO    188
00273 #define EM_MICROBLAZE 189
00274 #define EM_TILEGX     191
00275 #define EM_NUM        192
00276
00277 #if 0
00278 #define EM_ALPHA      0x9026 /* interim value used by Linux until the
00279                               committee comes up with a final number */
00280 #define EM_S390       0xA390 /* interim value used for IBM S390 */
00281 #endif
00282
00283 /* object file version - page 1-4 (e_version) */
00284
00285 #define EV_NONE        0
00286 #define EV_CURRENT     1
00287 /* e_ident[] Identification Indexes - figure 1-4, page 1-5 */
00288
00289 #define EI_MAG0        0
00290 #define EI_MAG1        1
00291 #define EI_MAG2        2
00292 #define EI_MAG3        3
00293 #define EI_CLASS       4
00294 #define EI_DATA        5
00295 #define EI_VERSION     6
00296 #define EI_OSABI       7
00297 #define EI_ABIVERSION  8
00298 #define EI_PAD         9
00301 /* magic number - page 1-5 */
00302
00303 #define ELFMAG0        0x7f
00304 #define ELFMAG1        'E'
00305 #define ELFMAG2        'L'
00306 #define ELFMAG3        'F'
00308 /* file class or capacity - page 1-6 */
00309
00310 #define ELFCLASSNONE   0

```

```
00311 #define ELFCLASS32 1
00312 #define ELFCLASS64 2
00314 /* data encoding - page 1-6 */
00315
00316 #define ELFDATANONE 0
00317 #define ELFDATA2LSB 1
00318 #define ELFDATA2MSB 2
00320 /* Identify operating system and ABI to which the object is targeted */
00321
00322 #define ELFOSABI_SYSV 0
00323 #define ELFOSABI_HPUX 1
00324 #define ELFOSABI_STANDALONE 255
00327 /******
00328 /* Sections - page 1-8 */
00329 /******
00330
00331 /* special section indexes */
00332
00333 #define SHN_UNDEF 0
00334 #define SHN_LORESERVE 0xfff00
00335 #define SHN_LOPROC 0xfff00
00336 #define SHN_HIPROC 0xfff1f
00337 #define SHN_ABS 0xffff1
00338 #define SHN_COMMON 0xffff2
00339 #define SHN_HIRESERVE 0xfffff
00342 typedef struct {
00343     Elf32_Word sh_name;
00344     Elf32_Word sh_type;
00345     Elf32_Word sh_flags;
00346     Elf32_Addr sh_addr;
00347     Elf32_Off sh_offset;
00348     Elf32_Word sh_size;
00349     Elf32_Word sh_link;
00350     Elf32_Word sh_info;
00351     Elf32_Word sh_addralign;
00352     Elf32_Word sh_entsize;
00353 } Elf32_Shdr;
00354
00356 typedef struct {
00357     Elf64_Word sh_name;
00358     Elf64_Word sh_type;
00359     Elf64_Xword sh_flags;
00360     Elf64_Addr sh_addr;
00361     Elf64_Off sh_offset;
00362     Elf64_Xword sh_size;
00363     Elf64_Word sh_link;
00364     Elf64_Word sh_info;
00365     Elf64_Xword sh_addralign;
00366     Elf64_Xword sh_entsize;
00367 } Elf64_Shdr;
00368
00369 /* section type - figure 1-10, page 1-10 */
00370
00371 #define SHT_NULL 0
00372 #define SHT_PROGBITS 1
00373 #define SHT_SYMTAB 2
00374 #define SHT_STRTAB 3
00375 #define SHT_RELA 4
00376 #define SHT_HASH 5
00377 #define SHT_DYNAMIC 6
00378 #define SHT_NOTE 7
00379 #define SHT_NOBITS 8
00380 #define SHT_REL 9
00381 #define SHT_SHLIB 10
00382 #define SHT_DYNSYM 11
00383 #define SHT_INIT_ARRAY 14
00384 #define SHT_FINI_ARRAY 15
00385 #define SHT_PREINIT_ARRAY 16
00386 #define SHT_GROUP 17
00387 #define SHT_SYMTAB_SHNDX 18
00388 #define SHT_NUM 19
00389 #define SHT_LOOS 0x60000000
00390 #define SHT_HIOS 0x6fffffff
00391 #define SHT_LOPROC 0x70000000
00392 #define SHT_HIPROC 0x7fffffff
00393 #define SHT_LOUSER 0x80000000
00394 #define SHT_HIUSER 0xffffffff
00395
00396 /* section attribute flags - page 1-12, figure 1-12 */
00397
00398 #define SHF_WRITE 0x1
00399 #define SHF_ALLOC 0x2
00400 #define SHF_EXECINSTR 0x4
00401 #define SHF_MERGE 0x10
00402 #define SHF_STRINGS 0x20
00403 #define SHF_INFO_LINK 0x40
00404 #define SHF_LINK_ORDER 0x80
```

```

00405 #define SHF_OS_NONCONFORMING 0x100
00407 #define SHF_GROUP 0x200
00408 #define SHF_TLS 0x400
00409 #define SHF_MASKOS 0x0ff00000
00410 #define SHF_MASKPROC 0xf0000000
00413 /*****
00414  /* Program Header - figure 2-1, page 2-2 */
00415  *****/
00416
00418 typedef struct {
00419     Elf32_Word    p_type;
00420     Elf32_Off     p_offset;
00421     Elf32_Addr    p_vaddr;
00422     Elf32_Addr    p_paddr;
00423     Elf32_Word    p_filesz;
00424     Elf32_Word    p_memsz;
00425     Elf32_Word    p_flags;
00426     Elf32_Word    p_align;
00427 } Elf32_Phdr;
00428
00430 typedef struct {
00431     Elf64_Word    p_type;
00432     Elf64_Word    p_flags;
00433     Elf64_Off     p_offset;
00434     Elf64_Addr    p_vaddr;
00435     Elf64_Addr    p_paddr;
00436     Elf64_Xword   p_filesz;
00437     Elf64_Xword   p_memsz;
00438     Elf64_Xword   p_align;
00439 } Elf64_Phdr;
00440
00441 /* segment types - figure 2-2, page 2-3 */
00442
00443 #define PT_NULL 0
00444 #define PT_LOAD 1
00445 #define PT_DYNAMIC 2
00446 #define PT_INTERP 3
00447 #define PT_NOTE 4
00448 #define PT_SHLIB 5
00449 #define PT_PHDR 6
00450 #define PT_TLS 7
00451 #define PT_NUM 8
00452 #define PT_LOOS 0x60000000
00453 #define PT_HIOS 0x6fffffff
00454 #define PT_LOPROC 0x70000000
00455 #define PT_HIPROC 0x7fffffff
00457 #define PT_GNU_EH_FRAME (PT_LOOS + 0x474e550)
00458 #define PT_GNU_STACK (PT_LOOS + 0x474e551)
00459 #define PT_GNU_RELRO (PT_LOOS + 0x474e552)
00461 #define PT_L4_STACK (PT_LOOS + 0x12)
00462 #define PT_L4_KIP (PT_LOOS + 0x13)
00463 #define PT_L4_AUX (PT_LOOS + 0x14)
00465 /* segment permissions - page 2-3 */
00466
00467 #define PF_X 0x1
00468 #define PF_W 0x2
00469 #define PF_R 0x4
00470 #define PF_MASKOS 0x0ff00000
00471 #define PF_MASKPROC 0x7fffffff
00472
00473 /* Legal values for note segment descriptor types for core files. */
00474
00475 #define NT_PRSTATUS 1
00476 #define NT_FPREGSET 2
00477 #define NT_PRPSINFO 3
00478 #define NT_PRXREG 4
00479 #define NT_TASKSTRUCT 4
00480 #define NT_PLATFORM 5
00481 #define NT_AUXV 6
00482 #define NT_GWINDOWS 7
00483 #define NT_ASRS 8
00484 #define NT_PSTATUS 10
00485 #define NT_PSINFO 13
00486 #define NT_PRCRED 14
00487 #define NT_UTSNAME 15
00488 #define NT_LWPSTATUS 16
00489 #define NT_LWPSINFO 17
00490 #define NT_PRFPXREG 20
00492 /* Legal values for the note segment descriptor types for object files. */
00493
00494 #define NT_VERSION 1
00496 /* Dynamic structure - figure 2-9, page 2-12 */
00497
00499 typedef struct {
00500     Elf32_Sword    d_tag;
00501     union {
00502         Elf32_Word    d_val;

```



```
00503 Elf32_Addr d_ptr;
00504     } d_un;
00505 } Elf32_Dyn;
00506
00508 typedef struct {
00509     Elf64_Sxword d_tag;
00510     union {
00511         Elf64_Xword d_val;
00512         Elf64_Addr d_ptr;
00513     } d_un;
00514 } Elf64_Dyn;
00515
00518 #define DT_NULL 0
00519 #define DT_NEEDED 1
00520 #define DT_PLTRELSZ 2
00521 #define DT_PLTGOT 3
00522 #define DT_HASH 4
00523 #define DT_STRTAB 5
00524 #define DT_SYMTAB 6
00525 #define DT_RELA 7
00526 #define DT_RELASZ 8
00527 #define DT_RELAENT 9
00528 #define DT_STRSZ 10
00529 #define DT_SYMENT 11
00530 #define DT_INIT 12
00531 #define DT_FINI 13
00532 #define DT_SONAME 14
00533 #define DT_RPATH 15
00534 #define DT_SYMBOLIC 16
00535 #define DT_REL 17
00536 #define DT_RELSZ 18
00537 #define DT_RELENT 19
00538 #define DT_PTRREL 20
00539 #define DT_DEBUG 21
00540 #define DT_TEXTREL 22
00541 #define DT_JMPREL 23
00542 #define DT_BIND_NOW 24
00543 #define DT_INIT_ARRAY 25
00544 #define DT_FINI_ARRAY 26
00545 #define DT_INIT_ARRAYSZ 27
00546 #define DT_FINI_ARRAYSZ 28
00547 #define DT_RUNPATH 29
00548 #define DT_FLAGS 30
00549 #define DT_ENCODING 32
00550 #define DT_PREINIT_ARRAY 32
00551 #define DT_PREINIT_ARRAYSZ 33
00552 #define DT_NUM 34
00553 #define DT_LOOS 0x6000000d
00554 #define DT_HIOS 0x6ffff000
00555 #define DT_LOPROC 0x70000000
00556 #define DT_HIPROC 0x7fffffff
00558 /* Values of 'd_un.d_val' in the DT_FLAGS entry. */
00559 #define DF_ORIGIN 0x00000001
00560 #define DF_SYMBOLIC 0x00000002
00561 #define DF_TEXTREL 0x00000004
00562 #define DF_BIND_NOW 0x00000008
00563 #define DF_STATIC_TLS 0x00000010
00565 /* State flags selectable in the 'd_un.d_val' element of the DT_FLAGS_1
00566     entry in the dynamic section. */
00567 #define DF_1_NOW 0x00000001
00568 #define DF_1_GLOBAL 0x00000002
00569 #define DF_1_GROUP 0x00000004
00570 #define DF_1_NODELETE 0x00000008
00571 #define DF_1_LOADFLTR 0x00000010
00572 #define DF_1_INITFIRST 0x00000020
00573 #define DF_1_NOOPEN 0x00000040
00574 #define DF_1_ORIGIN 0x00000080
00575 #define DF_1_DIRECT 0x00000100
00576 #define DF_1_TRANS 0x00000200
00577 #define DF_1_INTERPOSE 0x00000400
00578 #define DF_1_NODEFLIB 0x00000800
00579 #define DF_1_NODUMP 0x00001000
00580 #define DF_1_CONFALT 0x00002000
00581 #define DF_1_ENDFILTEE 0x00004000
00582 #define DF_1_DISPRELDNE 0x00008000
00583 #define DF_1_DISPRELPND 0x00010000
00585 /* Flags for the feature selection in DT_FEATURE_1. */
00586 #define DTF_1_PARINIT 0x00000001
00587 #define DTF_1_CONFEXP 0x00000002
00588
00589 /* Flags in the DT_POSFLAG_1 entry effecting only the next DT_* entry. */
00590 #define DF_1_LAZYLOAD 0x00000001
00591 #define DF_1_GROUPPERM 0x00000002
00594 /* Relocation - page 1-21, figure 1-20 */
00595
00596 typedef struct {
00597     Elf32_Addr r_offset;
```

```

00598     Elf32_Word    r_info;
00599 } Elf32_Rel;
00600
00601 typedef struct {
00602     Elf32_Addr    r_offset;
00603     Elf32_Word    r_info;
00604     Elf32_Sword    r_addend;
00605 } Elf32_Rela;
00606
00607 typedef struct {
00608     Elf64_Addr    r_offset;
00609     Elf64_Xword    r_info;
00610 } Elf64_Rel;
00611
00612 typedef struct {
00613     Elf64_Addr    r_offset;
00614     Elf64_Xword    r_info;
00615     Elf64_Sxword    r_addend;
00616 } Elf64_Rela;
00617
00618 #define ELF32_R_SYM(i)    ((i)>>8)
00619 #define ELF32_R_TYPE(i)    ((unsigned char)(i))
00620 #define ELF32_R_INFO(s,t) (((s)<<8)+(unsigned char)(t))
00621
00622 #define ELF64_R_SYM(i)    ((i)>>32)
00623 #define ELF64_R_TYPE(i)    ((i)&0xffffffffL)
00624 #define ELF64_R_INFO(s,t) (((s)<<32)+(t)&0xffffffffL)
00625
00626 /* Relocation types (processor specific) - page 1-23, figure 1-22 */
00627
00628 #define R_386_NONE    0
00629 #define R_386_32    1
00630 #define R_386_PC32    2
00631 #define R_386_GOT32    3
00632 #define R_386_PLT32    4
00633 #define R_386_COPY    5
00634 #define R_386_GLOB_DAT    6
00635 #define R_386_JMP_SLOT    7
00636 #define R_386_RELATIVE    8
00637 #define R_386_GOTOFF    9
00638 #define R_386_GOTPC    10
00639 #define R_386_32PLT    11
00640 #define R_386_TLS_TPOFF    14 /* Offset in static TLS block */
00641 #define R_386_TLS_IE    15 /* Address of GOT entry for static TLS
00642     block offset */
00643 #define R_386_TLS_GOTIE    16 /* GOT entry for static TLS block
00644     offset */
00645 #define R_386_TLS_LE    17 /* Offset relative to static TLS
00646     block */
00647 #define R_386_TLS_GD    18 /* Direct 32 bit for GNU version of
00648     general dynamic thread local data */
00649 #define R_386_TLS_LDM    19 /* Direct 32 bit for GNU version of
00650     local dynamic thread local data
00651     in LE code */
00652 #define R_386_16    20
00653 #define R_386_PC16    21
00654 #define R_386_8    22
00655 #define R_386_PC8    23
00656 #define R_386_TLS_GD_32    24 /* Direct 32 bit for general dynamic
00657     thread local data */
00658 #define R_386_TLS_GD_PUSH    25 /* Tag for pushl in GD TLS code */
00659 #define R_386_TLS_GD_CALL    26 /* Relocation for call to
00660     __tls_get_addr() */
00661 #define R_386_TLS_GD_POP    27 /* Tag for popl in GD TLS code */
00662 #define R_386_TLS_LDM_32    28 /* Direct 32 bit for local dynamic
00663     thread local data in LE code */
00664 #define R_386_TLS_LDM_PUSH    29 /* Tag for pushl in LDM TLS code */
00665 #define R_386_TLS_LDM_CALL    30 /* Relocation for call to
00666     __tls_get_addr() in LDM code */
00667 #define R_386_TLS_LDM_POP    31 /* Tag for popl in LDM TLS code */
00668 #define R_386_TLS_LDO_32    32 /* Offset relative to TLS block */
00669 #define R_386_TLS_IE_32    33 /* GOT entry for negated static TLS
00670     block offset */
00671 #define R_386_TLS_LE_32    34 /* Negated offset relative to static
00672     TLS block */
00673 #define R_386_TLS_DTPMOD32    35 /* ID of module containing symbol */
00674 #define R_386_TLS_DTPOFF32    36 /* Offset in TLS block */
00675 #define R_386_TLS_TPOFF32    37 /* Negated offset in static TLS block */
00676 /* Keep this the last entry. */
00677 #define R_386_NUM    38
00678
00679 /* ARM specific declarations */
00680
00681 /* Processor specific flags for the ELF header e_flags field. */
00682 #define EF_ARM_RELEXEC    0x01
00683 #define EF_ARM_HASENTRY    0x02
00684 #define EF_ARM_INTERWORK    0x04

```

```
00685 #define EF_ARM_APCS_26      0x08
00686 #define EF_ARM_APCS_FLOAT    0x10
00687 #define EF_ARM_PIC           0x20
00688 #define EF_ARM_ALIGN8        0x40 /* 8-bit structure alignment is in use */
00689 #define EF_ARM_NEW_ABI        0x80
00690 #define EF_ARM_OLD_ABI        0x100
00691
00692 /* Other constants defined in the ARM ELF spec. version B-01. */
00693 /* NB. These conflict with values defined above. */
00694 #define EF_ARM_SYMSARESORTED  0x04
00695 #define EF_ARM_DYNSYMSUSESEGIDX 0x08
00696 #define EF_ARM_MAPSYMSFIRST  0x10
00697 #define EF_ARM_EABIMASK       0xFF000000
00698
00699 #define EF_ARM_EABI_VERSION(flags) ((flags) & EF_ARM_EABIMASK)
00700 #define EF_ARM_EABI_UNKNOWN    0x00000000
00701 #define EF_ARM_EABI_VER1       0x01000000
00702 #define EF_ARM_EABI_VER2       0x02000000
00703
00704 /* Additional symbol types for Thumb */
00705 #define STT_ARM_TFUNC          0xd
00706
00707 /* ARM-specific values for sh_flags */
00708 #define SHF_ARM_ENTRYSECT      0x10000000 /* Section contains an entry point */
00709 #define SHF_ARM_COMDEF         0x80000000 /* Section may be multiply defined
00710                                     in the input to a link step */
00711
00712 /* ARM-specific program header flags */
00713 #define PF_ARM_SB              0x10000000 /* Segment contains the location
00714                                     addressed by the static base */
00715
00716 /* ARM relocs. */
00717 #define R_ARM_NONE             0 /* No reloc */
00718 #define R_ARM_PC24             1 /* PC relative 26 bit branch */
00719 #define R_ARM_ABS32            2 /* Direct 32 bit */
00720 #define R_ARM_REL32            3 /* PC relative 32 bit */
00721 #define R_ARM_PC13            4
00722 #define R_ARM_ABS16            5 /* Direct 16 bit */
00723 #define R_ARM_ABS12            6 /* Direct 12 bit */
00724 #define R_ARM_THM_ABS5        7
00725 #define R_ARM_ABS8            8 /* Direct 8 bit */
00726 #define R_ARM_SBREL32          9
00727 #define R_ARM_THM_PC22        10
00728 #define R_ARM_THM_PC8         11
00729 #define R_ARM_AMP_VCALL9      12
00730 #define R_ARM_SWI24           13
00731 #define R_ARM_THM_SWI8        14
00732 #define R_ARM_XPC25           15
00733 #define R_ARM_THM_XPC22       16
00734 #define R_ARM_COPY            20 /* Copy symbol at runtime */
00735 #define R_ARM_GLOB_DAT         21 /* Create GOT entry */
00736 #define R_ARM_JUMP_SLOT        22 /* Create PLT entry */
00737 #define R_ARM_RELATIVE         23 /* Adjust by program base */
00738 #define R_ARM_GOTOFF           24 /* 32 bit offset to GOT */
00739 #define R_ARM_GOTPC            25 /* 32 bit PC relative offset to GOT */
00740 #define R_ARM_GOT32            26 /* 32 bit GOT entry */
00741 #define R_ARM_PLT32            27 /* 32 bit PLT address */
00742 #define R_ARM_ALU_PCREL_7_0    32
00743 #define R_ARM_ALU_PCREL_15_8   33
00744 #define R_ARM_ALU_PCREL_23_15  34
00745 #define R_ARM_LDR_SBREL_11_0   35
00746 #define R_ARM_ALU_SBREL_19_12  36
00747 #define R_ARM_ALU_SBREL_27_20  37
00748 #define R_ARM_GNU_VTENTRY     100
00749 #define R_ARM_GNU_VTINHERIT   101
00750 #define R_ARM_THM_PC11         102 /* thumb unconditional branch */
00751 #define R_ARM_THM_PC9          103 /* thumb conditional branch */
00752 #define R_ARM_RXPC25           249
00753 #define R_ARM_RSBREL32         250
00754 #define R_ARM_THM_RPC22        251
00755 #define R_ARM_RREL32           252
00756 #define R_ARM_RABS22           253
00757 #define R_ARM_RPC24            254
00758 #define R_ARM_RBASE            255
00759 /* Keep this the last entry. */
00760 #define R_ARM_NUM              256
00761
00762 /* AMD x86-64 relocations. */
00763 #define R_X86_64_NONE          0 /* No reloc */
00764 #define R_X86_64_64           1 /* Direct 64 bit */
00765 #define R_X86_64_PC32          2 /* PC relative 32 bit signed */
00766 #define R_X86_64_GOT32         3 /* 32 bit GOT entry */
00767 #define R_X86_64_PLT32         4 /* 32 bit PLT address */
00768 #define R_X86_64_COPY          5 /* Copy symbol at runtime */
00769 #define R_X86_64_GLOB_DAT      6 /* Create GOT entry */
00770 #define R_X86_64_JUMP_SLOT     7 /* Create PLT entry */
00771 #define R_X86_64_RELATIVE      8 /* Adjust by program base */
```

```

00772 #define R_X86_64_GOTPCREL 9 /* 32 bit signed PC relative
00773         offset to GOT */
00774 #define R_X86_64_32 10 /* Direct 32 bit zero extended */
00775 #define R_X86_64_32S 11 /* Direct 32 bit sign extended */
00776 #define R_X86_64_16 12 /* Direct 16 bit zero extended */
00777 #define R_X86_64_PC16 13 /* 16 bit sign extended pc relative */
00778 #define R_X86_64_8 14 /* Direct 8 bit sign extended */
00779 #define R_X86_64_PC8 15 /* 8 bit sign extended pc relative */
00780 #define R_X86_64_DTPMOD64 16 /* ID of module containing symbol */
00781 #define R_X86_64_DTPOFF64 17 /* Offset in module's TLS block */
00782 #define R_X86_64_TPOFF64 18 /* Offset in initial TLS block */
00783 #define R_X86_64_TLSGD 19 /* 32 bit signed PC relative offset
00784         to two GOT entries for GD symbol */
00785 #define R_X86_64_TLSLD 20 /* 32 bit signed PC relative offset
00786         to two GOT entries for LD symbol */
00787 #define R_X86_64_DTPOFF32 21 /* Offset in TLS block */
00788 #define R_X86_64_GOTTPOFF 22 /* 32 bit signed PC relative offset
00789         to GOT entry for IE symbol */
00790 #define R_X86_64_TPOFF32 23 /* Offset in initial TLS block */
00791
00792 #define R_X86_64_NUM 24
00793
00794 /* Symbol Table Entry - page 1-17, figure 1-16 */
00795
00796 #define STN_UNDEF 0
00797
00799 typedef struct {
00800     Elf32_Word st_name;
00801     Elf32_Addr st_value;
00802     Elf32_Word st_size;
00803     unsigned char st_info;
00804     unsigned char st_other;
00805     Elf32_Half st_shndx;
00806 } Elf32_Sym;
00807
00809 typedef struct {
00810     Elf64_Word st_name;
00811     unsigned char st_info;
00812     unsigned char st_other;
00813     Elf64_Half st_shndx;
00814     Elf64_Addr st_value;
00815     Elf64_Xword st_size;
00816 } Elf64_Sym;
00817
00818 #define ELF32_ST_BIND(i) ((i)>4)
00819 #define ELF32_ST_TYPE(i) ((i)&0xf)
00820 #define ELF32_ST_INFO(b,t) (((b)<4)+((t)&0xf))
00821
00822 #define ELF64_ST_BIND(i) ((i)>4)
00823 #define ELF64_ST_TYPE(i) ((i)&0xf)
00824 #define ELF64_ST_INFO(b,t) (((b)<4)+((t)&0xf))
00825
00826 /* Symbol Binding - page 1-18, figure 1-17 */
00827
00828 #define STB_LOCAL 0
00829 #define STB_GLOBAL 1
00830 #define STB_WEAK 2
00831 #define STB_LOOS 10
00832 #define STB_HIOS 12
00833 #define STB_LOPROC 13
00834 #define STB_HIPROC 15
00836 /* Symbol Types - page 1-19, figure 1-18 */
00837
00838 #define STT_NOTYPE 0
00839 #define STT_OBJECT 1
00840 #define STT_FUNC 2
00841 #define STT_SECTION 3
00842 #define STT_FILE 4
00843 #define STT_LOOS 10
00844 #define STT_HIOS 12
00845 #define STT_LOPROC 13
00846 #define STT_HIPROC 15
00848 enum Elf_ATs
00849 {
00850     AT_NULL = 0,
00851     AT_IGNORE = 1,
00852     AT_EXECD = 2,
00853     AT_PHDR = 3,
00854     AT_PHENT = 4,
00855     AT_PHNUM = 5,
00856     AT_PAGESZ = 6,
00857     AT_BASE = 7,
00858     AT_FLAGS = 8,
00859     AT_ENTRY = 9,
00860     AT_NOTELF = 10,
00861     AT_UID = 11,
00862     AT_EUID = 12,

```

```

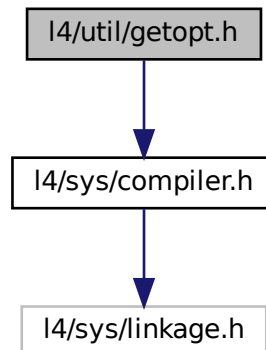
00863     AT_GID          = 13,
00864     AT_EGID         = 14,
00865
00866     AT_L4_AUX       = 0xf0,
00867     AT_L4_ENV       = 0xf1,
00868 };
00869
00870 typedef struct Elf32_Auxv
00871 {
00872     Elf32_Word atype;
00873     Elf32_Word avalue;
00874 } Elf32_Auxv;
00875
00876 typedef struct Elf64_Auxv
00877 {
00878     Elf64_Word atype;
00879     Elf64_Word avalue;
00880 } Elf64_Auxv;
00881
00882 /* Some helpers */
00883 static inline int l4util_elf_check_magic(ElfW(Ehdr) const *hdr);
00884 static inline int l4util_elf_check_arch(ElfW(Ehdr) const *hdr);
00885 static inline ElfW(Phdr) *l4util_elf_phdr(ElfW(Ehdr) const *hdr);
00886
00887 /* Implementations */
00888 static inline
00889 int l4util_elf_check_magic(ElfW(Ehdr) const *hdr)
00890 {
00891     return    hdr->e_ident[EI_MAG0] == ELFMAG0
00892             && hdr->e_ident[EI_MAG1] == ELFMAG1
00893             && hdr->e_ident[EI_MAG2] == ELFMAG2
00894             && hdr->e_ident[EI_MAG3] == ELFMAG3;
00895 }
00896
00897 static inline
00898 int l4util_elf_check_arch(ElfW(Ehdr) const *hdr)
00899 {
00900     return    hdr->e_ident[EI_CLASS] == L4_ARCH_EI_CLASS
00901             && hdr->e_ident[EI_DATA]  == L4_ARCH_EI_DATA
00902             && hdr->e_machine        == L4_ARCH_E_MACHINE;
00903 }
00904
00905 static inline
00906 ElfW(Phdr) *l4util_elf_phdr(ElfW(Ehdr) const *hdr)
00907 {
00908     return (ElfW(Phdr) *) ((char *)hdr + hdr->e_phoff);
00909 }
00910
00913 #endif /* _L4_EXEC_ELF_H */

```

16.397 l4/util/getopt.h File Reference

getopt

```
#include <l4/sys/compiler.h>
Include dependency graph for getopt.h:
```



16.397.1 Detailed Description

getopt

Definition in file [getopt.h](#).

16.398 getopt.h

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU Lesser General Public License 2.1.
00011  * Please see the COPYING-LGPL-2.1 file for details.
00012  */
00013 #ifndef _GETOPT_H
00014 #define _GETOPT_H
00015
00016 #ifndef NULL
00017 #define NULL 0
00018 #endif
00019
00020 #include <l4/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.

```

```

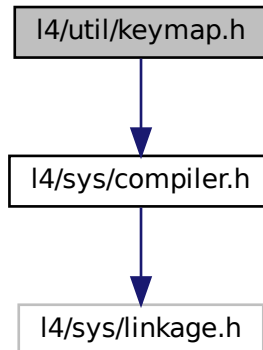
00040
00041     Otherwise, 'optind' communicates from one call to the next
00042     how much of ARGV has been scanned so far.  */
00043
00044 extern int optind;
00045
00046 /* Callers store zero here to inhibit the error message 'getopt' prints
00047    for unrecognized options.  */
00048
00049 extern 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 */

```

16.399 l4/util/keymap.h File Reference

Event to ASCII key mapping.

```
#include <l4/sys/compiler.h>
Include dependency graph for keymap.h:
```



16.399.1 Detailed Description

Event to ASCII key mapping.

Definition in file [keymap.h](#).

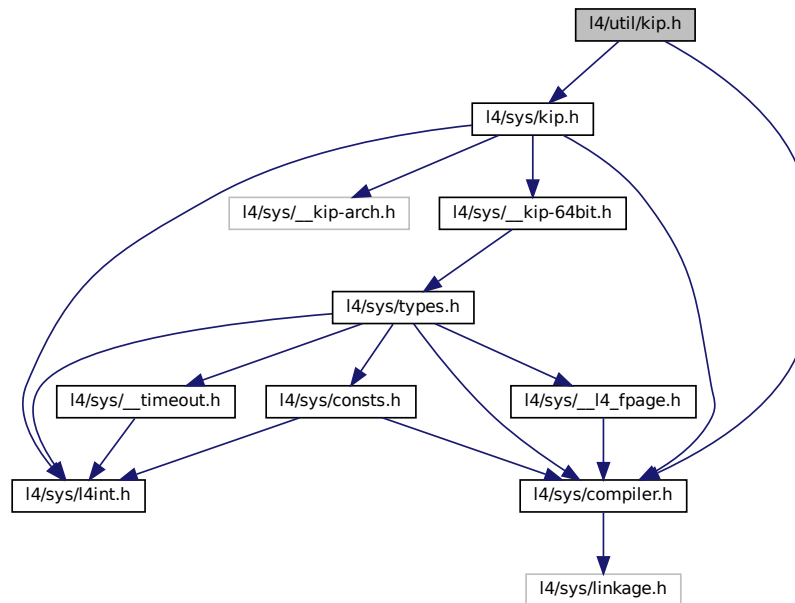
16.400 keymap.h

```
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__ */
```

16.401 l4/util/kip.h File Reference

```
#include <l4/sys/kip.h>
#include <l4/sys/compiler.h>
```


Include dependency graph for kip.h:



Macros

- `#define l4util_kip_for_each_feature(s)` for (s += strlen(s) + 1; *s; s += strlen(s) + 1)
Cycle through kernel features given in the KIP.

Functions

- `int l4util_kip_kernel_is_ux (l4_kernel_info_t *)`
Return whether the kernel is running natively or under UX.
- `int l4util_kip_kernel_has_feature (l4_kernel_info_t *, const char *str)`
Check if kernel supports a feature.
- `unsigned long l4util_kip_kernel_abi_version (l4_kernel_info_t *)`
Return kernel ABI version.

16.401.1 Macro Definition Documentation

16.401.1.1 l4util_kip_for_each_feature

```
#define l4util_kip_for_each_feature(
    s )  for (s += strlen(s) + 1; *s; s += 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 61 of file `kip.h`.

16.401.2 Function Documentation

16.401.2.1 l4util_kip_kernel_abi_version()

```
unsigned long l4util_kip_kernel_abi_version (
    l4_kernel_info_t * )
```

Return kernel ABI version.

Returns

Kernel ABI version.

16.401.2.2 l4util_kip_kernel_has_feature()

```
int l4util_kip_kernel_has_feature (
    l4_kernel_info_t * ,
    const char * str )
```

Check if kernel supports a feature.

Parameters

<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.

16.401.2.3 l4util_kip_kernel_is_ux()

```
int l4util_kip_kernel_is_ux (
    l4_kernel_info_t * )
```

Return whether the kernel is running natively or under UX.

Returns

1 when running under UX, 0 if not running under UX.

Examples

[examples/sys/ux-vhw/main.c](#).

16.402 kip.h

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU Lesser General Public License 2.1.
00011  * Please see the COPYING-LGPL-2.1 file for details.
00012  */
00013
00014 #pragma once
00015
00016 #include <l4/sys/kip.h>
00017 #include <l4/sys/compiler.h>
00018
00024
00025
00026 EXTERN_C_BEGIN
00027
00033 L4_CV int l4util_kip_kernel_is_ux(l4_kernel_info_t *);
00034
00044 L4_CV int l4util_kip_kernel_has_feature(l4_kernel_info_t *, const char *str);
00045
00051 L4_CV unsigned long l4util_kip_kernel_abi_version(l4_kernel_info_t *);
00052
00053 EXTERN_C_END
00054
00061 #define l4util_kip_for_each_feature(s) \
00062     for (s += strlen(s) + 1; *s; s += strlen(s) + 1)
00063

```

16.403 l4/sys/kip.h File Reference

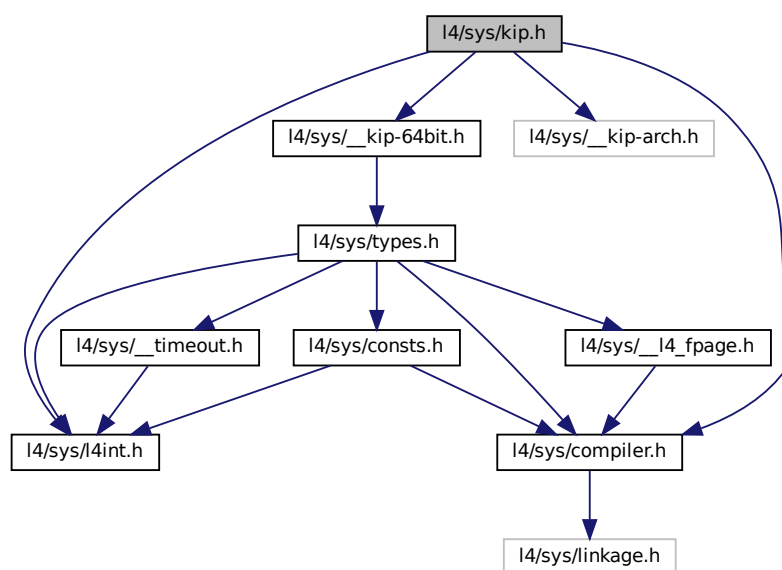
Kernel Info Page access functions.

```

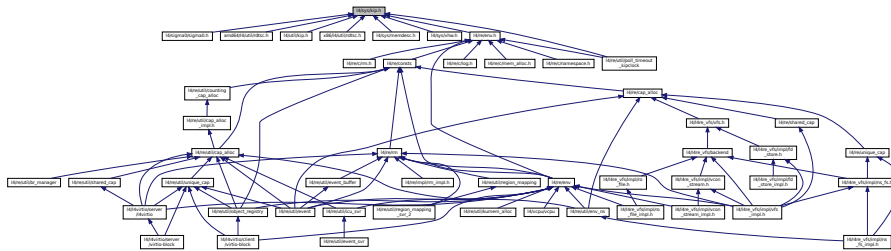
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
#include <l4/sys/__kip-arch.h>
#include <l4/sys/__kip-64bit.h>

```

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").

Functions

- `l4_umword_t l4_kip_version (l4_kernel_info_t *kip) L4_NOTHROW`
Get the kernel version.
- `const char * l4_kip_version_string (l4_kernel_info_t *kip) L4_NOTHROW`
Get the kernel version string.
- `int l4_kernel_info_version_offset (l4_kernel_info_t *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 *kip) L4_NOTHROW`
Return clock value from the KIP.
- `l4_umword_t l4_kip_clock_lw (l4_kernel_info_t *kip) L4_NOTHROW`
Return least significant machine word of clock value from the KIP.

16.403.1 Detailed Description

Kernel Info Page access functions.

Definition in file [kip.h](#).

16.403.2 Function Documentation

16.403.2.1 l4_kernel_info_version_offset()

```
int l4_kernel_info_version_offset (
    l4_kernel_info_t * 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).
------------	--

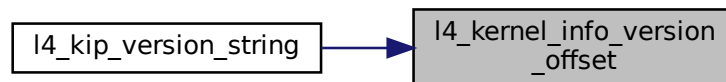
Returns

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

Definition at line 138 of file [kip.h](#).

Referenced by [l4_kip_version_string\(\)](#).

Here is the caller graph for this function:



16.403.2.2 l4_kip_clock()

```
l4_cpu_time_t l4_kip_clock (  
    l4_kernel_info_t * 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.

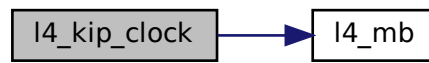
The KIP clock always contains the current (relative) time in micro seconds independently of the CPU frequency. The clock is only guaranteed to be accurate within the scheduling granularity announced in the KIP.

Definition at line 142 of file [kip.h](#).

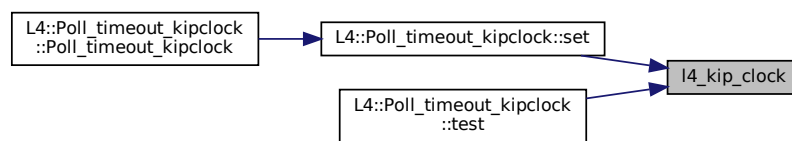
References [l4_mb\(\)](#).

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:



16.403.2.3 l4_kip_clock_lw()

```

l4_umword_t l4_kip_clock_lw (
    l4_kernel_info_t * 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.

Definition at line 164 of file [kip.h](#).

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

Here is the call graph for this function:



16.403.2.4 l4_kip_version()

```
l4_umword_t l4_kip_version (  
    l4_kernel_info_t * 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 130 of file [kip.h](#).

16.403.2.5 l4_kip_version_string()

```
const char * l4_kip_version_string (  
    l4_kernel_info_t * kip ) [inline]
```

Get the kernel version string.

Parameters

<i>kip</i>	Kernel Info Page.
------------	-------------------

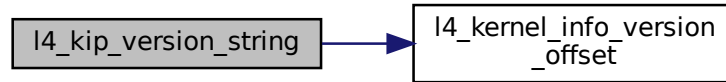
Returns

Kernel version string.

Definition at line 134 of file [kip.h](#).

References [l4_kernel_info_version_offset\(\)](#).

Here is the call graph for this function:



16.404 kip.h

```

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
00054
00058 enum l4_kernel_info_consts_t
00059 {
00060     L4_KIP_VERSION_FIASCO      = 0x87004444,
00061     L4_KIP_VERSION_FIASCO_MASK = 0xff00ffff,
00062 };
00063
00067 #define L4_KERNEL_INFO_MAGIC (0x4BE6344CL) /* "L4pK" */
00068
00069
00077 L4_INLINE l4_umword_t l4_kip_version(l4_kernel_info_t *kip) L4_NOTHROW;
00078
00086 L4_INLINE const char *l4_kip_version_string(l4_kernel_info_t *kip) L4_NOTHROW;
00087
00096 L4_INLINE int
00097 l4_kernel_info_version_offset(l4_kernel_info_t *kip) L4_NOTHROW;
00098
00110 L4_INLINE l4_cpu_time_t
00111 l4_kip_clock(l4_kernel_info_t *kip) L4_NOTHROW;
00112
00120 L4_INLINE l4_umword_t
  
```



```

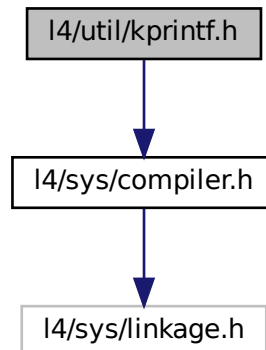
00121 l4_kip_clock_lw(l4_kernel_info_t *kip) L4_NOTHROW;
00122
00125 /*****
00126  * Implementations
00127  *****/
00128
00129 L4_INLINE l4_umword_t
00130 l4_kip_version(l4_kernel_info_t *kip) L4_NOTHROW
00131 { return kip->version & L4_KIP_VERSION_FIASCO_MASK; }
00132
00133 L4_INLINE const char*
00134 l4_kip_version_string(l4_kernel_info_t *k) L4_NOTHROW
00135 { return (const char *)k + l4_kernel_info_version_offset(k); }
00136
00137 L4_INLINE int
00138 l4_kernel_info_version_offset(l4_kernel_info_t *kip) L4_NOTHROW
00139 { return kip->offset_version_strings « 4; }
00140
00141 L4_INLINE l4_cpu_time_t
00142 l4_kip_clock(l4_kernel_info_t *kip) L4_NOTHROW
00143 {
00144     unsigned long h1, l;
00145     unsigned long *c;
00146
00147     if (sizeof(unsigned long) == 8)
00148         return kip->_clock_val;
00149
00150     c = (unsigned long *)&kip->_clock_val;
00151     do
00152     {
00153         h1 = c[1];
00154         l4_mb();
00155         l = c[0];
00156         l4_mb();
00157     }
00158     while (h1 != c[1]);
00159
00160     return ((unsigned long long)h1 « 32) | l;
00161 }
00162
00163 L4_INLINE l4_umword_t
00164 l4_kip_clock_lw(l4_kernel_info_t *kip) L4_NOTHROW
00165 {
00166     /* We do the casting because the clock field is volatile */
00167     unsigned long *c = (unsigned long *)&kip->_clock_val;
00168     l4_mb();
00169     return c[0];
00170 }

```

16.405 l4/util/kprintf.h File Reference

printf using the kernel debugger

```
#include <l4/sys/compiler.h>
Include dependency graph for kprintf.h:
```



16.405.1 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](#).

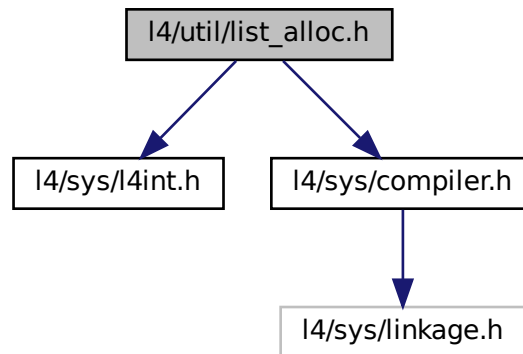
16.406 kprintf.h

```
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__ */
```

16.407 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.

16.407.1 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](#).

16.407.2 Function Documentation

16.407.2.1 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

16.407.2.2 l4la_avail()

```
l4_size_t l4la_avail (
    l4la_free_t ** first )
```

Show available memory in pool.

Parameters

<i>first</i>	list identifier
--------------	-----------------

16.407.2.3 l4la_dump()

```
void l4la_dump (
    l4la_free_t ** first )
```

Show all list members.

Parameters

<i>first</i>	list identifier
--------------	-----------------

16.407.2.4 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

16.407.2.5 l4la_init()

```
void l4la_init (
    l4la_free_t ** first )
```

Init memory pool.

Parameters

<i>first</i>	list identifier
--------------	-----------------

16.408 list_alloc.h

```
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

```

16.409 I4/util/lock.h File Reference

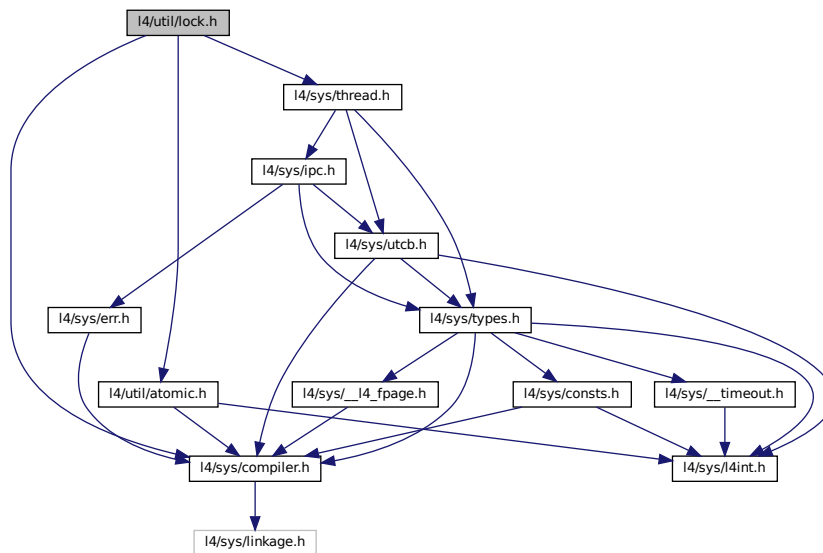
Simple lock implementation.

```

#include <l4/sys/thread.h>
#include <l4/sys/compiler.h>
#include <l4/util/atomic.h>

```

Include dependency graph for lock.h:



16.409.1 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](#).

16.410 lock.h

```

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

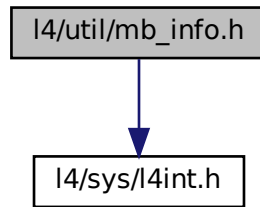
```

16.411 l4/util/mb_info.h File Reference

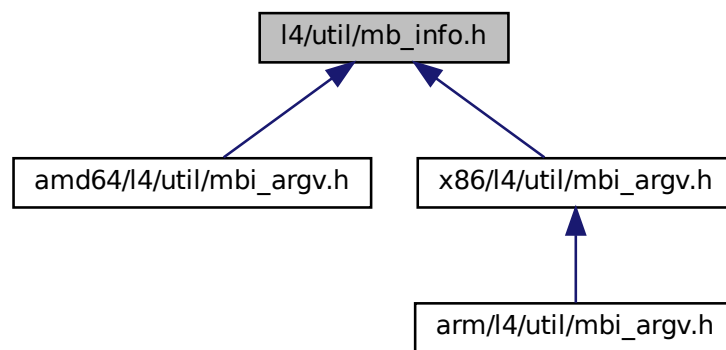
Multiboot info structure as defined by GRUB.

```
#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.

16.411.1 Detailed Description

Multiboot info structure as defined by GRUB.

Definition in file [mb_info.h](#).

16.411.2 Macro Definition Documentation

16.411.2.1 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 286 of file [mb_info.h](#).

16.411.2.2 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 298 of file [mb_info.h](#).

16.411.2.3 MB_ARD_MEMORY

```
#define MB_ARD_MEMORY 1
```

usable memory "Type", all others are reserved.

Definition at line 59 of file [mb_info.h](#).

16.411.2.4 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 65 of file [mb_info.h](#).

16.412 mb_info.h

```
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/l4int.h>
00024
00025 /*
00026  * \defgroup l4util_mb_mod Multiboot v1
00027  * \ingroup l4util_api
00028  */
00029
00034 typedef struct
00035 {
00036     l4_uint32_t mod_start;
00037     l4_uint32_t mod_end;
00038     l4_uint32_t cmdline;
00039     l4_uint32_t pad;
00040 } l4util_mb_mod_t;
00041
00042
00049 typedef struct __attribute__((packed))
00050 {
00051     l4_uint32_t struct_size;
00052     l4_uint64_t addr;
00053     l4_uint64_t size;
00054     l4_uint32_t type;
00055     /* unspecified optional padding... */
00056 } l4util_mb_addr_range_t;
00057
00059 #define MB_ARD_MEMORY 1
00060
00065 #define MB_ART_MEMORY 1
00066 #define MB_ART_RESERVED 2
00067 #define MB_ART_ACPI 3
00069 #define MB_ART_NVS 4
00070 #define MB_ART_UNUSABLE 5
00074 typedef struct
00075 {
00076     l4_uint32_t size;
00077     l4_uint8_t drive_number;
00078     l4_uint8_t drive_mode;
00079     l4_uint16_t drive_cylinders;
00080     l4_uint8_t drive_heads;
00081     l4_uint8_t drive_sectors;
00082     l4_uint16_t drive_ports[0];
00083 } l4util_mb_drive_t;
00084
00085 /* Drive Mode. */
00086 #define MB_DI_CHS_MODE 0
00087 #define MB_DI_LBA_MODE 1
00088
```

```

00089
00091 typedef struct
00092 {
00093     14_uint16_t version;
00094     14_uint16_t cseg;
00095     14_uint32_t offset;
00096     14_uint16_t cseg_16;
00097     14_uint16_t dseg_16;
00098     14_uint16_t cseg_len;
00099     14_uint16_t cseg_16_len;
00100     14_uint16_t dseg_16_len;
00101 } 14util_mb_apm_t;
00102
00103
00105 typedef struct
00106 {
00107     14_uint8_t signature[4];
00108     14_uint16_t version;
00109     14_uint32_t oem_string;
00110     14_uint32_t capabilities;
00111     14_uint32_t video_mode;
00112     14_uint16_t total_memory;
00113     14_uint16_t oem_software_rev;
00114     14_uint32_t oem_vendor_name;
00115     14_uint32_t oem_product_name;
00116     14_uint32_t oem_product_rev;
00117     14_uint8_t reserved[222];
00118     14_uint8_t oem_data[256];
00119 } __attribute__((packed)) 14util_mb_vbe_ctrl_t;
00120
00121
00123 typedef struct
00124 {
00127     14_uint16_t mode_attributes;
00128     14_uint8_t win_a_attributes;
00129     14_uint8_t win_b_attributes;
00130     14_uint16_t win_granularity;
00131     14_uint16_t win_size;
00132     14_uint16_t win_a_segment;
00133     14_uint16_t win_b_segment;
00134     14_uint32_t win_func;
00135     14_uint16_t bytes_per_scanline;
00140     14_uint16_t x_resolution;
00141     14_uint16_t y_resolution;
00142     14_uint8_t x_char_size;
00143     14_uint8_t y_char_size;
00144     14_uint8_t number_of_planes;
00145     14_uint8_t bits_per_pixel;
00146     14_uint8_t number_of_banks;
00147     14_uint8_t memory_model;
00148     14_uint8_t bank_size;
00149     14_uint8_t number_of_image_pages;
00150     14_uint8_t reserved0;
00155     14_uint8_t red_mask_size;
00156     14_uint8_t red_field_position;
00157     14_uint8_t green_mask_size;
00158     14_uint8_t green_field_position;
00159     14_uint8_t blue_mask_size;
00160     14_uint8_t blue_field_position;
00161     14_uint8_t reserved_mask_size;
00162     14_uint8_t reserved_field_position;
00163     14_uint8_t direct_color_mode_info;
00168     14_uint32_t phys_base;
00169     14_uint32_t reserved1;
00170     14_uint16_t reserved2;
00175     14_uint16_t linear_bytes_per_scanline;
00176     14_uint8_t banked_number_of_image_pages;
00177     14_uint8_t linear_number_of_image_pages;
00178     14_uint8_t linear_red_mask_size;
00179     14_uint8_t linear_red_field_position;
00180     14_uint8_t linear_green_mask_size;
00181     14_uint8_t linear_green_field_position;
00182     14_uint8_t linear_blue_mask_size;
00183     14_uint8_t linear_blue_field_position;
00184     14_uint8_t linear_reserved_mask_size;
00185     14_uint8_t linear_reserved_field_position;
00186     14_uint32_t max_pixel_clock;
00187
00188     /* The VBE spec says this structure should have a size of 256 bytes but
00189      * the described structure layout is only 255 bytes... */
00190     14_uint8_t reserved3[189 + 1];
00192 } __attribute__((packed)) 14util_mb_vbe_mode_t;
00193
00194
00202 typedef struct
00203 {
00204     14_uint32_t flags;

```

```

00205     l4_uint32_t mem_lower;
00206     l4_uint32_t mem_upper;
00207     l4_uint32_t boot_device;
00208     l4_uint32_t cmdline;
00209     l4_uint32_t mods_count;
00210     l4_uint32_t mods_addr;
00212     union
00213     {
00214         struct
00215         {
00217             l4_uint32_t tabsize;
00218             l4_uint32_t strsize;
00219             l4_uint32_t addr;
00220             l4_uint32_t pad;
00221         }
00222         a;
00223
00224         struct
00225         {
00227             l4_uint32_t num;
00228             l4_uint32_t size;
00229             l4_uint32_t addr;
00230             l4_uint32_t shndx;
00231         }
00232         e;
00233     }
00234     syms;
00235
00236     l4_uint32_t mmap_length;
00237     l4_uint32_t mmap_addr;
00238     l4_uint32_t drives_length;
00239     l4_uint32_t drives_addr;
00240     l4_uint32_t config_table;
00241     l4_uint32_t boot_loader_name;
00242     l4_uint32_t apm_table;
00243     l4_uint32_t vbe_ctrl_info;
00244     l4_uint32_t vbe_mode_info;
00245     l4_uint16_t vbe_mode;
00246     l4_uint16_t vbe_interface_seg;
00247     l4_uint16_t vbe_interface_off;
00248     l4_uint16_t vbe_interface_len;
00249 } l4util_mb_info_t;
00250
00257 static inline l4util_mb_addr_range_t *
00258 l4util_mb_first_mmap_entry(l4util_mb_info_t *mbi)
00259 {
00260     return (l4util_mb_addr_range_t *) (l4_addr_t) mbi->mmap_addr;
00261 }
00262
00272 static inline l4util_mb_addr_range_t *
00273 l4util_mb_next_mmap_entry(l4util_mb_addr_range_t *e)
00274 {
00275     return (l4util_mb_addr_range_t *) ((l4_addr_t) e + e->struct_size
00276                                         + sizeof(e->struct_size));
00277 }
00278
00286 #define l4util_mb_for_each_mmap_entry(i, mbi)           \
00287     for (i = l4util_mb_first_mmap_entry(mbi);         \
00288          (unsigned long) i < (unsigned long) mbi->mmap_addr + mbi->mmap_length; \
00289          i = l4util_mb_next_mmap_entry(i))
00290
00291 #endif /* ! __ASSEMBLY__ */
00292
00298 #define L4UTIL_MB_MEMORY      0x00000001
00299
00301 #define L4UTIL_MB_BOOTDEV     0x00000002
00302
00304 #define L4UTIL_MB_CMDLINE     0x00000004
00305
00307 #define L4UTIL_MB_MODS        0x00000008
00308
00309 /* These next two are mutually exclusive */
00311 #define L4UTIL_MB_AOUT_SYMS    0x00000010
00312
00314 #define L4UTIL_MB_ELF_SHDR     0x00000020
00315
00317 #define L4UTIL_MB_MEM_MAP      0x00000040
00318
00320 #define L4UTIL_MB_DRIVE_INFO   0x00000080
00321
00323 #define L4UTIL_MB_CONFIG_TABLE 0x00000100
00324
00326 #define L4UTIL_MB_BOOT_LOADER_NAME 0x00000200
00327
00329 #define L4UTIL_MB_APM_TABLE    0x00000400
00330
00332 #define L4UTIL_MB_VIDEO_INFO   0x00000800

```

```

00333
00334
00336 #define L4UTIL_MB_VALID      0x2BADB002UL
00337 #define L4UTIL_MB_VALID_ASM   0x2BADB002
00338
00339
00340 /*****
00341  * Multiboot2
00342  *****/
00343
00344 #ifndef __ASSEMBLY__
00345
00346 typedef struct
00347 {
00348     l4_uint32_t total_size;
00349     l4_uint32_t reserved;
00350 } __attribute__((packed)) l4util_mb2_info_t;
00351
00352 typedef struct
00353 {
00354     char string[0];
00355 } __attribute__((packed)) l4util_mb2_cmdline_tag_t;
00356
00357 typedef struct
00358 {
00359     l4_uint32_t mod_start;
00360     l4_uint32_t mod_end;
00361     char string[];
00362 } __attribute__((packed)) l4util_mb2_module_tag_t;
00363
00364 typedef struct
00365 {
00366     l4_uint64_t base_addr;
00367     l4_uint64_t length;
00368     l4_uint32_t type;
00369     l4_uint32_t reserved;
00370 } __attribute__((packed)) l4util_mb2_memmap_entry_t;
00371
00372 typedef struct
00373 {
00374     l4_uint32_t entry_size;
00375     l4_uint32_t entry_version;
00376     l4util_mb2_memmap_entry_t entries[];
00377 } __attribute__((packed)) l4util_mb2_memmap_tag_t;
00378
00379 typedef struct
00380 {
00381     char data[0];
00382 } __attribute__((packed)) l4util_mb2_rsdp_tag_t;
00383
00384 typedef struct
00385 {
00386     l4_uint32_t type;
00387     l4_uint32_t size;
00388
00389     union
00390     {
00391         l4util_mb2_cmdline_tag_t cmdline;
00392         l4util_mb2_module_tag_t module;
00393         l4util_mb2_memmap_tag_t memmap;
00394         l4util_mb2_rsdp_tag_t rsdp;
00395     };
00396 } __attribute__((packed)) l4util_mb2_tag_t;
00397
00398 #endif /* ! __ASSEMBLY__ */
00399
00400
00401 #define L4UTIL_MB2_MAGIC      0xE85250D6
00402 #define L4UTIL_MB2_ARCH_I386  0x0
00403
00404 #define L4UTIL_MB2_TERMINATOR_HEADER_TAG  0
00405 #define L4UTIL_MB2_INFO_REQUEST_HEADER_TAG 1
00406 #define L4UTIL_MB2_ENTRY_ADDRESS_HEADER_TAG 3
00407 #define L4UTIL_MB2_RELOCATABLE_HEADER_TAG 10
00408
00409 #define L4UTIL_MB2_TAG_FLAG_REQUIRED 0
00410
00411 #define L4UTIL_MB2_TAG_ALIGN_SHIFT 3
00412 #define L4UTIL_MB2_TAG_ALIGN      8
00413
00414 #define L4UTIL_MB2_TERMINATOR_INFO_TAG      0
00415 #define L4UTIL_MB2_BOOT_CMDLINE_INFO_TAG    1
00416 #define L4UTIL_MB2_MODULE_INFO_TAG          3
00417 #define L4UTIL_MB2_MEMORY_MAP_INFO_TAG      6
00418 #define L4UTIL_MB2_RSDP_OLD_INFO_TAG        14
00419 #define L4UTIL_MB2_RSDP_NEW_INFO_TAG        15
00420 #define L4UTIL_MB2_IMAGE_LOAD_BASE_PHYS_INFO_TAG 21

```

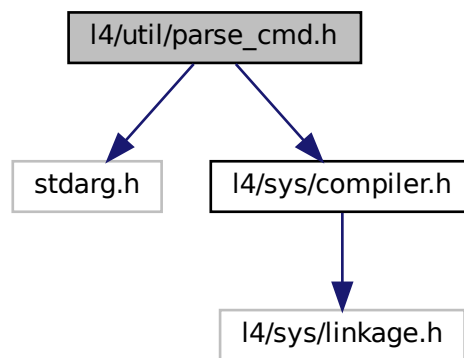
```
00421
00422 #define L4UTIL_MB2_RELO_PREFERRED_NONE 0
00423 #define L4UTIL_MB2_RELO_PREFERRED_MIN 1
00424 #define L4UTIL_MB2_RELO_PREFERRED_MAX 2
00425
00426 #endif
```

16.413 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, char arg0,...)
Parse the command-line for specified arguments and store the values into variables.

16.413.1 Detailed Description

comfortable command-line parsing

Date

2002

Author

Jork Loeser jork.loeser@inf.tu-dresden.de

Definition in file [parse_cmd.h](#).

16.413.2 Function Documentation

16.413.2.1 `parse_cmdline()`

```
int parse_cmdline (  
    int * argc,  
    const char *** argv,  
    char arg0,  
    ... )
```

Parse the command-line for specified arguments and store the values into variables.

This Functions gets the command-line, and a list of command-descriptors. Then, the command-line is parsed according to the given descriptors, storing strings, switches and numeric arguments at given addresses, and possibly calling specified functions. A default help descriptor is added. Its purpose is to present a short command overview in the case the given command-line does not fit to the descriptors.

Each command-descriptor has the following form:

short option char, long option name, comment, type, val, addr.

The *short option char* specifies the short form of the described option. The short form will be recognized after a single dash, or in a group of short options preceeded by a single dash. Specify '' if no short form should be used.

The *long option name* specifies the long form of the described option. The long form will be recognized after two dashes. Specify 0 if no long form should be used for this option.

The *comment* is a string that will be used when presenting the short command-line help.

The *type* specifies, if the option should be recognized as

- a number (`PARSE_CMD_INT`),
- a switch (`PARSE_CMD_SWITCH`),
- a string (`PARSE_CMD_STRING`),
- a function call (`PARSE_CMD_FN`, `PARSE_CMD_FN_ARG`),

- an increment/decrement operator (PARSE_CMD_INC, PARSE_CMD_DEC).

If type is PARSE_CMD_INT, the option requires a second argument on the command-line after the option. This argument is parsed as a number. It can be preceeded by 0x to present a hex-value or by 0 to present an octal form. *addr* is interpreted as an int-pointer. The scanned argument from the command-line is stored in this pointer.

If *type* is PARSE_CMD_SWITCH, *addr* must be a pointer to int, and the value from *val* is stored at this pointer.

With PARSE_CMD_STRING, an additional argument is expected at the cmdline. *addr* must be a pointer to const char*, and a pointer to the argument on the command line is stored at this pointer. The value in *val* is a default value, which is stored at *addr* if the corresponding option is not given on the command line.

With PARSE_CMD_FN_ARG, *addr* is interpreted as a function pointer of type [parse_cmd_fn_t](#). It will be called with *val* as argument if the corresponding option is found.

If *type* is PARSE_CMD_FN_ARG, *addr* is as a function pointer of type [parse_cmd_fn_arg_t](#), and handled similar to PARSE_CMD_FN. An additional argument is expected at the command line, however. It is given to the called function as 2nd argument, and parsed as an integer as with PARSE_CMD_INT as a third argument.

If *type* is PARSE_CMD_INC or PARSE_CMD_DEC, *addr* is interpreted as an int-pointer. The value of *val* is stored to this pointer first. For every occurrence of the option in the command line, the integer referenced by *addr* is incremented or decremented, respectively.

The list of command-descriptors is terminated by specifying a binary 0 for the short option char.

Note: The short option char 'h' and the long option name "help" must not be specified. They are used for the default help descriptor and produce a short command-options help when specified on the command-line.

Parameters

<i>argc</i>	pointer to number of command line parameters as passed to main
<i>argv</i>	pointer to array of command line parameters as passed to main
<i>arg0</i>	format list describing the command line options to parse for

Returns

0 if the command-line was successfully parsed, otherwise:

- -1 if the given descriptors are somehow wrong.
- -2 if not enough memory was available to hold temporary structs.
- -3 if the given command-line args did not meet the specified set.
- -4 if the help-option was given.

Upon return, *argc* and *argv* point to a list of arguments that were not scanned as arguments. See [getoptlong](#) for details on scanning.

16.414 parse_cmd.h

```
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 <l4/sys/compiler.h>
00022
00028
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, char arg0, ...);
00141 L4_CV int parse_cmdlinev(int *argc, const char***argv, char arg0, va_list va);
00142 L4_CV int parse_cmdline_extra(const char*argv0, const char*line, char delim,
00143                             char arg0,...);
00144
00145 EXTERN_C_END
00148 #endif
00149

```

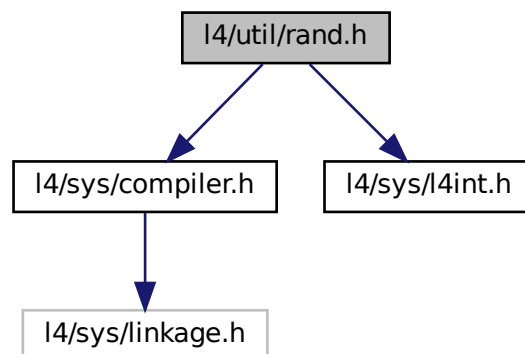
16.415 l4/util/rand.h File Reference

Simple Pseudo-Random Number Generator.

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

```
#include <l4/sys/l4int.h>
```

Include dependency graph for rand.h:



Functions

- [l4_uint32_t l4util_rand](#) (void)
Deliver next random number.
- void [l4util_srand](#) ([l4_uint32_t](#) seed)
Initialize random number generator.

16.415.1 Detailed Description

Simple Pseudo-Random Number Generator.

Date

1998

Author

Lars Reuther reuther@os.inf.tu-dresden.de

Definition in file [rand.h](#).

16.416 rand.h

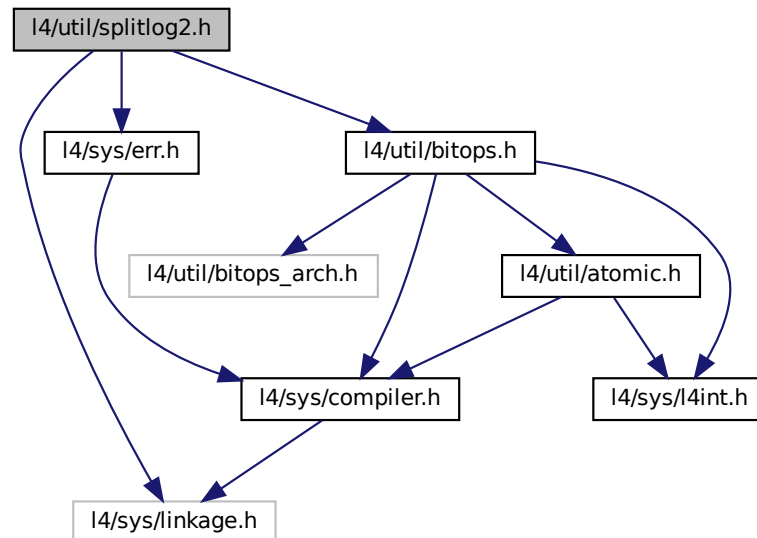
```
00001
00008 /*
00009  * (c) 2008-2009 Author(s)
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU Lesser General Public License 2.1.
00013  * Please see the COPYING-LGPL-2.1 file for details.
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 */
```

16.417 l4/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.

16.417.1 Detailed Description

Split a range in log2 aligned and size-aligned chunks.

Definition in file [splitlog2.h](#).

16.418 splitlog2.h

```

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
00071 L4_INLINE l4_addr_t
00072 l4util_splitlog2_size(l4_addr_t start, l4_addr_t end)
00073 {
00074     int start_bits = l4util_bsf(start);
00075     int len_bits = l4util_bsr(end - start + 1);
00076     if (start_bits != -1 && len_bits > start_bits)
00077         len_bits = start_bits;
00078     return len_bits;
00079 }
00080
00081
00082 #endif /* ! __L4UTIL__INCLUDE__SPLITLOG2_H__ */

```

16.419 l4/util/thread.h File Reference

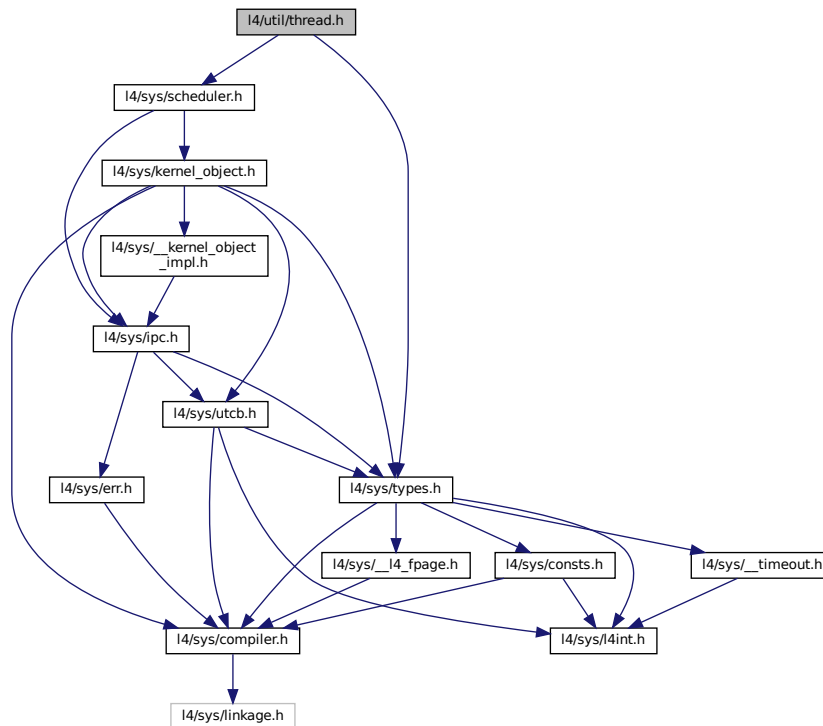
Low-level Thread Functions.

```

#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.

16.419.1 Detailed Description

Low-level Thread Functions.

Date

1997

Author

Sebastian Schönberg

Definition in file [thread.h](#).

16.419.2 Macro Definition Documentation

16.419.2.1 __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) { for(;;); }
```

```
thread_cap->ex_regs((l4_umword_t)helper_thread, stack_addr);
```

Definition at line 72 of file [thread.h](#).

16.420 thread.h

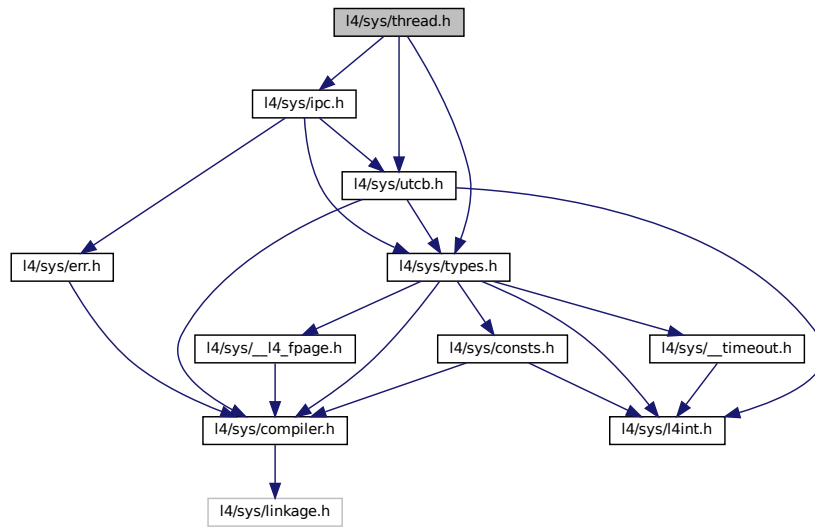
```
00001
00008 /*
00009  * (c) 2003-2009 Author(s)
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  * This file is part of TUD:OS and distributed under the terms of the
00012  * GNU Lesser General Public License 2.1.
00013  * Please see the COPYING-LGPL-2.1 file for details.
00014  */
00015
00016 #ifndef __L4_THREAD_H
00017 #define __L4_THREAD_H
00018
00019 #include <l4/sys/types.h>
00020 #include <l4/sys/scheduler.h>
00021
00022 EXTERN_C_BEGIN
00023
00046 L4_CV long
00047 l4util_create_thread(l4_cap_idx_t id, l4_utcb_t *thread_utcb,
00048                     l4_cap_idx_t factory,
00049                     l4_umword_t pc, l4_umword_t sp, l4_cap_idx_t pager,
00050                     l4_cap_idx_t task,
00051                     l4_cap_idx_t scheduler, l4_sched_param_t scp) L4_NOTHROW;
00052
00053 EXTERN_C_END
00054
00055 #ifndef L4UTIL_THREAD_FUNC
00072 #define __L4UTIL_THREAD_FUNC(name) void L4_NORETURN name(void)
00073 #define L4UTIL_THREAD_FUNC(name) __L4UTIL_THREAD_FUNC(name)
00074 #define __L4UTIL_THREAD_STATIC_FUNC(name) static L4_NORETURN void name(void)
00075 #define L4UTIL_THREAD_STATIC_FUNC(name) __L4UTIL_THREAD_STATIC_FUNC(name)
00076 #endif
00077
00078 #endif /* __L4_THREAD_H */
```

16.421 l4/sys/thread.h File Reference

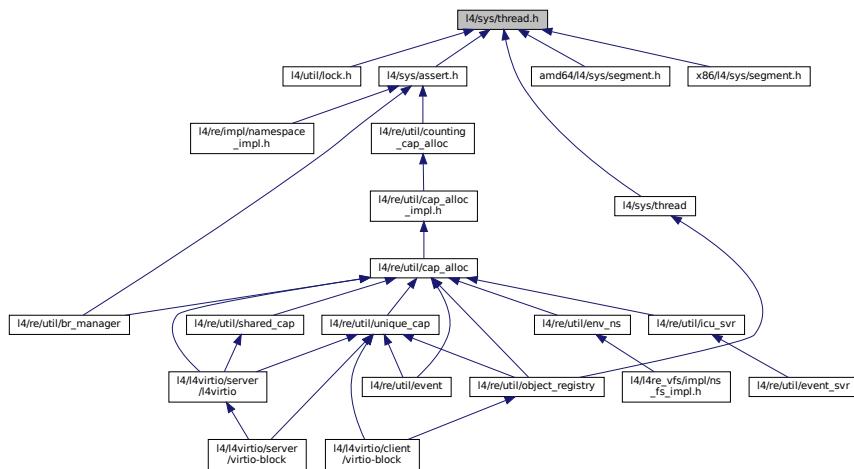
Common thread related definitions.

```
#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 or disable 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 or disable 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 or disable 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 or disable 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.

16.421.1 Detailed Description

Common thread related definitions.

Definition in file [thread.h](#).

16.422 thread.h

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *      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
00087 L4_INLINE l4_msgtag_t
00088 l4_thread_ex_regs(l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp,
00089                  l4_umword_t flags) L4_NOTHROW;
00090
00097 L4_INLINE l4_msgtag_t

```

```

00098 l4_thread_ex_regs_u(l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp,
00099                      l4_umword_t flags, l4_utcb_t *utcb) L4_NOTHROW;
00100
00124 L4_INLINE l4_msgtag_t
00125 l4_thread_ex_regs_ret(l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp,
00126                      l4_umword_t *flags) L4_NOTHROW;
00127
00134 L4_INLINE l4_msgtag_t
00135 l4_thread_ex_regs_ret_u(l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp,
00136                        l4_umword_t *flags, l4_utcb_t *utcb) L4_NOTHROW;
00137
00138
00139
00187 L4_INLINE void
00188 l4_thread_control_start(void) L4_NOTHROW;
00189
00194 L4_INLINE void
00195 l4_thread_control_start_u(l4_utcb_t *utcb) L4_NOTHROW;
00196
00206 L4_INLINE void
00207 l4_thread_control_pager(l4_cap_idx_t pager) L4_NOTHROW;
00208
00213 L4_INLINE void
00214 l4_thread_control_pager_u(l4_cap_idx_t pager, l4_utcb_t *utcb) L4_NOTHROW;
00215
00225 L4_INLINE void
00226 l4_thread_control_exc_handler(l4_cap_idx_t exc_handler) L4_NOTHROW;
00227
00232 L4_INLINE void
00233 l4_thread_control_exc_handler_u(l4_cap_idx_t exc_handler,
00234                                l4_utcb_t *utcb) L4_NOTHROW;
00235
00256 L4_INLINE void
00257 l4_thread_control_bind(l4_utcb_t *thread_utcb,
00258                       l4_cap_idx_t task) L4_NOTHROW;
00259
00264 L4_INLINE void
00265 l4_thread_control_bind_u(l4_utcb_t *thread_utcb,
00266                          l4_cap_idx_t task, l4_utcb_t *utcb) L4_NOTHROW;
00267
00282 L4_INLINE void
00283 l4_thread_control_alien(int on) L4_NOTHROW;
00284
00289 L4_INLINE void
00290 l4_thread_control_alien_u(l4_utcb_t *utcb, int on) L4_NOTHROW;
00291
00302 L4_INLINE void
00303 l4_thread_control_ux_host_syscall(int on) L4_NOTHROW;
00304
00309 L4_INLINE void
00310 l4_thread_control_ux_host_syscall_u(l4_utcb_t *utcb, int on) L4_NOTHROW;
00311
00312
00313
00321 L4_INLINE l4_msgtag_t
00322 l4_thread_control_commit(l4_cap_idx_t thread) L4_NOTHROW;
00323
00328 L4_INLINE l4_msgtag_t
00329 l4_thread_control_commit_u(l4_cap_idx_t thread, l4_utcb_t *utcb) L4_NOTHROW;
00330
00337 L4_INLINE l4_msgtag_t
00338 l4_thread_yield(void) L4_NOTHROW;
00339
00348 L4_INLINE l4_msgtag_t
00349 l4_thread_switch(l4_cap_idx_t to_thread) L4_NOTHROW;
00350
00355 L4_INLINE l4_msgtag_t
00356 l4_thread_switch_u(l4_cap_idx_t to_thread, l4_utcb_t *utcb) L4_NOTHROW;
00357
00358
00359
00369 L4_INLINE l4_msgtag_t
00370 l4_thread_stats_time(l4_cap_idx_t thread, l4_kernel_clock_t *us) L4_NOTHROW;
00371
00376 L4_INLINE l4_msgtag_t
00377 l4_thread_stats_time_u(l4_cap_idx_t thread, l4_kernel_clock_t *us,
00378                       l4_utcb_t *utcb) L4_NOTHROW;
00379
00380
00391 L4_INLINE l4_msgtag_t
00392 l4_thread_vcpu_resume_start(void) L4_NOTHROW;
00393
00398 L4_INLINE l4_msgtag_t
00399 l4_thread_vcpu_resume_start_u(l4_utcb_t *utcb) L4_NOTHROW;
00400
00427 L4_INLINE l4_msgtag_t
00428 l4_thread_vcpu_resume_commit(l4_cap_idx_t thread,

```

```

00429             l4_msgtag_t tag) L4_NOTHROW;
00430
00435 L4_INLINE l4_msgtag_t
00436 l4_thread_vcpu_resume_commit_u(l4_cap_idx_t thread,
00437                                l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW;
00438
00439
00456 L4_INLINE l4_msgtag_t
00457 l4_thread_vcpu_control(l4_cap_idx_t thread, l4_addr_t vcpu_state) L4_NOTHROW;
00458
00466 L4_INLINE l4_msgtag_t
00467 l4_thread_vcpu_control_u(l4_cap_idx_t thread, l4_addr_t vcpu_state,
00468                          l4_utcb_t *utcb) L4_NOTHROW;
00469
00492 L4_INLINE l4_msgtag_t
00493 l4_thread_vcpu_control_ext(l4_cap_idx_t thread, l4_addr_t ext_vcpu_state) L4_NOTHROW;
00494
00502 L4_INLINE l4_msgtag_t
00503 l4_thread_vcpu_control_ext_u(l4_cap_idx_t thread, l4_addr_t ext_vcpu_state,
00504                              l4_utcb_t *utcb) L4_NOTHROW;
00505
00506
00519 L4_INLINE l4_msgtag_t
00520 l4_thread_register_del_irq(l4_cap_idx_t thread, l4_cap_idx_t irq) L4_NOTHROW;
00521
00526 L4_INLINE l4_msgtag_t
00527 l4_thread_register_del_irq_u(l4_cap_idx_t thread, l4_cap_idx_t irq,
00528                              l4_utcb_t *utcb) L4_NOTHROW;
00529
00541 L4_INLINE l4_msgtag_t
00542 l4_thread_modify_sender_start(void) L4_NOTHROW;
00543
00548 L4_INLINE l4_msgtag_t
00549 l4_thread_modify_sender_start_u(l4_utcb_t *u) L4_NOTHROW;
00550
00575 L4_INLINE int
00576 l4_thread_modify_sender_add(l4_umword_t match_mask,
00577                              l4_umword_t match,
00578                              l4_umword_t del_bits,
00579                              l4_umword_t add_bits,
00580                              l4_msgtag_t *tag) L4_NOTHROW;
00581
00586 L4_INLINE int
00587 l4_thread_modify_sender_add_u(l4_umword_t match_mask,
00588                               l4_umword_t match,
00589                               l4_umword_t del_bits,
00590                               l4_umword_t add_bits,
00591                               l4_msgtag_t *tag, l4_utcb_t *u) L4_NOTHROW;
00592
00604 L4_INLINE l4_msgtag_t
00605 l4_thread_modify_sender_commit(l4_cap_idx_t thread, l4_msgtag_t tag) L4_NOTHROW;
00606
00611 L4_INLINE l4_msgtag_t
00612 l4_thread_modify_sender_commit_u(l4_cap_idx_t thread, l4_msgtag_t tag,
00613                                  l4_utcb_t *u) L4_NOTHROW;
00614
00621 enum L4_thread_ops
00622 {
00623     L4_THREAD_CONTROL_OP           = 0UL,
00624     L4_THREAD_EX_REGS_OP          = 1UL,
00625     L4_THREAD_SWITCH_OP           = 2UL,
00626     L4_THREAD_STATS_OP            = 3UL,
00627     L4_THREAD_VCPU_RESUME_OP      = 4UL,
00628     L4_THREAD_REGISTER_DELETE_IRQ_OP = 5UL,
00629     L4_THREAD_MODIFY_SENDER_OP     = 6UL,
00630     L4_THREAD_VCPU_CONTROL_OP      = 7UL,
00631     L4_THREAD_VCPU_CONTROL_EXT_OP  = L4_THREAD_VCPU_CONTROL_OP | 0x10000,
00632     L4_THREAD_X86_GDT_OP          = 0x10UL,
00633     L4_THREAD_ARM_TPIDRURO_OP     = 0x10UL,
00634     L4_THREAD_AMD64_SET_SEGMENT_BASE_OP = 0x12UL,
00635     L4_THREAD_AMD64_GET_SEGMENT_INFO_OP = 0x13UL,
00636     L4_THREAD_OPCODE_MASK         = 0xffff,
00637 };
00638
00649 enum L4_thread_control_flags
00650 {
00652     L4_THREAD_CONTROL_SET_PAGER      = 0x00100000,
00654     L4_THREAD_CONTROL_BIND_TASK     = 0x02000000,
00656     L4_THREAD_CONTROL_ALIEN         = 0x04000000,
00658     L4_THREAD_CONTROL_UX_NATIVE     = 0x08000000,
00660     L4_THREAD_CONTROL_SET_EXC_HANDLER = 0x10000000,
00661 };
00662
00672 enum L4_thread_control_mr_indices
00673 {
00674     L4_THREAD_CONTROL_MR_IDX_FLAGS = 0,
00675     L4_THREAD_CONTROL_MR_IDX_PAGER = 1,

```

```

00676 L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER = 2,
00677 L4_THREAD_CONTROL_MR_IDX_FLAG_VALS   = 4,
00678 L4_THREAD_CONTROL_MR_IDX_BIND_UTCB   = 5,
00679 L4_THREAD_CONTROL_MR_IDX_BIND_TASK   = 6,
00680 };
00681
00682 enum L4_thread_ex_regs_flags
00683 {
00684     L4_THREAD_EX_REGS_CANCEL           = 0x10000UL,
00685     L4_THREAD_EX_REGS_TRIGGER_EXCEPTION = 0x20000UL,
00686 };
00687
00688 /* IMPLEMENTATION ----- */
00689 #include <l4/sys/ipc.h>
00690 #include <l4/sys/types.h>
00691
00692 L4_INLINE l4_msgtag_t
00693 l4_thread_ex_regs_u(l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp,
00694                    l4_umword_t flags, l4_utcb_t *utcb) L4_NOTHROW
00695 {
00696     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00697     v->mr[0] = L4_THREAD_EX_REGS_OP | flags;
00698     v->mr[1] = ip;
00699     v->mr[2] = sp;
00700     return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 3, 0, 0), L4_IPC_NEVER);
00701 }
00702
00703 L4_INLINE l4_msgtag_t
00704 l4_thread_ex_regs_ret_u(l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp,
00705                        l4_umword_t *flags, l4_utcb_t *utcb) L4_NOTHROW
00706 {
00707     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00708     l4_msgtag_t ret = l4_thread_ex_regs_u(thread, *ip, *sp, *flags, utcb);
00709     if (l4_error_u(ret, utcb))
00710         return ret;
00711
00712     *flags = v->mr[0];
00713     *ip    = v->mr[1];
00714     *sp    = v->mr[2];
00715     return ret;
00716 }
00717
00718 L4_INLINE void
00719 l4_thread_control_start_u(l4_utcb_t *utcb) L4_NOTHROW
00720 {
00721     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00722     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] = L4_THREAD_CONTROL_OP;
00723 }
00724
00725 L4_INLINE void
00726 l4_thread_control_pager_u(l4_cap_idx_t pager, l4_utcb_t *utcb) L4_NOTHROW
00727 {
00728     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00729     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |= L4_THREAD_CONTROL_SET_PAGER;
00730     v->mr[L4_THREAD_CONTROL_MR_IDX_PAGER] = pager;
00731 }
00732
00733 L4_INLINE void
00734 l4_thread_control_exc_handler_u(l4_cap_idx_t exc_handler,
00735                                l4_utcb_t *utcb) L4_NOTHROW
00736 {
00737     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00738     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |= L4_THREAD_CONTROL_SET_EXC_HANDLER;
00739     v->mr[L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER] = exc_handler;
00740 }
00741
00742 L4_INLINE void
00743 l4_thread_control_bind_u(l4_utcb_t *thread_utcb, l4_cap_idx_t task,
00744                        l4_utcb_t *utcb) L4_NOTHROW
00745 {
00746     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00747     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |= L4_THREAD_CONTROL_BIND_TASK;
00748     v->mr[L4_THREAD_CONTROL_MR_IDX_BIND_UTCB] = (l4_addr_t)thread_utcb;
00749     v->mr[L4_THREAD_CONTROL_MR_IDX_BIND_TASK] = L4_ITEM_MAP;
00750     v->mr[L4_THREAD_CONTROL_MR_IDX_BIND_TASK + 1] = l4_obj_fpage(task, 0, L4_CAP_FPAGE_RWS).raw;
00751 }
00752
00753 L4_INLINE void
00754 l4_thread_control_alien_u(l4_utcb_t *utcb, int on) L4_NOTHROW
00755 {
00756     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00757     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |= L4_THREAD_CONTROL_ALIEN;
00758     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAG_VALS] |= on ? L4_THREAD_CONTROL_ALIEN : 0;
00759 }
00760
00761
00762

```

```

00768 L4_INLINE void
00769 l4_thread_control_ux_host_syscall_u(l4_utcb_t *utcb, int on) L4_NOTHROW
00770 {
00771     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00772     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |= L4_THREAD_CONTROL_UX_NATIVE;
00773     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAG_VALS] |= on ? L4_THREAD_CONTROL_UX_NATIVE : 0;
00774 }
00775
00776 L4_INLINE l4_msgtag_t
00777 l4_thread_control_commit_u(l4_cap_idx_t thread, l4_utcb_t *utcb) L4_NOTHROW
00778 {
00779     int items = 0;
00780     if (l4_utcb_mr_u(utcb)->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] & L4_THREAD_CONTROL_BIND_TASK)
00781         items = 1;
00782     return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 6, items, 0), L4_IPC_NEVER);
00783 }
00784
00785
00786 L4_INLINE l4_msgtag_t
00787 l4_thread_yield(void) L4_NOTHROW
00788 {
00789     l4_ipc_receive(L4_INVALID_CAP, NULL, L4_IPC_BOTH_TIMEOUT_0);
00790     return l4_msgtag(0, 0, 0, 0);
00791 }
00792
00793 /* Preliminary, to be changed */
00794 L4_INLINE l4_msgtag_t
00795 l4_thread_switch_u(l4_cap_idx_t to_thread, l4_utcb_t *utcb) L4_NOTHROW
00796 {
00797     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00798     v->mr[0] = L4_THREAD_SWITCH_OP;
00799     return l4_ipc_call(to_thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER);
00800 }
00801
00802
00803 L4_INLINE l4_msgtag_t
00804 l4_thread_stats_time_u(l4_cap_idx_t thread, l4_kernel_clock_t *us,
00805                        l4_utcb_t *utcb) L4_NOTHROW
00806 {
00807     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00808     l4_msgtag_t res;
00809
00810     v->mr[0] = L4_THREAD_STATS_OP;
00811
00812     res = l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER);
00813
00814     if (l4_msgtag_has_error(res))
00815         return res;
00816
00817     *us = v->mr64[l4_utcb_mr64_idx(0)];
00818
00819     return res;
00820 }
00821
00822 L4_INLINE l4_msgtag_t
00823 l4_thread_vcpu_resume_start_u(l4_utcb_t *utcb) L4_NOTHROW
00824 {
00825     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00826     v->mr[0] = L4_THREAD_VCPU_RESUME_OP;
00827     return l4_msgtag(L4_PROTO_THREAD, 1, 0, 0);
00828 }
00829
00830 L4_INLINE l4_msgtag_t
00831 l4_thread_vcpu_resume_commit_u(l4_cap_idx_t thread,
00832                                l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW
00833 {
00834     return l4_ipc_call(thread, utcb, tag, L4_IPC_NEVER);
00835 }
00836
00837 L4_INLINE l4_msgtag_t
00838 l4_thread_ex_regs(l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp,
00839                  l4_umword_t flags) L4_NOTHROW
00840 {
00841     return l4_thread_ex_regs_u(thread, ip, sp, flags, l4_utcb());
00842 }
00843
00844 L4_INLINE l4_msgtag_t
00845 l4_thread_ex_regs_ret(l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp,
00846                      l4_umword_t *flags) L4_NOTHROW
00847 {
00848     return l4_thread_ex_regs_ret_u(thread, ip, sp, flags, l4_utcb());
00849 }
00850
00851 L4_INLINE void
00852 l4_thread_control_start(void) L4_NOTHROW
00853 {
00854     l4_thread_control_start_u(l4_utcb());

```

```

00855 }
00856
00857 L4_INLINE void
00858 l4_thread_control_pager(l4_cap_idx_t pager) L4_NOTHROW
00859 {
00860     l4_thread_control_pager_u(pager, l4_utcb());
00861 }
00862
00863 L4_INLINE void
00864 l4_thread_control_exc_handler(l4_cap_idx_t exc_handler) L4_NOTHROW
00865 {
00866     l4_thread_control_exc_handler_u(exc_handler, l4_utcb());
00867 }
00868
00869
00870 L4_INLINE void
00871 l4_thread_control_bind(l4_utcb_t *thread_utcb, l4_cap_idx_t task) L4_NOTHROW
00872 {
00873     l4_thread_control_bind_u(thread_utcb, task, l4_utcb());
00874 }
00875
00876 L4_INLINE void
00877 l4_thread_control_alien(int on) L4_NOTHROW
00878 {
00879     l4_thread_control_alien_u(l4_utcb(), on);
00880 }
00881
00882 L4_INLINE void
00883 l4_thread_control_ux_host_syscall(int on) L4_NOTHROW
00884 {
00885     l4_thread_control_ux_host_syscall_u(l4_utcb(), on);
00886 }
00887
00888 L4_INLINE l4_msgtag_t
00889 l4_thread_control_commit(l4_cap_idx_t thread) L4_NOTHROW
00890 {
00891     return l4_thread_control_commit_u(thread, l4_utcb());
00892 }
00893
00894
00895
00896
00897 L4_INLINE l4_msgtag_t
00898 l4_thread_switch(l4_cap_idx_t to_thread) L4_NOTHROW
00899 {
00900     return l4_thread_switch_u(to_thread, l4_utcb());
00901 }
00902
00903
00904
00905
00906 L4_INLINE l4_msgtag_t
00907 l4_thread_stats_time(l4_cap_idx_t thread, l4_kernel_clock_t *us) L4_NOTHROW
00908 {
00909     return l4_thread_stats_time_u(thread, us, l4_utcb());
00910 }
00911
00912 L4_INLINE l4_msgtag_t
00913 l4_thread_vcpu_resume_start(void) L4_NOTHROW
00914 {
00915     return l4_thread_vcpu_resume_start_u(l4_utcb());
00916 }
00917
00918 L4_INLINE l4_msgtag_t
00919 l4_thread_vcpu_resume_commit(l4_cap_idx_t thread,
00920                               l4_msgtag_t tag) L4_NOTHROW
00921 {
00922     return l4_thread_vcpu_resume_commit_u(thread, tag, l4_utcb());
00923 }
00924
00925
00926 L4_INLINE l4_msgtag_t
00927 l4_thread_register_del_irq_u(l4_cap_idx_t thread, l4_cap_idx_t irq,
00928                               l4_utcb_t *u) L4_NOTHROW
00929 {
00930     l4_msg_regs_t *m = l4_utcb_mr_u(u);
00931     m->mr[0] = L4_THREAD_REGISTER_DELETE_IRQ_OP;
00932     m->mr[1] = l4_map_obj_control(0, 0);
00933     m->mr[2] = l4_obj_fpage(irq, 0, L4_CAP_FPAGE_RWS).raw;
00934     return l4_ipc_call(thread, u, l4_msgtag(L4_PROTO_THREAD, 1, 1, 0), L4_IPC_NEVER);
00935 }
00936
00937
00938 L4_INLINE l4_msgtag_t
00939 l4_thread_register_del_irq(l4_cap_idx_t thread, l4_cap_idx_t irq) L4_NOTHROW
00940 {
00941     return l4_thread_register_del_irq_u(thread, irq, l4_utcb());

```

```

00942 }
00943
00944
00945 L4_INLINE l4_msgtag_t
00946 l4_thread_vcpu_control_u(l4_cap_idx_t thread, l4_addr_t vcpu_state,
00947                          l4_utcb_t *utcb) L4_NOTHROW
00948 {
00949     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00950     v->mr[0] = L4_THREAD_VCPU_CONTROL_OP;
00951     v->mr[1] = vcpu_state;
00952     return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0), L4_IPC_NEVER);
00953 }
00954
00955 L4_INLINE l4_msgtag_t
00956 l4_thread_vcpu_control(l4_cap_idx_t thread, l4_addr_t vcpu_state) L4_NOTHROW
00957 { return l4_thread_vcpu_control_u(thread, vcpu_state, l4_utcb()); }
00958
00959
00960 L4_INLINE l4_msgtag_t
00961 l4_thread_vcpu_control_ext_u(l4_cap_idx_t thread, l4_addr_t ext_vcpu_state,
00962                              l4_utcb_t *utcb) L4_NOTHROW
00963 {
00964     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00965     v->mr[0] = L4_THREAD_VCPU_CONTROL_EXT_OP;
00966     v->mr[1] = ext_vcpu_state;
00967     return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0), L4_IPC_NEVER);
00968 }
00969
00970 L4_INLINE l4_msgtag_t
00971 l4_thread_vcpu_control_ext(l4_cap_idx_t thread, l4_addr_t ext_vcpu_state) L4_NOTHROW
00972 { return l4_thread_vcpu_control_ext_u(thread, ext_vcpu_state, l4_utcb()); }
00973
00974 L4_INLINE l4_msgtag_t
00975 l4_thread_modify_sender_start_u(l4_utcb_t *u) L4_NOTHROW
00976 {
00977     l4_msg_regs_t *m = l4_utcb_mr_u(u);
00978     m->mr[0] = L4_THREAD_MODIFY_SENDER_OP;
00979     return l4_msgtag(L4_PROTO_THREAD, 1, 0, 0);
00980 }
00981
00982 L4_INLINE int
00983 l4_thread_modify_sender_add_u(l4_umword_t match_mask,
00984                               l4_umword_t match,
00985                               l4_umword_t del_bits,
00986                               l4_umword_t add_bits,
00987                               l4_msgtag_t *tag, l4_utcb_t *u) L4_NOTHROW
00988 {
00989     l4_msg_regs_t *m = l4_utcb_mr_u(u);
00990     unsigned w = l4_msgtag_words(*tag);
00991     if (w >= L4_UTCB_GENERIC_DATA_SIZE - 4)
00992         return -L4_ENOMEM;
00993
00994     m->mr[w] = match_mask;
00995     m->mr[w+1] = match;
00996     m->mr[w+2] = del_bits;
00997     m->mr[w+3] = add_bits;
00998
00999     *tag = l4_msgtag(l4_msgtag_label(*tag), w + 4, 0, 0);
01000
01001     return 0;
01002 }
01003
01004 L4_INLINE l4_msgtag_t
01005 l4_thread_modify_sender_commit_u(l4_cap_idx_t thread, l4_msgtag_t tag,
01006                                  l4_utcb_t *u) L4_NOTHROW
01007 {
01008     return l4_ipc_call(thread, u, tag, L4_IPC_NEVER);
01009 }
01010
01011 L4_INLINE l4_msgtag_t
01012 l4_thread_modify_sender_start(void) L4_NOTHROW
01013 {
01014     return l4_thread_modify_sender_start_u(l4_utcb());
01015 }
01016
01017 L4_INLINE int
01018 l4_thread_modify_sender_add(l4_umword_t match_mask,
01019                             l4_umword_t match,
01020                             l4_umword_t del_bits,
01021                             l4_umword_t add_bits,
01022                             l4_msgtag_t *tag) L4_NOTHROW
01023 {
01024     return l4_thread_modify_sender_add_u(match_mask, match,
01025                                           del_bits, add_bits, tag, l4_utcb());
01026 }
01027
01028 L4_INLINE l4_msgtag_t

```



```

01029 l4_thread_modify_sender_commit(l4_cap_idx_t thread, l4_msgtag_t tag) L4_NOTHROW
01030 {
01031     return l4_thread_modify_sender_commit_u(thread, tag, l4_utcb());
01032 }

```

16.423 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.

16.423.1 Detailed Description

ARM-specific thread related definitions.

Definition in file [thread.h](#).

16.424 thread.h

```

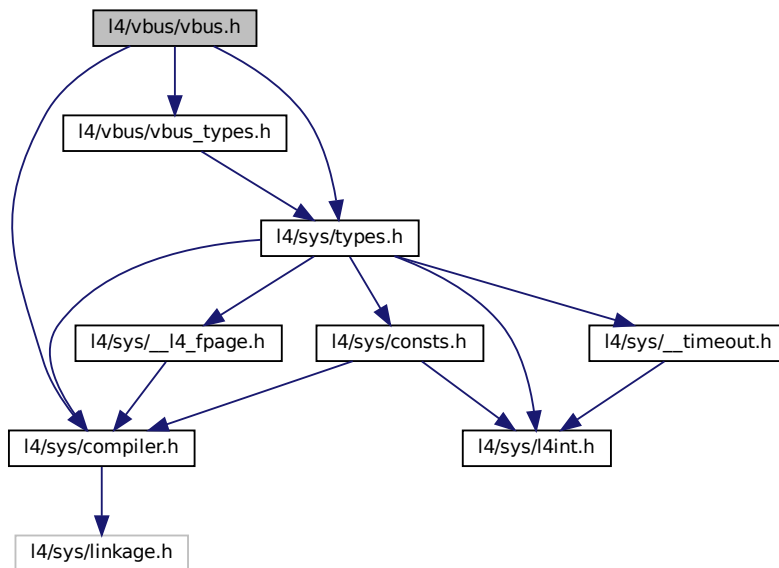
00001
00002 /*
00003  * (c) 2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * This file is part of TUD:OS and distributed under the terms of the
00007  * GNU General Public License 2.
00008  * Please see the COPYING-GPL-2 file for details.
00009  *
00010  * As a special exception, you may use this file as part of a free software
00011  * library without restriction. Specifically, if other files instantiate
00012  * templates or use macros or inline functions from this file, or you compile
00013  * this file and link it with other files to produce an executable, this
00014  * file does not by itself cause the resulting executable to be covered by
00015  * the GNU General Public License. This exception does not however
00016  * invalidate any other reasons why the executable file might be covered by
00017  * the GNU General Public License.
00018  */
00019 #pragma once
00020
00021 #include_next <l4/sys/thread.h>
00022
00023 L4_INLINE l4_msgtag_t
00024 l4_thread_arm_set_tpidruro(l4_cap_idx_t thread, l4_addr_t tpidruro) L4_NOTHROW;
00025
00026 L4_INLINE l4_msgtag_t
00027 l4_thread_arm_set_tpidruro_u(l4_cap_idx_t thread, l4_addr_t tpidruro,
00028                             l4_utcb_t *utcb) L4_NOTHROW;
00029
00030 /* IMPLEMENTATION ----- */
00031
00032 L4_INLINE l4_msgtag_t
00033 l4_thread_arm_set_tpidruro_u(l4_cap_idx_t thread, l4_addr_t tpidruro,
00034                             l4_utcb_t *utcb) L4_NOTHROW
00035 {
00036     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00037     v->mr[0] = L4_THREAD_ARM_TPIDRURO_OP;
00038     v->mr[1] = tpidruro;
00039     return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0),
00040                       L4_IPC_NEVER);
00041 }
00042
00043 L4_INLINE l4_msgtag_t
00044 l4_thread_arm_set_tpidruro(l4_cap_idx_t thread, l4_addr_t tpidruro) L4_NOTHROW
00045 {
00046     return l4_thread_arm_set_tpidruro_u(thread, tpidruro, l4_utcb());
00047 }

```

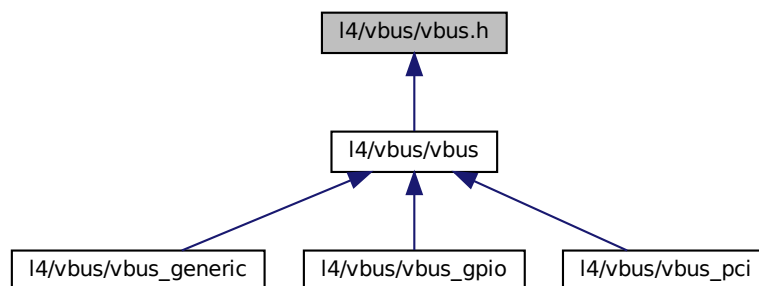
16.425 l4/vbus/vbus.h File Reference

Description of the vbus C API.

```
#include <l4/sys/compiler.h>
#include <l4/vbus/vbus_types.h>
#include <l4/sys/types.h>
Include dependency graph for vbus.h:
```



This graph shows which files directly or indirectly include this file:



Enumerations

- enum { `L4VBUS_NULL` = (`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_request_ioport](#) ([l4_cap_idx_t](#) vbus, [l4vbus_resource_t](#) const *res)

Request an IO port resource.

- int [l4vbus_request_resource](#) ([l4_cap_idx_t](#) vbus, [l4vbus_resource_t](#) const *res, int flags)

Request a resource of a specific type.

- 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 ([L4::Task](#)) 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_release_resource](#) ([l4_cap_idx_t](#) vbus, [l4vbus_resource_t](#) const *res)

Release a previously requested 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.

16.425.1 Detailed Description

Description of the vbus C API.

Definition in file [vbus.h](#).

16.425.2 Enumeration Type Documentation

16.425.2.1 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](#).

16.425.2.2 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 l4_icu_msi_info() to denote that the ICU should interpret the source ID as a device handle.
---------------------------	---

Definition at line 28 of file [vbus.h](#).

16.426 vbus.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  */
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
00157 int L4_CV
00158 l4vbus_request_ioport(l4_cap_idx_t vbus, l4vbus_resource_t const *res);
00159
00175 int L4_CV
00176 L4_DEPRECATED("use l4vbus_request_ioport")
00177 l4vbus_request_resource(l4_cap_idx_t vbus, l4vbus_resource_t const *res,
00178                        int flags);
00179
00183 enum L4vbus_dma_domain_assign_flags
00184 {
00186     L4VBUS_DMAD_UNBIND = 0,
00188     L4VBUS_DMAD_BIND   = 1,
00190     L4VBUS_DMAD_L4RE_DMA_SPACE = 0,
00192     L4VBUS_DMAD_KERNEL_DMA_SPACE = 2,
00193 };
00194
00215 int L4_CV
00216 l4vbus_assign_dma_domain(l4_cap_idx_t vbus, unsigned domain_id,
00217                          unsigned flags, l4_cap_idx_t dma_space);
00218
00227 int L4_CV
00228 l4vbus_release_ioport(l4_cap_idx_t vbus, l4vbus_resource_t const *res);
00229
00241 int L4_CV
00242 L4_DEPRECATED("use l4vbus_release_ioport")
00243 l4vbus_release_resource(l4_cap_idx_t vbus, l4vbus_resource_t const *res);
00244
00254 int L4_CV
00255 l4vbus_vicu_get_cap(l4_cap_idx_t vbus, l4vbus_device_handle_t icu,
00256                    l4_cap_idx_t cap);
00257
00258 __END_DECLS
00259

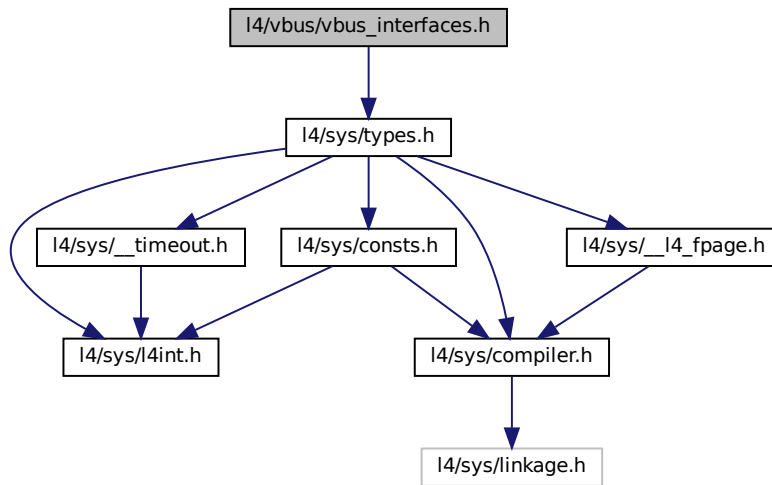
```

16.427 l4/vbus/vbus_interfaces.h File Reference

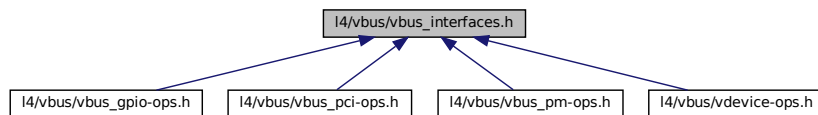
This header contains the definition of VBUS sub-interfaces and convenience functions to work with the interface IDs.

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

Include dependency graph for vbus_interfaces.h:



This graph shows which files directly or indirectly include this file:



Enumerations

- enum `l4vbus_iface_type_t`
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.

16.427.1 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](#).

16.427.2 Enumeration Type Documentation

16.427.2.1 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 42 of file [vbus_interfaces.h](#).

16.427.2.2 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.

Definition at line 31 of file [vbus_interfaces.h](#).

16.427.3 Function Documentation

16.427.3.1 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 93 of file [vbus_interfaces.h](#).

16.428 vbus_interfaces.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  */
00013 #pragma once
00014
00015 #include <l4/sys/types.h>
00016
00031 enum l4vbus_iface_type_t {
00032     L4VBUS_INTERFACE_ICU = 0,
00033     L4VBUS_INTERFACE_GPIO,
00034     L4VBUS_INTERFACE_PCI,
00035     L4VBUS_INTERFACE_PCIDEV,
00036     L4VBUS_INTERFACE_PM,
00037     L4VBUS_INTERFACE_BUS,
00038     L4VBUS_INTERFACE_GENERIC = 0x20
00039 };
00040
00041
00042 enum {
00050     L4VBUS_IFACE_SHIFT = 26
00051 };
00052
00064 L4_INLINE unsigned l4vbus_subinterface(unsigned opcode)
00065 {
00066     return opcode » L4VBUS_IFACE_SHIFT;
00067 }
00068
00080 L4_INLINE unsigned l4vbus_interface_opcode(unsigned opcode)
00081 {
00082     return opcode & ((1 « L4VBUS_IFACE_SHIFT) - 1);
00083 }
00084
00093 L4_INLINE int l4vbus_subinterface_supported(l4_uint32_t dev_type,
00094                                             l4vbus_iface_type_t iface_type)
00095 {
00096     if (iface_type == L4VBUS_INTERFACE_GENERIC)
00097         return 1;
00098
00099     return (dev_type & (1 « iface_type)) ? 1 : 0;
00100 }

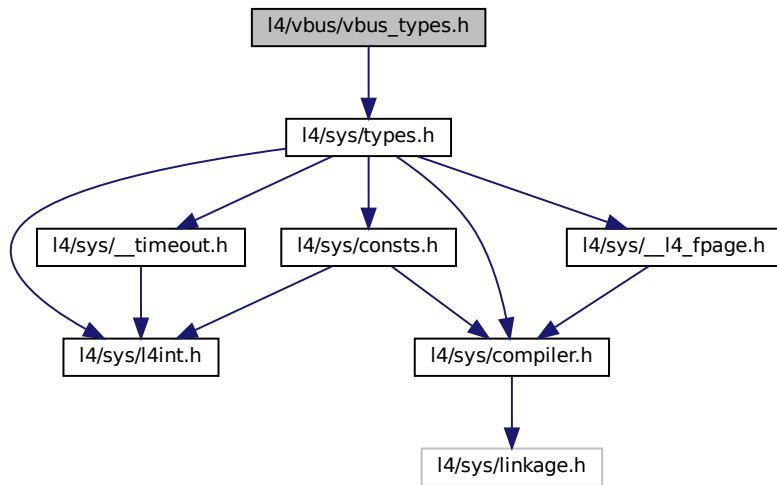
```

16.429 l4/vbus/vbus_types.h File Reference

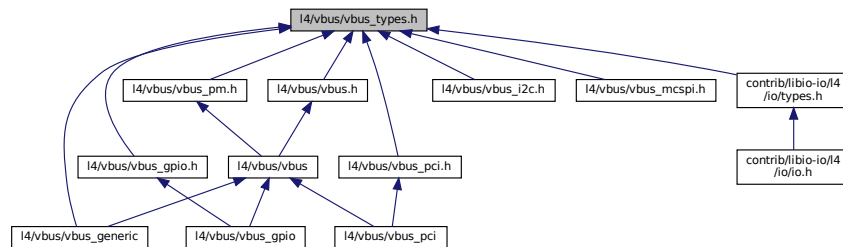
This header file contains descriptions of vbus related data types and constants.


```
#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.

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_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](#).

16.429.1 Detailed Description

This header file contains descriptions of vbus related data types and constants.

Definition in file [vbus_types.h](#).

16.429.2 Enumeration Type Documentation

16.429.2.1 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 81 of file [vbus_types.h](#).

16.429.2.2 l4vbus_resource_flags_t

enum [l4vbus_resource_flags_t](#)

Description of vbus resource flags.

At the moment these flags are mostly place holders as the IO server does not return any resource flag at all. These two flags were introduced for custom servers implementing the vbus protocol.

Enumerator

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 56 of file [vbus_types.h](#).

16.429.2.3 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 39 of file [vbus_types.h](#).

16.430 vbus_types.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  */
00015 #pragma once
00016
00017 #include <l4/sys/types.h>
00018
00019 typedef l4_mword_t l4vbus_device_handle_t;
00020 typedef l4_addr_t l4vbus_paddr_t;
00021
00023 typedef struct {
00025     l4_uint16_t    type;
00027     l4_uint16_t    flags;
00029     l4vbus_paddr_t start;
00031     l4vbus_paddr_t end;
00033     l4vbus_device_handle_t provider;
00035     l4_uint32_t id;
00036 } l4vbus_resource_t;
00037
00039 enum l4vbus_resource_type_t {
00040     L4VBUS_RESOURCE_INVALID = 0,
00041     L4VBUS_RESOURCE_IRQ,
00042     L4VBUS_RESOURCE_MEM,
00043     L4VBUS_RESOURCE_PORT,
00044     L4VBUS_RESOURCE_BUS,
00045     L4VBUS_RESOURCE_GPIO,
00046     L4VBUS_RESOURCE_DMA_DOMAIN,
00047     L4VBUS_RESOURCE_MAX,
00048 };
00049
00056 enum l4vbus_resource_flags_t {
00058     L4VBUS_RESOURCE_F_MEM_MMIO_READ = 0x2000,
00060     L4VBUS_RESOURCE_F_MEM_MMIO_WRITE = 0x4000,
00061 };

```

```
00062
00063 enum l4vbus_consts_t {
00064     L4VBUS_DEV_NAME_LEN = 64,
00065     L4VBUS_MAX_DEPTH = 100,
00066 };
00067
00069 typedef struct {
00071     l4_uint32_t    type;
00073     char           name[L4VBUS_DEV_NAME_LEN];
00075     unsigned       num_resources;
00077     unsigned       flags;
00078 } l4vbus_device_t;
00079
00081 enum l4vbus_device_flags_t {
00082     L4VBUS_DEVICE_F_CHILDREN = 0x10,
00083 };
```

Chapter 17

Example Documentation

17.1 hello/server/src/main.c

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. 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.

```

# 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.

```

/*
 * (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"
    "thread:\n\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(tl,
                       (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, tl, &sp);
if (l4_error(tag))
    return 4;
/* Receive initial exception from just started thread */
tag = l4_ipc_receive(tl, 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.

```

/*
 * (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.

```

/*
 * (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.
 */
#if 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 */
#if 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);
    }
#ifdef 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");
}
#ifdef 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");
}
#ifdef 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 (l4_is_invalid_cap(alien = l4re_util_cap_alloc()))
        return 1;
    l4_touch_rw(alien_thread_stack, sizeof(alien_thread_stack));
    tag = l4_factory_create_thread(l4re_env()->factory, alien);
    if (l4_error(tag))
        return 2;
    l4_debugger_set_object_name(alien, "alienth");
    l4_addr_t kumem;
    if (l4re_util_kumem_alloc(&kumem, 0, L4RE_THIS_TASK_CAP, l4re_env()->rm)
        return 3;
    l4_thread_control_start();
    l4_thread_control_pager(l4re_env()->main_thread);
    l4_thread_control_exc_handler(l4re_env()->main_thread);
    l4_thread_control_bind((l4_utcb_t *)kumem, L4RE_THIS_TASK_CAP);
    l4_thread_control_alien(1);
    tag = l4_thread_control_commit(alien);
    if (l4_error(tag))
        return 4;
    tag = l4_thread_ex_regs(alien,
                           (l4_umword_t)alien_thread,
                           (l4_umword_t)alien_thread_stack + sizeof(alien_thread_stack),
                           0);
    if (l4_error(tag))
        return 5;
    l4_sched_param_t sp = l4_sched_param(1, 0);
    tag = l4_scheduler_run_thread(l4re_env()->scheduler, alien, &sp);
    if (l4_error(tag))
        return 6;
    calibrate_timer();
    /* Pager/Exception loop */
    if (l4_msgtag_has_error(tag = l4_ipc_receive(alien, u, L4_IPC_NEVER)))
    {
        printf("l4_ipc_receive failed");
        return 7;
    }
    memcpy(&exc, l4_utcb_exc(), sizeof(exc));
    mr0 = l4_utcb_mr()->mr[0];
    mr1 = l4_utcb_mr()->mr[1];
    for (;;)
    {
        s = l4_rdtsc();
        if (l4_msgtag_is_exception(tag))
        {
            print_exc_state(&exc);
            tag = l4_msgtag_is_alien_after_call(&exc)
                ? 0 : L4_PROTO_ALLOW_SYSCALL,
                L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);

```

```

    }
    else
    {
        printf("Umm, non-handled request (like PF): %lx %lx\n", mr0, mr1);
        memcpy(l4_utcb_exc(), &exc, sizeof(exc));
        /* Reply and wait */
        if (l4_msgtag_has_error(tag = l4_ipc_call(alien, u, tag, L4_IPC_NEVER)))
        {
            printf("l4_ipc_call failed\n");
            return 8;
        }
        memcpy(&exc, l4_utcb_exc(), sizeof(exc));
        mr0 = l4_utcb_mr()->mr[0];
        mr1 = l4_utcb_mr()->mr[1];
        print_timediff(s);
    }
    return 0;
}

```

17.7 examples/sys/utcb-ipc/main.c

This example shows how to send IPC using the UTCB to store payload.

```

/*
 * (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/re/env.h>
#include <l4/re/c/util/cap_alloc.h>
#include <l4/util/thread.h>
#include <stdio.h>
#include <string.h>
static unsigned char stack2[8 « 10];
static l4_cap_idx_t thread1_cap, thread2_cap;
static void thread1(void)
{
    l4_msg_regs_t *mr = l4_utcb_mr();
    l4_msgtag_t tag;
    int i, j;
    printf("Thread1 up (%p)\n", l4_utcb());
    for (i = 0; i < 10; i++)
    {
        for (j = 0; j < L4_UTCB_GENERIC_DATA_SIZE; j++)
            mr->mr[j] = 'A' + (i + j) % ('~' - 'A' + 1);
        tag = l4_msgtag(0, L4_UTCB_GENERIC_DATA_SIZE, 0, 0);
        if (l4_msgtag_has_error(l4_ipc_send(thread2_cap, l4_utcb(), tag, L4_IPC_NEVER)))
            printf("IPC-send error\n");
    }
}
L4UTIL_THREAD_STATIC_FUNC(thread2)
{
    l4_msgtag_t tag;
    l4_msg_regs_t mr;
    unsigned i;
    printf("Thread2 up (%p)\n", l4_utcb());
    while (1)
    {
        if (l4_msgtag_has_error(tag = l4_ipc_receive(thread1_cap, l4_utcb(), L4_IPC_NEVER)))
            printf("IPC receive error\n");
        memcpy(&mr, l4_utcb_mr(), sizeof(mr));
        printf("Thread2 receive (%d): ", l4_msgtag_words(tag));
        for (i = 0; i < l4_msgtag_words(tag); i++)
            printf("%c", (char)mr.mr[i]);
        printf("\n");
    }
    __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))

```

```

    return 1;
tag = l4_factory_create_thread(l4re_env()->factory, thread2_cap);
if (l4_error(tag))
    return 1;
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 *)l4re_env()->first_free_utcb,
                       L4RE_THIS_TASK_CAP);
tag = l4_thread_control_commit(thread2_cap);
if (l4_error(tag))
    return 2;
tag = l4_thread_ex_regs(thread2_cap,
                       (l4_umword_t)thread2,
                       (l4_umword_t)(stack2 + sizeof(stack2)), 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, thread2_cap, &sp);
if (l4_error(tag))
    return 4;
thread1();
return 0;
}

```

17.8 examples/sys/ux-vhw/main.c

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 *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.

```

/*
 * (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. The interrupt number of the virtual key interrupt can be
 * found in the kernel info page.
 */
#include <l4/re/c/util/cap_alloc.h>
#include <l4/re/c/namespace.h>
#include <l4/sys/utcb.h>
#include <l4/sys/irq.h>
#include <l4/sys/factory.h>
#include <l4/sys/icu.h>
#include <stdio.h>
int main(void)
{
    int irqno = 1;
    l4_cap_idx_t irqcap, icucap;
    l4_msgtag_t tag;
    int err;
    icucap = l4re_env_get_cap("icu");
    /* Get a free capability slot for the ICU capability */
    if (l4_is_invalid_cap(icucap))
    {
        printf("Did not find an ICU\n");
        return 1;
    }
    /* Get another free capability slot for the corresponding IRQ object */
    if (l4_is_invalid_cap(irqcap = l4re_util_cap_alloc()))
        return 1;
    /* Create IRQ object */
    if (l4_error(tag = l4_factory_create_irq(l4re_global_env->factory, irqcap)))
    {
        printf("Could not create IRQ object: %lx\n", l4_error(tag));
        return 1;
    }
    /*
     * Bind the recently allocated IRQ object to the IRQ number irqno
     * as provided by the ICU.
     */
    if (l4_error(l4_icu_bind(icucap, irqno, irqcap)))
    {
        printf("Binding IRQ%d to the ICU failed\n", irqno);
        return 1;
    }
    /* Bind ourselves to the IRQ */
    tag = l4_rcv_ep_bind_thread(irqcap, l4re_env()->main_thread, 0xDEAD);
    if ((err = l4_error(tag)))
    {
        printf("Error binding to IRQ %d: %d\n", irqno, err);
        return 1;
    }
    printf("Attached to key IRQ %d\nPress keys now, Shift-Q to exit\n", irqno);
    /* IRQ receive loop */
    while (1)
    {
        unsigned long label = 0;
        /* Wait for the interrupt to happen */
        tag = l4_irq_receive(irqcap, L4_IPC_NEVER);
        if ((err = l4_ipc_error(tag, l4_utcb())))
            printf("Error on IRQ receive: %d\n", err);
        else
        {
            /* Process the interrupt -- may do a 'break' */
            printf("Got IRQ with label 0x%lX\n", label);
        }
    }
    /* We're done, detach from the interrupt. */
    tag = l4_irq_detach(irqcap);
    if ((err = l4_error(tag)))
        printf("Error detach from IRQ: %d\n", err);
    return 0;
}

```

17.10 examples/clntsrv/server.cc

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.

```

/*
 * (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 vall = 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.

```

-- 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.

```

/*
 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 * economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4re/c/mem_alloc.h>
#include <l4re/c/rm.h>
#include <l4re/c/util/cap_alloc.h>
#include <l4/sys/err.h>
#include <stdio.h>
#include <string.h>
static int allocate_mem(unsigned long size_in_bytes, unsigned long flags,
    void **virt_addr)
{
    int r;
    l4re_ds_t ds;
    /* Allocate a free capability index for our data space */
    ds = l4re_util_cap_alloc();
    if (l4_is_invalid_cap(ds))
        return -L4_ENOMEM;
    size_in_bytes = l4_trunc_page(size_in_bytes);
    /* Allocate memory via a dataspace */
    if ((r = l4re_ma_alloc(size_in_bytes, ds, flags)))
        return r;
    /* Make the dataspace visible in our address space */
    *virt_addr = 0;
    if ((r = l4re_rm_attach(virt_addr, size_in_bytes,
        L4RE_RM_F_SEARCH_ADDR | L4RE_RM_F_RWX, ds, 0,
        flags & L4RE_MA_SUPER_PAGES
        ? L4_SUPERPAGESHIFT : L4_PAGESHIFT)))
    {

```

```

        /* Free dataspace again */
        l4re_util_cap_free_um(ds);
        return r;
    }
    /* Done, virtual address is in virt_addr */
    return 0;
}
static int free_mem(void *virt_addr)
{
    int r;
    l4re_ds_t ds;
    /* Detach memory from our address space */
    if ((r = l4re_rm_detach_ds(virt_addr, &ds)))
        return r;
    /* Free memory at our memory allocator */
    l4re_util_cap_free_um(ds);
    /* All went ok */
    return 0;
}
int main(void)
{
    void *virt;
    /* Allocate memory: 16k Bytes (usually) */
    if (allocate_mem(4 * L4_PAGESIZE, 0, &virt))
        return 1;
    printf("Allocated memory.\n");
    /* Do something with the memory */
    memset(virt, 0x12, 4 * L4_PAGESIZE);
    printf("Touched memory.\n");
    /* Free memory */
    if (free_mem(virt))
        return 2;
    printf("Freed and done. Bye.\n");
    return 0;
}

```

17.14 examples/libs/l4re/c++/mem_alloc/ma+rm.cc

Coarse grained memory allocation, in C++.

```

/*
 * (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;
    L4Re::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,
                                           L4Re::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.

```

/*
 * (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 <cstdio>
#include <unistd.h>
#include "interface.h"
int main()
{
    /*
     * Try to get server interface cap.
     */
    L4::Cap<My_interface> svr = L4Re::Env::env()->get_cap<My_interface>("shm");
    if (!svr.is_valid())
    {
        printf("Could not get the server capability\n");
        return 1;
    }
    /*
     * Alloc data space cap slot
     */
    L4::Cap<L4Re::Dataspace> ds = L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>();
    if (!ds.is_valid())
    {
        printf("Could not get capability slot!\n");
        return 1;
    }
    /*
     * Alloc server notifier IRQ cap slot
     */
    L4::Cap<L4::Irq> irq = L4Re::Util::cap_alloc.alloc<L4::Irq>();
    if (!irq.is_valid())
    {
        printf("Could not get capability slot!\n");
        return 1;
    }
    /*
     * Request shared data-space cap.
     */
    if (svr->get_shared_buffer(ds, irq))

```

```

    {
        printf("Could not get shared memory dataspace!\n");
        return 1;
    }
    /*
     * Attach to arbitrary region
     */
    char *addr = 0;
    int err = L4Re::Env::env()->rm()->attach(&addr, ds->size(),
                                           L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
                                           L4::Ipc::make_cap_rw(ds));

    if (err < 0)
    {
        printf("Error attaching data space: %s\n", l4sys_errtostr(err));
        return 1;
    }
    printf("Content: %s\n", addr);
    // wait a bit for the demo effect
    printf("Sleeping a bit...\n");
    sleep(1);
    /*
     * Fill in new stuff
     */
    memset(addr, 0, ds->size());
    char const * const msg = "Hello from client, too!";
    printf("Setting new content in shared memory\n");
    snprintf(addr, strlen(msg)+1, msg);
    l4_cache_clean_data((unsigned long)addr,
                       (unsigned long)addr + strlen(msg) + 1);
    // notify the server
    irq->trigger();
    /*
     * Detach region containing addr, result should be Detached_ds (other results
     * only apply if we split regions etc.).
     */
    err = L4Re::Env::env()->rm()->detach(addr, 0);
    if (err)
        printf("Failed to detach region\n");
    /* Free objects and capabilities, just for completeness. */
    L4Re::Util::cap_alloc.free(ds, L4Re::This_task);
    L4Re::Util::cap_alloc.free(irq, L4Re::This_task);
    return 0;
}

```

17.16 examples/libs/l4re/c++/shared_ds/ds_srv.cc

Sharing memory between applications, server/creator side.

```

/*
 * (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.

```

-- 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. Note that there's also a shared memory library that supplies this functionality in more convenient way.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <stdio.h>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>
#include <l4/re/util/object_registry>
#include <l4/cxx/ipc_server>
#include "shared.h"
static char page_to_map[L4_PAGESIZE] __attribute__((aligned(L4_PAGESIZE)));
static L4Re::Util::Registry_server<> server;
class Smap_server : public L4::Server_object_t<Mapper>
{
public:
    int dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios);
};
int
Smap_server::dispatch(l4_umword_t, L4::Ipc::Iostream &ios)
{
    l4_msgtag_t t;
    ios » t;
}

```

```
// We're only talking the Map_example protocol
if (t.label() != Mapper::Protocol)
    return -L4_EBADPROTO;
L4::Opcode opcode;
ios » opcode;
switch (opcode)
{
    case Mapper::Do_map:
        l4_addr_t snd_base;
        ios » snd_base;
        // put something into the page to read it out at the other side
        sprintf(page_to_map, sizeof(page_to_map), "Hello from the server!");
        printf("Sending to client\n");
        // send page
        ios « L4::Ipc::Snd_fpage::mem((l4_addr_t)page_to_map, L4_PAGESHIFT,
                                     L4_FPAGE_RO, snd_base);
        return L4_EOK;
    default:
        return -L4_ENOSYS;
}
}
int
main()
{
    static Smap_server smap;
    // Register server
    if (!server.registry()->register_obj(&smap, "smap").is_valid())
    {
        printf("Could not register my service, read-only namespace?\n");
        return 1;
    }
    printf("Welcome to the memory map example server!\n");
    // Wait for client requests
    server.loop();
    return 0;
}
```

17.19 examples/libs/l4re/streammap/client.cc

Client/Server example showing how to map a page to another task – Client implementation. 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.

```

-- 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 'smmap' channel in server
-- mode. The log prefix will be 'server', colored yellow.
L4.default_loader:start({ caps = { smmap = smmap_channel:svr() },
                        log = { "server", "yellow" } },
                        "rom/ex_smap-server");
-- The client program.
-- It is given the 'smmap' channel to be able to talk to the server.
-- The log prefix will be 'client', colored green.
L4.default_loader:start({ caps = { smmap = smmap_channel },
                        log = { "client", "green" } },
                        "rom/ex_smap-client");

```

17.21 examples/libs/libirq/loop.c

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.

```
/*
 * (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.

```
/*
 * (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.

```

-- 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](#).

```

/*
 * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU Lesser General Public License 2.1.
 * Please see the COPYING-LGPL-2.1 file for details.
 */
#include <l4/l4re_vfs/backend>
#include <l4/cxx/string>
#include <l4/cxx/avl_tree>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <dirent.h>
#include <cstdio>
#include <cstdlib>
#include <cstring>
namespace {
using namespace L4Re::Vfs;
using cxx::Ref_ptr;
class File_data
{
public:
    File_data() : _buf(0), _size(0) {}
    unsigned long put(unsigned long offset,
                     unsigned long bufsize, void *srcbuf);
    unsigned long get(unsigned long offset,
                     unsigned long bufsize, void *dstbuf);
    unsigned long size(unsigned long offset);
    unsigned long size() const { return _size; }
    ~File_data() throw() { free(_buf); }
private:
    void *_buf;
    unsigned long _size;
};
unsigned long
File_data::put(unsigned long offset, unsigned long bufsize, void *srcbuf)
{
    if (offset + bufsize > _size)
        size(offset + bufsize);
    if (!_buf)
        return 0;
    memcpy((char *)_buf + offset, srcbuf, bufsize);
    return bufsize;
}
unsigned long
File_data::get(unsigned long offset, unsigned long bufsize, void *dstbuf)
{
    unsigned long s = bufsize;
    if (offset > _size)
        return 0;
    if (offset + bufsize > _size)
        s = _size - offset;
    memcpy(dstbuf, (char *)_buf + offset, s);
    return s;
}
unsigned long
File_data::size(unsigned long offset)
{
    if (offset != _size)
    {
        _size = offset;
        _buf = realloc(_buf, _size);
    }
    if (!_buf)
        return 0;
    return -ENOSPC;
}
class Node : public cxx::Avl_tree_node
{
public:
    Node(const char *path, mode_t mode)
        : _ref_cnt(0), _path(strdup(path))
    {
        memset(&_info, 0, sizeof(_info));
        _info.st_mode = mode;
    }
    const char *path() const { return _path; }
    struct stat64 *info() { return &_info; }
    void add_ref() throw() { ++_ref_cnt; }
    int remove_ref() throw() { return --_ref_cnt; }
    bool is_dir() const { return S_ISDIR(_info.st_mode); }
}

```

```

    virtual ~Node() { free(_path); }
private:
    int      _ref_cnt;
    char     *_path;
    struct stat64 _info;
};
struct Node_get_key
{
    typedef cxx::String Key_type;
    static Key_type key_of(Node const *n)
    { return n->path(); }
};
struct Path_avl_tree_compare
{
    bool operator () (const char *l, const char *r) const
    { return strcmp(l, r) < 0; }
    bool operator () (const cxx::String l, const cxx::String r) const
    {
        int v = strncmp(l.start(), r.start(), cxx::min(l.len(), r.len()));
        return v < 0 || (v == 0 && l.len() < r.len());
    }
};
class Pers_file : public Node
{
public:
    Pers_file(const char *name, mode_t mode)
        : Node(name, (mode & 0777) | __S_IFREG) {}
    File_data const &data() const { return _data; }
    File_data &data() { return _data; }
private:
    File_data _data;
};
class Pers_dir : public Node
{
private:
    typedef cxx::Avl_tree<Node, Node_get_key, Path_avl_tree_compare> Tree;
    Tree _tree;
public:
    Pers_dir(const char *name, mode_t mode)
        : Node(name, (mode & 0777) | __S_IFDIR) {}
    Ref_ptr<Node> find_path(cxx::String);
    bool add_node(Ref_ptr<Node> const &);
    typedef Tree::Const_iterator Const_iterator;
    Const_iterator begin() const { return _tree.begin(); }
    Const_iterator end() const { return _tree.end(); }
};
Ref_ptr<Node> Pers_dir::find_path(cxx::String path)
{
    return cxx::ref_ptr(_tree.find_node(path));
}
bool Pers_dir::add_node(Ref_ptr<Node> const &n)
{
    bool e = _tree.insert(n.ptr()).second;
    if (e)
        n->add_ref();
    return e;
}
class Tmpfs_dir : public Be_file
{
public:
    explicit Tmpfs_dir(Ref_ptr<Pers_dir> const &d) throw()
        : _dir(d), _getdents_state(false) {}
    int get_entry(const char *, int, mode_t, Ref_ptr<File> *) throw();
    ssize_t getdents(char *, size_t) throw();
    int fstat64(struct stat64 *buf) const throw();
    int utime(const struct utimbuf *) throw();
    int fchmod(mode_t) throw();
    int mkdir(const char *, mode_t) throw();
    int unlink(const char *) throw();
    int rename(const char *, const char *) throw();
    int faccessat(const char *, int, int) throw();
private:
    int walk_path(cxx::String const &s,
        Ref_ptr<Node> *ret, cxx::String *remaining = 0);
    Ref_ptr<Pers_dir> _dir;
    bool _getdents_state;
    Pers_dir::Const_iterator _getdents_iter;
};
class Tmpfs_file : public Be_file_pos
{
public:
    explicit Tmpfs_file(Ref_ptr<Pers_file> const &f) throw()
        : Be_file_pos(), _file(f) {}
    off64_t size() const throw();
    int fstat64(struct stat64 *buf) const throw();
    int ftruncate64(off64_t p) throw();
    int ioctl(unsigned long, va_list) throw();
};

```

```

    int utime(const struct utimbuf *) throw();
    int fchmod(mode_t) throw();
private:
    ssize_t preadv(const struct iovec *v, int iovcnt, off64_t p) throw();
    ssize_t pwritev(const struct iovec *v, int iovcnt, off64_t p) throw();
    Ref_ptr<Pers_file> _file;
};

ssize_t Tmpfs_file::preadv(const struct iovec *v, int iovcnt, off64_t p) throw()
{
    if (iovcnt < 0)
        return -EINVAL;
    ssize_t sum = 0;
    for (int i = 0; i < iovcnt; ++i)
    {
        sum += _file->data().get(p, v[i].iov_len, v[i].iov_base);
        p += v[i].iov_len;
    }
    return sum;
}

ssize_t Tmpfs_file::pwritev(const struct iovec *v, int iovcnt, off64_t p) throw()
{
    if (iovcnt < 0)
        return -EINVAL;
    ssize_t sum = 0;
    for (int i = 0; i < iovcnt; ++i)
    {
        sum += _file->data().put(p, v[i].iov_len, v[i].iov_base);
        p += v[i].iov_len;
    }
    return sum;
}

int Tmpfs_file::fstat64(struct stat64 *buf) const throw()
{
    _file->info()->st_size = _file->data().size();
    memcpy(buf, _file->info(), sizeof(*buf));
    return 0;
}

int Tmpfs_file::ftruncate64(off64_t p) throw()
{
    if (p < 0)
        return -EINVAL;
    if (_file->data().size(p) == 0)
        return 0;
    return -EIO; // most likely ENOSPC, but can't report that
}

off64_t Tmpfs_file::size() const throw()
{
    return _file->data().size();
}

int
Tmpfs_file::ioctl(unsigned long v, va_list args) throw()
{
    switch (v)
    {
        case FIONREAD: // return amount of data still available
            int *available = va_arg(args, int *);
            *available = _file->data().size() - pos();
            return 0;
    };
    return -EINVAL;
}

int
Tmpfs_file::utime(const struct utimbuf *times) throw()
{
    _file->info()->st_atime = times->actime;
    _file->info()->st_mtime = times->modtime;
    return 0;
}

int
Tmpfs_file::fchmod(mode_t m) throw()
{
    _file->info()->st_mode = m;
    return 0;
}

int
Tmpfs_dir::faccessat(const char *path, int mode, int) throw()
{
    Ref_ptr<Node> node;
    cxx::String name = path;
    int err = walk_path(name, &node, &name);
    if (err < 0)
        return err;
    if (mode == F_OK) // existence check
        return 0;
    struct stat64 *stats = node->info();
    if ((mode & R_OK) && !(stats->st_mode & S_IRUSR))
        return -EACCES;
    if ((mode & W_OK) && !(stats->st_mode & S_IWUSR))
        return -EACCES;
}

```

```

    if ((mode & X_OK) && !(stats->st_mode & S_IXUSR))
        return -EACCES;
    return 0;
}
int
Tmpfs_dir::get_entry(const char *name, int flags, mode_t mode,
                    Ref_ptr<File> *file) throw()
{
    Ref_ptr<Node> path;
    if (!*name)
    {
        *file = cxx::ref_ptr(this);
        return 0;
    }
    cxx::String n = name;
    int e = walk_path(n, &path, &n);
    if (e == -ENOTDIR)
        return e;
    if (!(flags & O_CREAT) && e < 0)
        return e;
    if ((flags & O_CREAT) && e == -ENOENT)
    {
        Ref_ptr<Node> node(new Pers_file(n.start(), mode));
        // when ENOENT is return, path is always a directory
        bool e = cxx::ref_ptr_static_cast<Pers_dir>(path)->add_node(node);
        if (!e)
            return -ENOMEM;
        path = node;
    }
    if (path->is_dir())
        *file = cxx::ref_ptr(new Tmpfs_dir(cxx::ref_ptr_static_cast<Pers_dir>(path)));
    else
        *file = cxx::ref_ptr(new Tmpfs_file(cxx::ref_ptr_static_cast<Pers_file>(path)));
    if (!*file)
        return -ENOMEM;
    return 0;
}
ssize_t
Tmpfs_dir::getdents(char *buf, size_t sz) throw()
{
    struct dirent64 *d = (struct dirent64 *)buf;
    ssize_t ret = 0;
    if (!_getdents_state)
    {
        _getdents_iter = _dir->begin();
        _getdents_state = true;
    }
    else if (_getdents_iter == _dir->end())
    {
        _getdents_state = false;
        return 0;
    }
    for (; _getdents_iter != _dir->end(); ++_getdents_iter)
    {
        unsigned l = strlen(_getdents_iter->path()) + 1;
        if (l > sizeof(d->d_name))
            l = sizeof(d->d_name);
        unsigned n = offsetof(struct dirent64, d_name) + l;
        n = (n + sizeof(long) - 1) & ~(sizeof(long) - 1);
        if (n > sz)
            break;
        d->d_ino = 1;
        d->d_off = 0;
        memcpy(d->d_name, _getdents_iter->path(), l);
        d->d_reclen = n;
        d->d_type = DT_REG;
        ret += n;
        sz -= n;
        d = (struct dirent64 *)((unsigned long)d + n);
    }
    return ret;
}
int
Tmpfs_dir::fstat64(struct stat64 *buf) const throw()
{
    memcpy(buf, _dir->info(), sizeof(*buf));
    return 0;
}
int
Tmpfs_dir::utime(const struct utimbuf *times) throw()
{
    _dir->info()->st_atime = times->actime;
    _dir->info()->st_mtime = times->modtime;
    return 0;
}
int
Tmpfs_dir::fchmod(mode_t m) throw()

```

```

{
    _dir->info()->st_mode = m;
    return 0;
}
int
Tmpfs_dir::walk_path(cxx::String const &s,
                    Ref_ptr<Node> *ret, cxx::String *remaining)
{
    Ref_ptr<Pers_dir> p = _dir;
    cxx::String s = _s;
    Ref_ptr<Node> n;
    while (1)
    {
        if (s.len() == 0)
        {
            *ret = p;
            return 0;
        }
        cxx::String::Index sep = s.find("/");
        if (sep - s.start() == 1 && *s.start() == '.')
        {
            s = s.substr(s.start() + 2);
            continue;
        }
        n = p->find_path(s.head(sep - s.start()));
        if (!n)
        {
            *ret = p;
            if (remaining)
                *remaining = s.head(sep - s.start());
            return -ENOENT;
        }
        if (sep == s.end())
        {
            *ret = n;
            return 0;
        }
        if (!n->is_dir())
            return -ENOTDIR;
        s = s.substr(sep + 1);
        p = cxx::ref_ptr_static_cast<Pers_dir>(n);
    }
    *ret = n;
    return 0;
}
int
Tmpfs_dir::mkdir(const char *name, mode_t mode) throw()
{
    Ref_ptr<Node> node = _dir;
    cxx::String p = cxx::String(name);
    cxx::String path, last = p;
    cxx::String::Index s = p.rfind("/");
    // trim /'s at the end
    while (p.len() && s == p.end() - 1)
    {
        p.len(p.len() - 1);
        s = p.rfind("/");
    }
    //printf("MKDIR '%s' p=%p %p\n", name, p.start(), s);
    if (s != p.end())
    {
        path = p.head(s);
        last = p.substr(s + 1, p.end() - s);
        int e = walk_path(path, &node);
        if (e < 0)
            return e;
    }
    if (!node->is_dir())
        return -ENOTDIR;
    // due to path walking we can end up with an empty name
    if (p.len() == 0 || p == cxx::String("."))
        return 0;
    Ref_ptr<Pers_dir> dnode = cxx::ref_ptr_static_cast<Pers_dir>(node);
    Ref_ptr<Pers_dir> dir(new Pers_dir(last.start(), mode));
    return dnode->add_node(dir) ? 0 : -EEXIST;
}
int
Tmpfs_dir::unlink(const char *name) throw()
{
    cxx::Ref_ptr<Node> n;
    int e = walk_path(name, &n);
    if (e < 0)
        return -ENOENT;
    printf("Unimplemented (if file exists): %s(%s)\n", __func__, name);
    return -ENOMEM;
}
int

```

```

Tmpfs_dir::rename(const char *old, const char *newn) throw()
{
    printf("Unimplemented: %s(%s, %s)\n", __func__, old, newn);
    return -ENOMEM;
}
class Tmpfs_fs : public Be_file_system
{
public:
    Tmpfs_fs() : Be_file_system("tmpfs") {}
    int mount(char const *source, unsigned long mountflags,
              void const *data, cxx::Ref_ptr<File> *dir) throw()
    {
        (void)mountflags;
        (void)source;
        (void)data;
        *dir = cxx::ref_ptr(new Tmpfs_dir(cxx::ref_ptr(new Pers_dir("root", 0777))));
        if (!*dir)
            return -ENOMEM;
        return 0;
    }
};
static Tmpfs_fs _tmpfs L4RE_VFS_FILE_SYSTEM_ATTRIBUTE;
}

```

17.26 examples/libs/shmc/prodcons.c

Simple shared memory example.

```

/*
 * (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>
// a small helper
#define CHK(func) if (func) { printf("failure: %d\n", __LINE__); return (void *)-1; }
static const char some_data[] = "Hi consumer!";
static void *thread_producer(void *d)
{
    (void)d;
    l4shmc_chunk_t p_one;
    l4shmc_signal_t s_one, s_done;
    l4shmc_area_t shmarea;
    // 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, "prod", &s_one));
    CHK(l4shmc_attach_signal_to(&shmarea, "done",
                               pthread_l4_cap(pthread_self()), 10000, &s_done));
    // connect chunk and signal
    CHK(l4shmc_connect_chunk_signal(&p_one, &s_one));
    printf("PRODUCER: 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)));
        printf("PRODUCER: Sent data\n");
        CHK(l4shmc_wait_signal(&s_done));
    }
    l4_sleep_forever();
    return NULL;
}
static void *thread_consume(void *d)
{
    (void)d;
    l4shmc_area_t shmarea;

```



```

l4shmc_chunk_t p_one;
l4shmc_signal_t s_one, s_done;
// 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, "done", &s_done));
// attach signal to this thread
CHK(l4shmc_attach_signal_to(&shmarea, "prod",
                           pthread_l4_cap(pthread_self()), 10000, &s_one));

// connect chunk and signal
CHK(l4shmc_connect_chunk_signal(&p_one, &s_one));
while (1)
{
    CHK(l4shmc_wait_chunk(&p_one));
    printf("CONSUMER: 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;
    // create new shared memory area, 8K in size
    if (l4shmc_create("testshm", 8192))
        return 1;
    // create two threads, one for producer, one for consumer
    pthread_create(&one, 0, thread_producer, 0);
    pthread_create(&two, 0, thread_consume, 0);
    // now sleep, the two threads are doing the work
    l4_sleep_forever();
    return 0;
}

```


Index

`__iface`
 L4::Kobject_2t< Derived, Base1, Base2, PROTO,
 S_DEMAND >, [1020](#)
 L4::Kobject_3t< Derived, Base1, Base2, Base3,
 PROTO, S_DEMAND >, [1025](#)

`__iface_list`
 L4::Kobject_2t< Derived, Base1, Base2, PROTO,
 S_DEMAND >, [1021](#)
 L4::Kobject_3t< Derived, Base1, Base2, Base3,
 PROTO, S_DEMAND >, [1025](#)

`__L4UTIL_THREAD_FUNC`
 thread.h, [2318](#)

`__check_protocols__`
 L4::Kobject_2t< Derived, Base1, Base2, PROTO,
 S_DEMAND >, [1021](#)
 L4::Kobject_3t< Derived, Base1, Base2, Base3,
 PROTO, S_DEMAND >, [1026](#)

`__strcpy_maxlen`
 debugger.h, [2096](#)

`_bus`
 L4vbus::Device, [1476](#)

`~Be_file_system`
 L4Re::Vfs::Be_file_system, [1382](#)

`~H_list_item_t`
 cxx::H_list_item_t< ELEM_TYPE >, [710](#)

a
 L4Re::Video::Pixel_info, [1427](#), [1428](#)

`Access_width`
 L4Re::Mmio_space, [1277](#)

`acquire`
 L4Re::Inhibitor, [1264](#)

`add`
 L4::lpc_svr::Timeout_queue, [976](#)
 L4::Thread::Modify_senders, [1118](#)
 L4vcpu::State, [1517](#)
 L4virtio::Svr::Driver_mem_list_t< DATA >, [1615](#)

`add_block`
 L4virtio::Driver::Block_device, [1540](#)

`add_ku_mem`
 L4::Task, [1098](#)

`add_timeout`
 L4::lpc_svr::Server_iface, [968](#)
 L4::lpc_svr::Timeout_queue_hooks< HOOKS,
 BR_MAN >, [984](#)

`align_to`
 L4::lpc::Msg, [558](#)

`all`
 L4::Kip::Mem_desc, [1008](#), [1009](#)

`alloc`
 cxx::Base_slab< Obj_size, Slab_size, Max_free,
 Alloc >, [616](#)
 cxx::Base_slab_static< Obj_size, Slab_size,
 Max_free, Alloc >, [624](#)
 cxx::List_alloc, [718](#)
 cxx::Slab< Type, Slab_size, Max_free, Alloc >,
 [745](#)
 cxx::Slab_static< Type, Slab_size, Max_free, Alloc
 >, [750](#)
 L4Re::Cap_alloc, [1218](#)
 L4Re::Mem_alloc, [1273](#)
 L4Re::Util::Counting_cap_alloc< COUNTERTYPE
 >, [1325](#), [1326](#)

`alloc_buffer_demand`
 L4::lpc_svr::Br_manager_no_buffers, [952](#)
 L4::lpc_svr::Server_iface, [968](#)
 L4Re::Util::Br_manager, [1314](#)

`alloc_descriptor`
 L4virtio::Driver::Virtqueue, [1562](#)

`alloc_dynamic_entry`
 L4Re::Util::Names::Name_space, [1355](#)

`alloc_fd`
 L4Re::Vfs::Fs, [1395](#)

`alloc_max`
 cxx::List_alloc, [718](#)

`allocate`
 L4Re::Dataspace, [1225](#)
 L4Re::Util::Dataspace_svr, [1333](#)

`amd64 Virtual Registers (UTCB)`, [523](#)

`amd64/4/sys/cache.h`, [2039](#), [2040](#)

`amd64/4/sys/consts.h`, [2055](#)

`amd64/4/sys/l4int.h`, [2148](#), [2149](#)

`amd64/4/sys/linkage.h`, [1764](#)

`amd64/4/sys/segment.h`, [1740](#), [1743](#)

`amd64/4/sys/utcb.h`, [2207](#), [2208](#)

`amd64/4/util/apic.h`, [1681](#), [1682](#)

`amd64/4/util/bitops_arch.h`, [1770](#), [1771](#)

`amd64/4/util/cpu.h`, [1778](#), [1781](#)

`amd64/4/util/idt.h`, [1692](#), [1693](#)

`amd64/4/util/irq.h`, [1868](#), [1870](#)

`amd64/4/util/l4_macros.h`, [1786](#), [1787](#)

`amd64/4/util/mbi_argv.h`, [1790](#)

`amd64/4/util/perform.h`, [1695](#), [1696](#)

`amd64/4/util/port_io.h`, [1750](#)

`amd64/4/util/rdtsc.h`, [1707](#), [1716](#)

`amd64/4/util/spin.h`, [1729](#), [1730](#)

`amd64/4/util/util.h`, [1731](#), [1733](#)

`amd64/4/f/4/sys/segment.h`, [1737](#), [1739](#)

`amd64/4/f/4/util/port_io.h`, [1749](#), [1750](#)

- Arch
 - L4::Kip::Mem_desc, 1008
- Area
 - L4Re::Rm, 1297
- ARM Virtual Registers (UTCB), 105
- arm/l4/sys/atomic.h, 2236, 2237
- arm/l4/sys/cache.h, 2037, 2038
- arm/l4/sys/consts.h, 2053, 2054
- arm/l4/sys/l4int.h, 2148
- arm/l4/sys/linkage.h, 1763
- arm/l4/sys/mem_op.h, 1766, 1767
- arm/l4/sys/thread.h, 2329
- arm/l4/sys/utcb.h, 2204, 2206
- arm/l4/sys/vm.h, 1768, 1769
- arm/l4/util/bitops_arch.h, 1770
- arm/l4/util/cpu.h, 1777, 1778
- arm/l4/util/irq.h, 1867, 1868
- arm/l4/util/l4_macros.h, 1786
- arm/l4/util/mbi_argv.h, 1789
- arm/l4f/l4/sys/ipc.h, 2128, 2129
- arm/l4f/l4/sys/syscall_defs.h, 1792, 1793
- assert.h
 - l4_assert, 2228
- assign_dma_domain
 - L4vbus::Vbus, 1508, 1509
- associate
 - L4Re::Dma_space, 1239
- Atomic Instructions, 106
 - l4util_add16, 108
 - l4util_add16_res, 108
 - l4util_add32, 108
 - l4util_add32_res, 109
 - l4util_add8, 109
 - l4util_add8_res, 109
 - l4util_and16, 110
 - l4util_and16_res, 110
 - l4util_and32, 110
 - l4util_and32_res, 111
 - l4util_and8, 111
 - l4util_and8_res, 111
 - l4util_atomic_add, 112
 - l4util_atomic_inc, 112
 - l4util_cmpxchg, 112
 - l4util_cmpxchg16, 113
 - l4util_cmpxchg32, 114
 - l4util_cmpxchg64, 114
 - l4util_cmpxchg8, 115
 - l4util_dec16, 115
 - l4util_dec16_res, 116
 - l4util_dec32, 116
 - l4util_dec32_res, 116
 - l4util_dec8, 117
 - l4util_dec8_res, 117
 - l4util_inc16, 117
 - l4util_inc16_res, 117
 - l4util_inc32, 118
 - l4util_inc32_res, 118
 - l4util_inc8, 118
 - l4util_inc8_res, 119
 - l4util_or16, 119
 - l4util_or16_res, 119
 - l4util_or32, 120
 - l4util_or32_res, 120
 - l4util_or8, 120
 - l4util_or8_res, 121
 - l4util_sub16, 121
 - l4util_sub16_res, 121
 - l4util_sub32, 122
 - l4util_sub32_res, 122
 - l4util_sub8, 123
 - l4util_sub8_res, 123
 - l4util_xchg, 123
 - l4util_xchg16, 124
 - l4util_xchg32, 124
 - l4util_xchg8, 124
- attach
 - L4Re::Rm, 1298, 1299
 - L4Re::Util::Event_buffer_t< PAYLOAD >, 1343
- Attach_flags
 - L4Re::Rm::F, 1307
- Attach_mask
 - L4Re::Rm::F, 1308
- Attr
 - L4::Thread::Attr, 1115
- Attribute
 - L4Re::Dma_space, 1238
- Attributes
 - L4Re::Dma_space, 1237
- Auto_ptr
 - cxx::Auto_ptr< T >, 596, 597
- Auxiliary data, 125
- avail
 - cxx::List_alloc, 719
- avail_align
 - L4virtio::Virtqueue, 1647
- avail_size
 - L4virtio::Virtqueue, 1647
- Avl_map
 - cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >, 601
- ax
 - l4_vcpu_regs_t, 1204
- b
 - L4Re::Video::Pixel_info, 1428
- backtrace.h
 - l4util_backtrace, 2239
- Bad_address
 - L4virtio::Svr::Bad_descriptor, 1573
- Bad_descriptor
 - L4virtio::Svr::Bad_descriptor, 1573
- Bad_flags
 - L4virtio::Svr::Bad_descriptor, 1573
- Bad_next
 - L4virtio::Svr::Bad_descriptor, 1573
- Bad_rights
 - L4virtio::Svr::Bad_descriptor, 1573

- Bad_size
 - L4virtio::Svr::Bad_descriptor, [1573](#)
- Base API, [126](#)
- Base_avl_set
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [662](#)
- Basic Macros, [128](#)
- Be_file_system
 - L4Re::Vfs::Be_file_system, [1381](#)
- begin
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [663](#), [664](#)
 - cxx::Bits::Bst< Node, Get_key, Compare >, [681](#), [682](#)
- Bidirectional
 - L4Re::Dma_space, [1238](#)
- bind
 - L4::lcu, [848](#)
 - L4::lommu, [864](#)
 - L4::Thread::Attr, [1116](#)
- bind_notification_irq
 - L4virtio::Driver::Device, [1549](#)
- bind_thread
 - L4::Rcv_endpoint, [1056](#)
- bit
 - cxx::Bitmap_base, [649](#)
- Bit Manipulation, [129](#)
 - l4util_bsf, [130](#)
 - l4util_bsr, [130](#)
 - l4util_btc, [131](#)
 - l4util_btr, [131](#)
 - l4util_bts, [132](#)
 - l4util_clear_bit, [133](#)
 - l4util_complement_bit, [133](#)
 - l4util_find_first_set_bit, [134](#)
 - l4util_find_first_zero_bit, [134](#)
 - l4util_next_power2, [134](#)
 - l4util_set_bit, [135](#)
 - l4util_test_bit, [135](#)
- bit_index
 - cxx::Bitmap_base, [650](#)
- Bits
 - cxx::Bitfield< T, LSB, MSB >, [629](#)
- bits_per_pixel
 - L4Re::Video::Pixel_info, [1429](#)
- Bits_type
 - cxx::Bitfield< T, LSB, MSB >, [628](#)
- Block_dev_base
 - L4virtio::Svr::Block_dev_base< Ds_data >, [1578](#)
- Bootloader
 - L4::Kip::Mem_desc, [1008](#)
- bp
 - l4_vcpu_regs_t, [1205](#)
- Br_bytes
 - L4::lpc::Msg, [557](#)
- buf
 - L4Re::Util::Event_buffer_t< PAYLOAD >, [1344](#)
- buf_cp_in
 - L4::lpc, [550](#)
- buf_cp_out
 - L4::lpc, [551](#)
- buf_in
 - L4::lpc, [551](#)
- buffer
 - L4Re::Util::Event_t< PAYLOAD >, [1348](#)
- Buffer Registers (BRs), [136](#)
 - L4_BDR_IO_SHIFT, [137](#)
 - L4_BDR_MEM_SHIFT, [137](#)
 - L4_BDR_OBJ_SHIFT, [137](#)
 - l4_buffer_desc_consts_t, [137](#)
- Bufferable
 - L4Re::Dataspace::F, [1232](#)
- bus_cap
 - L4vbus::Device, [1470](#)
- bx
 - l4_vcpu_regs_t, [1205](#)
- bytes_per_pixel
 - L4Re::Video::Pixel_info, [1429](#)
- c
 - L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >, [1021](#)
 - L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >, [1026](#)
- C++ Exceptions, [137](#)
- C++ IPC Interface Definition., [138](#)
- C_bits
 - cxx::Bitmap_base, [648](#)
- Cache Consistency, [138](#)
 - l4_cache_clean_data, [139](#)
 - l4_cache_coherent, [140](#)
 - l4_cache_dma_coherent, [140](#)
 - l4_cache_flush_data, [141](#)
 - l4_cache_inv_data, [141](#)
- Cache_buffered
 - L4Re::Rm::F, [1308](#)
- Cache_normal
 - L4Re::Rm::F, [1308](#)
- Cache_uncached
 - L4Re::Rm::F, [1308](#)
- Cacheable
 - L4Re::Dataspace::F, [1232](#)
- Caching_mask
 - L4Re::Dataspace::F, [1232](#)
 - L4Re::Rm::F, [1308](#)
- Caching_shift
 - L4Re::Dataspace::F, [1232](#)
 - L4Re::Rm, [1298](#)
- call
 - L4::Arm_smccc, [777](#)
 - L4::lpc::loststream, [887](#)
- Cap
 - L4::Cap< T >, [787](#), [788](#)
 - L4::lpc::Cap< T >, [878](#)
- cap
 - L4::Cap_base, [794](#)
 - L4::Invalid_capability, [858](#)

- L4::Kobject, [1016](#)
- cap_alloc
 - L4Re::Util, [592](#)
- Cap_base
 - L4::Cap_base, [793](#)
- cap_cast
 - L4, [542](#)
- cap_dynamic_cast
 - L4, [543](#)
- cap_equal
 - L4::Task, [1099](#)
- Cap_mask
 - L4::lpc::Cap< T >, [878](#)
- cap_received
 - L4::lpc::Gen_fpage< T >, [881](#)
- cap_reinterpret_cast
 - L4, [544](#), [545](#)
- Cap_type
 - L4::Cap_base, [792](#)
- cap_valid
 - L4::Task, [1099](#)
- Capabilities, [142](#)
 - L4_BASE_ARM_SMCCC_CAP, [144](#)
 - L4_BASE_CAPS_LAST, [144](#)
 - L4_BASE_DEBUGGER_CAP, [144](#)
 - L4_BASE_FACTORY_CAP, [144](#)
 - L4_BASE_ICU_CAP, [144](#)
 - L4_BASE_IOMMU_CAP, [144](#)
 - L4_BASE_LOG_CAP, [144](#)
 - L4_BASE_PAGER_CAP, [144](#)
 - L4_BASE_SCHEDULER_CAP, [144](#)
 - L4_BASE_TASK_CAP, [144](#)
 - L4_BASE_THREAD_CAP, [144](#)
 - l4_cap_consts_t, [143](#)
 - L4_CAP_MASK, [143](#)
 - L4_CAP_SHIFT, [143](#)
 - L4_CAP_SIZE, [143](#)
 - l4_capability_equal, [144](#)
 - l4_default_caps_t, [143](#)
 - L4_INVALID_CAP, [143](#)
 - l4_is_invalid_cap, [144](#)
 - l4_is_valid_cap, [145](#)
- capability
 - L4_DISABLE_COPY, [2043](#)
- Capability allocator, [146](#)
 - l4re_util_cap_last, [146](#)
- Caps
 - L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >, [1130](#)
- cast
 - L4vcpu::Vcpu, [1522](#)
- cfg_read
 - L4vbus::Pci_dev, [1496](#)
 - L4vbus::Pci_host_bridge, [1501](#)
- cfg_write
 - L4vbus::Pci_dev, [1496](#)
 - L4vbus::Pci_host_bridge, [1502](#)
- chain
 - L4::lirq_mux, [1000](#)
- change_queue_config
 - L4virtio::Svr::Dev_config, [1592](#)
- check_size
 - L4::lpc::Msg, [559](#), [560](#)
- chkcapi
 - L4Re, [569](#)
- chkipc
 - L4Re, [570](#)
- chksys
 - L4Re, [571](#), [573](#), [574](#)
- Chunks, [147](#)
 - l4shmc_add_chunk, [148](#)
 - l4shmc_chunk_capacity, [148](#)
 - l4shmc_chunk_ptr, [149](#)
 - l4shmc_chunk_signal, [149](#)
 - l4shmc_get_chunk, [149](#)
 - l4shmc_get_chunk_to, [150](#)
 - l4shmc_iterate_chunk, [150](#)
- Class
 - L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >, [1021](#)
 - L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >, [1025](#)
- clear
 - cxx::Bits::Basic_list< POLICY >, [677](#)
 - L4::Types::Flags< BITS_ENUM, UNDERLYING >, [1156](#)
 - L4Re::Dataspace, [1226](#)
 - L4Re::Util::Dataspace_svr, [1333](#)
 - L4vcpu::State, [1518](#)
- clear_bit
 - cxx::Bitmap_base, [650](#)
- Coherent
 - L4Re::Dma_space, [1239](#)
- Color_component
 - L4Re::Video::Color_component, [1415](#)
- Com_error
 - L4::Com_error, [804](#)
- Comfortable Command Line Parsing, [151](#)
- compiler.h
 - L4_ALWAYS_INLINE, [2047](#)
 - L4_HIDDEN, [2047](#)
 - L4_NOTHROW, [2047](#)
- config_get
 - L4vbus::Gpio_pin, [1486](#)
- config_pad
 - L4vbus::Gpio_module, [1479](#)
 - L4vbus::Gpio_pin, [1487](#)
- config_pull
 - L4vbus::Gpio_pin, [1487](#)
- config_queue
 - L4virtio::Device, [1533](#)
 - L4virtio::Driver::Device, [1550](#)
- Console API, [151](#)
- consumed
 - L4virtio::Svr::Virtqueue, [1638](#)
- Consumer, [152](#), [155](#)

- l4shmc_chunk_consumed, [156](#)
- l4shmc_chunk_size, [156](#)
- l4shmc_enable_chunk, [157](#)
- l4shmc_enable_signal, [152](#)
- l4shmc_is_chunk_ready, [157](#)
- l4shmc_wait_any, [153](#)
- l4shmc_wait_any_to, [153](#)
- l4shmc_wait_any_try, [154](#)
- l4shmc_wait_chunk, [158](#)
- l4shmc_wait_chunk_to, [158](#)
- l4shmc_wait_chunk_try, [159](#)
- l4shmc_wait_signal, [154](#)
- l4shmc_wait_signal_to, [154](#)
- l4shmc_wait_signal_try, [155](#)
- contains
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [1624](#)
- Continuous
 - L4Re::Mem_alloc, [1272](#)
- contrib/libio-io/l4/io/io.h, [1793](#), [1796](#)
- control
 - L4::Thread, [1107](#)
- Conventional
 - L4::Kip::Mem_desc, [1008](#)
- copy
 - L4::Cap< T >, [789](#)
 - L4::Cap_base, [795](#)
 - L4Re::Util::Dataspace_svr, [1334](#)
- copy_in
 - L4Re::Dataspace, [1226](#)
- copy_receive_cap
 - L4Re::Util::Names::Name_space, [1355](#)
- copy_to
 - L4virtio::Svr::Data_buffer, [1587](#)
- count
 - L4::Kip::Mem_desc, [1010](#)
- Counting_cap_alloc
 - L4Re::Util::Counting_cap_alloc< COUNTERTYPE >, [1325](#)
- CPU related functions, [138](#)
- cpu.h
 - l4util_cpu_capabilities, [1779](#), [1783](#)
 - l4util_cpu_capabilities_nocheck, [1779](#), [1783](#)
 - l4util_cpu_has_cpuid, [1780](#), [1784](#)
- Cpu_disable
 - L4::Platform_control, [1047](#)
- cpu_disable
 - L4::Platform_control, [1047](#)
- Cpu_enable
 - L4::Platform_control, [1047](#)
- cpu_enable
 - L4::Platform_control, [1048](#)
- create
 - L4::Factory, [830](#), [831](#)
- create_buffer
 - L4Re::Video::Goos, [1421](#)
- create_factory
 - L4::Factory, [832](#)
- create_gate
 - L4::Factory, [833](#)
- create_irq
 - L4::Factory, [835](#)
- create_task
 - L4::Factory, [836](#)
- create_thread
 - L4::Factory, [837](#)
- create_view
 - L4Re::Video::Goos, [1421](#)
- create_vm
 - L4::Factory, [838](#)
- current_flags
 - L4virtio::Svr::Request_processor, [1630](#)
- cx
 - l4_vcpu_regs_t, [1205](#)
- cxx, [535](#)
- cxx::Auto_ptr< T >, [595](#)
 - Auto_ptr, [596](#), [597](#)
 - get, [597](#)
 - operator=, [597](#)
 - release, [598](#)
- cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >, [599](#)
 - Avl_map, [601](#)
 - insert, [602](#)
 - operator[], [603](#)
- cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >, [604](#)
- cxx::Avl_tree< Node, Get_key, Compare >, [607](#)
 - insert, [610](#)
 - remove, [611](#)
- cxx::Avl_tree_node, [612](#)
- cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, [614](#)
 - alloc, [616](#)
 - free, [617](#)
 - free_objects, [618](#)
 - max_free_slabs, [616](#)
 - object_size, [616](#)
 - objects_per_slab, [616](#)
 - slab_size, [616](#)
 - total_objects, [619](#)
- cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i, [620](#)
- cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, [621](#)
 - alloc, [624](#)
 - free, [624](#)
 - free_objects, [625](#)
 - max_free_slabs, [624](#)
 - object_size, [624](#)
 - objects_per_slab, [624](#)
 - slab_size, [624](#)
 - total_objects, [626](#)
- cxx::Bitfield< T, LSB, MSB >, [627](#)
 - Bits, [629](#)
 - Bits_type, [628](#)
 - get, [629](#)

- get_unshifted, 630
- Low_mask, 629
- Lsb, 629
- Mask, 629
- Masks, 629
- Msb, 629
- set, 631
- set_dirty, 631
- set_unshifted, 632
- set_unshifted_dirty, 633
- Shift_type, 628
- val, 634
- val_dirty, 636
- val_unshifted, 637
- cxx::Bitfield< T, LSB, MSB >::Value< TT >, 638
- cxx::Bitfield< T, LSB, MSB >::Value_base< TT >, 639
- cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >, 641
- cxx::Bitmap< BITS >, 642
 - scan_zero, 645
- cxx::Bitmap_base, 646
 - bit, 649
 - bit_index, 650
 - C_bits, 648
 - clear_bit, 650
 - operator[], 651
 - scan_zero, 652
 - set_bit, 653
 - W_bits, 648
 - word_index, 653
- cxx::Bitmap_base::Bit, 654
- cxx::Bitmap_base::Char< BITS >, 655
- cxx::Bitmap_base::Word< BITS >, 655
- cxx::Bits, 537
- cxx::Bits::Avl_map_get_key< KEY_TYPE >, 657
- cxx::Bits::Avl_set_get_key< KEY_TYPE >, 657
- cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, 658
 - Base_avl_set, 662
 - begin, 663, 664
 - E_exist, 662
 - E_inval, 662
 - E_noent, 662
 - E_nomem, 662
 - end, 665
 - erase, 666
 - find_node, 667
 - insert, 667
 - lower_bound_node, 668
 - rbegin, 669
 - remove, 670
 - rend, 671
- cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node, 672
 - operator*, 673
 - operator->, 673
 - valid, 674
- cxx::Bits::Basic_list< POLICY >, 674
 - clear, 677
 - iter, 677
- cxx::Bits::Bst< Node, Get_key, Compare >, 678
 - begin, 681, 682
 - dir, 683
 - end, 684
 - find, 685
 - find_node, 686
 - lower_bound_node, 687
 - rbegin, 688
 - remove_all, 689
 - remove_tree, 690
 - rend, 691
- cxx::Bits::Bst_node, 692
- cxx::Bits::Direction, 694
 - Direction_e, 695
 - L, 695
 - N, 695
 - operator!, 695
 - R, 695
- cxx::Bits::Smart_ptr_list< ITEM >, 696
 - pop_front, 697
- cxx::Bits::Smart_ptr_list_item< T, STORE_T >, 698
- cxx::H_list< T, POLICY >, 699
 - erase, 702
 - insert, 703
 - insert_after, 704
 - insert_before, 705
 - iter, 705
 - pop_front, 706
 - remove, 707
 - replace, 708
- cxx::H_list_item_t< ELEM_TYPE >, 708
 - ~H_list_item_t, 710
 - H_list_item_t, 710
- cxx::H_list_t< T >, 711
- cxx::List< D, Alloc >, 714
 - operator[], 715
- cxx::List< D, Alloc >::lter, 716
- cxx::List_alloc, 717
 - alloc, 718
 - alloc_max, 718
 - avail, 719
 - free, 719
 - List_alloc, 717
- cxx::List_item, 720
 - push_back, 721
 - push_front, 722
 - remove, 722
- cxx::List_item::lter, 723
- cxx::List_item::T_iter< T, Poly >, 725
- cxx::Lt_functor< Obj >, 728
- cxx::New_allocator< _Type >, 728
- cxx::Nothrow, 730
- cxx::Pair< First, Second >, 730
 - Pair, 731
- cxx::Pair_first_compare< Cmp, Typ >, 732
 - operator(), 733

- Pair_first_compare, [733](#)
- cxx::Ref_obj_list_item< T >, [734](#)
- cxx::Ref_ptr< T, CNT >, [735](#)
 - get, [738](#)
 - ptr, [738](#)
 - Ref_ptr, [737](#), [738](#)
 - release, [739](#)
- cxx::S_list< T, POLICY >, [739](#)
 - pop_front, [742](#)
- cxx::Slab< Type, Slab_size, Max_free, Alloc >, [743](#)
 - alloc, [745](#)
 - free, [745](#)
- cxx::Slab_static< Type, Slab_size, Max_free, Alloc >, [746](#)
 - alloc, [750](#)
- cxx::static_vector< T, IDX >, [751](#)
- cxx::String, [752](#)
 - find, [755](#)
 - from_dec, [756](#)
 - from_hex, [757](#)
 - starts_with, [758](#)
 - String, [755](#)
- cxx::Weak_ref< T >, [759](#)
- cxx::Weak_ref_base, [761](#)
- data
 - L4::lpc::Varg, [938](#)
- Data_buffer
 - L4virtio::Svr::Data_buffer, [1587](#)
- data_size
 - L4virtio::Svr::Block_request< Ds_data >, [1584](#)
- data_space
 - L4Re::Vfs::Be_file, [1379](#)
 - L4Re::Vfs::Regular_file, [1408](#)
- Dataspace interface, [163](#)
 - l4re_ds_allocate, [164](#)
 - l4re_ds_clear, [164](#)
 - l4re_ds_copy_in, [165](#)
 - L4RE_DS_F_BUFFERABLE, [164](#)
 - L4RE_DS_F_CACHEABLE, [164](#)
 - L4RE_DS_F_CACHING_MASK, [164](#)
 - L4RE_DS_F_CACHING_SHIFT, [164](#)
 - L4RE_DS_F_NORMAL, [164](#)
 - L4RE_DS_F_UNCACHEABLE, [164](#)
 - l4re_ds_flags, [165](#)
 - l4re_ds_info, [165](#)
 - l4re_ds_map_flags, [164](#)
 - l4re_ds_size, [166](#)
- debug
 - L4Re::Debug_obj, [1236](#)
- Debug interface, [166](#)
 - l4re_debug_obj_debug, [166](#)
- debugger.h
 - __strcpy_maxlen, [2096](#)
 - l4_debugger_get_object_name, [2097](#)
 - l4_debugger_query_log_name, [2097](#)
 - l4_debugger_query_log_typeid, [2098](#)
 - l4_debugger_switch_log, [2098](#)
- Debugging API, [167](#)
- dec_refcnt
 - L4::Kobject, [1018](#)
- Dedicated
 - L4::Kip::Mem_desc, [1008](#)
- delete_buffer
 - L4Re::Video::Goos, [1421](#)
- delete_obj
 - L4::Task, [1100](#)
- delete_view
 - L4Re::Video::Goos, [1422](#)
- Demand
 - L4::Kobject_typeid< T >, [1030](#)
 - L4::Type_info::Demand, [1126](#)
- demand
 - L4::Kobject_typeid< T >, [1030](#)
- desc
 - L4virtio::Driver::Virtqueue, [1563](#)
 - L4virtio::Svr::Virtqueue, [1639](#)
 - L4virtio::Svr::Virtqueue::Head_desc, [1642](#)
- desc_align
 - L4virtio::Virtqueue, [1648](#)
- desc_avail
 - L4virtio::Svr::Virtqueue, [1639](#)
- desc_size
 - L4virtio::Virtqueue, [1648](#)
- detach
 - L4::Irq, [989](#)
 - L4Re::Rm, [1300](#)–[1302](#)
 - L4Re::Util::Event_buffer_t< PAYLOAD >, [1344](#)
- Detach_again
 - L4Re::Rm, [1298](#)
- Detach_exact
 - L4Re::Rm, [1298](#)
- Detach_flags
 - L4Re::Rm, [1297](#)
- Detach_free
 - L4Re::Rm::F, [1308](#)
- Detach_keep
 - L4Re::Rm, [1298](#)
- Detach_overlap
 - L4Re::Rm, [1298](#)
- Detach_result
 - L4Re::Rm, [1298](#)
- Detached_ds
 - L4Re::Rm, [1298](#)
- Dev_config
 - L4virtio::Svr::Dev_config, [1591](#), [1592](#)
- dev_handle
 - L4vbus::Device, [1471](#)
- device
 - L4vbus::Device, [1471](#)
- device_by_hid
 - L4vbus::Device, [1472](#)
- device_config
 - L4virtio::Device, [1534](#)
- device_error
 - L4virtio::Svr::Device_t< DATA >, [1607](#)
- device_notification_irq

- L4virtio::Device, [1534](#)
- device_notify_irq
 - L4virtio::Svr::Device_t< DATA >, [1608](#)
- DF_1_DIRECT
 - elf.h, [2263](#)
- DF_1_DISPRELPND
 - elf.h, [2263](#)
- DF_1_GLOBAL
 - elf.h, [2263](#)
- DF_1_GROUP
 - elf.h, [2264](#)
- DF_1_INTERPOSE
 - elf.h, [2264](#)
- DF_1_NODEFLIB
 - elf.h, [2264](#)
- DF_1_NODUMP
 - elf.h, [2264](#)
- DF_1_NOOPEN
 - elf.h, [2265](#)
- DF_1_NOW
 - elf.h, [2265](#)
- DF_1_ORIGIN
 - elf.h, [2265](#)
- DF_P1_GROUPPERM
 - elf.h, [2265](#)
- DF_P1_LAZYLOAD
 - elf.h, [2266](#)
- di
 - l4_vcpu_regs_t, [1205](#)
- dir
 - cxx::Bits::Bst< Node, Get_key, Compare >, [683](#)
- Direction
 - L4Re::Dma_space, [1238](#)
- Direction_e
 - cxx::Bits::Direction, [695](#)
- disable
 - L4virtio::Virtqueue, [1650](#)
- disable_notify
 - L4virtio::Svr::Virtqueue, [1640](#)
- disassociate
 - L4Re::Dma_space, [1239](#)
- dispatch
 - L4::Basic_registry, [781](#)
 - L4::Epiface, [816](#)
 - L4::Epiface_t< Derived, IFACE, BASE, bool >, [821](#)
 - L4::Irqp_t< Derived, BASE, bool >, [1004](#)
 - L4::Server_object, [1082](#)
- dispatch_meta_request
 - L4::Server_object_t< IFACE, BASE >, [1086](#)
- DMA Space Interface, [159](#)
 - l4re_dma_space_associate, [160](#)
 - l4re_dma_space_disassociate, [161](#)
 - l4re_dma_space_map, [161](#)
 - l4re_dma_space_t, [160](#)
 - l4re_dma_space_unmap, [162](#)
- dma_space.h
 - L4RE_DMA_SPACE_BIDIRECTIONAL, [1884](#)
 - L4RE_DMA_SPACE_COHERENT, [1885](#)
- l4re_dma_space_direction, [1884](#)
- L4RE_DMA_SPACE_FROM_DEVICE, [1884](#)
- L4RE_DMA_SPACE_NONE, [1884](#)
- L4RE_DMA_SPACE_PHYS_SPACE, [1885](#)
- l4re_dma_space_space_attribs, [1884](#)
- L4RE_DMA_SPACE_TO_DEVICE, [1884](#)
- done
 - L4virtio::Svr::Data_buffer, [1588](#)
- down
 - L4::Semaphore, [1073](#)
- drive_cylinders
 - l4util_mb_drive_t, [1461](#)
- drive_mode
 - l4util_mb_drive_t, [1461](#)
- drive_number
 - l4util_mb_drive_t, [1461](#)
- driver_acknowledge
 - L4virtio::Driver::Device, [1551](#)
- driver_connect
 - L4virtio::Driver::Device, [1551](#)
- Driver_mem_region_t
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [1624](#)
- drv_base
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [1625](#)
- ds
 - L4virtio::Svr::Dev_config, [1593](#)
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [1625](#)
- Ds_map_mask
 - L4Re::Rm::F, [1308](#)
- ds_offset
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [1626](#)
- DT_NULL
 - elf.h, [2266](#)
- dump
 - L4Re::Video::Color_component, [1415](#)
 - L4Re::Video::Pixel_info, [1430](#)
 - L4virtio::Virtqueue, [1651](#)
- dx
 - l4_vcpu_regs_t, [1206](#)
- E_exist
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [662](#)
- E_inval
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [662](#)
- E_noent
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [662](#)
- E_nomem
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [662](#)
- e_phnum
 - Elf32_Ehdr, [766](#)
 - Elf64_Ehdr, [772](#)

- e_shnum
 - Elf32_Ehdr, [766](#)
 - Elf64_Ehdr, [772](#)
- Eager_map
 - L4Re::Rm::F, [1308](#)
- EDID parsing functionality, [167](#)
 - libedid_block_size, [168](#)
 - libedid_check_header, [168](#)
 - libedid_checksum, [169](#)
 - libedid_consts, [168](#)
 - libedid_dump, [169](#)
 - libedid_dump_standard_timings, [169](#)
 - libedid_num_ext_blocks, [170](#)
 - libedid_pnp_id, [170](#)
 - libedid_prefered_resolution, [170](#)
 - libedid_revision, [171](#)
 - libedid_version, [171](#)
- EI_CLASS
 - elf.h, [2266](#)
- EI_DATA
 - elf.h, [2267](#)
- EI_OSABI
 - elf.h, [2267](#)
- EI_PAD
 - elf.h, [2268](#)
- EI_VERSION
 - elf.h, [2268](#)
- ELF binary format, [172](#)
- elf.h
 - DF_1_DIRECT, [2263](#)
 - DF_1_DISPRELPND, [2263](#)
 - DF_1_GLOBAL, [2263](#)
 - DF_1_GROUP, [2264](#)
 - DF_1_INTERPOSE, [2264](#)
 - DF_1_NODEFLIB, [2264](#)
 - DF_1_NODUMP, [2264](#)
 - DF_1_NOOPEN, [2265](#)
 - DF_1_NOW, [2265](#)
 - DF_1_ORIGIN, [2265](#)
 - DF_P1_GROUPPERM, [2265](#)
 - DF_P1_LAZYLOAD, [2266](#)
 - DT_NULL, [2266](#)
 - EI_CLASS, [2266](#)
 - EI_DATA, [2267](#)
 - EI_OSABI, [2267](#)
 - EI_PAD, [2268](#)
 - EI_VERSION, [2268](#)
 - ELFCLASSNONE, [2269](#)
 - ELFDATA2LSB, [2269](#)
 - ELFDATA2MSB, [2270](#)
 - ELFDATANONE, [2270](#)
 - ELFOSABI_AIX, [2271](#)
 - ELFOSABI_FREEBSD, [2271](#)
 - ELFOSABI_HPUX, [2271](#)
 - ELFOSABI_IRIX, [2272](#)
 - ELFOSABI_LINUX, [2272](#)
 - ELFOSABI_MODESTO, [2272](#)
 - ELFOSABI_NETBSD, [2272](#)
 - ELFOSABI_OPENBSD, [2273](#)
 - ELFOSABI_SOLARIS, [2273](#)
 - ELFOSABI_SYSV, [2273](#)
 - ELFOSABI_TRU64, [2274](#)
 - NT_VERSION, [2274](#)
 - SHF_GROUP, [2274](#)
 - SHF_MASKOS, [2274](#)
 - SHF_TLS, [2275](#)
 - SHT_NUM, [2275](#)
- Elf32_Dyn, [764](#)
- Elf32_Ehdr, [765](#)
 - e_phnum, [766](#)
 - e_shnum, [766](#)
- Elf32_Phdr, [767](#)
- Elf32_Shdr, [768](#)
- Elf32_Sym, [769](#)
- Elf64_Dyn, [770](#)
- Elf64_Ehdr, [771](#)
 - e_phnum, [772](#)
 - e_shnum, [772](#)
- Elf64_Phdr, [773](#)
- Elf64_Shdr, [774](#)
- Elf64_Sym, [775](#)
- elf_aux.h
 - L4RE_ELF_AUX_ELEM, [1930](#)
 - L4RE_ELF_AUX_ELEM_T, [1930](#)
 - L4RE_ELF_AUX_T_KIP_ADDR, [1931](#)
 - L4RE_ELF_AUX_T_NONE, [1931](#)
 - L4RE_ELF_AUX_T_STACK_ADDR, [1931](#)
 - L4RE_ELF_AUX_T_STACK_SIZE, [1931](#)
 - L4RE_ELF_AUX_T_VMA, [1931](#)
- ELFCLASSNONE
 - elf.h, [2269](#)
- ELFDATA2LSB
 - elf.h, [2269](#)
- ELFDATA2MSB
 - elf.h, [2270](#)
- ELFDATANONE
 - elf.h, [2270](#)
- ELFOSABI_AIX
 - elf.h, [2271](#)
- ELFOSABI_FREEBSD
 - elf.h, [2271](#)
- ELFOSABI_HPUX
 - elf.h, [2271](#)
- ELFOSABI_IRIX
 - elf.h, [2272](#)
- ELFOSABI_LINUX
 - elf.h, [2272](#)
- ELFOSABI_MODESTO
 - elf.h, [2272](#)
- ELFOSABI_NETBSD
 - elf.h, [2272](#)
- ELFOSABI_OPENBSD
 - elf.h, [2273](#)
- ELFOSABI_SOLARIS
 - elf.h, [2273](#)
- ELFOSABI_SYSV

- elf.h, [2273](#)
- ELFOSABI_TRU64
 - elf.h, [2274](#)
- empty
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [1626](#)
- enable_notify
 - L4virtio::Svr::Virtqueue, [1640](#)
- end
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [665](#)
 - cxx::Bits::Bst< Node, Get_key, Compare >, [684](#)
 - L4::Kip::Mem_desc, [1011](#)
- enqueue_descriptor
 - L4virtio::Driver::Virtqueue, [1564](#)
- entry_ip
 - L4vcpu::Vcpu, [1522](#)
- entry_sp
 - L4vcpu::Vcpu, [1523](#)
- env
 - L4Re::Env, [1244](#)
- env.h
 - l4re_env_t, [1936](#)
- erase
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [666](#)
 - cxx::H_list< T, POLICY >, [702](#)
- err_no
 - L4::Runtime_error, [1064](#)
- Error
 - L4virtio::Svr::Bad_descriptor, [1573](#)
- Error codes, [179](#)
 - L4_E2BIG, [180](#)
 - L4_EACCESS, [180](#)
 - L4_EADDRNOTAVAIL, [180](#)
 - L4_EAGAIN, [180](#)
 - L4_EBADPROTO, [180](#)
 - L4_EBUSY, [180](#)
 - L4_EEXIST, [180](#)
 - L4_EFAULT, [180](#)
 - L4_EINVAL, [180](#)
 - L4_EIO, [180](#)
 - L4_EIPC_HI, [180](#)
 - L4_EIPC_LO, [180](#)
 - L4_EMSGMISSARG, [180](#)
 - L4_EMSGTOOLONG, [180](#)
 - L4_EMSGTOOSHORT, [180](#)
 - L4_ENAMETOOLONG, [180](#)
 - L4_ENODEV, [180](#)
 - L4_ENOENT, [180](#)
 - L4_ENOMEM, [180](#)
 - L4_ENOREPLY, [180](#)
 - L4_ENOSPC, [180](#)
 - L4_ENOSYS, [180](#)
 - L4_ENXIO, [180](#)
 - L4_EOK, [180](#)
 - L4_EPERM, [180](#)
 - L4_ERANGE, [180](#)
 - L4_ERRNOMAX, [180](#)
 - l4_error_code_t, [179](#)
- Error Handling, [173](#)
 - l4_error, [174](#)
 - L4_IPC_ENOT_EXISTENT, [174](#)
 - l4_ipc_error, [176](#)
 - l4_ipc_error_code, [177](#)
 - L4_IPC_ERROR_MASK, [174](#)
 - l4_ipc_is_rcv_error, [178](#)
 - l4_ipc_is_snd_error, [178](#)
 - L4_IPC_REABORTED, [174](#)
 - L4_IPC_RECANCELED, [174](#)
 - L4_IPC_REMAPFAILED, [174](#)
 - L4_IPC_REMSGCUT, [174](#)
 - L4_IPC_RERCVPFTO, [174](#)
 - L4_IPC_RESNDPFTO, [174](#)
 - L4_IPC_RETIMEOUT, [174](#)
 - L4_IPC_SEABORTED, [174](#)
 - L4_IPC_SECANCELED, [174](#)
 - L4_IPC_SEMAPFAILED, [174](#)
 - L4_IPC_SEMSGCUT, [174](#)
 - L4_IPC_SERCVPFTO, [174](#)
 - L4_IPC_SESNDPFTO, [174](#)
 - L4_IPC_SETIMEOUT, [174](#)
 - L4_IPC_SND_ERR_MASK, [174](#)
 - l4_ipc_tcr_error_t, [174](#)
- Event API, [180](#)
- Event interface, [181](#)
 - l4re_event_get_axis_info, [182](#)
 - l4re_event_get_buffer, [182](#)
 - l4re_event_get_num_streams, [183](#)
 - l4re_event_get_stream_info, [183](#)
 - l4re_event_get_stream_info_for_id, [184](#)
- Event_buffer_t
 - L4Re::Event_buffer_t< PAYLOAD >, [1259](#)
- ex_regs
 - L4::Thread, [1107](#), [1108](#)
- exc_handler
 - L4::Thread::Attr, [1116](#)
- Exception registers, [184](#)
 - l4_utcb_exc, [185](#)
 - l4_utcb_exc_is_ex_regs_exception, [185](#)
 - l4_utcb_exc_is_pf, [186](#)
 - l4_utcb_exc_pc, [186](#)
 - l4_utcb_exc_pc_set, [187](#)
- execute
 - L4Re::Ned::Cmd_control, [1286](#)
- expired
 - L4::lpc_svr::Timeout, [974](#)
- ext_alloc
 - L4vcpu::Vcpu, [1523](#)
- Extended vCPU support, [187](#)
 - l4vcpu_ext_alloc, [188](#)
- extra_str
 - L4::Runtime_error, [1064](#)
- F_above
 - L4Re::Video::View, [1434](#)
- F_auto_refresh

- L4Re::Video::Goos, [1420](#)
- F_dyn_allocated
 - L4Re::Video::View, [1434](#)
- F_dynamic_buffers
 - L4Re::Video::Goos, [1420](#)
- F_dynamic_views
 - L4Re::Video::Goos, [1420](#)
- F_flags_mask
 - L4Re::Video::View, [1434](#)
- F_fully_dynamic
 - L4Re::Video::View, [1434](#)
- F_l4re_video_goos_auto_refresh
 - Video API, [499](#)
- F_l4re_video_goos_dynamic_buffers
 - Video API, [499](#)
- F_l4re_video_goos_dynamic_views
 - Video API, [499](#)
- F_l4re_video_goos_pointer
 - Video API, [499](#)
- F_l4re_video_view_above
 - Video API, [499](#)
- F_l4re_video_view_dyn_allocated
 - Video API, [499](#)
- F_l4re_video_view_flags_mask
 - Video API, [499](#)
- F_l4re_video_view_none
 - Video API, [499](#)
- F_l4re_video_view_set_background
 - Video API, [499](#)
- F_l4re_video_view_set_buffer
 - Video API, [499](#)
- F_l4re_video_view_set_buffer_offset
 - Video API, [499](#)
- F_l4re_video_view_set_bytes_per_line
 - Video API, [499](#)
- F_l4re_video_view_set_flags
 - Video API, [499](#)
- F_l4re_video_view_set_pixel
 - Video API, [499](#)
- F_l4re_video_view_set_position
 - Video API, [499](#)
- F_none
 - L4Re::Video::View, [1434](#)
- F_pointer
 - L4Re::Video::Goos, [1420](#)
- F_set_background
 - L4Re::Video::View, [1434](#)
- F_set_buffer
 - L4Re::Video::View, [1434](#)
- F_set_buffer_offset
 - L4Re::Video::View, [1434](#)
- F_set_bytes_per_line
 - L4Re::Video::View, [1434](#)
- F_set_flags
 - L4Re::Video::View, [1434](#)
- F_set_pixel
 - L4Re::Video::View, [1434](#)
- F_set_position
 - L4Re::Video::View, [1434](#)
- faccessat
 - L4Re::Vfs::Directory, [1385](#)
- Factory, [188](#)
 - l4_factory_create_factory, [190](#)
 - l4_factory_create_factory_u, [191](#)
 - l4_factory_create_gate, [191](#)
 - l4_factory_create_gate_u, [192](#)
 - l4_factory_create_irq, [194](#)
 - l4_factory_create_irq_u, [195](#)
 - l4_factory_create_task, [196](#)
 - l4_factory_create_task_u, [197](#)
 - l4_factory_create_thread, [198](#)
 - l4_factory_create_thread_u, [199](#)
 - l4_factory_create_vm, [200](#)
 - l4_factory_create_vm_u, [201](#)
- factory
 - L4Re::Env, [1244](#), [1245](#)
- fchmod
 - L4Re::Vfs::Generic_file, [1399](#)
- fdatsync
 - L4Re::Vfs::Regular_file, [1408](#)
- features
 - l4_icu_info_t, [1186](#)
- Fiasco extensions, [202](#)
 - fiasco_gdt_get_entry_offset, [203](#)
 - fiasco_gdt_set, [203](#)
 - fiasco_ldt_set, [204](#)
 - fiasco_tbuf_log, [204](#)
 - fiasco_tbuf_log_3val, [205](#)
 - fiasco_tbuf_log_binary, [205](#)
- Fiasco real time scheduling extensions, [206](#)
- Fiasco-UX Virtual devices, [207](#)
 - L4_TYPE_VHW_FRAMEBUFFER, [208](#)
 - L4_TYPE_VHW_INPUT, [208](#)
 - L4_TYPE_VHW_NET, [208](#)
 - L4_TYPE_VHW_NONE, [208](#)
 - l4_vhw_entry_type, [207](#)
- fiasco_amd64_segment_info
 - segment.h, [1742](#)
- fiasco_amd64_set_fs
 - segment.h, [1738](#), [1742](#)
- fiasco_amd64_set_segment_base
 - segment.h, [1738](#), [1743](#)
- fiasco_gdt_get_entry_offset
 - Fiasco extensions, [203](#)
- fiasco_gdt_set
 - Fiasco extensions, [203](#)
- fiasco_ldt_set
 - Fiasco extensions, [204](#)
- fiasco_tbuf_log
 - Fiasco extensions, [204](#)
- fiasco_tbuf_log_3val
 - Fiasco extensions, [205](#)
- fiasco_tbuf_log_binary
 - Fiasco extensions, [205](#)
- finalize_request
 - L4virtio::Svr::Block_dev_base< Ds_data >, [1578](#)

- find
 - cxx::Bits::Bst< Node, Get_key, Compare >, 685
 - cxx::String, 755
 - L4::Basic_registry, 781
 - L4Re::Rm, 1302
 - L4virtio::Svr::Driver_mem_list_t< DATA >, 1616
- find_next_used
 - L4virtio::Driver::Virtqueue, 1564
- find_node
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, 667
 - cxx::Bits::Bst< Node, Get_key, Compare >, 686
- first
 - L4::Kip::Mem_desc, 1012
- first_free_cap
 - L4Re::Env, 1245
- first_free_utcb
 - L4Re::Env, 1246
- Flags
 - L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >, 1130
 - L4::Types::Flags< BITS_ENUM, UNDERLYING >, 1155
 - L4Re::Dataspace::F, 1232
 - L4Re::Video::Goos, 1420
 - L4Re::Video::View, 1434
- flags
 - l4_exc_regs_t, 1183
 - l4_msgtag_t, 1192
 - L4Re::Dataspace, 1228
 - l4re_env_cap_entry_t, 1445
 - L4virtio::Svr::Driver_mem_region_t< DATA >, 1626
- Flex pages, 208
 - L4_CAP_FPAGE_D, 212
 - L4_CAP_FPAGE_R, 212
 - L4_cap_fpage_rights, 212
 - L4_CAP_FPAGE_RO, 212
 - L4_CAP_FPAGE_RS, 213
 - L4_CAP_FPAGE_RSD, 213
 - L4_CAP_FPAGE_RW, 212
 - L4_CAP_FPAGE_RWD, 213
 - L4_CAP_FPAGE_RWS, 213
 - L4_CAP_FPAGE_RWSD, 213
 - L4_CAP_FPAGE_S, 212
 - L4_CAP_FPAGE_W, 212
 - l4_fpage, 215
 - L4_FPAGE_ADDR_BITS, 214
 - L4_FPAGE_ADDR_SHIFT, 214
 - l4_fpage_all, 215
 - L4_FPAGE_BUFFERABLE, 213
 - L4_FPAGE_C_IPCGATE_SVR, 215
 - L4_FPAGE_C_NO_REF_CNT, 215
 - L4_FPAGE_C_OBJ_RIGHT1, 215
 - L4_FPAGE_C_OBJ_RIGHT2, 215
 - L4_FPAGE_C_OBJ_RIGHT3, 215
 - L4_FPAGE_C_OBJ_RIGHTS, 215
 - L4_FPAGE_C_REF_CNT, 215
 - L4_FPAGE_CACHE_OPT, 213
 - l4_fpage_cacheability_opt_t, 213
 - L4_FPAGE_CACHEABLE, 213
 - l4_fpage_consts, 213
 - l4_fpage_contains, 216
 - l4_fpage_invalid, 216
 - l4_fpage_ioport, 217
 - l4_fpage_max_order, 217
 - l4_fpage_memaddr, 218
 - l4_fpage_obj, 219
 - l4_fpage_page, 219
 - L4_fpage_rights, 214
 - l4_fpage_rights, 220
 - L4_FPAGE_RIGHTS_BITS, 214
 - L4_FPAGE_RIGHTS_MASK, 214
 - L4_FPAGE_RIGHTS_SHIFT, 214
 - L4_FPAGE_RO, 214
 - L4_FPAGE_RW, 214
 - L4_FPAGE_RWX, 214
 - L4_FPAGE_RX, 214
 - l4_fpage_set_rights, 221
 - l4_fpage_size, 221
 - L4_FPAGE_SIZE_BITS, 214
 - L4_FPAGE_SIZE_SHIFT, 214
 - l4_fpage_type, 222
 - L4_FPAGE_TYPE_BITS, 214
 - L4_FPAGE_TYPE_SHIFT, 214
 - L4_FPAGE_UNCACHEABLE, 213
 - L4_FPAGE_W, 214
 - L4_FPAGE_X, 214
 - l4_iofpage, 222
 - L4_IOPORT_MAX, 212
 - l4_is_fpage_writable, 223
 - l4_obj_fpage, 224
 - L4_obj_fpage_ctl, 214
 - L4_WHOLE_ADDRESS_SPACE, 210
 - L4_WHOLE_IOADDRESS_SPACE, 212
- foreach_available_event
 - L4Re::Util::Event_buffer_consumer_t< PAYLOAD >, 1340
- fpage
 - L4::Cap_base, 796
- free
 - cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, 617
 - cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, 624
 - cxx::List_alloc, 719
 - cxx::Slab< Type, Slab_size, Max_free, Alloc >, 745
 - L4Re::Cap_alloc, 1219
 - L4Re::Mem_alloc, 1273
 - L4Re::Util::Counting_cap_alloc< COUNTERTYPE >, 1327
- free_area
 - L4Re::Rm, 1303
- free_capability
 - L4Re::Util::Names::Name_space, 1355

- free_descriptor
 - L4virtio::Driver::Virtqueue, [1565](#)
- free_dynamic_entry
 - L4Re::Util::Names::Name_space, [1356](#)
- free_epiface
 - L4Re::Util::Names::Name_space, [1356](#)
- free_fd
 - L4Re::Vfs::Fs, [1395](#)
- free_objects
 - cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, [618](#)
 - cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, [625](#)
- from_ci
 - L4::lpc::Cap< T >, [879](#)
- from_dec
 - cxx::String, [756](#)
- From_device
 - L4Re::Dma_space, [1238](#)
- from_hex
 - cxx::String, [757](#)
- from_raw
 - L4::Types::Flags< BITS_ENUM, UNDERLYING >, [1156](#)
- fstat64
 - L4Re::Vfs::Be_file, [1379](#)
 - L4Re::Vfs::Generic_file, [1400](#)
- fsync
 - L4Re::Vfs::Regular_file, [1408](#)
- ftruncate64
 - L4Re::Vfs::Regular_file, [1408](#)
- full
 - L4virtio::Svr::Driver_mem_list_t< DATA >, [1616](#)
- Functions to manipulate the local IDT, [225](#)
- g
 - L4Re::Video::Pixel_info, [1430](#)
- get
 - cxx::Auto_ptr< T >, [597](#)
 - cxx::Bitfield< T, LSB, MSB >, [629](#)
 - cxx::Ref_ptr< T, CNT >, [738](#)
 - L4::lpc::Iostream, [888](#), [889](#)
 - L4::lpc::Istream, [896](#), [897](#)
 - L4Re::Env, [1246](#)
 - L4Re::Video::Color_component, [1415](#)
 - L4vbus::Gpio_module, [1480](#)
 - L4vbus::Gpio_pin, [1488](#)
 - L4virtio::Ptr< T >, [1570](#)
- get_areas
 - L4Re::Rm, [1304](#)
- get_attr
 - L4::Vcon, [1172](#)
- get_avail_idx
 - L4virtio::Virtqueue, [1651](#)
- get_buffer
 - L4Re::Event, [1257](#)
- get_buffer_demand
 - L4::Epiface, [817](#)
 - L4::Server_object_t< IFACE, BASE >, [1086](#)
- get_cap
 - L4Re::Env, [1247](#)
- get_cap_alloc
 - L4Re::Cap_alloc, [1220](#)
- get_cmd
 - L4virtio::Svr::Dev_config, [1593](#)
- get_epiface
 - L4Re::Util::Names::Name_space, [1356](#)
- get_fb
 - L4Re::Util::Video::Goos_svr, [1373](#)
- get_file
 - L4Re::Vfs::Fs, [1395](#)
- get_infos
 - L4::lpc_gate, [949](#)
- get_lock
 - L4Re::Vfs::Regular_file, [1409](#)
- get_object_name
 - L4::Debugger, [807](#)
- get_random
 - L4Re::Random, [1293](#)
- get_rcv_cap
 - L4::lpc_svr::Server_iface, [969](#)
 - L4Re::Util::Br_manager, [1314](#)
- get_regions
 - L4Re::Rm, [1304](#)
- get_resource
 - L4vbus::Device, [1473](#)
- get_static_buffer
 - L4Re::Video::Goos, [1422](#)
- get_status_flags
 - L4Re::Vfs::Generic_file, [1400](#)
- get_tail_avail_idx
 - L4virtio::Virtqueue, [1652](#)
- get_unshifted
 - cxx::Bitfield< T, LSB, MSB >, [630](#)
- get_value
 - L4::lpc::Varg, [938](#)
- get_writeback
 - L4virtio::Svr::Block_dev_base< Ds_data >, [1579](#)
- global_id
 - L4::Debugger, [808](#)
- gran_offset
 - l4_sched_cpu_set_t, [1195](#)
- granularity
 - l4_sched_cpu_set_t, [1193](#)
- guest_features
 - L4virtio::Svr::Dev_config, [1594](#)
- H_list_item_t
 - cxx::H_list_item_t< ELEM_TYPE >, [710](#)
- handle_expired_timeouts
 - L4::lpc_svr::Timeout_queue, [977](#)
- handle_mem_cmd_write
 - L4virtio::Svr::Device_t< DATA >, [1608](#)
- has_alpha
 - L4Re::Video::Pixel_info, [1431](#)
- has_more
 - L4virtio::Svr::Request_processor, [1630](#)
- hdr

- L4virtio::Svr::Dev_config, [1595](#)
- i
- L4vcpu::Vcpu, [1524](#)
- IA32 Port I/O API, [225](#)
- id
- L4::Kobject_typeid< T >, [1030](#)
- id_received
- L4::lpc::Gen_fpage< T >, [881](#)
- idle_time
- L4::Scheduler, [1067](#)
- In_area
- L4Re::Rm::F, [1308](#)
- indirect_bfm_t
- L4virtio::Virtqueue::Desc::Flags, [1666](#)
- Info
- L4::Kip::Mem_desc, [1008](#)
- info
- L4::lcu, [849](#)
 - L4::Scheduler, [1068](#)
 - L4Re::Dataspace, [1228](#)
 - L4Re::Video::Goos, [1423](#)
 - L4Re::Video::View, [1435](#)
- Info_acpi_rsdp
- L4::Kip::Mem_desc, [1007](#)
- Info_sub_type
- L4::Kip::Mem_desc, [1007](#)
- init
- L4Re::Util::Event_t< PAYLOAD >, [1349](#)
 - L4virtio::Svr::Driver_mem_list_t< DATA >, [1617](#)
- init_infos
- L4Re::Util::Video::Goos_svr, [1373](#)
- init_mem_info
- L4virtio::Svr::Device_t< DATA >, [1609](#)
- init_poll
- L4Re::Util::Event_t< PAYLOAD >, [1350](#)
- init_queue
- L4virtio::Driver::Virtqueue, [1566](#)
- Initial Environment, [249](#)
- l4re_env, [250](#)
 - l4re_env_get_cap, [251](#)
 - l4re_env_get_cap_e, [251](#)
 - l4re_env_get_cap_l, [252](#)
 - l4re_kip, [253](#)
- initial_caps
- L4Re::Env, [1248](#)
- initialize_rings
- L4virtio::Driver::Virtqueue, [1567](#)
- insert
- cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >, [602](#)
 - cxx::Avl_tree< Node, Get_key, Compare >, [610](#)
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [667](#)
 - cxx::H_list< T, POLICY >, [703](#)
- insert_after
- cxx::H_list< T, POLICY >, [704](#)
- insert_before
- cxx::H_list< T, POLICY >, [705](#)
- Integer Types, [254](#)
- interface
- L4::Meta, [1037](#)
- Interface for asynchronous ISR handlers with a given IRQ capability., [255](#)
- l4irq_request_cap, [255](#)
- Interface for asynchronous ISR handlers., [256](#)
- l4irq_release, [257](#)
 - l4irq_request, [257](#)
- Interface using direct functionality., [258](#), [260](#)
- l4irq_attach, [261](#)
 - l4irq_attach_cap, [258](#)
 - l4irq_attach_cap_ft, [259](#)
 - l4irq_attach_ft, [261](#)
 - l4irq_attach_thread, [262](#)
 - l4irq_attach_thread_cap, [259](#)
 - l4irq_attach_thread_cap_ft, [260](#)
 - l4irq_attach_thread_ft, [262](#)
 - l4irq_detach, [263](#)
 - l4irq_unmask, [263](#)
 - l4irq_unmask_and_wait_any, [264](#)
 - l4irq_wait, [264](#)
 - l4irq_wait_any, [264](#)
- Internal constants, [265](#)
- Internal functions, [266](#)
- Internal Helpers, [265](#)
- internal_loop
- L4::Server< LOOP_HOOKS >, [1077](#)
- Interrupt controller, [266](#)
- l4_icu_bind, [268](#)
 - l4_icu_bind_u, [269](#)
 - L4_ICU_FLAG_MSI, [268](#)
 - L4_icu_flags, [268](#)
 - l4_icu_info, [270](#)
 - l4_icu_info_t, [268](#)
 - l4_icu_info_u, [271](#)
 - l4_icu_mask, [272](#)
 - l4_icu_mask_u, [272](#)
 - l4_icu_msi_info, [273](#)
 - l4_icu_msi_info_u, [274](#)
 - l4_icu_set_mode, [275](#)
 - l4_icu_set_mode_u, [275](#)
 - l4_icu_unbind, [276](#)
 - l4_icu_unbind_u, [277](#)
 - l4_icu_unmask, [278](#)
 - l4_icu_unmask_u, [278](#)
- Invalid
- L4::Cap_base, [793](#)
 - L4virtio::Ptr< T >, [1570](#)
- Invalid_capability
- L4::Invalid_capability, [858](#)
- Invalid_type
- L4virtio::Ptr< T >, [1570](#)
- IO interface, [225](#)
- L4IO_DEVICE_ANY, [228](#)
 - L4IO_DEVICE_INVALID, [228](#)
 - L4IO_DEVICE_OTHER, [228](#)
 - L4IO_DEVICE_PCI, [228](#)

- l4io_device_types_t, 227
- L4IO_DEVICE_USB, 228
- l4io_has_resource, 229
- l4io_iomem_flags_t, 228
- l4io_lookup_device, 229
- l4io_lookup_resource, 229
- L4IO_MEM_CACHED, 228
- L4IO_MEM_EAGER_MAP, 228
- L4IO_MEM_NONCACHED, 228
- L4IO_MEM_USE_MTRR, 228
- L4IO_MEM_USE_RESERVED_AREA, 228
- l4io_release_iomem, 230
- l4io_release_ioport, 230
- l4io_request_iomem, 231
- l4io_request_iomem_region, 231
- l4io_request_ioport, 232
- l4io_request_resource_iomem, 233
- L4IO_RESOURCE_ANY, 228
- L4IO_RESOURCE_INVALID, 228
- L4IO_RESOURCE_IRQ, 228
- L4IO_RESOURCE_MEM, 228
- L4IO_RESOURCE_PORT, 228
- l4io_resource_t, 226
- l4io_resource_types_t, 228
- io.h
 - l4io_get_root_device, 1795
 - l4io_iterate_devices, 1795
 - l4io_request_all_ioports, 1796
 - l4io_request_icu, 1796
- io_page_fault
 - L4::Io::Pager, 861
- ioctl
 - L4Re::Vfs::Special_file, 1413
- lostream
 - L4::Ipc::lostream, 887
- IPC-Gate API, 233
 - l4_ipc_gate_bind_thread, 234
 - l4_ipc_gate_get_infos, 235
 - l4_rcv_ep_bind_thread, 235
- ipc.h
 - l4_ipc_to_errno, 2124
- ipc_client
 - L4_RPC_DEF, 2062
- ipc_iface
 - L4_INLINE_RPC, 2065
 - L4_INLINE_RPC_NF, 2066
 - L4_INLINE_RPC_NF_OP, 2066
 - L4_INLINE_RPC_OP, 2067
 - L4_RPC, 2067
 - L4_RPC_NF, 2068
 - L4_RPC_NF_OP, 2068
 - L4_RPC_OP, 2069
- ipc_stream
 - operator<<, 1837–1839
 - operator>>, 1840–1844
- irq
 - L4Re::Util::Event_t< PAYLOAD >, 1351
- IRQ handling library, 236
- irq.h
 - l4util_irq_acknowledge, 1869, 1872
- irq_disable_save
 - L4vcpu::Vcpu, 1524
- irq_enable
 - L4vbus::Pci_dev, 1497
 - L4vbus::Pci_host_bridge, 1503
 - L4vcpu::Vcpu, 1525
- irq_restore
 - L4vcpu::Vcpu, 1526
- IRQs, 236
 - l4_irq_detach, 239
 - l4_irq_detach_u, 239
 - L4_IRQ_F_BOTH, 238
 - L4_IRQ_F_BOTH_EDGE, 238
 - L4_IRQ_F_CLEAR_WAKEUP, 238
 - L4_IRQ_F_EDGE, 238
 - L4_IRQ_F_LEVEL, 238
 - L4_IRQ_F_LEVEL_HIGH, 238
 - L4_IRQ_F_LEVEL_LOW, 238
 - L4_IRQ_F_MASK, 238
 - L4_IRQ_F_NEG, 238
 - L4_IRQ_F_NEG_EDGE, 238
 - L4_IRQ_F_NONE, 238
 - L4_IRQ_F_POS, 238
 - L4_IRQ_F_POS_EDGE, 238
 - L4_IRQ_F_SET_WAKEUP, 238
 - L4_irq_mode, 238
 - l4_irq_mux_chain, 240
 - l4_irq_mux_chain_u, 241
 - l4_irq_receive, 242
 - l4_irq_receive_u, 243
 - l4_irq_trigger, 244
 - l4_irq_trigger_u, 245
 - l4_irq_unmask, 246
 - l4_irq_unmask_u, 246
 - l4_irq_wait, 247
 - l4_irq_wait_u, 248
- is_compatible
 - L4vbus::Device, 1474
- is_compound
 - L4::Ipc::Gen_fpage< T >, 881
- is_irq_entry
 - L4vcpu::Vcpu, 1526
- is_nil
 - L4::Ipc::Varg, 939
- is_of
 - L4::Ipc::Varg, 939
- is_of_int
 - L4::Ipc::Varg, 940
- is_online
 - L4::Scheduler, 1068
- is_page_fault_entry
 - L4vcpu::Vcpu, 1527
- is_static
 - L4Re::Util::Dataspace_svr, 1334
- is_valid
 - L4::Cap_base, 797

- L4virtio::Ptr< T >, [1571](#)
- is_virtual
 - L4::Kip::Mem_desc, [1013](#)
- is_writable
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [1626](#)
- Istream
 - L4::lpc::Istream, [896](#)
- Item_bytes
 - L4::lpc::Msg, [557](#)
- Item_words
 - L4::lpc::Msg, [557](#)
- iter
 - cxx::Bits::Basic_list< POLICY >, [677](#)
 - cxx::H_list< T, POLICY >, [705](#)
- Kept_ds
 - L4Re::Rm, [1298](#)
- Kernel Debugger, [279](#)
 - l4_debugger_global_id, [280](#)
 - l4_debugger_kobj_to_id, [280](#)
 - l4_debugger_set_object_name, [281](#)
- Kernel Interface Page, [281](#)
- Kernel Interface Page API, [283](#)
- Kernel Objects, [283](#)
- kip.h
 - l4_kernel_info_version_offset, [2292](#)
 - l4_kip_clock, [2293](#)
 - l4_kip_clock_lw, [2294](#)
 - l4_kip_version, [2295](#)
 - l4_kip_version_string, [2295](#)
 - l4util_kip_for_each_feature, [2289](#)
 - l4util_kip_kernel_abi_version, [2290](#)
 - l4util_kip_kernel_has_feature, [2290](#)
 - l4util_kip_kernel_is_ux, [2290](#)
- kobj_to_id
 - L4::Debugger, [808](#)
- kobject_typeid
 - L4 kernel object type information, [314](#)
- Kumem allocator utility, [285](#)
- Kumem utilities, [285](#)
- kumem_alloc
 - L4Re::Util, [588](#)
- kumem_alloc.h
 - l4re_util_kumem_alloc, [1902](#)
- L
 - cxx::Bits::Direction, [695](#)
- L4, [538](#)
 - cap_cast, [542](#)
 - cap_dynamic_cast, [543](#)
 - cap_reinterpret_cast, [544](#), [545](#)
 - PROTO_ANY, [542](#)
 - PROTO_EMPTY, [542](#)
 - round_order, [545](#)
 - throw_ipc_exception, [546](#), [547](#)
 - trunc_order, [547](#)
- L4 IPC Opcodes, [286](#)
 - L4_ICU_OP_BIND, [287](#)
 - L4_ICU_OP_INFO, [287](#)
 - L4_ICU_OP_MASK, [287](#)
 - L4_ICU_OP_MSI_INFO, [287](#)
 - L4_ICU_OP_SET_MODE, [287](#)
 - L4_ICU_OP_UNBIND, [287](#)
 - L4_ICU_OP_UNMASK, [287](#)
 - L4_icu_opcode, [286](#)
 - L4_IPC_GATE_BIND_OP, [287](#)
 - L4_IPC_GATE_GET_INFO_OP, [287](#)
 - L4_ipc_gate_ops, [287](#)
 - L4_PLATFORM_CTL_CPU_DISABLE_OP, [288](#)
 - L4_PLATFORM_CTL_CPU_ENABLE_OP, [288](#)
 - L4_platform_ctl_ops, [288](#)
 - L4_PLATFORM_CTL_SYS_SHUTDOWN_OP, [288](#)
 - L4_PLATFORM_CTL_SYS_SUSPEND_OP, [288](#)
 - L4_TASK_ADD_KU_MEM_OP, [288](#)
 - L4_TASK_CAP_INFO_OP, [288](#)
 - L4_TASK_LDT_SET_X86_OP, [288](#)
 - L4_TASK_MAP_OP, [288](#)
 - L4_TASK_MAP_VGICC_ARM_OP, [288](#)
 - L4_task_ops, [288](#)
 - L4_TASK_UNMAP_OP, [288](#)
 - L4_THREAD_AMD64_GET_SEGMENT_INFO_OP, [289](#)
 - L4_THREAD_AMD64_SET_SEGMENT_BASE_OP, [289](#)
 - L4_THREAD_ARM_TPIDRURO_OP, [289](#)
 - L4_THREAD_CONTROL_OP, [289](#)
 - L4_THREAD_EX_REGS_OP, [289](#)
 - L4_THREAD_MODIFY_SENDER_OP, [289](#)
 - L4_THREAD_OPCODE_MASK, [289](#)
 - L4_thread_ops, [288](#)
 - L4_THREAD_REGISTER_DELETE_IRQ_OP, [289](#)
 - L4_THREAD_STATS_OP, [289](#)
 - L4_THREAD_SWITCH_OP, [289](#)
 - L4_THREAD_VCPU_CONTROL_OP, [289](#)
 - L4_THREAD_VCPU_RESUME_OP, [289](#)
 - L4_THREAD_X86_GDT_OP, [289](#)
 - L4_VCON_GET_ATTR_OP, [289](#)
 - L4_vcon_ops, [289](#)
 - L4_VCON_READ_OP, [289](#)
 - L4_VCON_SET_ATTR_OP, [289](#)
 - L4_VCON_WRITE_OP, [289](#)
- L4 kernel object type information, [313](#)
 - kobject_typeid, [314](#)
- L4 Vbus functions, [301](#)
 - l4vbus_assign_dma_domain, [303](#)
 - L4vbus_dma_domain_assign_flags, [303](#)
 - L4VBUS_DMAD_BIND, [303](#)
 - L4VBUS_DMAD_KERNEL_DMA_SPACE, [303](#)
 - L4VBUS_DMAD_L4RE_DMA_SPACE, [303](#)
 - L4VBUS_DMAD_UNBIND, [303](#)
 - l4vbus_get_device, [304](#)
 - l4vbus_get_device_by_hid, [305](#)
 - l4vbus_get_hid, [306](#)
 - l4vbus_get_next_device, [306](#)
 - l4vbus_get_resource, [308](#)

- l4vbus_is_compatible, 309
- l4vbus_release_ioport, 309
- l4vbus_release_resource, 310
- l4vbus_request_ioport, 311
- l4vbus_request_resource, 311
- l4vbus_vicu_get_cap, 312
- L4 VIRTIO Block Device, 289
 - L4virtio_block_operations, 290
 - L4VIRTIO_BLOCK_S_IOERR, 291
 - L4VIRTIO_BLOCK_S_OK, 291
 - L4VIRTIO_BLOCK_S_UNSUPP, 291
 - L4virtio_block_status, 291
 - L4VIRTIO_BLOCK_T_DISCARD, 290
 - L4VIRTIO_BLOCK_T_FLUSH, 290
 - L4VIRTIO_BLOCK_T_GET_ID, 290
 - L4VIRTIO_BLOCK_T_IN, 290
 - L4VIRTIO_BLOCK_T_OUT, 290
 - L4VIRTIO_BLOCK_T_WRITE_ZEROES, 290
- L4 VIRTIO Input Device, 291
- L4 VIRTIO Interface, 292
- L4 VIRTIO Transport Layer, 293
 - L4_virtio_cmd, 295
 - L4_virtio_irq_status, 295
 - L4_virtio_opcodes, 296
 - L4VIRTIO_CMD_CFG_QUEUE, 295
 - L4VIRTIO_CMD_MASK, 295
 - L4VIRTIO_CMD_NONE, 295
 - L4VIRTIO_CMD_SET_STATUS, 295
 - l4virtio_config_queue, 298
 - l4virtio_config_queue_t, 295
 - l4virtio_config_queues, 298
 - l4virtio_device_config, 299
 - l4virtio_device_config_ds, 299
 - L4virtio_device_ids, 296
 - l4virtio_device_notification_irq, 299
 - L4virtio_device_status, 297
 - L4virtio_feature_bits, 297
 - L4VIRTIO_FEATURE_CMD_CONFIG, 297
 - L4VIRTIO_FEATURE_VERSION_1, 297
 - L4VIRTIO_ID_9P, 296
 - L4VIRTIO_ID_BALLOON, 296
 - L4VIRTIO_ID_BLOCK, 296
 - L4VIRTIO_ID_CAIF, 297
 - L4VIRTIO_ID_CONSOLE, 296
 - L4VIRTIO_ID_CRYPTOP, 297
 - L4VIRTIO_ID_GPU, 297
 - L4VIRTIO_ID_INPUT, 297
 - L4VIRTIO_ID_NET, 296
 - L4VIRTIO_ID_RNG, 296
 - L4VIRTIO_ID_RPMSG, 296
 - L4VIRTIO_ID_RPROC_SERIAL, 297
 - L4VIRTIO_ID_SCSI, 296
 - L4VIRTIO_ID_SOCKET, 297
 - L4VIRTIO_ID_VSOCKET, 297
 - L4VIRTIO_IRQ_STATUS_CONFIG, 296
 - L4VIRTIO_IRQ_STATUS_VRING, 296
 - L4VIRTIO_OP_CONFIG_QUEUE, 296
 - L4VIRTIO_OP_REGISTER_DS, 296
 - L4VIRTIO_OP_REGISTER_IFACE, 296
 - L4VIRTIO_OP_SET_STATUS, 296
 - l4virtio_register_ds, 300
 - l4virtio_register_iface, 300
 - l4virtio_set_status, 301
 - L4VIRTIO_STATUS_ACKNOWLEDGE, 297
 - L4VIRTIO_STATUS_DEVICE_NEEDS_RESET, 297
 - L4VIRTIO_STATUS_DRIVER, 297
 - L4VIRTIO_STATUS_DRIVER_OK, 297
 - L4VIRTIO_STATUS_FAILED, 297
 - L4VIRTIO_STATUS_FEATURES_OK, 297
- l4/cxx/alloc.h, 1797, 1798
- l4/cxx/atomic.h, 2237, 2238
- l4/cxx/avl_map, 1798, 1800
- l4/cxx/avl_set, 1801, 1803
- l4/cxx/avl_tree, 1806, 1808
- l4/cxx/basic_ostream, 1812, 1813
- l4/cxx/basic_vector.h, 1816, 1817
- l4/cxx/bits/bst.h, 1817, 1819
- l4/cxx/bits/bst_base.h, 1821, 1823
- l4/cxx/bits/bst_iter.h, 1824, 1826
- l4/cxx/exceptions, 1827, 1829
- l4/cxx/iostream, 1831, 1833
- l4/cxx/ipc_helper, 1833, 1834
- l4/cxx/ipc_server, 2074, 2076
- l4/cxx/ipc_stream, 1835, 1845
- l4/cxx/l4iostream, 1854, 1855
- l4/cxx/l4types.h, 1856, 1857
- l4/cxx/main_thread, 1857, 1858
- l4/cxx/pair, 1859, 1860
- l4/cxx/std_exc_io, 1860, 1861
- l4/cxx/string.h, 1862, 1864
- l4/cxx/thread, 2190, 2192
- l4/irq/irq.h, 1864, 1866
- l4/libedid/edid.h, 1877, 1878
- l4/re/c/dataspace.h, 1879, 1880
- l4/re/c/debug.h, 1881, 1882
- l4/re/c/dma_space.h, 1883, 1885
- l4/re/c/event.h, 1886, 1887
- l4/re/c/log.h, 1890, 1891
- l4/re/c/mem_alloc.h, 1891, 1893
- l4/re/c/namespace.h, 1894, 1895
- l4/re/c/rm.h, 1896, 1897
- l4/re/c/util/cap_alloc.h, 1900, 1901
- l4/re/c/util/kumem_alloc.h, 1901, 1903
- l4/re/c/util/video/goos_fb.h, 1903, 1904
- l4/re/c/video/colors.h, 1905, 1907
- l4/re/c/video/goos.h, 1907, 1909
- l4/re/c/video/view.h, 1910, 1913
- l4/re/cap_alloc, 1914, 1915
- l4/re/consts, 2058, 2060
- l4/re/consts.h, 2056, 2058
- l4/re/dataspace, 1920, 1922
- l4/re/dataspace-sys.h, 1923, 1924
- l4/re/debug, 1924, 1926
- l4/re/dma_space, 1926, 1928
- l4/re/elf_aux.h, 1929, 1931

l4/re/env, 1932, 1934
 l4/re/env.h, 1935, 1937
 l4/re/error_helper, 1938, 1940
 l4/re/event.h, 1888, 1889
 l4/re/impl/dataspace_impl.h, 1944, 1945
 l4/re/impl/mem_alloc_impl.h, 1946, 1947
 l4/re/impl/namespace_impl.h, 1948, 1949
 l4/re/impl/rm_impl.h, 1950, 1951
 l4/re/l4aux.h, 1952, 1954
 l4/re/log, 1954, 1956
 l4/re/log-sys.h, 1956, 1957
 l4/re/mem_alloc, 1957, 1958
 l4/re/mem_alloc-sys.h, 1959, 1960
 l4/re/namespace, 1960, 1962
 l4/re/namespace-sys.h, 1963, 1964
 l4/re/parent, 1964, 1966
 l4/re/parent-sys.h, 1966, 1967
 l4/re/protocols.h, 1967, 1968
 l4/re/rm, 1969, 1970
 l4/re/rm-sys.h, 1974, 1975
 l4/re/shared_cap, 1975, 1977
 l4/re/unique_cap, 1980, 1982
 l4/re/util/bitmap_cap_alloc, 1985, 1986
 l4/re/util/cap, 1987, 1989
 l4/re/util/cap_alloc, 1917, 1919
 l4/re/util/cap_alloc_impl.h, 1989, 1990
 l4/re/util/counting_cap_alloc, 1991, 1992
 l4/re/util/event, 1941, 1942
 l4/re/util/item_alloc, 1994, 1995
 l4/re/util/kumem_alloc, 1996, 1997
 l4/re/util/region_mapping, 1998, 1999
 l4/re/util/shared_cap, 1977, 1979
 l4/re/util/unique_cap, 1982, 1984
 l4/re/video/goos-sys.h, 2004, 2005
 l4/shmc/shmc.h, 2005, 2008
 l4/sigma0/sigma0.h, 2010, 2015
 l4/sys/__kernel_object_impl.h, 2017
 l4/sys/__ktrace-impl.h, 2018, 2019
 l4/sys/__typeinfo.h, 2020, 2023
 l4/sys/__vm-arm.h, 2033, 2034
 l4/sys/assert.h, 2226, 2228
 l4/sys/cache.h, 2035, 2036
 l4/sys/capability, 2042, 2044
 l4/sys/compiler.h, 2045, 2048
 l4/sys/consts.h, 2049, 2052
 l4/sys/cxx/ipc_client, 2060, 2062
 l4/sys/cxx/ipc_iface, 2063, 2069
 l4/sys/cxx/ipc_types, 2077, 2079
 l4/sys/cxx/smart_capability_1x, 2085, 2086
 l4/sys/cxx/types, 2089, 2090
 l4/sys/debugger, 2093, 2094
 l4/sys/debugger.h, 2095, 2099
 l4/sys/err.h, 2102, 2103
 l4/sys/exception, 2103, 2105
 l4/sys/factory, 2105, 2107
 l4/sys/factory.h, 2109, 2111
 l4/sys/icu, 2115, 2116
 l4/sys/icu.h, 2116, 2119
 l4/sys/ipc.h, 2122, 2125
 l4/sys/ipc_gate, 2131, 2132
 l4/sys/ipc_gate.h, 2133, 2135
 l4/sys/irq, 2136, 2137
 l4/sys/irq.h, 1873, 1875
 l4/sys/kernel_object.h, 2139, 2140
 l4/sys/kip, 2141, 2143
 l4/sys/kip.h, 2291, 2296
 l4/sys/ktrace.h, 2144, 2146
 l4/sys/l4int.h, 2146, 2147
 l4/sys/memdesc.h, 2150, 2152
 l4/sys/meta, 2154, 2156
 l4/sys/pager, 2156, 2158
 l4/sys/platform_control, 2158, 2160
 l4/sys/platform_control.h, 2160, 2162
 l4/sys/rcv_endpoint, 2164, 2165
 l4/sys/rcv_endpoint.h, 2166, 2167
 l4/sys/scheduler, 2168, 2169
 l4/sys/scheduler.h, 2170, 2172
 l4/sys/semaphore, 2174, 2176
 l4/sys/smart_capability, 2176, 2178
 l4/sys/task, 2180, 2182
 l4/sys/task.h, 2183, 2184
 l4/sys/thread, 2187, 2189
 l4/sys/thread.h, 2319, 2322
 l4/sys/typeinfo_svr, 2192, 2194
 l4/sys/types.h, 2195, 2197
 l4/sys/utcb.h, 2200, 2202
 l4/sys/vcon, 2212, 2214
 l4/sys/vcon.h, 2214, 2217
 l4/sys/vhw.h, 2220, 2221
 l4/sys/vm, 2222, 2224
 l4/util/assert.h, 2224, 2225
 l4/util/atomic.h, 2229, 2232
 l4/util/backtrace.h, 2238, 2240
 l4/util/base64.h, 2240, 2242
 l4/util/bitops.h, 2242, 2244
 l4/util/elf.h, 2247, 2276
 l4/util/getopt.h, 2285, 2286
 l4/util/keymap.h, 2287, 2288
 l4/util/kip.h, 2288, 2291
 l4/util/kprintf.h, 2297, 2298
 l4/util/l4_macros.h, 1787, 1788
 l4/util/list_alloc.h, 2299, 2301
 l4/util/lock.h, 2302, 2303
 l4/util/mb_info.h, 2303, 2307
 l4/util/parse_cmd.h, 2311, 2313
 l4/util/rand.h, 2314, 2315
 l4/util/splitlog2.h, 2315, 2316
 l4/util/thread.h, 2317, 2319
 l4/vbus/vbus.h, 2330, 2332
 l4/vbus/vbus_interfaces.h, 2333, 2336
 l4/vbus/vbus_types.h, 2336, 2339
 L4::Alloc_list, 776
 L4::Arm_smccc, 777
 call, 777
 L4::Base_exception, 778
 L4::Basic_registry, 780

- dispatch, 781
- find, 781
- L4::Bounds_error, 782
- L4::Cap< T >, 785
 - Cap, 787, 788
 - copy, 789
 - move, 789
- L4::Cap_base, 790
 - cap, 794
 - Cap_base, 793
 - Cap_type, 792
 - copy, 795
 - fpage, 796
 - Invalid, 793
 - is_valid, 797
 - move, 798
 - No_init, 793
 - No_init_type, 793
 - snd_base, 799
 - validate, 800
- L4::Com_error, 801
 - Com_error, 804
- L4::Debugger, 804
 - get_object_name, 807
 - global_id, 808
 - kobj_to_id, 808
 - query_log_name, 808
 - query_log_typeid, 809
 - set_object_name, 810
 - switch_log, 810
- L4::Element_already_exists, 811
- L4::Element_not_found, 813
- L4::Epiface, 815
 - dispatch, 816
 - get_buffer_demand, 817
 - obj_cap, 817
 - server_iface, 817
 - set_server, 818
- L4::Epiface_t< Derived, IFACE, BASE, bool >, 819
 - dispatch, 821
- L4::Epiface_t0< RPC_IFACE, BASE >, 822
 - obj_cap, 824
- L4::Exception, 824
 - Rpcs, 825
- L4::Exception_tracer, 825
- L4::Factory, 827
 - create, 830, 831
 - create_factory, 832
 - create_gate, 833
 - create_irq, 835
 - create_task, 836
 - create_thread, 837
 - create_vm, 838
- L4::Factory::Lstr, 839
 - Lstr, 840
- L4::Factory::Nil, 840
- L4::Factory::S, 841
 - operator l4_msgtag_t, 843
- put, 843–845
 - S, 842, 843
- L4::lcu, 846
 - bind, 848
 - info, 849
 - mask, 851
 - msi_info, 852
 - set_mode, 852
 - unbind, 853
- L4::lcu::Info, 854
- L4::Invalid_capability, 855
 - cap, 858
 - Invalid_capability, 858
- L4::lo_pager, 859
 - io_page_fault, 861
- L4::lomu, 861
 - bind, 864
 - unbind, 864
- L4::IOModifier, 864
- L4::lpc, 548
 - buf_cp_in, 550
 - buf_cp_out, 551
 - buf_in, 551
 - make_cap, 552
 - make_cap_full, 552
 - make_cap_rw, 553
 - make_cap_rws, 554
 - msg_ptr, 555
 - read, 555
 - str_cp_in, 555
- L4::lpc::Array< ELEM_TYPE, LEN_TYPE >, 865
- L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >, 868
- L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE >, 869
- L4::lpc::As_value< T >, 871
- L4::lpc::Buf_item, 872
- L4::lpc::Call, 873
- L4::lpc::Call_t< RIGHTS >, 874
- L4::lpc::Call_zero_send_timeout, 875
- L4::lpc::Cap< T >, 877
 - Cap, 878
 - Cap_mask, 878
 - from_ci, 879
 - Rights_mask, 878
- L4::lpc::Gen_fpage< T >, 879
 - cap_received, 881
 - id_received, 881
 - is_compound, 881
 - local_id_received, 882
- L4::lpc::In_out< T >, 882
- L4::lpc::lostream, 883
 - call, 887
 - get, 888, 889
 - lostream, 887
 - put, 889, 890
 - reply_and_wait, 890, 891
 - reset, 891
- L4::lpc::Istream, 892

- get, [896](#), [897](#)
- Istream, [896](#)
- receive, [898](#)
- reset, [898](#)
- skip, [899](#)
- tag, [899](#)
- wait, [900](#), [901](#)
- L4::lpc::Msg, [556](#)
 - align_to, [558](#)
 - Br_bytes, [557](#)
 - check_size, [559](#), [560](#)
 - Item_bytes, [557](#)
 - Item_words, [557](#)
 - Mr_bytes, [557](#)
 - Mr_words, [557](#)
 - msg_add, [560](#)
 - msg_get, [561](#)
 - Word_bytes, [557](#)
- L4::lpc::Msg::Cnt_val_ops< MTYPE, DIR, CLASS >, [902](#)
- L4::lpc::Msg::Cls_buffer, [904](#)
- L4::lpc::Msg::Cls_data, [905](#)
- L4::lpc::Msg::Cls_item, [906](#)
- L4::lpc::Msg::Dir_in, [907](#)
- L4::lpc::Msg::Dir_out, [908](#)
- L4::lpc::Msg::Do_in_data, [909](#)
- L4::lpc::Msg::Do_in_items, [910](#)
- L4::lpc::Msg::Do_out_data, [911](#)
- L4::lpc::Msg::Do_out_items, [912](#)
- L4::lpc::Msg::Do_rcv_buffers, [913](#)
- L4::lpc::Msg::Elem< Array< A, LEN > >, [916](#)
- L4::lpc::Msg::Elem< Array< A, LEN > & >, [915](#)
- L4::lpc::Msg::Elem< Array_ref< A, LEN > & >, [917](#)
- L4::lpc::Msg::Is_valid_rpc_type< T >, [918](#)
- L4::lpc::Msg::Svr_arg_pack< IPC_TYPE >, [920](#)
- L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >, [920](#)
- L4::lpc::Msg_ptr< T >, [922](#)
 - Msg_ptr, [922](#)
- L4::lpc::Opt< T >, [923](#)
- L4::lpc::Ostream, [924](#)
 - put, [927](#), [928](#)
 - send, [928](#)
 - tag, [929](#)
- L4::lpc::Out< T >, [930](#)
- L4::lpc::Ret_array< T >, [931](#)
- L4::lpc::Send_only, [932](#)
- L4::lpc::Small_buf, [933](#)
 - Small_buf, [933](#), [934](#)
- L4::lpc::Snd_item, [934](#)
- L4::lpc::Str_cp_in< T >, [935](#)
 - Str_cp_in, [936](#)
- L4::lpc::Varg, [936](#)
 - data, [938](#)
 - get_value, [938](#)
 - is_nil, [939](#)
 - is_of, [939](#)
 - is_of_int, [940](#)
 - length, [940](#)
 - tag, [941](#)
 - type, [941](#)
 - value, [941](#)
- L4::lpc::Varg_list< MAX >, [942](#)
- L4::lpc::Varg_list_ref, [944](#)
 - Varg_list_ref, [945](#)
- L4::lpc::Varg_list_ref::Iterator, [946](#)
- L4::lpc_gate, [947](#)
 - get_infos, [949](#)
- L4::lpc_svr, [561](#)
- L4::lpc_svr::Br_manager_no_buffers, [950](#)
 - alloc_buffer_demand, [952](#)
- L4::lpc_svr::Compound_reply, [953](#)
- L4::lpc_svr::Default_loop_hooks, [954](#)
- L4::lpc_svr::Default_setup_wait, [956](#)
- L4::lpc_svr::Default_timeout, [957](#)
- L4::lpc_svr::Direct_dispatch< R >, [959](#)
- L4::lpc_svr::Direct_dispatch< R * >, [961](#)
- L4::lpc_svr::Exc_dispatch< R, Exc >, [962](#)
- L4::lpc_svr::Ignore_errors, [963](#)
- L4::lpc_svr::Server_iface, [965](#)
 - add_timeout, [968](#)
 - alloc_buffer_demand, [968](#)
 - get_rcv_cap, [969](#)
 - rcv_cap, [969](#), [970](#)
 - realloc_rcv_cap, [971](#)
 - remove_timeout, [972](#)
- L4::lpc_svr::Timed_work< HOOKS >, [972](#)
- L4::lpc_svr::Timeout, [973](#)
 - expired, [974](#)
 - timeout, [975](#)
- L4::lpc_svr::Timeout_queue, [976](#)
 - add, [976](#)
 - handle_expired_timeouts, [977](#)
 - next_timeout, [978](#)
 - remove, [979](#)
 - timeout_expired, [980](#)
- L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >, [981](#)
 - add_timeout, [984](#)
 - remove_timeout, [985](#)
- L4::Irq, [986](#)
 - detach, [989](#)
 - receive, [990](#)
 - unmask, [991](#)
 - wait, [991](#)
- L4::Irq_eoi, [992](#)
 - unmask, [994](#)
- L4::Irq_handler_object, [994](#)
- L4::Irq_mux, [997](#)
 - chain, [1000](#)
- L4::Irqep_t< Derived, BASE, bool >, [1001](#)
 - dispatch, [1004](#)
 - obj_cap, [1005](#)
- L4::Kip::Mem_desc, [1005](#)
 - all, [1008](#), [1009](#)
 - Arch, [1008](#)

- Bootloader, [1008](#)
- Conventional, [1008](#)
- count, [1010](#)
- Dedicated, [1008](#)
- end, [1011](#)
- first, [1012](#)
- Info, [1008](#)
- Info_acpi_rsd, [1007](#)
- Info_sub_type, [1007](#)
- is_virtual, [1013](#)
- Mem_desc, [1008](#)
- Mem_type, [1007](#)
- Reserved, [1008](#)
- set, [1013](#)
- Shared, [1008](#)
- size, [1014](#)
- start, [1014](#)
- sub_type, [1014](#)
- type, [1015](#)
- Undefined, [1008](#)
- L4::Kobject, [1015](#)
 - cap, [1016](#)
 - dec_refcnt, [1018](#)
- L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >, [1018](#)
 - __iface, [1020](#)
 - __iface_list, [1021](#)
 - __check_protocols__, [1021](#)
 - c, [1021](#)
 - Class, [1021](#)
- L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >, [1022](#)
 - __iface, [1025](#)
 - __iface_list, [1025](#)
 - __check_protocols__, [1026](#)
 - c, [1026](#)
 - Class, [1025](#)
- L4::Kobject_demand< T >, [1026](#)
- L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >, [1027](#)
- L4::Kobject_typeid< T >, [1029](#)
 - Demand, [1030](#)
 - demand, [1030](#)
 - id, [1030](#)
 - proto_dispatch, [1031](#)
- L4::Kobject_typeid< void >, [1032](#)
- L4::Kobject_x< Derived, ARGS >, [1033](#)
- L4::Meta, [1034](#)
 - interface, [1037](#)
 - num_interfaces, [1037](#)
 - supports, [1037](#)
- L4::Out_of_memory, [1038](#)
- L4::Pager, [1041](#)
 - page_fault, [1044](#)
- L4::Platform_control, [1045](#)
 - Cpu_disable, [1047](#)
 - cpu_disable, [1047](#)
 - Cpu_enable, [1047](#)
 - cpu_enable, [1048](#)
 - Opcode, [1047](#)
 - Shutdown, [1047](#)
 - Suspend, [1047](#)
 - system_shutdown, [1048](#)
 - system_suspend, [1048](#)
- L4::Poll_timeout_kipclock, [1049](#)
 - Poll_timeout_kipclock, [1050](#)
 - set, [1050](#)
 - test, [1051](#)
 - timed_out, [1052](#)
- L4::Proto_t< P >, [1052](#)
- L4::Rcv_endpoint, [1053](#)
 - bind_thread, [1056](#)
- L4::Registry_iface, [1057](#)
 - register_irq_obj, [1058](#)
 - register_obj, [1059](#), [1060](#)
 - unregister_obj, [1061](#)
- L4::Runtime_error, [1061](#)
 - err_no, [1064](#)
 - extra_str, [1064](#)
 - Runtime_error, [1064](#)
- L4::Scheduler, [1065](#)
 - idle_time, [1067](#)
 - info, [1068](#)
 - is_online, [1068](#)
 - run_thread, [1069](#)
- L4::Semaphore, [1070](#)
 - down, [1073](#)
 - up, [1074](#)
- L4::Server< LOOP_HOOKS >, [1075](#)
 - internal_loop, [1077](#)
 - loop, [1078](#)
 - Server, [1077](#)
- L4::Server_object, [1079](#)
 - dispatch, [1082](#)
- L4::Server_object_t< IFACE, BASE >, [1084](#)
 - dispatch_meta_request, [1086](#)
 - get_buffer_demand, [1086](#)
 - proto_dispatch, [1087](#)
- L4::Server_object_x< Derived, IFACE, BASE >, [1088](#)
- L4::Smart_cap< T, SMART >, [1091](#)
 - Smart_cap, [1094](#)
- L4::String, [1094](#)
- L4::Task, [1095](#)
 - add_ku_mem, [1098](#)
 - cap_equal, [1099](#)
 - cap_valid, [1099](#)
 - delete_obj, [1100](#)
 - map, [1100](#)
 - release_cap, [1101](#)
 - unmap, [1101](#)
 - unmap_batch, [1102](#)
- L4::Thread, [1103](#)
 - control, [1107](#)
 - ex_regs, [1107](#), [1108](#)
 - modify_senders, [1109](#)
 - register_del_irq, [1110](#)

- stats_time, 1111
- switch_to, 1111
- vcpu_control, 1111
- vcpu_control_ext, 1112
- vcpu_resume_commit, 1113
- vcpu_resume_start, 1113
- L4::Thread::Attr, 1114
 - Attr, 1115
 - bind, 1116
 - exc_handler, 1116
 - pager, 1117
 - ux_host_syscall, 1117
- L4::Thread::Modify_senders, 1118
 - add, 1118
- L4::Triggerable, 1120
 - trigger, 1122
- L4::Type_info, 1124
- L4::Type_info::Demand, 1125
 - Demand, 1126
 - no_demand, 1127
- L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >, 1128
 - Caps, 1130
 - Flags, 1130
 - Mem, 1130
 - Ports, 1130
- L4::Type_info::Demand_union_t< D1, D2 >, 1130
- L4::Typeid, 562
- L4::Typeid::Detail::Rpc< OPCODE, O, Default_op< R > >::Rpc< Y >, 1134
- L4::Typeid::Detail::Rpc< OPCODE, O, R, X... >, 1135
- L4::Typeid::Detail::Rpc< OPCODE, O, R, X... >::Rpc< Y >, 1136
- L4::Typeid::Detail::Rpc< OPCODE, O, X >, 1133
- L4::Typeid::Detail::Rpc_end, 1137
- L4::Typeid::P_dispatch< LIST >, 1138
- L4::Typeid::Raw_ipc< CLASS >, 1139
- L4::Typeid::Rpc_nocode< OPERATION >, 1139
- L4::Typeid::Rpc< RPCS >, 1142
- L4::Typeid::Rpc_code< OPCODE_TYPE >, 1144
- L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS >, 1145
- L4::Typeid::Rpc_sys< ARG >, 1147
- L4::Types, 563
- L4::Types::Bool< V >, 1149
- L4::Types::False, 1151
- L4::Types::Flags< BITS_ENUM, UNDERLYING >, 1152
 - clear, 1156
 - Flags, 1155
 - from_raw, 1156
 - None, 1155
 - None_type, 1155
- L4::Types::Flags_ops_t< DT >, 1157
- L4::Types::Flags_t< DT, T >, 1159
- L4::Types::Int_for_size< SIZE, bool >, 1161
- L4::Types::Int_for_type< T >, 1162
- L4::Types::Same< A, B >, 1163
- L4::Types::True, 1165
- L4::Unknown_error, 1167
- L4::Vcon, 1170
 - get_attr, 1172
 - read, 1173
 - read_with_flags, 1174
 - send, 1174
 - set_attr, 1175
 - write, 1176
- L4::Vm, 1177
- l4_addr_consts_t
 - Memory related, 359
- L4_ALWAYS_INLINE
 - compiler.h, 2047
- L4_AMD64_SEGMENT_FS
 - segment.h, 1741
- L4_AMD64_SEGMENT_GS
 - segment.h, 1741
- l4_assert
 - assert.h, 2228
- L4_BASE_ARM_SMCCC_CAP
 - Capabilities, 144
- L4_BASE_CAPS_LAST
 - Capabilities, 144
- L4_BASE_DEBUGGER_CAP
 - Capabilities, 144
- L4_BASE_FACTORY_CAP
 - Capabilities, 144
- L4_BASE_ICU_CAP
 - Capabilities, 144
- L4_BASE_IOMMU_CAP
 - Capabilities, 144
- L4_BASE_LOG_CAP
 - Capabilities, 144
- L4_BASE_PAGER_CAP
 - Capabilities, 144
- L4_BASE_SCHEDULER_CAP
 - Capabilities, 144
- L4_BASE_TASK_CAP
 - Capabilities, 144
- L4_BASE_THREAD_CAP
 - Capabilities, 144
- L4_BDR_IO_SHIFT
 - Buffer Registers (BRs), 137
- L4_BDR_MEM_SHIFT
 - Buffer Registers (BRs), 137
- L4_BDR_OBJ_SHIFT
 - Buffer Registers (BRs), 137
- l4_buf_regs_t, 1180
- l4_buffer_desc_consts_t
 - Buffer Registers (BRs), 137
- l4_busy_wait_ns
 - rdtsc.h, 1709, 1720
- l4_busy_wait_us
 - rdtsc.h, 1710, 1721
- l4_bytes_to_mwords
 - Memory related, 359

- `l4_cache_clean_data`
 - Cache Consistency, [139](#)
- `l4_cache_coherent`
 - Cache Consistency, [140](#)
- `l4_cache_dma_coherent`
 - Cache Consistency, [140](#)
- `l4_cache_flush_data`
 - Cache Consistency, [141](#)
- `l4_cache_inv_data`
 - Cache Consistency, [141](#)
- `l4_calibrate_tsc`
 - `rdtsc.h`, [1710](#), [1722](#)
- `l4_cap_consts_t`
 - Capabilities, [143](#)
- `L4_CAP_FPAGE_D`
 - Flex pages, [212](#)
- `L4_CAP_FPAGE_R`
 - Flex pages, [212](#)
- `L4_cap_fpage_rights`
 - Flex pages, [212](#)
- `L4_CAP_FPAGE_RO`
 - Flex pages, [212](#)
- `L4_CAP_FPAGE_RS`
 - Flex pages, [213](#)
- `L4_CAP_FPAGE_RSD`
 - Flex pages, [213](#)
- `L4_CAP_FPAGE_RW`
 - Flex pages, [212](#)
- `L4_CAP_FPAGE_RWD`
 - Flex pages, [213](#)
- `L4_CAP_FPAGE_RWS`
 - Flex pages, [213](#)
- `L4_CAP_FPAGE_RWSD`
 - Flex pages, [213](#)
- `L4_CAP_FPAGE_S`
 - Flex pages, [212](#)
- `L4_CAP_FPAGE_W`
 - Flex pages, [212](#)
- `L4_CAP_MASK`
 - Capabilities, [143](#)
- `L4_CAP_SHIFT`
 - Capabilities, [143](#)
- `L4_CAP_SIZE`
 - Capabilities, [143](#)
- `l4_capability_equal`
 - Capabilities, [144](#)
- `l4_capability_next`
 - `types.h`, [2197](#)
- `l4_debugger_get_object_name`
 - `debugger.h`, [2097](#)
- `l4_debugger_global_id`
 - Kernel Debugger, [280](#)
- `l4_debugger_kobj_to_id`
 - Kernel Debugger, [280](#)
- `l4_debugger_query_log_name`
 - `debugger.h`, [2097](#)
- `l4_debugger_query_log_typeid`
 - `debugger.h`, [2098](#)
- `l4_debugger_set_object_name`
 - Kernel Debugger, [281](#)
- `l4_debugger_switch_log`
 - `debugger.h`, [2098](#)
- `l4_default_caps_t`
 - Capabilities, [143](#)
- `L4_DISABLE_COPY`
 - capability, [2043](#)
- `L4_E2BIG`
 - Error codes, [180](#)
- `L4_EACCESS`
 - Error codes, [180](#)
- `L4_EADDRNOTAVAIL`
 - Error codes, [180](#)
- `L4_EAGAIN`
 - Error codes, [180](#)
- `L4_EBADPROTO`
 - Error codes, [180](#)
- `L4_EBUSY`
 - Error codes, [180](#)
- `L4_EEXIST`
 - Error codes, [180](#)
- `L4_EFAULT`
 - Error codes, [180](#)
- `L4_EINVAL`
 - Error codes, [180](#)
- `L4_EIO`
 - Error codes, [180](#)
- `L4_EIPC_HI`
 - Error codes, [180](#)
- `L4_EIPC_LO`
 - Error codes, [180](#)
- `L4_EMMSGMISSARG`
 - Error codes, [180](#)
- `L4_EMMSGTOOLONG`
 - Error codes, [180](#)
- `L4_EMMSGTOOSHORT`
 - Error codes, [180](#)
- `L4_ENAMETOOLONG`
 - Error codes, [180](#)
- `L4_ENODEV`
 - Error codes, [180](#)
- `L4_ENOENT`
 - Error codes, [180](#)
- `L4_ENOMEM`
 - Error codes, [180](#)
- `L4_ENOREPLY`
 - Error codes, [180](#)
- `L4_ENOSPC`
 - Error codes, [180](#)
- `L4_ENOSYS`
 - Error codes, [180](#)
- `L4_ENXIO`
 - Error codes, [180](#)
- `L4_EOK`
 - Error codes, [180](#)
- `L4_EPERM`
 - Error codes, [180](#)

- L4_ERANGE
 - Error codes, [180](#)
- L4_ERRNOMAX
 - Error codes, [180](#)
- l4_error
 - Error Handling, [174](#)
- l4_error_code_t
 - Error codes, [179](#)
- l4_exc_regs_t, [1181](#)
 - flags, [1183](#)
 - ss, [1183](#)
- l4_factory_create_factory
 - Factory, [190](#)
- l4_factory_create_factory_u
 - Factory, [191](#)
- l4_factory_create_gate
 - Factory, [191](#)
- l4_factory_create_gate_u
 - Factory, [192](#)
- l4_factory_create_irq
 - Factory, [194](#)
- l4_factory_create_irq_u
 - Factory, [195](#)
- l4_factory_create_task
 - Factory, [196](#)
- l4_factory_create_task_u
 - Factory, [197](#)
- l4_factory_create_thread
 - Factory, [198](#)
- l4_factory_create_thread_u
 - Factory, [199](#)
- l4_factory_create_vm
 - Factory, [200](#)
- l4_factory_create_vm_u
 - Factory, [201](#)
- L4_FP_ALL_SPACES
 - Task, [437](#)
- L4_FP_DELETE_OBJ
 - Task, [437](#)
- L4_FP_OTHER_SPACES
 - Task, [437](#)
- l4_fpage
 - Flex pages, [215](#)
- L4_FPAGE_ADDR_BITS
 - Flex pages, [214](#)
- L4_FPAGE_ADDR_SHIFT
 - Flex pages, [214](#)
- l4_fpage_all
 - Flex pages, [215](#)
- L4_FPAGE_BUFFERABLE
 - Flex pages, [213](#)
- L4_FPAGE_C_IPCGATE_SVR
 - Flex pages, [215](#)
- L4_FPAGE_C_NO_REF_CNT
 - Flex pages, [215](#)
- L4_FPAGE_C_OBJ_RIGHT1
 - Flex pages, [215](#)
- L4_FPAGE_C_OBJ_RIGHT2
 - Flex pages, [215](#)
- L4_FPAGE_C_OBJ_RIGHT3
 - Flex pages, [215](#)
- L4_FPAGE_C_OBJ_RIGHTS
 - Flex pages, [215](#)
- L4_FPAGE_C_REF_CNT
 - Flex pages, [215](#)
- L4_FPAGE_CACHE_OPT
 - Flex pages, [213](#)
- l4_fpage_cacheability_opt_t
 - Flex pages, [213](#)
- L4_FPAGE_CACHEABLE
 - Flex pages, [213](#)
- l4_fpage_consts
 - Flex pages, [213](#)
- l4_fpage_contains
 - Flex pages, [216](#)
- l4_fpage_invalid
 - Flex pages, [216](#)
- l4_fpage_ioport
 - Flex pages, [217](#)
- l4_fpage_max_order
 - Flex pages, [217](#)
- l4_fpage_memaddr
 - Flex pages, [218](#)
- l4_fpage_obj
 - Flex pages, [219](#)
- l4_fpage_page
 - Flex pages, [219](#)
- L4_fpage_rights
 - Flex pages, [214](#)
- l4_fpage_rights
 - Flex pages, [220](#)
- L4_FPAGE_RIGHTS_BITS
 - Flex pages, [214](#)
- L4_FPAGE_RIGHTS_MASK
 - Flex pages, [214](#)
- L4_FPAGE_RIGHTS_SHIFT
 - Flex pages, [214](#)
- L4_FPAGE_RO
 - Flex pages, [214](#)
- L4_FPAGE_RW
 - Flex pages, [214](#)
- L4_FPAGE_RWX
 - Flex pages, [214](#)
- L4_FPAGE_RX
 - Flex pages, [214](#)
- l4_fpage_set_rights
 - Flex pages, [221](#)
- l4_fpage_size
 - Flex pages, [221](#)
- L4_FPAGE_SIZE_BITS
 - Flex pages, [214](#)
- L4_FPAGE_SIZE_SHIFT
 - Flex pages, [214](#)
- l4_fpage_t, [1184](#)
- l4_fpage_type
 - Flex pages, [222](#)

- L4_FPAGE_TYPE_BITS
 - Flex pages, [214](#)
- L4_FPAGE_TYPE_SHIFT
 - Flex pages, [214](#)
- L4_FPAGE_UNCACHEABLE
 - Flex pages, [213](#)
- L4_FPAGE_W
 - Flex pages, [214](#)
- L4_FPAGE_X
 - Flex pages, [214](#)
- l4_get_hz
 - rdtsc.h, [1711](#), [1722](#)
- L4_HIDDEN
 - compiler.h, [2047](#)
- l4_icu_bind
 - Interrupt controller, [268](#)
- l4_icu_bind_u
 - Interrupt controller, [269](#)
- L4_ICU_FLAG_MSI
 - Interrupt controller, [268](#)
- L4_icu_flags
 - Interrupt controller, [268](#)
- l4_icu_info
 - Interrupt controller, [270](#)
- l4_icu_info_t, [1184](#)
 - features, [1186](#)
 - Interrupt controller, [268](#)
- l4_icu_info_u
 - Interrupt controller, [271](#)
- l4_icu_mask
 - Interrupt controller, [272](#)
- l4_icu_mask_u
 - Interrupt controller, [272](#)
- l4_icu_msi_info
 - Interrupt controller, [273](#)
- l4_icu_msi_info_t, [1186](#)
- l4_icu_msi_info_u
 - Interrupt controller, [274](#)
- L4_ICU_OP_BIND
 - L4 IPC Opcodes, [287](#)
- L4_ICU_OP_INFO
 - L4 IPC Opcodes, [287](#)
- L4_ICU_OP_MASK
 - L4 IPC Opcodes, [287](#)
- L4_ICU_OP_MSI_INFO
 - L4 IPC Opcodes, [287](#)
- L4_ICU_OP_SET_MODE
 - L4 IPC Opcodes, [287](#)
- L4_ICU_OP_UNBIND
 - L4 IPC Opcodes, [287](#)
- L4_ICU_OP_UNMASK
 - L4 IPC Opcodes, [287](#)
- L4_icu_opcode
 - L4 IPC Opcodes, [286](#)
- l4_icu_set_mode
 - Interrupt controller, [275](#)
- l4_icu_set_mode_u
 - Interrupt controller, [275](#)
- l4_icu_unbind
 - Interrupt controller, [276](#)
- l4_icu_unbind_u
 - Interrupt controller, [277](#)
- l4_icu_unmask
 - Interrupt controller, [278](#)
- l4_icu_unmask_u
 - Interrupt controller, [278](#)
- L4_INLINE_RPC
 - ipc_iface, [2065](#)
- L4_INLINE_RPC_NF
 - ipc_iface, [2066](#)
- L4_INLINE_RPC_NF_OP
 - ipc_iface, [2066](#)
- L4_INLINE_RPC_OP
 - ipc_iface, [2067](#)
- L4_INVALID_ADDR
 - Memory related, [359](#)
- L4_INVALID_CAP
 - Capabilities, [143](#)
- l4_iofpage
 - Flex pages, [222](#)
- L4_IOPORT_MAX
 - Flex pages, [212](#)
- l4_ipc
 - Object Invocation, [386](#)
- l4_ipc_call
 - Object Invocation, [387](#)
- L4_IPC_ENOT_EXISTENT
 - Error Handling, [174](#)
- l4_ipc_error
 - Error Handling, [176](#)
- l4_ipc_error_code
 - Error Handling, [177](#)
- L4_IPC_ERROR_MASK
 - Error Handling, [174](#)
- L4_IPC_GATE_BIND_OP
 - L4 IPC Opcodes, [287](#)
- l4_ipc_gate_bind_thread
 - IPC-Gate API, [234](#)
- L4_IPC_GATE_GET_INFO_OP
 - L4 IPC Opcodes, [287](#)
- l4_ipc_gate_get_infos
 - IPC-Gate API, [235](#)
- L4_ipc_gate_ops
 - L4 IPC Opcodes, [287](#)
- l4_ipc_is_rcv_error
 - Error Handling, [178](#)
- l4_ipc_is_snd_error
 - Error Handling, [178](#)
- L4_IPC_REABORTED
 - Error Handling, [174](#)
- L4_IPC_RECANCELED
 - Error Handling, [174](#)
- l4_ipc_receive
 - Object Invocation, [388](#)
- L4_IPC_REMAPFAILED
 - Error Handling, [174](#)

- L4_IPC_REMSGCUT
 - Error Handling, [174](#)
- l4_ipc_reply_and_wait
 - Object Invocation, [389](#)
- L4_IPC_RERCVPFTO
 - Error Handling, [174](#)
- L4_IPC_RESNDPFTO
 - Error Handling, [174](#)
- L4_IPC_RETIMEOUT
 - Error Handling, [174](#)
- L4_IPC_SEABORTED
 - Error Handling, [174](#)
- L4_IPC_SECANCELED
 - Error Handling, [174](#)
- L4_IPC_SEMAPFAILED
 - Error Handling, [174](#)
- L4_IPC_SEMSGCUT
 - Error Handling, [174](#)
- l4_ipc_send
 - Object Invocation, [391](#)
- l4_ipc_send_and_wait
 - Object Invocation, [392](#)
- L4_IPC_SERCVPFTO
 - Error Handling, [174](#)
- L4_IPC_SESNDFPFTO
 - Error Handling, [174](#)
- L4_IPC_SETIMEOUT
 - Error Handling, [174](#)
- l4_ipc_sleep
 - Object Invocation, [393](#)
- L4_IPC_SND_ERR_MASK
 - Error Handling, [174](#)
- l4_ipc_tcr_error_t
 - Error Handling, [174](#)
- l4_ipc_timeout
 - Timeouts, [467](#)
- L4_IPC_TIMEOUT_0
 - Timeouts, [466](#)
- l4_ipc_to_errno
 - ipc.h, [2124](#)
- l4_ipc_wait
 - Object Invocation, [394](#)
- l4_irq_detach
 - IRQs, [239](#)
- l4_irq_detach_u
 - IRQs, [239](#)
- L4_IRQ_F_BOTH
 - IRQs, [238](#)
- L4_IRQ_F_BOTH_EDGE
 - IRQs, [238](#)
- L4_IRQ_F_CLEAR_WAKEUP
 - IRQs, [238](#)
- L4_IRQ_F_EDGE
 - IRQs, [238](#)
- L4_IRQ_F_LEVEL
 - IRQs, [238](#)
- L4_IRQ_F_LEVEL_HIGH
 - IRQs, [238](#)
- L4_IRQ_F_LEVEL_LOW
 - IRQs, [238](#)
- L4_IRQ_F_MASK
 - IRQs, [238](#)
- L4_IRQ_F_NEG
 - IRQs, [238](#)
- L4_IRQ_F_NEG_EDGE
 - IRQs, [238](#)
- L4_IRQ_F_NONE
 - IRQs, [238](#)
- L4_IRQ_F_POS
 - IRQs, [238](#)
- L4_IRQ_F_POS_EDGE
 - IRQs, [238](#)
- L4_IRQ_F_SET_WAKEUP
 - IRQs, [238](#)
- L4_irq_mode
 - IRQs, [238](#)
- l4_irq_mux_chain
 - IRQs, [240](#)
- l4_irq_mux_chain_u
 - IRQs, [241](#)
- l4_irq_receive
 - IRQs, [242](#)
- l4_irq_receive_u
 - IRQs, [243](#)
- l4_irq_trigger
 - IRQs, [244](#)
- l4_irq_trigger_u
 - IRQs, [245](#)
- l4_irq_unmask
 - IRQs, [246](#)
- l4_irq_unmask_u
 - IRQs, [246](#)
- l4_irq_wait
 - IRQs, [247](#)
- l4_irq_wait_u
 - IRQs, [248](#)
- l4_is_fpage_writable
 - Flex pages, [223](#)
- l4_is_invalid_cap
 - Capabilities, [144](#)
- l4_is_valid_cap
 - Capabilities, [145](#)
- L4_ITEM_CONT
 - Message Items, [363](#)
- L4_ITEM_MAP
 - Message Items, [363](#)
- l4_kernel_info_get_mem_desc_end
 - Memory descriptors (C version), [351](#)
- l4_kernel_info_get_mem_desc_is_virtual
 - Memory descriptors (C version), [351](#)
- l4_kernel_info_get_mem_desc_start
 - Memory descriptors (C version), [352](#)
- l4_kernel_info_get_mem_desc_subtype
 - Memory descriptors (C version), [352](#)
- l4_kernel_info_get_mem_desc_type
 - Memory descriptors (C version), [352](#)

- l4_kernel_info_get_num_mem_descs
 - Memory descriptors (C version), [353](#)
- l4_kernel_info_mem_desc_t, [1187](#)
 - Memory descriptors (C version), [350](#)
- l4_kernel_info_set_mem_desc
 - Memory descriptors (C version), [353](#)
- l4_kernel_info_t, [1188](#)
- l4_kernel_info_version_offset
 - kip.h, [2292](#)
- l4_kip_clock
 - kip.h, [2293](#)
- l4_kip_clock_lw
 - kip.h, [2294](#)
- l4_kip_version
 - kip.h, [2295](#)
- l4_kip_version_string
 - kip.h, [2295](#)
- L4_LOG2_PAGESIZE
 - Memory related, [358](#)
- L4_LOG2_SUPERPAGESIZE
 - Memory related, [358](#)
- l4_map_control
 - Message Items, [364](#)
- L4_MAP_ITEM_GRANT
 - Message Items, [363](#)
- L4_MAP_ITEM_MAP
 - Message Items, [363](#)
- l4_map_obj_control
 - Message Items, [364](#)
- l4_mem_info_acpi_rsdp
 - Memory descriptors (C version), [351](#)
- l4_mem_info_sub_type_t
 - Memory descriptors (C version), [350](#)
- L4_mem_op_widths
 - Memory operations., [354](#)
- l4_mem_read
 - Memory operations., [355](#)
- l4_mem_type_archspecific
 - Memory descriptors (C version), [351](#)
- l4_mem_type_bootloader
 - Memory descriptors (C version), [351](#)
- l4_mem_type_conventional
 - Memory descriptors (C version), [351](#)
- l4_mem_type_dedicated
 - Memory descriptors (C version), [351](#)
- l4_mem_type_info
 - Memory descriptors (C version), [351](#)
- l4_mem_type_reserved
 - Memory descriptors (C version), [351](#)
- l4_mem_type_shared
 - Memory descriptors (C version), [351](#)
- l4_mem_type_t
 - Memory descriptors (C version), [351](#)
- l4_mem_type_undefined
 - Memory descriptors (C version), [351](#)
- L4_MEM_WIDTH_1BYTE
 - Memory operations., [355](#)
- L4_MEM_WIDTH_2BYTE
 - Memory operations., [355](#)
- L4_MEM_WIDTH_4BYTE
 - Memory operations., [355](#)
- l4_mem_write
 - Memory operations., [355](#)
- l4_msg_item_consts_t
 - Message Items, [363](#)
- l4_msg_regs_t, [1190](#)
- l4_msgtag
 - Message Tag, [370](#)
- L4_MSGTAG_ERROR
 - Message Tag, [368](#)
- L4_MSGTAG_FLAGS
 - Message Tag, [368](#)
- l4_msgtag_flags
 - Message Tag, [368](#), [371](#)
- l4_msgtag_has_error
 - Message Tag, [371](#)
- l4_msgtag_is_exception
 - Message Tag, [372](#)
- l4_msgtag_is_io_page_fault
 - Message Tag, [373](#)
- l4_msgtag_is_page_fault
 - Message Tag, [373](#)
- l4_msgtag_is_preemption
 - Message Tag, [374](#)
- l4_msgtag_is_sigma0
 - Message Tag, [375](#)
- l4_msgtag_is_sys_exception
 - Message Tag, [375](#)
- l4_msgtag_items
 - Message Tag, [376](#)
- l4_msgtag_label
 - Message Tag, [377](#)
- L4_MSGTAG_PROPAGATE
 - Message Tag, [368](#)
- l4_msgtag_protocol
 - Message Tag, [369](#)
- L4_MSGTAG_SCHEDULE
 - Message Tag, [368](#)
- l4_msgtag_t, [1191](#)
 - flags, [1192](#)
 - Message Tag, [368](#)
- L4_MSGTAG_TRANSFER_FPU
 - Message Tag, [368](#)
- l4_msgtag_words
 - Message Tag, [378](#)
- L4_NOTHROW
 - compiler.h, [2047](#)
- l4_ns_to_tsc
 - rdtsc.h, [1711](#), [1722](#)
- l4_obj_fpage
 - Flex pages, [224](#)
- L4_obj_fpage_ctl
 - Flex pages, [214](#)
- L4_PAGEMASK
 - Memory related, [358](#)
- l4_platform_ctl_cpu_disable

- Platform Control C API, [397](#)
- L4_PLATFORM_CTL_CPU_DISABLE_OP
 - L4 IPC Opcodes, [288](#)
- l4_platform_ctl_cpu_enable
 - Platform Control C API, [398](#)
- L4_PLATFORM_CTL_CPU_ENABLE_OP
 - L4 IPC Opcodes, [288](#)
- L4_platform_ctl_ops
 - L4 IPC Opcodes, [288](#)
- L4_platform_ctl_proto
 - Message Tag, [369](#)
- L4_PLATFORM_CTL_SYS_SHUTDOWN_OP
 - L4 IPC Opcodes, [288](#)
- L4_PLATFORM_CTL_SYS_SUSPEND_OP
 - L4 IPC Opcodes, [288](#)
- l4_platform_ctl_system_shutdown
 - Platform Control C API, [398](#)
- l4_platform_ctl_system_suspend
 - Platform Control C API, [399](#)
- L4_PROTO_ALLOW_SYSCALL
 - Message Tag, [369](#)
- L4_PROTO_DEBUGGER
 - Message Tag, [369](#)
- L4_PROTO_DMA_SPACE
 - Message Tag, [369](#)
- L4_PROTO_EXCEPTION
 - Message Tag, [369](#)
- L4_PROTO_FACTORY
 - Message Tag, [369](#)
- L4_PROTO_IO_PAGE_FAULT
 - Message Tag, [369](#)
- L4_PROTO_IOMMU
 - Message Tag, [369](#)
- L4_PROTO_IRQ
 - Message Tag, [369](#)
- L4_PROTO_IRQ_MUX
 - Message Tag, [369](#)
- L4_PROTO_IRQ_SENDER
 - Message Tag, [369](#)
- L4_PROTO_KOBJECT
 - Message Tag, [369](#)
- L4_PROTO_LOG
 - Message Tag, [369](#)
- L4_PROTO_META
 - Message Tag, [369](#)
- L4_PROTO_NONE
 - Message Tag, [369](#)
- L4_PROTO_PAGE_FAULT
 - Message Tag, [369](#)
- L4_PROTO_PF_EXCEPTION
 - Message Tag, [369](#)
- L4_PROTO_PLATFORM_CTL
 - Message Tag, [370](#)
- L4_PROTO_PREEMPTION
 - Message Tag, [369](#)
- L4_PROTO_SCHEDULER
 - Message Tag, [369](#)
- L4_PROTO_SEMAPHORE
 - Message Tag, [369](#)
- L4_PROTO_SIGMA0
 - Message Tag, [369](#)
- L4_PROTO_SMCCC
 - Message Tag, [369](#)
- L4_PROTO_SYS_EXCEPTION
 - Message Tag, [369](#)
- L4_PROTO_TASK
 - Message Tag, [369](#)
- L4_PROTO_THREAD
 - Message Tag, [369](#)
- L4_PROTO_VM
 - Message Tag, [369](#)
- L4_RCV_EP_BIND_OP
 - rcv_endpoint.h, [2167](#)
- l4_rcv_ep_bind_thread
 - IPC-Gate API, [235](#)
- L4_rcv_ep_ops
 - rcv_endpoint.h, [2167](#)
- L4_RCV_ITEM_LOCAL_ID
 - Message Items, [363](#)
- L4_RCV_ITEM_SINGLE_CAP
 - Message Items, [363](#)
- l4_rcv_timeout
 - Timeouts, [467](#)
- l4_rdpmc
 - rdtsc.h, [1712](#), [1723](#)
- l4_rdpmc_32
 - rdtsc.h, [1712](#), [1723](#)
- l4_rdtsc
 - rdtsc.h, [1713](#), [1723](#)
- l4_rdtsc_32
 - rdtsc.h, [1713](#), [1724](#)
- l4_round_page
 - Memory related, [359](#)
- l4_round_size
 - Memory related, [360](#)
- L4_RPC
 - ipc_iface, [2067](#)
- L4_RPC_DEF
 - ipc_client, [2062](#)
- L4_RPC_NF
 - ipc_iface, [2068](#)
- L4_RPC_NF_OP
 - ipc_iface, [2068](#)
- L4_RPC_OP
 - ipc_iface, [2069](#)
- l4_sched_cpu_set
 - Scheduler, [418](#)
- l4_sched_cpu_set_t, [1193](#)
 - gran_offset, [1195](#)
 - granularity, [1193](#)
 - offset, [1194](#)
- l4_sched_param_t, [1195](#)
- l4_scheduler_idle_time
 - Scheduler, [419](#)
- L4_SCHEDULER_IDLE_TIME_OP
 - Scheduler, [418](#)

- l4_scheduler_info
 - Scheduler, [420](#)
- L4_SCHEDULER_INFO_OP
 - Scheduler, [418](#)
- l4_scheduler_is_online
 - Scheduler, [420](#)
- L4_scheduler_ops
 - Scheduler, [418](#)
- l4_scheduler_run_thread
 - Scheduler, [421](#)
- L4_SCHEDULER_RUN_THREAD_OP
 - Scheduler, [418](#)
- l4_sleep
 - util.h, [1732](#), [1735](#)
 - Utility Functions, [475](#)
- l4_snd_fpage_t, [1197](#)
- l4_snd_timeout
 - Timeouts, [468](#)
- l4_sndfpage_add
 - Object Invocation, [395](#)
- L4_SUPERPAGEMASK
 - Memory related, [358](#)
- L4_SUPERPAGESIZE
 - Memory related, [358](#)
- L4_sys_segment
 - segment.h, [1741](#)
- l4_syscall_flags_t
 - Object Invocation, [385](#)
- L4_SYSF_CALL
 - Object Invocation, [385](#)
- L4_SYSF_NONE
 - Object Invocation, [385](#)
- L4_SYSF_OPEN_WAIT
 - Object Invocation, [385](#)
- L4_SYSF_RECV
 - Object Invocation, [385](#)
- L4_SYSF_REPLY
 - Object Invocation, [385](#)
- L4_SYSF_REPLY_AND_WAIT
 - Object Invocation, [385](#)
- L4_SYSF_SEND
 - Object Invocation, [385](#)
- L4_SYSF_SEND_AND_WAIT
 - Object Invocation, [385](#)
- L4_SYSF_WAIT
 - Object Invocation, [385](#)
- l4_task_add_ku_mem
 - Task, [437](#)
- L4_TASK_ADD_KU_MEM_OP
 - L4 IPC Opcodes, [288](#)
- l4_task_cap_equal
 - Task, [438](#)
- L4_TASK_CAP_INFO_OP
 - L4 IPC Opcodes, [288](#)
- l4_task_cap_valid
 - Task, [438](#)
- l4_task_delete_obj
 - Task, [439](#)
- L4_TASK_LDT_SET_X86_OP
 - L4 IPC Opcodes, [288](#)
- L4_task_ldt_x86_consts
 - segment.h, [1741](#), [1748](#)
- L4_TASK_LDT_X86_ENTRY_SIZE
 - segment.h, [1742](#), [1748](#)
- L4_TASK_LDT_X86_MAX_ENTRIES
 - segment.h, [1742](#), [1748](#)
- l4_task_map
 - Task, [439](#)
- L4_TASK_MAP_OP
 - L4 IPC Opcodes, [288](#)
- L4_TASK_MAP_VGICC_ARM_OP
 - L4 IPC Opcodes, [288](#)
- L4_task_ops
 - L4 IPC Opcodes, [288](#)
- l4_task_release_cap
 - Task, [440](#)
- l4_task_unmap
 - Task, [441](#)
- l4_task_unmap_batch
 - Task, [441](#)
- L4_TASK_UNMAP_OP
 - L4 IPC Opcodes, [288](#)
- l4_task_vgicc_map
 - Task, [442](#)
- L4_THREAD_AMD64_GET_SEGMENT_INFO_OP
 - L4 IPC Opcodes, [289](#)
- L4_THREAD_AMD64_SET_SEGMENT_BASE_OP
 - L4 IPC Opcodes, [289](#)
- l4_thread_arm_set_tpidruro
 - Thread, [447](#)
- L4_THREAD_ARM_TPIDRURO_OP
 - L4 IPC Opcodes, [289](#)
- L4_THREAD_CONTROL_ALIEN
 - Thread, [446](#)
- l4_thread_control_alien
 - Thread control, [460](#)
- l4_thread_control_bind
 - Thread control, [461](#)
- L4_THREAD_CONTROL_BIND_TASK
 - Thread, [446](#)
- l4_thread_control_commit
 - Thread control, [461](#)
- l4_thread_control_exc_handler
 - Thread control, [463](#)
- L4_thread_control_flags
 - Thread, [445](#)
- L4_THREAD_CONTROL_MR_IDX_BIND_TASK
 - Thread, [446](#)
- L4_THREAD_CONTROL_MR_IDX_BIND_UTCB
 - Thread, [446](#)
- L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER
 - Thread, [446](#)
- L4_THREAD_CONTROL_MR_IDX_FLAG_VALS
 - Thread, [446](#)
- L4_THREAD_CONTROL_MR_IDX_FLAGS
 - Thread, [446](#)

- L4_THREAD_CONTROL_MR_IDX_PAGER
 - Thread, [446](#)
- L4_thread_control_mr_indices
 - Thread, [446](#)
- L4_THREAD_CONTROL_OP
 - L4 IPC Opcodes, [289](#)
- l4_thread_control_pager
 - Thread control, [463](#)
- L4_THREAD_CONTROL_SET_EXC_HANDLER
 - Thread, [446](#)
- L4_THREAD_CONTROL_SET_PAGER
 - Thread, [446](#)
- l4_thread_control_start
 - Thread control, [464](#)
- l4_thread_control_ux_host_syscall
 - Thread control, [464](#)
- L4_THREAD_CONTROL_UX_NATIVE
 - Thread, [446](#)
- l4_thread_ex_regs
 - Thread, [447](#)
- L4_THREAD_EX_REGS_CANCEL
 - Thread, [446](#)
- L4_thread_ex_regs_flags
 - Thread, [446](#)
- L4_THREAD_EX_REGS_OP
 - L4 IPC Opcodes, [289](#)
- l4_thread_ex_regs_ret
 - Thread, [448](#)
- l4_thread_ex_regs_ret_u
 - Thread, [449](#)
- L4_THREAD_EX_REGS_TRIGGER_EXCEPTION
 - Thread, [446](#)
- l4_thread_ex_regs_u
 - Thread, [450](#)
- l4_thread_modify_sender_add
 - Thread, [451](#)
- l4_thread_modify_sender_commit
 - Thread, [452](#)
- L4_THREAD_MODIFY_SENDER_OP
 - L4 IPC Opcodes, [289](#)
- l4_thread_modify_sender_start
 - Thread, [452](#)
- L4_THREAD_OPCODE_MASK
 - L4 IPC Opcodes, [289](#)
- L4_thread_ops
 - L4 IPC Opcodes, [288](#)
- l4_thread_register_del_irq
 - Thread, [452](#)
- L4_THREAD_REGISTER_DELETE_IRQ_OP
 - L4 IPC Opcodes, [289](#)
- l4_thread_regs_t, [1198](#)
- L4_THREAD_STATS_OP
 - L4 IPC Opcodes, [289](#)
- l4_thread_stats_time
 - Thread, [453](#)
- l4_thread_switch
 - Thread, [453](#)
- L4_THREAD_SWITCH_OP
 - L4 IPC Opcodes, [289](#)
- l4_thread_vcpu_control
 - Thread, [454](#)
- l4_thread_vcpu_control_ext
 - Thread, [454](#)
- l4_thread_vcpu_control_ext_u
 - Thread, [455](#)
- L4_THREAD_VCPU_CONTROL_OP
 - L4 IPC Opcodes, [289](#)
- l4_thread_vcpu_control_u
 - Thread, [456](#)
- l4_thread_vcpu_resume_commit
 - Thread, [457](#)
- L4_THREAD_VCPU_RESUME_OP
 - L4 IPC Opcodes, [289](#)
- l4_thread_vcpu_resume_start
 - Thread, [458](#)
- L4_THREAD_X86_GDT_OP
 - L4 IPC Opcodes, [289](#)
- l4_thread_yield
 - Thread, [458](#)
- l4_timeout
 - Timeouts, [468](#)
- l4_timeout_abs
 - Timeouts, [469](#)
- l4_timeout_get
 - Timeouts, [470](#)
- l4_timeout_is_absolute
 - Timeouts, [470](#)
- l4_timeout_rel
 - Timeouts, [471](#)
- l4_timeout_rel_get
 - Timeouts, [472](#)
- l4_timeout_s, [1199](#)
 - Timeouts, [466](#)
- l4_timeout_t, [1200](#)
 - Timeouts, [467](#)
- l4_touch_ro
 - Utility Functions, [476](#)
- l4_touch_rw
 - Utility Functions, [476](#)
- l4_trunc_page
 - Memory related, [361](#)
- l4_trunc_size
 - Memory related, [362](#)
- l4_tsc_init
 - rdtsc.h, [1714](#), [1724](#)
- l4_tsc_to_ns
 - rdtsc.h, [1715](#), [1725](#)
- l4_tsc_to_s_and_ns
 - rdtsc.h, [1715](#), [1725](#)
- l4_tsc_to_us
 - rdtsc.h, [1716](#), [1725](#)
- L4_TYPE_VHW_FRAMEBUFFER
 - Fiasco-UX Virtual devices, [208](#)
- L4_TYPE_VHW_INPUT
 - Fiasco-UX Virtual devices, [208](#)
- L4_TYPE_VHW_NET

- Fiasco-UX Virtual devices, [208](#)
- L4_TYPE_VHW_NONE
 - Fiasco-UX Virtual devices, [208](#)
- L4_TYPES_FLAGS_OPS_DEF
 - types, [2090](#)
- l4_unmap_flags_t
 - Task, [437](#)
- l4_usleep
 - Utility Functions, [477](#)
- l4_utcb_br
 - Virtual Registers (UTCBs), [521](#)
- L4_UTCB_BUF_REGS_OFFSET
 - x86 Virtual Registers (UTCB), [534](#)
- L4_utcb_consts_x86
 - x86 Virtual Registers (UTCB), [533](#)
- l4_utcb_exc
 - Exception registers, [185](#)
- l4_utcb_exc_is_ex_regs_exception
 - Exception registers, [185](#)
- l4_utcb_exc_is_pf
 - Exception registers, [186](#)
- l4_utcb_exc_pc
 - Exception registers, [186](#)
- l4_utcb_exc_pc_set
 - Exception registers, [187](#)
- L4_UTCB_EXCEPTION_REGS_SIZE
 - x86 Virtual Registers (UTCB), [534](#)
- L4_UTCB_GENERIC_BUFFERS_SIZE
 - x86 Virtual Registers (UTCB), [534](#)
- L4_UTCB_GENERIC_DATA_SIZE
 - x86 Virtual Registers (UTCB), [534](#)
- L4_UTCB_INHERIT_FPU
 - x86 Virtual Registers (UTCB), [534](#)
- l4_utcb_mr
 - Virtual Registers (UTCBs), [521](#)
- l4_utcb_mr64_idx
 - Timeouts, [472](#)
- L4_UTCB_MSG_REGS_OFFSET
 - x86 Virtual Registers (UTCB), [534](#)
- L4_UTCB_OFFSET
 - x86 Virtual Registers (UTCB), [534](#)
- l4_utcb_t
 - Virtual Registers (UTCBs), [521](#)
- l4_utcb_tcr
 - Virtual Registers (UTCBs), [522](#)
- L4_UTCB_THREAD_REGS_OFFSET
 - x86 Virtual Registers (UTCB), [534](#)
- l4_vcon_attr_t, [1201](#)
- L4_VCON_ECHO
 - Virtual Console, [508](#)
- l4_vcon_get_attr
 - Virtual Console, [509](#)
- L4_VCON_GET_ATTR_OP
 - L4 IPC Opcodes, [289](#)
- l4_vcon_get_attr_u
 - Virtual Console, [509](#)
- L4_vcon_i_flags
 - Virtual Console, [507](#)
- L4_VCON_ICANON
 - Virtual Console, [508](#)
- L4_VCON_ICRNL
 - Virtual Console, [508](#)
- L4_VCON_IGNCR
 - Virtual Console, [508](#)
- L4_VCON_INLCR
 - Virtual Console, [508](#)
- L4_vcon_l_flags
 - Virtual Console, [508](#)
- L4_vcon_o_flags
 - Virtual Console, [508](#)
- L4_VCON_OCRNL
 - Virtual Console, [508](#)
- L4_VCON_ONLCR
 - Virtual Console, [508](#)
- L4_VCON_ONLRET
 - Virtual Console, [508](#)
- L4_vcon_ops
 - L4 IPC Opcodes, [289](#)
- l4_vcon_read
 - Virtual Console, [510](#)
- L4_vcon_read_flags
 - vcon.h, [2217](#)
- L4_VCON_READ_OP
 - L4 IPC Opcodes, [289](#)
- L4_VCON_READ_SIZE
 - Virtual Console, [509](#)
- L4_VCON_READ_SIZE_MASK
 - vcon.h, [2217](#)
- L4_VCON_READ_STAT_BREAK
 - vcon.h, [2217](#)
- L4_VCON_READ_STAT_DONE
 - vcon.h, [2217](#)
- l4_vcon_read_u
 - Virtual Console, [511](#)
- l4_vcon_read_with_flags
 - Virtual Console, [512](#)
- l4_vcon_send
 - Virtual Console, [513](#)
- l4_vcon_send_u
 - Virtual Console, [514](#)
- l4_vcon_set_attr
 - Virtual Console, [515](#)
- L4_VCON_SET_ATTR_OP
 - L4 IPC Opcodes, [289](#)
- l4_vcon_set_attr_u
 - Virtual Console, [515](#)
- L4_vcon_size_consts
 - Virtual Console, [508](#)
- l4_vcon_write
 - Virtual Console, [516](#)
- L4_VCON_WRITE_OP
 - L4 IPC Opcodes, [289](#)
- L4_VCON_WRITE_SIZE
 - Virtual Console, [509](#)
- l4_vcon_write_u
 - Virtual Console, [517](#)

- L4_VCPU_F_DEBUG_EXC
 - vCPU API, [525](#)
- L4_VCPU_F_EXCEPTIONS
 - vCPU API, [525](#)
- L4_VCPU_F_FPU_ENABLED
 - vCPU API, [525](#)
- L4_VCPU_F_IRQ
 - vCPU API, [525](#)
- L4_VCPU_F_PAGE_FAULTS
 - vCPU API, [525](#)
- L4_VCPU_F_USER_MODE
 - vCPU API, [525](#)
- l4_vcpu_ipc_regs_t, [1202](#)
- L4_VCPU_OFFSET_EXT_INFOS
 - vCPU API, [525](#)
- L4_VCPU_OFFSET_EXT_STATE
 - vCPU API, [525](#)
- l4_vcpu_regs_t, [1203](#)
 - ax, [1204](#)
 - bp, [1205](#)
 - bx, [1205](#)
 - cx, [1205](#)
 - di, [1205](#)
 - dx, [1206](#)
 - si, [1206](#)
- L4_VCPU_SF_IRQ_PENDING
 - vCPU API, [525](#)
- L4_vcpu_state_flags
 - vCPU API, [525](#)
- L4_vcpu_state_offset
 - vCPU API, [525](#)
- l4_vcpu_state_t, [1207](#)
- L4_vcpu_sticky_flags
 - vCPU API, [525](#)
- l4_vhw_descriptor, [1209](#)
- l4_vhw_entry, [1211](#)
- l4_vhw_entry_type
 - Fiasco-UX Virtual devices, [207](#)
- L4_virtio_cmd
 - L4 VIRTIO Transport Layer, [295](#)
- L4_virtio_irq_status
 - L4 VIRTIO Transport Layer, [295](#)
- L4_virtio_opcodes
 - L4 VIRTIO Transport Layer, [296](#)
- l4_vm_svm_vmcb_control_area, [1212](#)
- l4_vm_svm_vmcb_state_save_area, [1213](#)
- l4_vm_svm_vmcb_state_save_area_seg, [1214](#)
- l4_vm_svm_vmcb_t, [1215](#)
- l4_vm_tz_state, [1216](#)
- L4_VM_VMX_BASIC_REG
 - VM API for VMX, [484](#)
- L4_vm_vmx_caps_regs
 - VM API for VMX, [483](#)
- l4_vm_vmx_clear
 - VM API for VMX, [484](#)
- L4_VM_VMX_CR0_FIXED0_REG
 - VM API for VMX, [484](#)
- L4_VM_VMX_CR0_FIXED1_REG
 - VM API for VMX, [484](#)
- L4_VM_VMX_CR4_FIXED0_REG
 - VM API for VMX, [484](#)
- L4_VM_VMX_CR4_FIXED1_REG
 - VM API for VMX, [484](#)
- l4_vm_vmx_dfl1_regs
 - VM API for VMX, [484](#)
- L4_VM_VMX_ENTRY_CTLS_DFL1_REG
 - VM API for VMX, [484](#)
- L4_VM_VMX_EPT_VPID_CAP_REG
 - VM API for VMX, [484](#)
- L4_VM_VMX_EXIT_CTLS_DFL1_REG
 - VM API for VMX, [484](#)
- l4_vm_vmx_field_len
 - VM API for VMX, [485](#)
- l4_vm_vmx_field_order
 - VM API for VMX, [486](#)
- l4_vm_vmx_get_caps
 - VM API for VMX, [486](#)
- l4_vm_vmx_get_caps_default1
 - VM API for VMX, [487](#)
- l4_vm_vmx_get_cr2_index
 - VM API for VMX, [487](#)
- L4_VM_VMX_MISC_REG
 - VM API for VMX, [484](#)
- L4_VM_VMX_NUM_CAPS_REGS
 - VM API for VMX, [484](#)
- L4_VM_VMX_NUM_DFL1_REGS
 - VM API for VMX, [484](#)
- L4_VM_VMX_PINBASED_CTLS_DFL1_REG
 - VM API for VMX, [484](#)
- L4_VM_VMX_PROCBASED_CTLS2_REG
 - VM API for VMX, [484](#)
- L4_VM_VMX_PROCBASED_CTLS_DFL1_REG
 - VM API for VMX, [484](#)
- l4_vm_vmx_ptr_load
 - VM API for VMX, [488](#)
- l4_vm_vmx_read
 - VM API for VMX, [488](#)
- l4_vm_vmx_read_16
 - VM API for VMX, [489](#)
- l4_vm_vmx_read_32
 - VM API for VMX, [490](#)
- l4_vm_vmx_read_64
 - VM API for VMX, [490](#)
- l4_vm_vmx_read_nat
 - VM API for VMX, [491](#)
- L4_VM_VMX_TRUE_ENTRY_CTLS_REG
 - VM API for VMX, [484](#)
- L4_VM_VMX_TRUE_EXIT_CTLS_REG
 - VM API for VMX, [484](#)
- L4_VM_VMX_TRUE_PINBASED_CTLS_REG
 - VM API for VMX, [484](#)
- L4_VM_VMX_TRUE_PROCBASED_CTLS_REG
 - VM API for VMX, [484](#)
- L4_VM_VMX_VMCS_CR2
 - VM API for VMX, [483](#)
- L4_VM_VMX_VMCS_ENUM_REG

- VM API for VMX, [484](#)
- L4_VM_VMX_VMCS_MSR_CSTAR
 - VM API for VMX, [483](#)
- L4_VM_VMX_VMCS_MSR_KERNEL_GS_BASE
 - VM API for VMX, [483](#)
- L4_VM_VMX_VMCS_MSR_LSTAR
 - VM API for VMX, [483](#)
- L4_VM_VMX_VMCS_MSR_STAR
 - VM API for VMX, [483](#)
- L4_VM_VMX_VMCS_MSR_SYSCALL_MASK
 - VM API for VMX, [483](#)
- L4_VM_VMX_VMCS_MSR_TSC_AUX
 - VM API for VMX, [483](#)
- L4_VM_VMX_VMCS_XCR0
 - VM API for VMX, [483](#)
- l4_vm_vmx_write
 - VM API for VMX, [492](#)
- l4_vm_vmx_write_16
 - VM API for VMX, [493](#)
- l4_vm_vmx_write_32
 - VM API for VMX, [494](#)
- l4_vm_vmx_write_64
 - VM API for VMX, [494](#)
- l4_vm_vmx_write_nat
 - VM API for VMX, [495](#)
- L4_WHOLE_ADDRESS_SPACE
 - Flex pages, [210](#)
- L4_WHOLE_IOADDRESS_SPACE
 - Flex pages, [212](#)
- L4IO_DEVICE_ANY
 - IO interface, [228](#)
- L4IO_DEVICE_INVALID
 - IO interface, [228](#)
- L4IO_DEVICE_OTHER
 - IO interface, [228](#)
- L4IO_DEVICE_PCI
 - IO interface, [228](#)
- l4io_device_types_t
 - IO interface, [227](#)
- L4IO_DEVICE_USB
 - IO interface, [228](#)
- l4io_get_root_device
 - io.h, [1795](#)
- l4io_has_resource
 - IO interface, [229](#)
- l4io_iomem_flags_t
 - IO interface, [228](#)
- l4io_iterate_devices
 - io.h, [1795](#)
- l4io_lookup_device
 - IO interface, [229](#)
- l4io_lookup_resource
 - IO interface, [229](#)
- L4IO_MEM_CACHED
 - IO interface, [228](#)
- L4IO_MEM_EAGER_MAP
 - IO interface, [228](#)
- L4IO_MEM_NONCACHED
 - IO interface, [228](#)
- L4IO_MEM_USE_MTRR
 - IO interface, [228](#)
- L4IO_MEM_USE_RESERVED_AREA
 - IO interface, [228](#)
- l4io_release_iomem
 - IO interface, [230](#)
- l4io_release_ioport
 - IO interface, [230](#)
- l4io_request_all_ioports
 - io.h, [1796](#)
- l4io_request_icu
 - io.h, [1796](#)
- l4io_request_iomem
 - IO interface, [231](#)
- l4io_request_iomem_region
 - IO interface, [231](#)
- l4io_request_ioport
 - IO interface, [232](#)
- l4io_request_resource_iomem
 - IO interface, [233](#)
- L4IO_RESOURCE_ANY
 - IO interface, [228](#)
- L4IO_RESOURCE_INVALID
 - IO interface, [228](#)
- L4IO_RESOURCE_IRQ
 - IO interface, [228](#)
- L4IO_RESOURCE_MEM
 - IO interface, [228](#)
- L4IO_RESOURCE_PORT
 - IO interface, [228](#)
- l4io_resource_t
 - IO interface, [226](#)
- l4io_resource_types_t
 - IO interface, [228](#)
- l4irq_attach
 - Interface using direct functionality., [261](#)
- l4irq_attach_cap
 - Interface using direct functionality., [258](#)
- l4irq_attach_cap_ft
 - Interface using direct functionality., [259](#)
- l4irq_attach_ft
 - Interface using direct functionality., [261](#)
- l4irq_attach_thread
 - Interface using direct functionality., [262](#)
- l4irq_attach_thread_cap
 - Interface using direct functionality., [259](#)
- l4irq_attach_thread_cap_ft
 - Interface using direct functionality., [260](#)
- l4irq_attach_thread_ft
 - Interface using direct functionality., [262](#)
- l4irq_detach
 - Interface using direct functionality., [263](#)
- l4irq_release
 - Interface for asynchronous ISR handlers., [257](#)
- l4irq_request
 - Interface for asynchronous ISR handlers., [257](#)
- l4irq_request_cap

- Interface for asynchronous ISR handlers with a given IRQ capability., 255
- l4irq_unmask
 - Interface using direct functionality., 263
- l4irq_unmask_and_wait_any
 - Interface using direct functionality., 264
- l4irq_wait
 - Interface using direct functionality., 264
- l4irq_wait_any
 - Interface using direct functionality., 264
- l4la_alloc
 - list_alloc.h, 2300
- l4la_avail
 - list_alloc.h, 2300
- l4la_dump
 - list_alloc.h, 2300
- l4la_free
 - list_alloc.h, 2300
- l4la_init
 - list_alloc.h, 2301
- L4Re, 564
 - chkcap, 569
 - chkipc, 570
 - chksys, 571, 573, 574
 - make_shared_cap, 575
 - make_shared_del_cap, 576
 - make_unique_cap, 577
 - make_unique_del_cap, 577
 - Shared_cap, 566
 - shared_cap, 566
 - Shared_del_cap, 567
 - shared_del_cap, 567
 - throw_error, 578
 - Unique_cap, 568
 - unique_cap, 568
 - Unique_del_cap, 568
 - unique_del_cap, 569
- L4Re C Interface, 314
 - l4re_inhibitor_acquire, 316
- L4Re C++ Interface, 317
- L4Re Capability API, 319
- L4Re ELF Auxiliary Information, 319
- L4Re Protocol identifiers, 320
- L4Re Util C Interface, 321
- L4Re Util C++ Interface, 321
- L4Re::Cap_alloc, 1217
 - alloc, 1218
 - free, 1219
 - get_cap_alloc, 1220
- L4Re::Console, 1221
- L4Re::Dataspace, 1223
 - allocate, 1225
 - clear, 1226
 - copy_in, 1226
 - flags, 1228
 - info, 1228
 - map, 1229
 - map_region, 1230
 - size, 1231
- L4Re::Dataspace::F, 1231
 - Bufferable, 1232
 - Cacheable, 1232
 - Caching_mask, 1232
 - Caching_shift, 1232
 - Flags, 1232
 - Normal, 1232
 - R, 1232
 - Rights_mask, 1232
 - Ro, 1232
 - RW, 1232
 - Uncacheable, 1232
 - W, 1232
- L4Re::Dataspace::Stats, 1233
- L4Re::Debug_obj, 1233
 - debug, 1236
- L4Re::Dma_space, 1236
 - associate, 1239
 - Attribute, 1238
 - Attributes, 1237
 - Bidirectional, 1238
 - Coherent, 1239
 - Direction, 1238
 - disassociate, 1239
 - From_device, 1238
 - map, 1240
 - No_sync, 1238
 - None, 1238
 - Phys_space, 1239
 - Space_attr, 1239
 - To_device, 1238
 - unmap, 1240
- L4Re::Env, 1241
 - env, 1244
 - factory, 1244, 1245
 - first_free_cap, 1245
 - first_free_utcb, 1246
 - get, 1246
 - get_cap, 1247
 - initial_caps, 1248
 - log, 1249
 - main_thread, 1249, 1250
 - mem_alloc, 1250
 - parent, 1251
 - rm, 1251, 1252
 - scheduler, 1252
 - task, 1253
 - utcb_area, 1253
- L4Re::Event, 1254
 - get_buffer, 1257
- L4Re::Event_buffer_t< PAYLOAD >, 1257
 - Event_buffer_t, 1259
 - next, 1259
 - put, 1259
- L4Re::Event_buffer_t< PAYLOAD >::Event, 1260
- L4Re::Inhibitor, 1261
 - acquire, 1264

- Name_max, [1264](#)
- next_lock_info, [1265](#)
- release, [1266](#)
- L4Re::Log, [1266](#)
 - print, [1269](#)
 - println, [1269](#)
- L4Re::Mem_alloc, [1269](#)
 - alloc, [1273](#)
 - Continuous, [1272](#)
 - free, [1273](#)
 - Mem_alloc_flags, [1272](#)
 - Pinned, [1272](#)
 - Super_pages, [1272](#)
- L4Re::Mmio_space, [1274](#)
 - Access_width, [1277](#)
 - mmio_read, [1277](#)
 - mmio_write, [1278](#)
 - Wd_16bit, [1277](#)
 - Wd_32bit, [1277](#)
 - Wd_64bit, [1277](#)
 - Wd_8bit, [1277](#)
- L4Re::Namespace, [1279](#)
 - Link, [1282](#)
 - Overwrite, [1282](#)
 - Partly_resolved, [1281](#)
 - query, [1282](#), [1283](#)
 - Query_result_flags, [1281](#)
 - Register_flags, [1281](#)
 - register_obj, [1284](#)
 - Ro, [1282](#)
 - Rs, [1282](#)
 - Rw, [1282](#)
 - Rws, [1282](#)
 - Strong, [1282](#)
 - Trusted, [1282](#)
 - unlink, [1284](#)
- L4Re::Ned::Cmd_control, [1285](#)
 - execute, [1286](#)
- L4Re::Parent, [1287](#)
 - signal, [1290](#)
- L4Re::Random, [1290](#)
 - get_random, [1293](#)
- L4Re::Rm, [1294](#)
 - Area, [1297](#)
 - attach, [1298](#), [1299](#)
 - Caching_shift, [1298](#)
 - detach, [1300–1302](#)
 - Detach_again, [1298](#)
 - Detach_exact, [1298](#)
 - Detach_flags, [1297](#)
 - Detach_keep, [1298](#)
 - Detach_overlap, [1298](#)
 - Detach_result, [1298](#)
 - Detached_ds, [1298](#)
 - find, [1302](#)
 - free_area, [1303](#)
 - get_areas, [1304](#)
 - get_regions, [1304](#)
 - Kept_ds, [1298](#)
 - Region, [1297](#)
 - Region_flag_shifts, [1298](#)
 - reserve_area, [1305](#), [1306](#)
 - Split_ds, [1298](#)
- L4Re::Rm::F, [1307](#)
 - Attach_flags, [1307](#)
 - Attach_mask, [1308](#)
 - Cache_buffered, [1308](#)
 - Cache_normal, [1308](#)
 - Cache_uncached, [1308](#)
 - Caching_mask, [1308](#)
 - Detach_free, [1308](#)
 - Ds_map_mask, [1308](#)
 - Eager_map, [1308](#)
 - In_area, [1308](#)
 - Pager, [1308](#)
 - Region_flags, [1308](#)
 - Region_flags_mask, [1308](#)
 - Reserved, [1308](#)
 - Rights_mask, [1308](#)
 - Search_addr, [1308](#)
- L4Re::Rm::Range, [1308](#)
- L4Re::Smart_cap_auto< Unmap_flags >, [1309](#)
- L4Re::Smart_count_cap< Unmap_flags >, [1310](#)
- L4Re::Util, [579](#)
 - cap_alloc, [592](#)
 - kumem_alloc, [588](#)
 - make_ref_cap, [588](#)
 - make_ref_del_cap, [589](#)
 - make_shared_cap, [589](#)
 - make_shared_del_cap, [590](#)
 - make_unique_cap, [590](#)
 - make_unique_del_cap, [591](#)
 - Shared_cap, [581](#)
 - shared_cap, [583](#)
 - Shared_del_cap, [584](#)
 - shared_del_cap, [584](#)
 - Unique_cap, [585](#)
 - unique_cap, [586](#)
 - Unique_del_cap, [586](#)
 - unique_del_cap, [587](#)
- L4Re::Util::Br_manager, [1311](#)
 - alloc_buffer_demand, [1314](#)
 - get_rcv_cap, [1314](#)
 - realloc_rcv_cap, [1315](#)
 - set_rcv_cap_flags, [1315](#)
- L4Re::Util::Br_manager_hooks, [1316](#)
- L4Re::Util::Br_manager_timeout_hooks, [1318](#)
- L4Re::Util::Cap_alloc_base, [1321](#)
- L4Re::Util::Counter< COUNTER >, [1322](#)
- L4Re::Util::Counting_cap_alloc< COUNTERTYPE >, [1323](#)
 - alloc, [1325](#), [1326](#)
 - Counting_cap_alloc, [1325](#)
 - free, [1327](#)
 - release, [1328](#)
 - setup, [1329](#)

- take, [1331](#)
- L4Re::Util::Dataspace_svr, [1332](#)
 - allocate, [1333](#)
 - clear, [1333](#)
 - copy, [1334](#)
 - is_static, [1334](#)
 - map, [1335](#)
 - map_hook, [1335](#)
 - page_shift, [1336](#)
 - release, [1336](#)
 - take, [1337](#)
- L4Re::Util::Event_buffer_consumer_t< PAYLOAD >, [1337](#)
 - foreach_available_event, [1340](#)
 - process, [1340](#)
- L4Re::Util::Event_buffer_t< PAYLOAD >, [1341](#)
 - attach, [1343](#)
 - buf, [1344](#)
 - detach, [1344](#)
- L4Re::Util::Event_svr< SVR >, [1345](#)
- L4Re::Util::Event_t< PAYLOAD >, [1347](#)
 - buffer, [1348](#)
 - init, [1349](#)
 - init_poll, [1350](#)
 - irq, [1351](#)
 - Mode, [1348](#)
 - Mode_irq, [1348](#)
 - Mode_polling, [1348](#)
- L4Re::Util::Item_alloc_base, [1351](#)
- L4Re::Util::Names::Name, [1352](#)
- L4Re::Util::Names::Name_space, [1353](#)
 - alloc_dynamic_entry, [1355](#)
 - copy_receive_cap, [1355](#)
 - free_capability, [1355](#)
 - free_dynamic_entry, [1356](#)
 - free_epiface, [1356](#)
 - get_epiface, [1356](#)
- L4Re::Util::Object_registry, [1357](#)
 - Object_registry, [1359](#)
 - register_irq_obj, [1360](#)
 - register_obj, [1360](#), [1361](#)
 - unregister_obj, [1362](#)
- L4Re::Util::Ref_cap< T >, [1363](#)
- L4Re::Util::Ref_del_cap< T >, [1364](#)
- L4Re::Util::Registry_server< LOOP_HOOKS >, [1365](#)
 - Registry_server, [1368](#)
- L4Re::Util::Smart_cap_auto< Unmap_flags >, [1369](#)
- L4Re::Util::Smart_count_cap< Unmap_flags >, [1370](#)
- L4Re::Util::Vcon_svr< SVR >, [1371](#)
- L4Re::Util::Video::Goos_svr, [1372](#)
 - get_fb, [1373](#)
 - init_infos, [1373](#)
 - refresh, [1374](#)
 - screen_info, [1374](#)
 - view_info, [1375](#)
- L4Re::Vfs, [592](#)
- L4Re::Vfs::Be_file, [1375](#)
 - data_space, [1379](#)
 - fstat64, [1379](#)
 - unlock_all_locks, [1379](#)
- L4Re::Vfs::Be_file_system, [1380](#)
 - ~Be_file_system, [1382](#)
 - Be_file_system, [1381](#)
 - type, [1382](#)
- L4Re::Vfs::Directory, [1383](#)
 - faccessat, [1385](#)
 - link, [1385](#)
 - mkdir, [1385](#)
 - rename, [1386](#)
 - rmdir, [1386](#)
 - symlink, [1387](#)
 - unlink, [1387](#)
- L4Re::Vfs::File, [1388](#)
- L4Re::Vfs::File_system, [1390](#)
 - mount, [1391](#)
 - type, [1392](#)
- L4Re::Vfs::Fs, [1392](#)
 - alloc_fd, [1395](#)
 - free_fd, [1395](#)
 - get_file, [1395](#)
 - mount, [1396](#)
 - set_fd, [1396](#)
- L4Re::Vfs::Generic_file, [1397](#)
 - fchmod, [1399](#)
 - fstat64, [1400](#)
 - get_status_flags, [1400](#)
 - set_status_flags, [1400](#)
 - unlock_all_locks, [1401](#)
- L4Re::Vfs::Mman, [1402](#)
- L4Re::Vfs::Ops, [1403](#)
- L4Re::Vfs::Regular_file, [1406](#)
 - data_space, [1408](#)
 - fdatasync, [1408](#)
 - fsync, [1408](#)
 - ftruncate64, [1408](#)
 - get_lock, [1409](#)
 - lseek64, [1409](#)
 - readv, [1409](#)
 - set_lock, [1410](#)
 - writev, [1410](#)
- L4Re::Vfs::Special_file, [1411](#)
 - ioctl, [1413](#)
- L4Re::Video::Color_component, [1414](#)
 - Color_component, [1415](#)
 - dump, [1415](#)
 - get, [1415](#)
 - operator==, [1416](#)
 - set, [1416](#)
 - shift, [1416](#)
 - size, [1417](#)
- L4Re::Video::Goos, [1418](#)
 - create_buffer, [1421](#)
 - create_view, [1421](#)
 - delete_buffer, [1421](#)
 - delete_view, [1422](#)
 - F_auto_refresh, [1420](#)

- F_dynamic_buffers, [1420](#)
- F_dynamic_views, [1420](#)
- F_pointer, [1420](#)
- Flags, [1420](#)
- get_static_buffer, [1422](#)
- info, [1423](#)
- view, [1423](#)
- L4Re::Video::Goos::Info, [1424](#)
- L4Re::Video::Pixel_info, [1425](#)
 - a, [1427](#), [1428](#)
 - b, [1428](#)
 - bits_per_pixel, [1429](#)
 - bytes_per_pixel, [1429](#)
 - dump, [1430](#)
 - g, [1430](#)
 - has_alpha, [1431](#)
 - operator==, [1431](#)
 - Pixel_info, [1426](#), [1427](#)
 - r, [1432](#)
- L4Re::Video::View, [1432](#)
 - F_above, [1434](#)
 - F_dyn_allocated, [1434](#)
 - F_flags_mask, [1434](#)
 - F_fully_dynamic, [1434](#)
 - F_none, [1434](#)
 - F_set_background, [1434](#)
 - F_set_buffer, [1434](#)
 - F_set_buffer_offset, [1434](#)
 - F_set_bytes_per_line, [1434](#)
 - F_set_flags, [1434](#)
 - F_set_pixel, [1434](#)
 - F_set_position, [1434](#)
 - Flags, [1434](#)
 - info, [1435](#)
 - refresh, [1435](#)
 - set_info, [1436](#)
 - set_viewport, [1436](#)
 - stack, [1437](#)
 - V_flags, [1434](#)
- L4Re::Video::View::Info, [1437](#)
- l4re_aux_t, [1440](#)
- l4re_debug_obj_debug
 - Debug interface, [166](#)
- l4re_dma_space_associate
 - DMA Space Interface, [160](#)
- L4RE_DMA_SPACE_BIDIRECTIONAL
 - dma_space.h, [1884](#)
- L4RE_DMA_SPACE_COHERENT
 - dma_space.h, [1885](#)
- l4re_dma_space_direction
 - dma_space.h, [1884](#)
- l4re_dma_space_disassociate
 - DMA Space Interface, [161](#)
- L4RE_DMA_SPACE_FROM_DEVICE
 - dma_space.h, [1884](#)
- l4re_dma_space_map
 - DMA Space Interface, [161](#)
- L4RE_DMA_SPACE_NONE
 - dma_space.h, [1884](#)
- L4RE_DMA_SPACE_PHYS_SPACE
 - dma_space.h, [1885](#)
- l4re_dma_space_space_attrs
 - dma_space.h, [1884](#)
- l4re_dma_space_t
 - DMA Space Interface, [160](#)
- L4RE_DMA_SPACE_TO_DEVICE
 - dma_space.h, [1884](#)
- l4re_dma_space_unmap
 - DMA Space Interface, [162](#)
- l4re_ds_allocate
 - Dataspace interface, [164](#)
- l4re_ds_clear
 - Dataspace interface, [164](#)
- l4re_ds_copy_in
 - Dataspace interface, [165](#)
- L4RE_DS_F_BUFFERABLE
 - Dataspace interface, [164](#)
- L4RE_DS_F_CACHEABLE
 - Dataspace interface, [164](#)
- L4RE_DS_F_CACHING_MASK
 - Dataspace interface, [164](#)
- L4RE_DS_F_CACHING_SHIFT
 - Dataspace interface, [164](#)
- L4RE_DS_F_NORMAL
 - Dataspace interface, [164](#)
- L4RE_DS_F_UNCACHEABLE
 - Dataspace interface, [164](#)
- l4re_ds_flags
 - Dataspace interface, [165](#)
- l4re_ds_info
 - Dataspace interface, [165](#)
- l4re_ds_map_flags
 - Dataspace interface, [164](#)
- l4re_ds_size
 - Dataspace interface, [166](#)
- l4re_ds_stats_t, [1441](#)
- L4RE_ELF_AUX_ELEM
 - elf_aux.h, [1930](#)
- L4RE_ELF_AUX_ELEM_T
 - elf_aux.h, [1930](#)
- l4re_elf_aux_mword_t, [1441](#)
- l4re_elf_aux_t, [1442](#)
- L4RE_ELF_AUX_T_KIP_ADDR
 - elf_aux.h, [1931](#)
- L4RE_ELF_AUX_T_NONE
 - elf_aux.h, [1931](#)
- L4RE_ELF_AUX_T_STACK_ADDR
 - elf_aux.h, [1931](#)
- L4RE_ELF_AUX_T_STACK_SIZE
 - elf_aux.h, [1931](#)
- L4RE_ELF_AUX_T_VMA
 - elf_aux.h, [1931](#)
- l4re_elf_aux_vma_t, [1443](#)
- l4re_env
 - Initial Environment, [250](#)
- l4re_env_cap_entry_t, [1444](#)

- flags, 1445
- l4re_env_cap_entry_t, 1444
- l4re_env_get_cap
 - Initial Environment, 251
- l4re_env_get_cap_e
 - Initial Environment, 251
- l4re_env_get_cap_l
 - Initial Environment, 252
- l4re_env_t, 1446
 - env.h, 1936
- l4re_event_get_axis_info
 - Event interface, 182
- l4re_event_get_buffer
 - Event interface, 182
- l4re_event_get_num_streams
 - Event interface, 183
- l4re_event_get_stream_info
 - Event interface, 183
- l4re_event_get_stream_info_for_id
 - Event interface, 184
- l4re_event_t, 1447
- l4re_inhibitor_acquire
 - L4Re C Interface, 316
- l4re_kip
 - Initial Environment, 253
- l4re_log_print
 - Log interface, 339
- l4re_log_print_srv
 - Log interface, 339
- l4re_log_printn
 - Log interface, 340
- l4re_log_printn_srv
 - Log interface, 341
- l4re_ma_alloc
 - Memory allocator, 344
- l4re_ma_alloc_align
 - Memory allocator, 345
- l4re_ma_alloc_align_srv
 - Memory allocator, 346
- l4re_ma_flags
 - Memory allocator, 343
- l4re_ma_free
 - Memory allocator, 347
- l4re_ma_free_srv
 - Memory allocator, 348
- l4re_ns_query_srv
 - Namespace interface, 380
- l4re_ns_query_to_srv
 - Namespace interface, 381
- l4re_ns_register_flags
 - Namespace interface, 380
- l4re_ns_register_obj_srv
 - Namespace interface, 382
- L4RE_PROTO_DATASPACE
 - protocols.h, 1968
- L4RE_PROTO_DEBUG
 - protocols.h, 1968
- L4RE_PROTO_DMA_SPACE
 - protocols.h, 1968
- L4RE_PROTO_EVENT
 - protocols.h, 1968
- L4RE_PROTO_GOOS
 - protocols.h, 1968
- L4RE_PROTO_INHIBITOR
 - protocols.h, 1968
- L4RE_PROTO_MMIO_SPACE
 - protocols.h, 1968
- L4RE_PROTO_NAMESPACE
 - protocols.h, 1968
- L4RE_PROTO_PARENT
 - protocols.h, 1968
- L4RE_PROTO_RM
 - protocols.h, 1968
- L4RE_PROTO_RSVD_1
 - protocols.h, 1968
- L4re_protocols
 - protocols.h, 1968
- l4re_rm_attach
 - Region map interface, 407
- l4re_rm_attach_srv
 - Region map interface, 408
- L4RE_RM_CACHING_SHIFT
 - Region map interface, 406
- l4re_rm_detach
 - Region map interface, 409
- l4re_rm_detach_ds
 - Region map interface, 410
- l4re_rm_detach_ds_unmap
 - Region map interface, 410
- l4re_rm_detach_srv
 - Region map interface, 411
- l4re_rm_detach_unmap
 - Region map interface, 412
- L4RE_RM_F_ATTACH_FLAGS
 - Region map interface, 406
- L4RE_RM_F_CACHE_BUFFERED
 - Region map interface, 406
- L4RE_RM_F_CACHE_NORMAL
 - Region map interface, 406
- L4RE_RM_F_CACHE_UNCACHED
 - Region map interface, 406
- L4RE_RM_F_CACHING
 - Region map interface, 406
- L4RE_RM_F_EAGER_MAP
 - Region map interface, 406
- L4RE_RM_F_IN_AREA
 - Region map interface, 406
- L4RE_RM_F_NO_ALIAS
 - Region map interface, 406
- L4RE_RM_F_PAGER
 - Region map interface, 406
- L4RE_RM_F_R
 - Region map interface, 406
- L4RE_RM_F_RESERVED
 - Region map interface, 406
- L4RE_RM_F_SEARCH_ADDR

- Region map interface, [406](#)
- `l4re_rm_find`
 - Region map interface, [413](#)
- `l4re_rm_find_srv`
 - Region map interface, [413](#)
- `l4re_rm_flags_values`
 - Region map interface, [406](#)
- `l4re_rm_free_area`
 - Region map interface, [414](#)
- `l4re_rm_free_area_srv`
 - Region map interface, [414](#)
- `L4RE_RM_REGION_FLAGS`
 - Region map interface, [406](#)
- `l4re_rm_reserve_area`
 - Region map interface, [415](#)
- `l4re_rm_reserve_area_srv`
 - Region map interface, [415](#)
- `l4re_rm_show_lists`
 - Region map interface, [416](#)
- `l4re_util_cap_last`
 - Capability allocator, [146](#)
- `l4re_util_kumem_alloc`
 - `kumem_alloc.h`, [1902](#)
- `l4re_video_color_component_t`, [1448](#)
- `l4re_video_goos_create_buffer`
 - Video API, [499](#)
- `l4re_video_goos_create_view`
 - Video API, [500](#)
- `l4re_video_goos_delete_buffer`
 - Video API, [500](#)
- `l4re_video_goos_delete_view`
 - Video API, [502](#)
- `l4re_video_goos_get_static_buffer`
 - Video API, [502](#)
- `l4re_video_goos_get_view`
 - Video API, [502](#)
- `l4re_video_goos_info`
 - Video API, [503](#)
- `l4re_video_goos_info_flags_t`
 - Video API, [499](#)
- `l4re_video_goos_info_t`, [1449](#)
- `l4re_video_goos_refresh`
 - Video API, [503](#)
- `l4re_video_pixel_info_t`, [1450](#)
- `l4re_video_view_get_info`
 - Video API, [504](#)
- `l4re_video_view_info_flags_t`
 - Video API, [499](#)
- `l4re_video_view_info_t`, [1451](#)
- `l4re_video_view_refresh`
 - Video API, [504](#)
- `l4re_video_view_set_info`
 - Video API, [504](#)
- `l4re_video_view_set_viewport`
 - Video API, [505](#)
- `l4re_video_view_stack`
 - Video API, [505](#)
- `l4re_video_view_t`, [1453](#)
- Video API, [498](#)
- `l4shmc_add_chunk`
 - Chunks, [148](#)
- `l4shmc_add_signal`
 - Signals, [429](#)
- `l4shmc_area_overhead`
 - Shared Memory Library, [424](#)
- `l4shmc_area_size`
 - Shared Memory Library, [425](#)
- `l4shmc_area_size_free`
 - Shared Memory Library, [425](#)
- `l4shmc_attach`
 - Shared Memory Library, [426](#)
- `l4shmc_attach_signal`
 - Signals, [430](#)
- `l4shmc_attach_signal_to`
 - Signals, [430](#)
- `l4shmc_attach_to`
 - Shared Memory Library, [426](#)
- `l4shmc_check_magic`
 - Signals, [431](#)
- `l4shmc_chunk_capacity`
 - Chunks, [148](#)
- `l4shmc_chunk_consumed`
 - Consumer, [156](#)
- `l4shmc_chunk_overhead`
 - Shared Memory Library, [427](#)
- `l4shmc_chunk_ptr`
 - Chunks, [149](#)
- `l4shmc_chunk_ready`
 - Producer, [401](#)
- `l4shmc_chunk_ready_sig`
 - Producer, [401](#)
- `l4shmc_chunk_signal`
 - Chunks, [149](#)
- `l4shmc_chunk_size`
 - Consumer, [156](#)
- `l4shmc_chunk_try_to_take`
 - Producer, [401](#)
- `l4shmc_connect_chunk_signal`
 - Shared Memory Library, [427](#)
- `l4shmc_create`
 - Shared Memory Library, [427](#)
- `l4shmc_enable_chunk`
 - Consumer, [157](#)
- `l4shmc_enable_signal`
 - Consumer, [152](#)
- `l4shmc_get_chunk`
 - Chunks, [149](#)
- `l4shmc_get_chunk_to`
 - Chunks, [150](#)
- `l4shmc_get_signal_to`
 - Signals, [431](#)
- `l4shmc_is_chunk_clear`
 - Producer, [402](#)
- `l4shmc_is_chunk_ready`
 - Consumer, [157](#)
- `l4shmc_iterate_chunk`

- Chunks, [150](#)
- l4shmc_signal_cap
 - Signals, [432](#)
- l4shmc_trigger
 - Producer, [400](#)
- l4shmc_wait_any
 - Consumer, [153](#)
- l4shmc_wait_any_to
 - Consumer, [153](#)
- l4shmc_wait_any_try
 - Consumer, [154](#)
- l4shmc_wait_chunk
 - Consumer, [158](#)
- l4shmc_wait_chunk_to
 - Consumer, [158](#)
- l4shmc_wait_chunk_try
 - Consumer, [159](#)
- l4shmc_wait_signal
 - Consumer, [154](#)
- l4shmc_wait_signal_to
 - Consumer, [154](#)
- l4shmc_wait_signal_try
 - Consumer, [155](#)
- l4sigma0_debug_dump
 - sigma0.h, [2012](#)
- L4SIGMA0_IPCERROR
 - sigma0.h, [2012](#)
- l4sigma0_map_anypage
 - sigma0.h, [2012](#)
- l4sigma0_map_errstr
 - sigma0.h, [2013](#)
- l4sigma0_map_iomem
 - sigma0.h, [2013](#)
- l4sigma0_map_kip
 - sigma0.h, [2014](#)
- l4sigma0_map_mem
 - sigma0.h, [2014](#)
- l4sigma0_new_client
 - sigma0.h, [2015](#)
- L4SIGMA0_NOFPAGE
 - sigma0.h, [2012](#)
- L4SIGMA0_NOTALIGNED
 - sigma0.h, [2012](#)
- L4SIGMA0_OK
 - sigma0.h, [2012](#)
- l4sigma0_return_flags_t
 - sigma0.h, [2012](#)
- L4SIGMA0_SMALLERFPAGE
 - sigma0.h, [2012](#)
- l4util_add16
 - Atomic Instructions, [108](#)
- l4util_add16_res
 - Atomic Instructions, [108](#)
- l4util_add32
 - Atomic Instructions, [108](#)
- l4util_add32_res
 - Atomic Instructions, [109](#)
- l4util_add8
 - Atomic Instructions, [109](#)
- l4util_add8_res
 - Atomic Instructions, [109](#)
- l4util_and16
 - Atomic Instructions, [110](#)
- l4util_and16_res
 - Atomic Instructions, [110](#)
- l4util_and32
 - Atomic Instructions, [110](#)
- l4util_and32_res
 - Atomic Instructions, [111](#)
- l4util_and8
 - Atomic Instructions, [111](#)
- l4util_and8_res
 - Atomic Instructions, [111](#)
- l4util_atomic_add
 - Atomic Instructions, [112](#)
- l4util_atomic_inc
 - Atomic Instructions, [112](#)
- l4util_backtrace
 - backtrace.h, [2239](#)
- l4util_bsf
 - Bit Manipulation, [130](#)
- l4util_bsr
 - Bit Manipulation, [130](#)
- l4util_btc
 - Bit Manipulation, [131](#)
- l4util_btr
 - Bit Manipulation, [131](#)
- l4util_bts
 - Bit Manipulation, [132](#)
- l4util_clear_bit
 - Bit Manipulation, [133](#)
- l4util_cmpxchg
 - Atomic Instructions, [112](#)
- l4util_cmpxchg16
 - Atomic Instructions, [113](#)
- l4util_cmpxchg32
 - Atomic Instructions, [114](#)
- l4util_cmpxchg64
 - Atomic Instructions, [114](#)
- l4util_cmpxchg8
 - Atomic Instructions, [115](#)
- l4util_complement_bit
 - Bit Manipulation, [133](#)
- l4util_cpu_capabilities
 - cpu.h, [1779](#), [1783](#)
- l4util_cpu_capabilities_nocheck
 - cpu.h, [1779](#), [1783](#)
- l4util_cpu_has_cpuid
 - cpu.h, [1780](#), [1784](#)
- l4util_dec16
 - Atomic Instructions, [115](#)
- l4util_dec16_res
 - Atomic Instructions, [116](#)
- l4util_dec32
 - Atomic Instructions, [116](#)
- l4util_dec32_res

- Atomic Instructions, [116](#)
- l4util_dec8
 - Atomic Instructions, [117](#)
- l4util_dec8_res
 - Atomic Instructions, [117](#)
- l4util_find_first_set_bit
 - Bit Manipulation, [134](#)
- l4util_find_first_zero_bit
 - Bit Manipulation, [134](#)
- l4util_idt_desc_t, [1454](#)
- l4util_idt_header_t, [1455](#)
- l4util_in16
 - port_io.h, [1756](#)
- l4util_in32
 - port_io.h, [1756](#)
- l4util_in8
 - port_io.h, [1757](#)
- l4util_inc16
 - Atomic Instructions, [117](#)
- l4util_inc16_res
 - Atomic Instructions, [117](#)
- l4util_inc32
 - Atomic Instructions, [118](#)
- l4util_inc32_res
 - Atomic Instructions, [118](#)
- l4util_inc8
 - Atomic Instructions, [118](#)
- l4util_inc8_res
 - Atomic Instructions, [119](#)
- l4util_ins16
 - port_io.h, [1757](#)
- l4util_ins32
 - port_io.h, [1758](#)
- l4util_ins8
 - port_io.h, [1758](#)
- l4util_ioport_map
 - port_io.h, [1752](#)
- l4util_irq_acknowledge
 - irq.h, [1869](#), [1872](#)
- l4util_kip_for_each_feature
 - kip.h, [2289](#)
- l4util_kip_kernel_abi_version
 - kip.h, [2290](#)
- l4util_kip_kernel_has_feature
 - kip.h, [2290](#)
- l4util_kip_kernel_is_ux
 - kip.h, [2290](#)
- l4util_l4mod_info, [1456](#)
 - vbe_ctrl_info, [1456](#)
- l4util_l4mod_mod, [1457](#)
- l4util_mb_addr_range_t, [1458](#)
- l4util_mb_apm_t, [1459](#)
- l4util_mb_drive_t, [1460](#)
 - drive_cylinders, [1461](#)
 - drive_mode, [1461](#)
 - drive_number, [1461](#)
- l4util_mb_for_each_mmap_entry
 - mb_info.h, [2306](#)
- l4util_mb_info_t, [1462](#)
- L4UTIL_MB_MEMORY
 - mb_info.h, [2306](#)
- l4util_mb_mod_t, [1463](#)
- l4util_mb_vbe_ctrl_t, [1464](#)
- l4util_mb_vbe_mode_t, [1465](#)
- l4util_micros2l4to
 - util.h, [1733](#), [1736](#)
 - Utility Functions, [478](#)
- l4util_next_power2
 - Bit Manipulation, [134](#)
- l4util_or16
 - Atomic Instructions, [119](#)
- l4util_or16_res
 - Atomic Instructions, [119](#)
- l4util_or32
 - Atomic Instructions, [120](#)
- l4util_or32_res
 - Atomic Instructions, [120](#)
- l4util_or8
 - Atomic Instructions, [120](#)
- l4util_or8_res
 - Atomic Instructions, [121](#)
- l4util_out16
 - port_io.h, [1758](#)
- l4util_out32
 - port_io.h, [1759](#)
- l4util_out8
 - port_io.h, [1759](#)
- l4util_outs16
 - port_io.h, [1760](#)
- l4util_outs32
 - port_io.h, [1760](#)
- l4util_outs8
 - port_io.h, [1760](#)
- l4util_rand
 - Random number support, [403](#)
- l4util_set_bit
 - Bit Manipulation, [135](#)
- l4util_splitlog2_hdl
 - Utility Functions, [478](#)
- l4util_splitlog2_size
 - Utility Functions, [479](#)
- l4util_srand
 - Random number support, [403](#)
- l4util_sub16
 - Atomic Instructions, [121](#)
- l4util_sub16_res
 - Atomic Instructions, [121](#)
- l4util_sub32
 - Atomic Instructions, [122](#)
- l4util_sub32_res
 - Atomic Instructions, [122](#)
- l4util_sub8
 - Atomic Instructions, [123](#)
- l4util_sub8_res
 - Atomic Instructions, [123](#)
- l4util_test_bit

- Bit Manipulation, [135](#)
- l4util_xchg
 - Atomic Instructions, [123](#)
- l4util_xchg16
 - Atomic Instructions, [124](#)
- l4util_xchg32
 - Atomic Instructions, [124](#)
- l4util_xchg8
 - Atomic Instructions, [124](#)
- L4vbus, [593](#)
- L4vbus GPIO functions, [322](#)
 - l4vbus_gpio_config_get, [324](#)
 - l4vbus_gpio_config_pad, [325](#)
 - l4vbus_gpio_config_pull, [325](#)
 - L4vbus_gpio_generic_func, [323](#)
 - l4vbus_gpio_get, [326](#)
 - l4vbus_gpio_multi_config_pad, [327](#)
 - l4vbus_gpio_multi_get, [328](#)
 - l4vbus_gpio_multi_set, [328](#)
 - l4vbus_gpio_multi_setup, [329](#)
 - L4VBUS_GPIO_PIN_PULL_DOWN, [324](#)
 - L4VBUS_GPIO_PIN_PULL_NONE, [324](#)
 - L4VBUS_GPIO_PIN_PULL_UP, [324](#)
 - L4vbus_gpio_pull_modes, [324](#)
 - l4vbus_gpio_set, [330](#)
 - l4vbus_gpio_setup, [331](#)
 - L4VBUS_GPIO_SETUP_INPUT, [323](#)
 - L4VBUS_GPIO_SETUP_IRQ, [323](#)
 - L4VBUS_GPIO_SETUP_OUTPUT, [323](#)
 - l4vbus_gpio_to_irq, [332](#)
- L4vbus PCI functions, [333](#)
 - l4vbus_pci_cfg_read, [333](#)
 - l4vbus_pci_cfg_write, [334](#)
 - l4vbus_pci_irq_enable, [335](#)
 - l4vbus_pciddev_cfg_read, [336](#)
 - l4vbus_pciddev_cfg_write, [336](#)
 - l4vbus_pciddev_irq_enable, [337](#)
- L4vbus::Device, [1468](#)
 - _bus, [1476](#)
 - bus_cap, [1470](#)
 - dev_handle, [1471](#)
 - device, [1471](#)
 - device_by_hid, [1472](#)
 - get_resource, [1473](#)
 - is_compatible, [1474](#)
 - next_device, [1474](#)
 - operator!=, [1475](#)
 - operator==, [1475](#)
- L4vbus::Gpio_module, [1476](#)
 - config_pad, [1479](#)
 - get, [1480](#)
 - pin, [1480](#)
 - set, [1481](#)
 - setup, [1481](#)
- L4vbus::Gpio_module::Pin_slice, [1482](#)
- L4vbus::Gpio_pin, [1483](#)
 - config_get, [1486](#)
 - config_pad, [1487](#)
 - config_pull, [1487](#)
 - get, [1488](#)
 - pin, [1488](#)
 - set, [1489](#)
 - setup, [1489](#)
 - to_irq, [1490](#)
- L4vbus::lcu, [1491](#)
 - Src_dev_handle, [1493](#)
 - Src_types, [1493](#)
- L4vbus::Pci_dev, [1493](#)
 - cfg_read, [1496](#)
 - cfg_write, [1496](#)
 - irq_enable, [1497](#)
- L4vbus::Pci_host_bridge, [1498](#)
 - cfg_read, [1501](#)
 - cfg_write, [1502](#)
 - irq_enable, [1503](#)
- L4vbus::Pm< DEC >, [1504](#)
- L4vbus::Vbus, [1505](#)
 - assign_dma_domain, [1508](#), [1509](#)
 - release_ioport, [1510](#)
 - release_resource, [1511](#)
 - request_ioport, [1511](#)
 - request_resource, [1513](#)
 - root, [1514](#)
- l4vbus_assign_dma_domain
 - L4 Vbus functions, [303](#)
- L4VBUS_DEVICE_F_CHILDREN
 - vbus_types.h, [2338](#)
- l4vbus_device_flags_t
 - vbus_types.h, [2338](#)
- l4vbus_device_t, [1514](#)
- L4vbus_dma_domain_assign_flags
 - L4 Vbus functions, [303](#)
- L4VBUS_DMAD_BIND
 - L4 Vbus functions, [303](#)
- L4VBUS_DMAD_KERNEL_DMA_SPACE
 - L4 Vbus functions, [303](#)
- L4VBUS_DMAD_L4RE_DMA_SPACE
 - L4 Vbus functions, [303](#)
- L4VBUS_DMAD_UNBIND
 - L4 Vbus functions, [303](#)
- l4vbus_get_device
 - L4 Vbus functions, [304](#)
- l4vbus_get_device_by_hid
 - L4 Vbus functions, [305](#)
- l4vbus_get_hid
 - L4 Vbus functions, [306](#)
- l4vbus_get_next_device
 - L4 Vbus functions, [306](#)
- l4vbus_get_resource
 - L4 Vbus functions, [308](#)
- l4vbus_gpio_config_get
 - L4vbus GPIO functions, [324](#)
- l4vbus_gpio_config_pad
 - L4vbus GPIO functions, [325](#)
- l4vbus_gpio_config_pull
 - L4vbus GPIO functions, [325](#)

- L4vbus_gpio_generic_func
 - L4vbus GPIO functions, [323](#)
- l4vbus_gpio_get
 - L4vbus GPIO functions, [326](#)
- l4vbus_gpio_multi_config_pad
 - L4vbus GPIO functions, [327](#)
- l4vbus_gpio_multi_get
 - L4vbus GPIO functions, [328](#)
- l4vbus_gpio_multi_set
 - L4vbus GPIO functions, [328](#)
- l4vbus_gpio_multi_setup
 - L4vbus GPIO functions, [329](#)
- L4VBUS_GPIO_PIN_PULL_DOWN
 - L4vbus GPIO functions, [324](#)
- L4VBUS_GPIO_PIN_PULL_NONE
 - L4vbus GPIO functions, [324](#)
- L4VBUS_GPIO_PIN_PULL_UP
 - L4vbus GPIO functions, [324](#)
- L4vbus_gpio_pull_modes
 - L4vbus GPIO functions, [324](#)
- l4vbus_gpio_set
 - L4vbus GPIO functions, [330](#)
- l4vbus_gpio_setup
 - L4vbus GPIO functions, [331](#)
- L4VBUS_GPIO_SETUP_INPUT
 - L4vbus GPIO functions, [323](#)
- L4VBUS_GPIO_SETUP_IRQ
 - L4vbus GPIO functions, [323](#)
- L4VBUS_GPIO_SETUP_OUTPUT
 - L4vbus GPIO functions, [323](#)
- l4vbus_gpio_to_irq
 - L4vbus GPIO functions, [332](#)
- L4VBUS_ICU_SRC_DEV_HANDLE
 - vbus.h, [2332](#)
- l4vbus_icu_src_types
 - vbus.h, [2332](#)
- L4VBUS_IFACE_SHIFT
 - vbus_interfaces.h, [2335](#)
- l4vbus_iface_type_t
 - vbus_interfaces.h, [2335](#)
- l4vbus_is_compatible
 - L4 Vbus functions, [309](#)
- L4VBUS_NULL
 - vbus.h, [2332](#)
- l4vbus_pci_cfg_read
 - L4vbus PCI functions, [333](#)
- l4vbus_pci_cfg_write
 - L4vbus PCI functions, [334](#)
- l4vbus_pci_irq_enable
 - L4vbus PCI functions, [335](#)
- l4vbus_pcidv_cfg_read
 - L4vbus PCI functions, [336](#)
- l4vbus_pcidv_cfg_write
 - L4vbus PCI functions, [336](#)
- l4vbus_pcidv_irq_enable
 - L4vbus PCI functions, [337](#)
- l4vbus_release_ioport
 - L4 Vbus functions, [309](#)
- l4vbus_release_resource
 - L4 Vbus functions, [310](#)
- l4vbus_request_ioport
 - L4 Vbus functions, [311](#)
- l4vbus_request_resource
 - L4 Vbus functions, [311](#)
- L4VBUS_RESOURCE_BUS
 - vbus_types.h, [2339](#)
- L4VBUS_RESOURCE_DMA_DOMAIN
 - vbus_types.h, [2339](#)
- L4VBUS_RESOURCE_F_MEM_MMIO_READ
 - vbus_types.h, [2338](#)
- L4VBUS_RESOURCE_F_MEM_MMIO_WRITE
 - vbus_types.h, [2338](#)
- l4vbus_resource_flags_t
 - vbus_types.h, [2338](#)
- L4VBUS_RESOURCE_GPIO
 - vbus_types.h, [2339](#)
- L4VBUS_RESOURCE_INVALID
 - vbus_types.h, [2339](#)
- L4VBUS_RESOURCE_IRQ
 - vbus_types.h, [2339](#)
- L4VBUS_RESOURCE_MAX
 - vbus_types.h, [2339](#)
- L4VBUS_RESOURCE_MEM
 - vbus_types.h, [2339](#)
- L4VBUS_RESOURCE_PORT
 - vbus_types.h, [2339](#)
- l4vbus_resource_t, [1515](#)
- l4vbus_resource_type_t
 - vbus_types.h, [2339](#)
- L4VBUS_ROOT_BUS
 - vbus.h, [2332](#)
- l4vbus_subinterface_supported
 - vbus_interfaces.h, [2335](#)
- l4vbus_vicu_get_cap
 - L4 Vbus functions, [312](#)
- L4vcpu::State, [1516](#)
 - add, [1517](#)
 - clear, [1518](#)
 - set, [1518](#)
 - State, [1517](#)
- L4vcpu::Vcpu, [1518](#)
 - cast, [1522](#)
 - entry_ip, [1522](#)
 - entry_sp, [1523](#)
 - ext_alloc, [1523](#)
 - i, [1524](#)
 - irq_disable_save, [1524](#)
 - irq_enable, [1525](#)
 - irq_restore, [1526](#)
 - is_irq_entry, [1526](#)
 - is_page_fault_entry, [1527](#)
 - r, [1527](#)
 - saved_state, [1528](#)
 - state, [1528](#), [1529](#)
 - task, [1529](#)
 - wait_for_event, [1529](#)

- l4vcpu_ext_alloc
 - Extended vCPU support, [188](#)
- l4vcpu_irq_disable
 - vCPU Support Library, [527](#)
- l4vcpu_irq_disable_save
 - vCPU Support Library, [527](#)
- l4vcpu_irq_enable
 - vCPU Support Library, [528](#)
- l4vcpu_irq_restore
 - vCPU Support Library, [529](#)
- l4vcpu_is_irq_entry
 - vCPU Support Library, [530](#)
- l4vcpu_is_page_fault_entry
 - vCPU Support Library, [531](#)
- l4vcpu_print_state
 - vCPU Support Library, [531](#)
- l4vcpu_wait_for_event
 - vCPU Support Library, [532](#)
- L4virtio, [594](#)
- L4virtio::Device, [1530](#)
 - config_queue, [1533](#)
 - device_config, [1534](#)
 - device_notification_irq, [1534](#)
 - register_ds, [1535](#)
 - register_iface, [1536](#)
 - set_status, [1536](#)
- L4virtio::Driver::Block_device, [1537](#)
 - add_block, [1540](#)
 - process_request, [1541](#)
 - process_used_queue, [1542](#)
 - send_request, [1542](#)
 - setup_device, [1543](#)
 - start_request, [1544](#)
- L4virtio::Driver::Block_device::Handle, [1545](#)
- L4virtio::Driver::Device, [1546](#)
 - bind_notification_irq, [1549](#)
 - config_queue, [1550](#)
 - driver_acknowledge, [1551](#)
 - driver_connect, [1551](#)
 - max_queue_size, [1553](#)
 - register_ds, [1554](#)
 - send, [1555](#)
 - send_and_wait, [1556](#)
 - wait, [1557](#)
 - wait_for_next_used, [1558](#)
- L4virtio::Driver::Virtqueue, [1559](#)
 - alloc_descriptor, [1562](#)
 - desc, [1563](#)
 - enqueue_descriptor, [1564](#)
 - find_next_used, [1564](#)
 - free_descriptor, [1565](#)
 - init_queue, [1566](#)
 - initialize_rings, [1567](#)
- L4virtio::Ptr< T >, [1568](#)
 - get, [1570](#)
 - Invalid, [1570](#)
 - Invalid_type, [1570](#)
 - is_valid, [1571](#)
- L4virtio::Svr::Bad_descriptor, [1572](#)
 - Bad_address, [1573](#)
 - Bad_descriptor, [1573](#)
 - Bad_flags, [1573](#)
 - Bad_next, [1573](#)
 - Bad_rights, [1573](#)
 - Bad_size, [1573](#)
 - Error, [1573](#)
 - message, [1574](#)
- L4virtio::Svr::Block_dev_base< Ds_data >, [1574](#)
 - Block_dev_base, [1578](#)
 - finalize_request, [1578](#)
 - get_writeback, [1579](#)
 - process_request, [1579](#)
 - set_blk_size, [1580](#)
 - set_config_wce, [1580](#)
 - set_discard, [1580](#)
 - set_size_max, [1582](#)
 - set_topology, [1582](#)
 - set_write_zeroes, [1582](#)
- L4virtio::Svr::Block_request< Ds_data >, [1583](#)
 - data_size, [1584](#)
 - next_block, [1585](#)
- L4virtio::Svr::Data_buffer, [1586](#)
 - copy_to, [1587](#)
 - Data_buffer, [1587](#)
 - done, [1588](#)
 - set, [1588](#)
 - skip, [1589](#)
- L4virtio::Svr::Dev_config, [1590](#)
 - change_queue_config, [1592](#)
 - Dev_config, [1591](#), [1592](#)
 - ds, [1593](#)
 - get_cmd, [1593](#)
 - guest_features, [1594](#)
 - hdr, [1595](#)
 - negotiated_features, [1595](#)
 - qconfig, [1596](#)
 - reset_cmd, [1596](#)
 - reset_queue, [1597](#)
 - set_device_needs_reset, [1598](#)
 - set_status, [1599](#)
 - status, [1600](#)
- L4virtio::Svr::Dev_features, [1601](#)
- L4virtio::Svr::Dev_status, [1602](#)
 - running, [1604](#)
- L4virtio::Svr::Device_t< DATA >, [1605](#)
 - device_error, [1607](#)
 - device_notify_irq, [1608](#)
 - handle_mem_cmd_write, [1608](#)
 - init_mem_info, [1609](#)
 - register_driver_irq, [1610](#)
 - reset_queue_config, [1610](#)
 - setup_queue, [1611](#)
- L4virtio::Svr::Driver_mem_list_t< DATA >, [1612](#)
 - add, [1615](#)
 - find, [1616](#)
 - full, [1616](#)

- init, 1617
- load_desc, 1618–1620
- remove, 1621
- L4virtio::Svr::Driver_mem_region_t< DATA >, 1622
 - contains, 1624
 - Driver_mem_region_t, 1624
 - drv_base, 1625
 - ds, 1625
 - ds_offset, 1626
 - empty, 1626
 - flags, 1626
 - is_writable, 1626
 - local, 1627
 - local_base, 1628
 - size, 1628
- L4virtio::Svr::Request_processor, 1629
 - current_flags, 1630
 - has_more, 1630
 - next, 1631
 - start, 1633, 1634
- L4virtio::Svr::Virtqueue, 1636
 - consumed, 1638
 - desc, 1639
 - desc_avail, 1639
 - disable_notify, 1640
 - enable_notify, 1640
 - next_avail, 1641
- L4virtio::Svr::Virtqueue::Head_desc, 1642
 - desc, 1642
 - operator Null_ptr_check const *, 1643
 - valid, 1643
- L4virtio::Virtqueue, 1644
 - avail_align, 1647
 - avail_size, 1647
 - desc_align, 1648
 - desc_size, 1648
 - disable, 1650
 - dump, 1651
 - get_avail_idx, 1651
 - get_tail_avail_idx, 1652
 - no_notify_guest, 1652
 - no_notify_host, 1653, 1654
 - num, 1654
 - ready, 1655
 - setup, 1656
 - setup_simple, 1657
 - total_size, 1658, 1659
 - used_align, 1659
 - used_size, 1660
- L4virtio::Virtqueue::Avail, 1661
- L4virtio::Virtqueue::Avail::Flags, 1662
 - no_irq_bfm_t, 1663
- L4virtio::Virtqueue::Desc, 1664
- L4virtio::Virtqueue::Desc::Flags, 1665
 - indirect_bfm_t, 1666
 - next_bfm_t, 1666
 - write_bfm_t, 1667
- L4virtio::Virtqueue::Used, 1667
- L4virtio::Virtqueue::Used::Flags, 1668
 - no_notify_bfm_t, 1669
- L4virtio::Virtqueue::Used_elem, 1670
 - Used_elem, 1670
- l4virtio_block_config_t, 1671
- l4virtio_block_discard_t, 1672
- l4virtio_block_header_t, 1673
- L4virtio_block_operations
 - L4 VIRTIO Block Device, 290
- L4VIRTIO_BLOCK_S_IOERR
 - L4 VIRTIO Block Device, 291
- L4VIRTIO_BLOCK_S_OK
 - L4 VIRTIO Block Device, 291
- L4VIRTIO_BLOCK_S_UNSUPP
 - L4 VIRTIO Block Device, 291
- L4virtio_block_status
 - L4 VIRTIO Block Device, 291
- L4VIRTIO_BLOCK_T_DISCARD
 - L4 VIRTIO Block Device, 290
- L4VIRTIO_BLOCK_T_FLUSH
 - L4 VIRTIO Block Device, 290
- L4VIRTIO_BLOCK_T_GET_ID
 - L4 VIRTIO Block Device, 290
- L4VIRTIO_BLOCK_T_IN
 - L4 VIRTIO Block Device, 290
- L4VIRTIO_BLOCK_T_OUT
 - L4 VIRTIO Block Device, 290
- L4VIRTIO_BLOCK_T_WRITE_ZEROES
 - L4 VIRTIO Block Device, 290
- L4VIRTIO_CMD_CFG_QUEUE
 - L4 VIRTIO Transport Layer, 295
- L4VIRTIO_CMD_MASK
 - L4 VIRTIO Transport Layer, 295
- L4VIRTIO_CMD_NONE
 - L4 VIRTIO Transport Layer, 295
- L4VIRTIO_CMD_SET_STATUS
 - L4 VIRTIO Transport Layer, 295
- l4virtio_config_hdr_t, 1674
 - status, 1675
- l4virtio_config_queue
 - L4 VIRTIO Transport Layer, 298
- l4virtio_config_queue_t, 1675
 - L4 VIRTIO Transport Layer, 295
- l4virtio_config_queues
 - L4 VIRTIO Transport Layer, 298
- l4virtio_device_config
 - L4 VIRTIO Transport Layer, 299
- l4virtio_device_config_ds
 - L4 VIRTIO Transport Layer, 299
- L4virtio_device_ids
 - L4 VIRTIO Transport Layer, 296
- l4virtio_device_notification_irq
 - L4 VIRTIO Transport Layer, 299
- L4virtio_device_status
 - L4 VIRTIO Transport Layer, 297
- L4virtio_feature_bits
 - L4 VIRTIO Transport Layer, 297
- L4VIRTIO_FEATURE_CMD_CONFIG

- L4 VIRTIO Transport Layer, [297](#)
- L4VIRTIO_FEATURE_VERSION_1
 - L4 VIRTIO Transport Layer, [297](#)
- L4VIRTIO_ID_9P
 - L4 VIRTIO Transport Layer, [296](#)
- L4VIRTIO_ID_BALLOON
 - L4 VIRTIO Transport Layer, [296](#)
- L4VIRTIO_ID_BLOCK
 - L4 VIRTIO Transport Layer, [296](#)
- L4VIRTIO_ID_CAIF
 - L4 VIRTIO Transport Layer, [297](#)
- L4VIRTIO_ID_CONSOLE
 - L4 VIRTIO Transport Layer, [296](#)
- L4VIRTIO_ID_CRYPTIO
 - L4 VIRTIO Transport Layer, [297](#)
- L4VIRTIO_ID_GPU
 - L4 VIRTIO Transport Layer, [297](#)
- L4VIRTIO_ID_INPUT
 - L4 VIRTIO Transport Layer, [297](#)
- L4VIRTIO_ID_NET
 - L4 VIRTIO Transport Layer, [296](#)
- L4VIRTIO_ID_RNG
 - L4 VIRTIO Transport Layer, [296](#)
- L4VIRTIO_ID_RPMSG
 - L4 VIRTIO Transport Layer, [296](#)
- L4VIRTIO_ID_RPROC_SERIAL
 - L4 VIRTIO Transport Layer, [297](#)
- L4VIRTIO_ID_SCSI
 - L4 VIRTIO Transport Layer, [296](#)
- L4VIRTIO_ID_SOCKET
 - L4 VIRTIO Transport Layer, [297](#)
- L4VIRTIO_ID_VSOCK
 - L4 VIRTIO Transport Layer, [297](#)
- l4virtio_input_absinfo_t, [1677](#)
- l4virtio_input_config_t, [1677](#)
- l4virtio_input_devids_t, [1678](#)
- l4virtio_input_event_t, [1679](#)
- L4VIRTIO_IRQ_STATUS_CONFIG
 - L4 VIRTIO Transport Layer, [296](#)
- L4VIRTIO_IRQ_STATUS_VRING
 - L4 VIRTIO Transport Layer, [296](#)
- L4VIRTIO_OP_CONFIG_QUEUE
 - L4 VIRTIO Transport Layer, [296](#)
- L4VIRTIO_OP_REGISTER_DS
 - L4 VIRTIO Transport Layer, [296](#)
- L4VIRTIO_OP_REGISTER_IFACE
 - L4 VIRTIO Transport Layer, [296](#)
- L4VIRTIO_OP_SET_STATUS
 - L4 VIRTIO Transport Layer, [296](#)
- l4virtio_register_ds
 - L4 VIRTIO Transport Layer, [300](#)
- l4virtio_register_iface
 - L4 VIRTIO Transport Layer, [300](#)
- l4virtio_set_status
 - L4 VIRTIO Transport Layer, [301](#)
- L4VIRTIO_STATUS_ACKNOWLEDGE
 - L4 VIRTIO Transport Layer, [297](#)
- L4VIRTIO_STATUS_DEVICE_NEEDS_RESET
 - L4 VIRTIO Transport Layer, [297](#)
- L4 VIRTIO Transport Layer, [297](#)
- L4VIRTIO_STATUS_DRIVER
 - L4 VIRTIO Transport Layer, [297](#)
- L4VIRTIO_STATUS_DRIVER_OK
 - L4 VIRTIO Transport Layer, [297](#)
- L4VIRTIO_STATUS_FAILED
 - L4 VIRTIO Transport Layer, [297](#)
- L4VIRTIO_STATUS_FEATURES_OK
 - L4 VIRTIO Transport Layer, [297](#)
- length
 - L4::lpc::Varg, [940](#)
- libedid_block_size
 - EDID parsing functionality, [168](#)
- libedid_check_header
 - EDID parsing functionality, [168](#)
- libedid_checksum
 - EDID parsing functionality, [169](#)
- Libedid_consts
 - EDID parsing functionality, [168](#)
- libedid_dump
 - EDID parsing functionality, [169](#)
- libedid_dump_standard_timings
 - EDID parsing functionality, [169](#)
- libedid_num_ext_blocks
 - EDID parsing functionality, [170](#)
- libedid_pnp_id
 - EDID parsing functionality, [170](#)
- libedid_prefered_resolution
 - EDID parsing functionality, [170](#)
- libedid_revision
 - EDID parsing functionality, [171](#)
- libedid_version
 - EDID parsing functionality, [171](#)
- Link
 - L4Re::Namespace, [1282](#)
- link
 - L4Re::Vfs::Directory, [1385](#)
- List_alloc
 - cxx::List_alloc, [717](#)
- list_alloc.h
 - l4la_alloc, [2300](#)
 - l4la_avail, [2300](#)
 - l4la_dump, [2300](#)
 - l4la_free, [2300](#)
 - l4la_init, [2301](#)
- load_desc
 - L4virtio::Svr::Driver_mem_list_t< DATA >, [1618](#)–[1620](#)
- local
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [1627](#)
- local_base
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [1628](#)
- local_id_received
 - L4::lpc::Gen_fpage< T >, [882](#)
- log
 - L4Re::Env, [1249](#)

- Log interface, [338](#)
 - [l4re_log_print](#), [339](#)
 - [l4re_log_print_srv](#), [339](#)
 - [l4re_log_printn](#), [340](#)
 - [l4re_log_printn_srv](#), [341](#)
- Logging interface, [342](#)
- loop
 - [L4::Server< LOOP_HOOKS >](#), [1078](#)
- Low-Level Thread Functions, [342](#)
- Low_mask
 - [cxx::Bitfield< T, LSB, MSB >](#), [629](#)
- lower_bound_node
 - [cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >](#), [668](#)
 - [cxx::Bits::Bst< Node, Get_key, Compare >](#), [687](#)
- Lsb
 - [cxx::Bitfield< T, LSB, MSB >](#), [629](#)
- lseek64
 - [L4Re::Vfs::Regular_file](#), [1409](#)
- Lstr
 - [L4::Factory::Lstr](#), [840](#)
- main_thread
 - [L4Re::Env](#), [1249](#), [1250](#)
- make_cap
 - [L4::lpc](#), [552](#)
- make_cap_full
 - [L4::lpc](#), [552](#)
- make_cap_rw
 - [L4::lpc](#), [553](#)
- make_cap_rws
 - [L4::lpc](#), [554](#)
- make_ref_cap
 - [L4Re::Util](#), [588](#)
- make_ref_del_cap
 - [L4Re::Util](#), [589](#)
- make_shared_cap
 - [L4Re](#), [575](#)
 - [L4Re::Util](#), [589](#)
- make_shared_del_cap
 - [L4Re](#), [576](#)
 - [L4Re::Util](#), [590](#)
- make_unique_cap
 - [L4Re](#), [577](#)
 - [L4Re::Util](#), [590](#)
- make_unique_del_cap
 - [L4Re](#), [577](#)
 - [L4Re::Util](#), [591](#)
- map
 - [L4::Task](#), [1100](#)
 - [L4Re::Dataspace](#), [1229](#)
 - [L4Re::Dma_space](#), [1240](#)
 - [L4Re::Util::Dataspace_srv](#), [1335](#)
- map_hook
 - [L4Re::Util::Dataspace_srv](#), [1335](#)
- map_region
 - [L4Re::Dataspace](#), [1230](#)
- Mask
 - [cxx::Bitfield< T, LSB, MSB >](#), [629](#)
- mask
 - [L4::lcu](#), [851](#)
- Masks
 - [cxx::Bitfield< T, LSB, MSB >](#), [629](#)
- max
 - Small C++ Template Library, [433](#)
- max_free_slabs
 - [cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >](#), [616](#)
 - [cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >](#), [624](#)
- max_queue_size
 - [L4virtio::Driver::Device](#), [1553](#)
- MB_ARD_MEMORY
 - [mb_info.h](#), [2306](#)
- MB_ART_MEMORY
 - [mb_info.h](#), [2306](#)
- mb_info.h
 - [l4util_mb_for_each_mmap_entry](#), [2306](#)
 - [L4UTIL_MB_MEMORY](#), [2306](#)
 - [MB_ARD_MEMORY](#), [2306](#)
 - [MB_ART_MEMORY](#), [2306](#)
- Mem
 - [L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >](#), [1130](#)
- mem_alloc
 - [L4Re::Env](#), [1250](#)
- Mem_alloc_flags
 - [L4Re::Mem_alloc](#), [1272](#)
- Mem_desc
 - [L4::Kip::Mem_desc](#), [1008](#)
- Mem_type
 - [L4::Kip::Mem_desc](#), [1007](#)
- Memory allocator, [343](#)
 - [l4re_ma_alloc](#), [344](#)
 - [l4re_ma_alloc_align](#), [345](#)
 - [l4re_ma_alloc_align_srv](#), [346](#)
 - [l4re_ma_flags](#), [343](#)
 - [l4re_ma_free](#), [347](#)
 - [l4re_ma_free_srv](#), [348](#)
- Memory descriptors (C version), [349](#)
 - [l4_kernel_info_get_mem_desc_end](#), [351](#)
 - [l4_kernel_info_get_mem_desc_is_virtual](#), [351](#)
 - [l4_kernel_info_get_mem_desc_start](#), [352](#)
 - [l4_kernel_info_get_mem_desc_subtype](#), [352](#)
 - [l4_kernel_info_get_mem_desc_type](#), [352](#)
 - [l4_kernel_info_get_num_mem_descs](#), [353](#)
 - [l4_kernel_info_mem_desc_t](#), [350](#)
 - [l4_kernel_info_set_mem_desc](#), [353](#)
 - [l4_mem_info_acpi_rsdp](#), [351](#)
 - [l4_mem_info_sub_type_t](#), [350](#)
 - [l4_mem_type_archspecific](#), [351](#)
 - [l4_mem_type_bootloader](#), [351](#)
 - [l4_mem_type_conventional](#), [351](#)
 - [l4_mem_type_dedicated](#), [351](#)
 - [l4_mem_type_info](#), [351](#)
 - [l4_mem_type_reserved](#), [351](#)
 - [l4_mem_type_shared](#), [351](#)

- l4_mem_type_t, [351](#)
- l4_mem_type_undefined, [351](#)
- Memory operations., [354](#)
 - l4_mem_op_widths, [354](#)
 - l4_mem_read, [355](#)
 - L4_MEM_WIDTH_1BYTE, [355](#)
 - L4_MEM_WIDTH_2BYTE, [355](#)
 - L4_MEM_WIDTH_4BYTE, [355](#)
 - l4_mem_write, [355](#)
- Memory related, [356](#)
 - l4_addr_consts_t, [359](#)
 - l4_bytes_to_mwords, [359](#)
 - L4_INVALID_ADDR, [359](#)
 - L4_LOG2_PAGESIZE, [358](#)
 - L4_LOG2_SUPERPAGESIZE, [358](#)
 - L4_PAGEMASK, [358](#)
 - l4_round_page, [359](#)
 - l4_round_size, [360](#)
 - L4_SUPERPAGEMASK, [358](#)
 - L4_SUPERPAGESIZE, [358](#)
 - l4_trunc_page, [361](#)
 - l4_trunc_size, [362](#)
- message
 - L4virtio::Svr::Bad_descriptor, [1574](#)
- Message Items, [362](#)
 - L4_ITEM_CONT, [363](#)
 - L4_ITEM_MAP, [363](#)
 - l4_map_control, [364](#)
 - L4_MAP_ITEM_GRANT, [363](#)
 - L4_MAP_ITEM_MAP, [363](#)
 - l4_map_obj_control, [364](#)
 - l4_msg_item_consts_t, [363](#)
 - L4_RCV_ITEM_LOCAL_ID, [363](#)
 - L4_RCV_ITEM_SINGLE_CAP, [363](#)
- Message Registers (MRs), [365](#)
- Message Tag, [366](#)
 - l4_msgtag, [370](#)
 - L4_MSGTAG_ERROR, [368](#)
 - L4_MSGTAG_FLAGS, [368](#)
 - l4_msgtag_flags, [368](#), [371](#)
 - l4_msgtag_has_error, [371](#)
 - l4_msgtag_is_exception, [372](#)
 - l4_msgtag_is_io_page_fault, [373](#)
 - l4_msgtag_is_page_fault, [373](#)
 - l4_msgtag_is_preemption, [374](#)
 - l4_msgtag_is_sigma0, [375](#)
 - l4_msgtag_is_sys_exception, [375](#)
 - l4_msgtag_items, [376](#)
 - l4_msgtag_label, [377](#)
 - L4_MSGTAG_PROPAGATE, [368](#)
 - l4_msgtag_protocol, [369](#)
 - L4_MSGTAG_SCHEDULE, [368](#)
 - l4_msgtag_t, [368](#)
 - L4_MSGTAG_TRANSFER_FPU, [368](#)
 - l4_msgtag_words, [378](#)
 - L4_platform_ctl_proto, [369](#)
 - L4_PROTO_ALLOW_SYSCALL, [369](#)
 - L4_PROTO_DEBUGGER, [369](#)
 - L4_PROTO_DMA_SPACE, [369](#)
 - L4_PROTO_EXCEPTION, [369](#)
 - L4_PROTO_FACTORY, [369](#)
 - L4_PROTO_IO_PAGE_FAULT, [369](#)
 - L4_PROTO_IOMMU, [369](#)
 - L4_PROTO_IRQ, [369](#)
 - L4_PROTO_IRQ_MUX, [369](#)
 - L4_PROTO_IRQ_SENDER, [369](#)
 - L4_PROTO_KOBJECT, [369](#)
 - L4_PROTO_LOG, [369](#)
 - L4_PROTO_META, [369](#)
 - L4_PROTO_NONE, [369](#)
 - L4_PROTO_PAGE_FAULT, [369](#)
 - L4_PROTO_PF_EXCEPTION, [369](#)
 - L4_PROTO_PLATFORM_CTL, [370](#)
 - L4_PROTO_PREEMPTION, [369](#)
 - L4_PROTO_SCHEDULER, [369](#)
 - L4_PROTO_SEMAPHORE, [369](#)
 - L4_PROTO_SIGMA0, [369](#)
 - L4_PROTO_SMCCC, [369](#)
 - L4_PROTO_SYS_EXCEPTION, [369](#)
 - L4_PROTO_TASK, [369](#)
 - L4_PROTO_THREAD, [369](#)
 - L4_PROTO_VM, [369](#)
- min
 - Small C++ Template Library, [434](#)
- mkdir
 - L4Re::Vfs::Directory, [1385](#)
- mmio_read
 - L4Re::Mmio_space, [1277](#)
- mmio_write
 - L4Re::Mmio_space, [1278](#)
- Mode
 - L4Re::Util::Event_t< PAYLOAD >, [1348](#)
- Mode_irq
 - L4Re::Util::Event_t< PAYLOAD >, [1348](#)
- Mode_polling
 - L4Re::Util::Event_t< PAYLOAD >, [1348](#)
- modify_senders
 - L4::Thread, [1109](#)
- mount
 - L4Re::Vfs::File_system, [1391](#)
 - L4Re::Vfs::Fs, [1396](#)
- move
 - L4::Cap< T >, [789](#)
 - L4::Cap_base, [798](#)
- Mr_bytes
 - L4::lpc::Msg, [557](#)
- Mr_words
 - L4::lpc::Msg, [557](#)
- Msb
 - cxx::Bitfield< T, LSB, MSB >, [629](#)
- msg_add
 - L4::lpc::Msg, [560](#)
- msg_get
 - L4::lpc::Msg, [561](#)
- Msg_ptr
 - L4::lpc::Msg_ptr< T >, [922](#)

- msg_ptr
 - L4::lpc, [555](#)
- msi_info
 - L4::lcu, [852](#)
- N
 - cxx::Bits::Direction, [695](#)
- Name-space API, [379](#)
- Name_max
 - L4Re::Inhibitor, [1264](#)
- Namespace interface, [379](#)
 - l4re_ns_query_srv, [380](#)
 - l4re_ns_query_to_srv, [381](#)
 - l4re_ns_register_flags, [380](#)
 - l4re_ns_register_obj_srv, [382](#)
- negotiated_features
 - L4virtio::Svr::Dev_config, [1595](#)
- next
 - L4Re::Event_buffer_t< PAYLOAD >, [1259](#)
 - L4virtio::Svr::Request_processor, [1631](#)
- next_avail
 - L4virtio::Svr::Virtqueue, [1641](#)
- next_bfm_t
 - L4virtio::Virtqueue::Desc::Flags, [1666](#)
- next_block
 - L4virtio::Svr::Block_request< Ds_data >, [1585](#)
- next_device
 - L4vbus::Device, [1474](#)
- next_lock_info
 - L4Re::Inhibitor, [1265](#)
- next_timeout
 - L4::lpc_svr::Timeout_queue, [978](#)
- no_demand
 - L4::Type_info::Demand, [1127](#)
- No_init
 - L4::Cap_base, [793](#)
- No_init_type
 - L4::Cap_base, [793](#)
- no_irq_bfm_t
 - L4virtio::Virtqueue::Avail::Flags, [1663](#)
- no_notify_bfm_t
 - L4virtio::Virtqueue::Used::Flags, [1669](#)
- no_notify_guest
 - L4virtio::Virtqueue, [1652](#)
- no_notify_host
 - L4virtio::Virtqueue, [1653](#), [1654](#)
- No_sync
 - L4Re::Dma_space, [1238](#)
- None
 - L4::Types::Flags< BITS_ENUM, UNDERLYING >, [1155](#)
 - L4Re::Dma_space, [1238](#)
- None_type
 - L4::Types::Flags< BITS_ENUM, UNDERLYING >, [1155](#)
- Normal
 - L4Re::Dataspace::F, [1232](#)
- NT_VERSION
 - elf.h, [2274](#)
- num
 - L4virtio::Virtqueue, [1654](#)
- num_interfaces
 - L4::Meta, [1037](#)
- obj_cap
 - L4::Epiface, [817](#)
 - L4::Epiface_t0< RPC_IFACE, BASE >, [824](#)
 - L4::lreqep_t< Derived, BASE, bool >, [1005](#)
- Object Invocation, [383](#)
 - l4_ipc, [386](#)
 - l4_ipc_call, [387](#)
 - l4_ipc_receive, [388](#)
 - l4_ipc_reply_and_wait, [389](#)
 - l4_ipc_send, [391](#)
 - l4_ipc_send_and_wait, [392](#)
 - l4_ipc_sleep, [393](#)
 - l4_ipc_wait, [394](#)
 - l4_sndfpage_add, [395](#)
 - l4_syscall_flags_t, [385](#)
 - L4_SYSF_CALL, [385](#)
 - L4_SYSF_NONE, [385](#)
 - L4_SYSF_OPEN_WAIT, [385](#)
 - L4_SYSF_RECV, [385](#)
 - L4_SYSF_REPLY, [385](#)
 - L4_SYSF_REPLY_AND_WAIT, [385](#)
 - L4_SYSF_SEND, [385](#)
 - L4_SYSF_SEND_AND_WAIT, [385](#)
 - L4_SYSF_WAIT, [385](#)
- Object_registry
 - L4Re::Util::Object_registry, [1359](#)
- object_size
 - cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, [616](#)
 - cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, [624](#)
- objects_per_slab
 - cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, [616](#)
 - cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, [624](#)
- offset
 - l4_sched_cpu_set_t, [1194](#)
- Opcode
 - L4::Platform_control, [1047](#)
- operator l4_msgtag_t
 - L4::Factory::S, [843](#)
- operator new
 - Small C++ Template Library, [435](#)
- operator Null_ptr_check const *
 - L4virtio::Svr::Virtqueue::Head_desc, [1643](#)
- operator!
 - cxx::Bits::Direction, [695](#)
- operator!=
 - L4vbus::Device, [1475](#)
- operator<<
 - ipc_stream, [1837–1839](#)
- operator>>
 - ipc_stream, [1840–1844](#)

- operator*
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COM-PARE, ALLOC, GET_KEY >::Node, [673](#)
- operator()
 - cxx::Pair_first_compare< Cmp, Typ >, [733](#)
- operator->
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COM-PARE, ALLOC, GET_KEY >::Node, [673](#)
- operator=
 - cxx::Auto_ptr< T >, [597](#)
- operator==
 - L4Re::Video::Color_component, [1416](#)
 - L4Re::Video::Pixel_info, [1431](#)
 - L4vbus::Device, [1475](#)
- operator[]
 - cxx::Avl_map< KEY_TYPE, DATA_TYPE, COM-PARE, ALLOC >, [603](#)
 - cxx::Bitmap_base, [651](#)
 - cxx::List< D, Alloc >, [715](#)
- Overwrite
 - L4Re::Namespace, [1282](#)
- page_fault
 - L4::Pager, [1044](#)
- page_shift
 - L4Re::Util::Dataspace_svr, [1336](#)
- Pager
 - L4Re::Rm::F, [1308](#)
- pager
 - L4::Thread::Attr, [1117](#)
- Pair
 - cxx::Pair< First, Second >, [731](#)
- Pair_first_compare
 - cxx::Pair_first_compare< Cmp, Typ >, [733](#)
- parent
 - L4Re::Env, [1251](#)
- Parent API, [396](#)
- parse_cmd.h
 - parse_cmdline, [2312](#)
- parse_cmdline
 - parse_cmd.h, [2312](#)
- Partly_resolved
 - L4Re::Namespace, [1281](#)
- Phys_space
 - L4Re::Dma_space, [1239](#)
- pin
 - L4vbus::Gpio_module, [1480](#)
 - L4vbus::Gpio_pin, [1488](#)
- Pinned
 - L4Re::Mem_alloc, [1272](#)
- Pixel_info
 - L4Re::Video::Pixel_info, [1426](#), [1427](#)
- Platform Control C API, [397](#)
 - I4_platform_ctl_cpu_disable, [397](#)
 - I4_platform_ctl_cpu_enable, [398](#)
 - I4_platform_ctl_system_shutdown, [398](#)
 - I4_platform_ctl_system_suspend, [399](#)
- Poll_timeout_kipclock
 - L4::Poll_timeout_kipclock, [1050](#)
- pop_front
 - cxx::Bits::Smart_ptr_list< ITEM >, [697](#)
 - cxx::H_list< T, POLICY >, [706](#)
 - cxx::S_list< T, POLICY >, [742](#)
- port_io.h
 - I4util_in16, [1756](#)
 - I4util_in32, [1756](#)
 - I4util_in8, [1757](#)
 - I4util_ins16, [1757](#)
 - I4util_ins32, [1758](#)
 - I4util_ins8, [1758](#)
 - I4util_ioport_map, [1752](#)
 - I4util_out16, [1758](#)
 - I4util_out32, [1759](#)
 - I4util_out8, [1759](#)
 - I4util_outs16, [1760](#)
 - I4util_outs32, [1760](#)
 - I4util_outs8, [1760](#)
- Ports
 - L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >, [1130](#)
- print
 - L4Re::Log, [1269](#)
- println
 - L4Re::Log, [1269](#)
- process
 - L4Re::Util::Event_buffer_consumer_t< PAYLOAD >, [1340](#)
- process_request
 - L4virtio::Driver::Block_device, [1541](#)
 - L4virtio::Svr::Block_dev_base< Ds_data >, [1579](#)
- process_used_queue
 - L4virtio::Driver::Block_device, [1542](#)
- Producer, [399](#), [400](#)
 - I4shmc_chunk_ready, [401](#)
 - I4shmc_chunk_ready_sig, [401](#)
 - I4shmc_chunk_try_to_take, [401](#)
 - I4shmc_is_chunk_clear, [402](#)
 - I4shmc_trigger, [400](#)
- PROTO_ANY
 - L4, [542](#)
- proto_dispatch
 - L4::Kobject_typeid< T >, [1031](#)
 - L4::Server_object_t< IFACE, BASE >, [1087](#)
- PROTO_EMPTY
 - L4, [542](#)
- protocols.h
 - L4RE_PROTO_DATASPACE, [1968](#)
 - L4RE_PROTO_DEBUG, [1968](#)
 - L4RE_PROTO_DMA_SPACE, [1968](#)
 - L4RE_PROTO_EVENT, [1968](#)
 - L4RE_PROTO_GOOS, [1968](#)
 - L4RE_PROTO_INHIBITOR, [1968](#)
 - L4RE_PROTO_MMIO_SPACE, [1968](#)
 - L4RE_PROTO_NAMESPACE, [1968](#)
 - L4RE_PROTO_PARENT, [1968](#)
 - L4RE_PROTO_RM, [1968](#)
 - L4RE_PROTO_RSVD_1, [1968](#)

- L4re_protocols, 1968
- ptr
 - cxx::Ref_ptr< T, CNT >, 738
- push_back
 - cxx::List_item, 721
- push_front
 - cxx::List_item, 722
- put
 - L4::Factory::S, 843–845
 - L4::lpc::Iostream, 889, 890
 - L4::lpc::Ostream, 927, 928
 - L4Re::Event_buffer_t< PAYLOAD >, 1259
- qconfig
 - L4virtio::Svr::Dev_config, 1596
- query
 - L4Re::Namespace, 1282, 1283
- query_log_name
 - L4::Debugger, 808
- query_log_typeid
 - L4::Debugger, 809
- Query_result_flags
 - L4Re::Namespace, 1281
- R
 - cxx::Bits::Direction, 695
 - L4Re::Dataspace::F, 1232
- r
 - L4Re::Video::Pixel_info, 1432
 - L4vcpu::Vcpu, 1527
- Random number support, 403
 - l4util_rand, 403
 - l4util_srand, 403
- rbegin
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, 669
 - cxx::Bits::Bst< Node, Get_key, Compare >, 688
- rcv_cap
 - L4::lpc_svr::Server_iface, 969, 970
- rcv_endpoint.h
 - L4_RCV_EP_BIND_OP, 2167
 - L4_rcv_ep_ops, 2167
- rdtsc.h
 - l4_busy_wait_ns, 1709, 1720
 - l4_busy_wait_us, 1710, 1721
 - l4_calibrate_tsc, 1710, 1722
 - l4_get_hz, 1711, 1722
 - l4_ns_to_tsc, 1711, 1722
 - l4_rdpmc, 1712, 1723
 - l4_rdpmc_32, 1712, 1723
 - l4_rdtsc, 1713, 1723
 - l4_rdtsc_32, 1713, 1724
 - l4_tsc_init, 1714, 1724
 - l4_tsc_to_ns, 1715, 1725
 - l4_tsc_to_s_and_ns, 1715, 1725
 - l4_tsc_to_us, 1716, 1725
- read
 - L4::lpc, 555
 - L4::Vcon, 1173
- read_with_flags
 - L4::Vcon, 1174
- readv
 - L4Re::Vfs::Regular_file, 1409
- ready
 - L4virtio::Virtqueue, 1655
- realloc_rcv_cap
 - L4::lpc_svr::Server_iface, 971
 - L4Re::Util::Br_manager, 1315
- Realtime API, 404
- receive
 - L4::lpc::Istream, 898
 - L4::Irq, 990
- Ref_ptr
 - cxx::Ref_ptr< T, CNT >, 737, 738
- refresh
 - L4Re::Util::Video::Goos_svr, 1374
 - L4Re::Video::View, 1435
- Region
 - L4Re::Rm, 1297
- Region map API, 404
- Region map interface, 405
 - l4re_rm_attach, 407
 - l4re_rm_attach_srv, 408
 - L4RE_RM_CACHING_SHIFT, 406
 - l4re_rm_detach, 409
 - l4re_rm_detach_ds, 410
 - l4re_rm_detach_ds_unmap, 410
 - l4re_rm_detach_srv, 411
 - l4re_rm_detach_unmap, 412
 - L4RE_RM_F_ATTACH_FLAGS, 406
 - L4RE_RM_F_CACHE_BUFFERED, 406
 - L4RE_RM_F_CACHE_NORMAL, 406
 - L4RE_RM_F_CACHE_UNCACHED, 406
 - L4RE_RM_F_CACHING, 406
 - L4RE_RM_F_EAGER_MAP, 406
 - L4RE_RM_F_IN_AREA, 406
 - L4RE_RM_F_NO_ALIAS, 406
 - L4RE_RM_F_PAGER, 406
 - L4RE_RM_F_R, 406
 - L4RE_RM_F_RESERVED, 406
 - L4RE_RM_F_SEARCH_ADDR, 406
 - l4re_rm_find, 413
 - l4re_rm_find_srv, 413
 - l4re_rm_flags_values, 406
 - l4re_rm_free_area, 414
 - l4re_rm_free_area_srv, 414
 - L4RE_RM_REGION_FLAGS, 406
 - l4re_rm_reserve_area, 415
 - l4re_rm_reserve_area_srv, 415
 - l4re_rm_show_lists, 416
- Region_flag_shifts
 - L4Re::Rm, 1298
- Region_flags
 - L4Re::Rm::F, 1308
- Region_flags_mask
 - L4Re::Rm::F, 1308
- register_del_irq

- L4::Thread, [1110](#)
- register_driver_irq
 - L4virtio::Svr::Device_t< DATA >, [1610](#)
- register_ds
 - L4virtio::Device, [1535](#)
 - L4virtio::Driver::Device, [1554](#)
- Register_flags
 - L4Re::Namespace, [1281](#)
- register_iface
 - L4virtio::Device, [1536](#)
- register_irq_obj
 - L4::Registry_iface, [1058](#)
 - L4Re::Util::Object_registry, [1360](#)
- register_obj
 - L4::Registry_iface, [1059](#), [1060](#)
 - L4Re::Namespace, [1284](#)
 - L4Re::Util::Object_registry, [1360](#), [1361](#)
- Registry_server
 - L4Re::Util::Registry_server< LOOP_HOOKS >, [1368](#)
- release
 - cxx::Auto_ptr< T >, [598](#)
 - cxx::Ref_ptr< T, CNT >, [739](#)
 - L4Re::Inhibitor, [1266](#)
 - L4Re::Util::Counting_cap_alloc< COUNTERTYPE >, [1328](#)
 - L4Re::Util::Dataspace_svr, [1336](#)
- release_cap
 - L4::Task, [1101](#)
- release_ioport
 - L4vbus::Vbus, [1510](#)
- release_resource
 - L4vbus::Vbus, [1511](#)
- remove
 - cxx::Avl_tree< Node, Get_key, Compare >, [611](#)
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [670](#)
 - cxx::H_list< T, POLICY >, [707](#)
 - cxx::List_item, [722](#)
 - L4::lpc_svr::Timeout_queue, [979](#)
 - L4virtio::Svr::Driver_mem_list_t< DATA >, [1621](#)
- remove_all
 - cxx::Bits::Bst< Node, Get_key, Compare >, [689](#)
- remove_timeout
 - L4::lpc_svr::Server_iface, [972](#)
 - L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >, [985](#)
- remove_tree
 - cxx::Bits::Bst< Node, Get_key, Compare >, [690](#)
- rename
 - L4Re::Vfs::Directory, [1386](#)
- rend
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [671](#)
 - cxx::Bits::Bst< Node, Get_key, Compare >, [691](#)
- replace
 - cxx::H_list< T, POLICY >, [708](#)
- reply_and_wait
 - L4::lpc::lostream, [890](#), [891](#)
- Reply_compound
 - Server-Side IPC framework, [423](#)
- Reply_mode
 - Server-Side IPC framework, [423](#)
- Reply_separate
 - Server-Side IPC framework, [423](#)
- request_ioport
 - L4vbus::Vbus, [1511](#)
- request_resource
 - L4vbus::Vbus, [1513](#)
- reserve_area
 - L4Re::Rm, [1305](#), [1306](#)
- Reserved
 - L4::Kip::Mem_desc, [1008](#)
 - L4Re::Rm::F, [1308](#)
- reset
 - L4::lpc::lostream, [891](#)
 - L4::lpc::lstream, [898](#)
- reset_cmd
 - L4virtio::Svr::Dev_config, [1596](#)
- reset_queue
 - L4virtio::Svr::Dev_config, [1597](#)
- reset_queue_config
 - L4virtio::Svr::Device_t< DATA >, [1610](#)
- Rights_mask
 - L4::lpc::Cap< T >, [878](#)
 - L4Re::Dataspace::F, [1232](#)
 - L4Re::Rm::F, [1308](#)
- rm
 - L4Re::Env, [1251](#), [1252](#)
- rmdir
 - L4Re::Vfs::Directory, [1386](#)
- Ro
 - L4Re::Dataspace::F, [1232](#)
 - L4Re::Namespace, [1282](#)
- root
 - L4vbus::Vbus, [1514](#)
- round_order
 - L4, [545](#)
- Rpcs
 - L4::Exception, [825](#)
- Rs
 - L4Re::Namespace, [1282](#)
- run_thread
 - L4::Scheduler, [1069](#)
- running
 - L4virtio::Svr::Dev_status, [1604](#)
- Runtime_error
 - L4::Runtime_error, [1064](#)
- RW
 - L4Re::Dataspace::F, [1232](#)
- Rw
 - L4Re::Namespace, [1282](#)
- Rws
 - L4Re::Namespace, [1282](#)
- S
 - L4::Factory::S, [842](#), [843](#)

- saved_state
 - L4vcpu::Vcpu, 1528
- scan_zero
 - cxx::Bitmap< BITS >, 645
 - cxx::Bitmap_base, 652
- Scheduler, 417
 - I4_sched_cpu_set, 418
 - I4_scheduler_idle_time, 419
 - L4_SCHEDULER_IDLE_TIME_OP, 418
 - I4_scheduler_info, 420
 - L4_SCHEDULER_INFO_OP, 418
 - I4_scheduler_is_online, 420
 - L4_scheduler_ops, 418
 - I4_scheduler_run_thread, 421
 - L4_SCHEDULER_RUN_THREAD_OP, 418
- scheduler
 - L4Re::Env, 1252
- screen_info
 - L4Re::Util::Video::Goos_svr, 1374
- Search_addr
 - L4Re::Rm::F, 1308
- segment.h
 - fiasco_amd64_segment_info, 1742
 - fiasco_amd64_set_fs, 1738, 1742
 - fiasco_amd64_set_segment_base, 1738, 1743
 - L4_AMD64_SEGMENT_FS, 1741
 - L4_AMD64_SEGMENT_GS, 1741
 - L4_sys_segment, 1741
 - L4_task_ldt_x86_consts, 1741, 1748
 - L4_TASK_LDT_X86_ENTRY_SIZE, 1742, 1748
 - L4_TASK_LDT_X86_MAX_ENTRIES, 1742, 1748
- send
 - L4::lpc::Ostream, 928
 - L4::Vcon, 1174
 - L4virtio::Driver::Device, 1555
- send_and_wait
 - L4virtio::Driver::Device, 1556
- send_request
 - L4virtio::Driver::Block_device, 1542
- Server
 - L4::Server< LOOP_HOOKS >, 1077
- Server-Side IPC framework, 422
 - Reply_compound, 423
 - Reply_mode, 423
 - Reply_separate, 423
- server_iface
 - L4::Epiface, 817
- set
 - cxx::Bitfield< T, LSB, MSB >, 631
 - L4::Kip::Mem_desc, 1013
 - L4::Poll_timeout_kipclock, 1050
 - L4Re::Video::Color_component, 1416
 - L4vbus::Gpio_module, 1481
 - L4vbus::Gpio_pin, 1489
 - L4vcpu::State, 1518
 - L4virtio::Svr::Data_buffer, 1588
- set_attr
 - L4::Vcon, 1175
- set_bit
 - cxx::Bitmap_base, 653
- set_blk_size
 - L4virtio::Svr::Block_dev_base< Ds_data >, 1580
- set_config_wce
 - L4virtio::Svr::Block_dev_base< Ds_data >, 1580
- set_device_needs_reset
 - L4virtio::Svr::Dev_config, 1598
- set_dirty
 - cxx::Bitfield< T, LSB, MSB >, 631
- set_discard
 - L4virtio::Svr::Block_dev_base< Ds_data >, 1580
- set_fd
 - L4Re::Vfs::Fs, 1396
- set_info
 - L4Re::Video::View, 1436
- set_lock
 - L4Re::Vfs::Regular_file, 1410
- set_mode
 - L4::lcu, 852
- set_object_name
 - L4::Debugger, 810
- set_rcv_cap_flags
 - L4Re::Util::Br_manager, 1315
- set_server
 - L4::Epiface, 818
- set_size_max
 - L4virtio::Svr::Block_dev_base< Ds_data >, 1582
- set_status
 - L4virtio::Device, 1536
 - L4virtio::Svr::Dev_config, 1599
- set_status_flags
 - L4Re::Vfs::Generic_file, 1400
- set_topology
 - L4virtio::Svr::Block_dev_base< Ds_data >, 1582
- set_unshifted
 - cxx::Bitfield< T, LSB, MSB >, 632
- set_unshifted_dirty
 - cxx::Bitfield< T, LSB, MSB >, 633
- set_viewport
 - L4Re::Video::View, 1436
- set_write_zeroes
 - L4virtio::Svr::Block_dev_base< Ds_data >, 1582
- setup
 - L4Re::Util::Counting_cap_alloc< COUNTERTYPE >, 1329
 - L4vbus::Gpio_module, 1481
 - L4vbus::Gpio_pin, 1489
 - L4virtio::Virtqueue, 1656
- setup_device
 - L4virtio::Driver::Block_device, 1543
- setup_queue
 - L4virtio::Svr::Device_t< DATA >, 1611
- setup_simple
 - L4virtio::Virtqueue, 1657
- Shared
 - L4::Kip::Mem_desc, 1008
- Shared Memory Library, 423

- l4shmc_area_overhead, [424](#)
- l4shmc_area_size, [425](#)
- l4shmc_area_size_free, [425](#)
- l4shmc_attach, [426](#)
- l4shmc_attach_to, [426](#)
- l4shmc_chunk_overhead, [427](#)
- l4shmc_connect_chunk_signal, [427](#)
- l4shmc_create, [427](#)
- Shared_cap
 - L4Re, [566](#)
 - L4Re::Util, [581](#)
- shared_cap
 - L4Re, [566](#)
 - L4Re::Util, [583](#)
- Shared_del_cap
 - L4Re, [567](#)
 - L4Re::Util, [584](#)
- shared_del_cap
 - L4Re, [567](#)
 - L4Re::Util, [584](#)
- SHF_GROUP
 - elf.h, [2274](#)
- SHF_MASKOS
 - elf.h, [2274](#)
- SHF_TLS
 - elf.h, [2275](#)
- shift
 - L4Re::Video::Color_component, [1416](#)
- Shift_type
 - cxx::Bitfield< T, LSB, MSB >, [628](#)
- SHT_NUM
 - elf.h, [2275](#)
- Shutdown
 - L4::Platform_control, [1047](#)
- si
 - l4_vcpu_regs_t, [1206](#)
- Sigma0 API, [428](#)
- sigma0.h
 - l4sigma0_debug_dump, [2012](#)
 - L4SIGMA0_IPCERROR, [2012](#)
 - l4sigma0_map_anypage, [2012](#)
 - l4sigma0_map_errstr, [2013](#)
 - l4sigma0_map_iomem, [2013](#)
 - l4sigma0_map_kip, [2014](#)
 - l4sigma0_map_mem, [2014](#)
 - l4sigma0_new_client, [2015](#)
 - L4SIGMA0_NOFPAGE, [2012](#)
 - L4SIGMA0_NOTALIGNED, [2012](#)
 - L4SIGMA0_OK, [2012](#)
 - l4sigma0_return_flags_t, [2012](#)
 - L4SIGMA0_SMALLERFPAGE, [2012](#)
- signal
 - L4Re::Parent, [1290](#)
- Signals, [429](#)
 - l4shmc_add_signal, [429](#)
 - l4shmc_attach_signal, [430](#)
 - l4shmc_attach_signal_to, [430](#)
 - l4shmc_check_magic, [431](#)
 - l4shmc_get_signal_to, [431](#)
 - l4shmc_signal_cap, [432](#)
- size
 - L4::Kip::Mem_desc, [1014](#)
 - L4Re::Dataspace, [1231](#)
 - L4Re::Video::Color_component, [1417](#)
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [1628](#)
- skip
 - L4::lpc::Istream, [899](#)
 - L4virtio::Svr::Data_buffer, [1589](#)
- slab_size
 - cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, [616](#)
 - cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, [624](#)
- Small C++ Template Library, [432](#)
 - max, [433](#)
 - min, [434](#)
 - operator new, [435](#)
- Small_buf
 - L4::lpc::Small_buf, [933](#), [934](#)
- Smart_cap
 - L4::Smart_cap< T, SMART >, [1094](#)
- snd_base
 - L4::Cap_base, [799](#)
- Space_attrb
 - L4Re::Dma_space, [1239](#)
- Split_ds
 - L4Re::Rm, [1298](#)
- Src_dev_handle
 - L4vbus::lcu, [1493](#)
- Src_types
 - L4vbus::lcu, [1493](#)
- ss
 - l4_exc_regs_t, [1183](#)
- stack
 - L4Re::Video::View, [1437](#)
- start
 - L4::Kip::Mem_desc, [1014](#)
 - L4virtio::Svr::Request_processor, [1633](#), [1634](#)
- start_request
 - L4virtio::Driver::Block_device, [1544](#)
- starts_with
 - cxx::String, [758](#)
- State
 - L4vcpu::State, [1517](#)
- state
 - L4vcpu::Vcpu, [1528](#), [1529](#)
- stats_time
 - L4::Thread, [1111](#)
- status
 - L4virtio::Svr::Dev_config, [1600](#)
 - l4virtio_config_hdr_t, [1675](#)
- Str_cp_in
 - L4::lpc::Str_cp_in< T >, [936](#)
- str_cp_in
 - L4::lpc, [555](#)

- String
 - `cx::String`, 755
- Strong
 - `L4Re::Namespace`, 1282
- `sub_type`
 - `L4::Kip::Mem_desc`, 1014
- Super_pages
 - `L4Re::Mem_alloc`, 1272
- supports
 - `L4::Meta`, 1037
- Suspend
 - `L4::Platform_control`, 1047
- `switch_log`
 - `L4::Debugger`, 810
- `switch_to`
 - `L4::Thread`, 1111
- symlink
 - `L4Re::Vfs::Directory`, 1387
- system_shutdown
 - `L4::Platform_control`, 1048
- system_suspend
 - `L4::Platform_control`, 1048
- tag
 - `L4::lpc::Istream`, 899
 - `L4::lpc::Ostream`, 929
 - `L4::lpc::Varg`, 941
- take
 - `L4Re::Util::Counting_cap_alloc< COUNTERTYPE >`, 1331
 - `L4Re::Util::Dataspace_svr`, 1337
- Task, 436
 - `L4_FP_ALL_SPACES`, 437
 - `L4_FP_DELETE_OBJ`, 437
 - `L4_FP_OTHER_SPACES`, 437
 - `I4_task_add_ku_mem`, 437
 - `I4_task_cap_equal`, 438
 - `I4_task_cap_valid`, 438
 - `I4_task_delete_obj`, 439
 - `I4_task_map`, 439
 - `I4_task_release_cap`, 440
 - `I4_task_unmap`, 441
 - `I4_task_unmap_batch`, 441
 - `I4_task_vgicc_map`, 442
 - `I4_unmap_flags_t`, 437
- task
 - `L4Re::Env`, 1253
 - `L4vcpu::Vcpu`, 1529
- test
 - `L4::Poll_timeout_kipclock`, 1051
- Thread, 443
 - `I4_thread_arm_set_tpidru0`, 447
 - `L4_THREAD_CONTROL_ALIEN`, 446
 - `L4_THREAD_CONTROL_BIND_TASK`, 446
 - `L4_thread_control_flags`, 445
 - `L4_THREAD_CONTROL_MR_IDX_BIND_TASK`, 446
 - `L4_THREAD_CONTROL_MR_IDX_BIND_UTCB`, 446
 - `L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER`, 446
 - `L4_THREAD_CONTROL_MR_IDX_FLAG_VALS`, 446
 - `L4_THREAD_CONTROL_MR_IDX_FLAGS`, 446
 - `L4_THREAD_CONTROL_MR_IDX_PAGER`, 446
 - `L4_thread_control_mr_indices`, 446
 - `L4_THREAD_CONTROL_SET_EXC_HANDLER`, 446
 - `L4_THREAD_CONTROL_SET_PAGER`, 446
 - `L4_THREAD_CONTROL_UX_NATIVE`, 446
 - `I4_thread_ex_regs`, 447
 - `L4_THREAD_EX_REGS_CANCEL`, 446
 - `L4_thread_ex_regs_flags`, 446
 - `I4_thread_ex_regs_ret`, 448
 - `I4_thread_ex_regs_ret_u`, 449
 - `L4_THREAD_EX_REGS_TRIGGER_EXCEPTION`, 446
 - `I4_thread_ex_regs_u`, 450
 - `I4_thread_modify_sender_add`, 451
 - `I4_thread_modify_sender_commit`, 452
 - `I4_thread_modify_sender_start`, 452
 - `I4_thread_register_del_irq`, 452
 - `I4_thread_stats_time`, 453
 - `I4_thread_switch`, 453
 - `I4_thread_vcpu_control`, 454
 - `I4_thread_vcpu_control_ext`, 454
 - `I4_thread_vcpu_control_ext_u`, 455
 - `I4_thread_vcpu_control_u`, 456
 - `I4_thread_vcpu_resume_commit`, 457
 - `I4_thread_vcpu_resume_start`, 458
 - `I4_thread_yield`, 458
- Thread control, 459
 - `I4_thread_control_alien`, 460
 - `I4_thread_control_bind`, 461
 - `I4_thread_control_commit`, 461
 - `I4_thread_control_exc_handler`, 463
 - `I4_thread_control_pager`, 463
 - `I4_thread_control_start`, 464
 - `I4_thread_control_ux_host_syscall`, 464
- Thread Control Registers (TCRs), 459
- `thread.h`
 - `__L4UTIL_THREAD_FUNC`, 2318
- `throw_error`
 - `L4Re`, 578
- `throw_ipc_exception`
 - `L4`, 546, 547
- `timed_out`
 - `L4::Poll_timeout_kipclock`, 1052
- timeout
 - `L4::lpc_svr::Timeout`, 975
- timeout_expired
 - `L4::lpc_svr::Timeout_queue`, 980
- Timeouts, 465
 - `I4_ipc_timeout`, 467
 - `L4_IPC_TIMEOUT_0`, 466
 - `I4_rcv_timeout`, 467
 - `I4_snd_timeout`, 468

- l4_timeout, 468
- l4_timeout_abs, 469
- l4_timeout_get, 470
- l4_timeout_is_absolute, 470
- l4_timeout_rel, 471
- l4_timeout_rel_get, 472
- l4_timeout_s, 466
- l4_timeout_t, 467
- l4_utcb_mr64_idx, 472
- Timestamp Counter, 473
- To_device
 - L4Re::Dma_space, 1238
- to_irq
 - L4vbus::Gpio_pin, 1490
- total_objects
 - cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, 619
 - cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, 626
- total_size
 - L4virtio::Virtqueue, 1658, 1659
- trigger
 - L4::Triggerable, 1122
- trunc_order
 - L4, 547
- Trusted
 - L4Re::Namespace, 1282
- type
 - L4::lpc::Varg, 941
 - L4::Kip::Mem_desc, 1015
 - L4Re::Vfs::Be_file_system, 1382
 - L4Re::Vfs::File_system, 1392
- types
 - L4_TYPES_FLAGS_OPS_DEF, 2090
- types.h
 - l4_capability_next, 2197
- unbind
 - L4::lcu, 853
 - L4::lommu, 864
- Uncacheable
 - L4Re::Dataspace::F, 1232
- Undefined
 - L4::Kip::Mem_desc, 1008
- Unique_cap
 - L4Re, 568
 - L4Re::Util, 585
- unique_cap
 - L4Re, 568
 - L4Re::Util, 586
- Unique_del_cap
 - L4Re, 568
 - L4Re::Util, 586
- unique_del_cap
 - L4Re, 569
 - L4Re::Util, 587
- unlink
 - L4Re::Namespace, 1284
 - L4Re::Vfs::Directory, 1387
- unlock_all_locks
 - L4Re::Vfs::Be_file, 1379
 - L4Re::Vfs::Generic_file, 1401
- unmap
 - L4::Task, 1101
 - L4Re::Dma_space, 1240
- unmap_batch
 - L4::Task, 1102
- unmask
 - L4::Irq, 991
 - L4::Irq_eoi, 994
- unregister_obj
 - L4::Registry_iface, 1061
 - L4Re::Util::Object_registry, 1362
- up
 - L4::Semaphore, 1074
- used_align
 - L4virtio::Virtqueue, 1659
- Used_elem
 - L4virtio::Virtqueue::Used_elem, 1670
- used_size
 - L4virtio::Virtqueue, 1660
- utcb_area
 - L4Re::Env, 1253
- util.h
 - l4_sleep, 1732, 1735
 - l4util_micros2l4to, 1733, 1736
- Utility Functions, 473
 - l4_sleep, 475
 - l4_touch_ro, 476
 - l4_touch_rw, 476
 - l4_usleep, 477
 - l4util_micros2l4to, 478
 - l4util_splitlog2_hdl, 478
 - l4util_splitlog2_size, 479
- ux_host_syscall
 - L4::Thread::Attr, 1117
- V_flags
 - L4Re::Video::View, 1434
- val
 - cxx::Bitfield< T, LSB, MSB >, 634
- val_dirty
 - cxx::Bitfield< T, LSB, MSB >, 636
- val_unshifted
 - cxx::Bitfield< T, LSB, MSB >, 637
- valid
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node, 674
 - L4virtio::Svr::Virtqueue::Head_desc, 1643
- validate
 - L4::Cap_base, 800
- value
 - L4::lpc::Varg, 941
- Varg_list_ref
 - L4::lpc::Varg_list_ref, 945
- vbe_ctrl_info
 - l4util_l4mod_info, 1456
- Vbus API, 495

- vbus.h
 - L4VBUS_ICU_SRC_DEV_HANDLE, [2332](#)
 - l4vbus_icu_src_types, [2332](#)
 - L4VBUS_NULL, [2332](#)
 - L4VBUS_ROOT_BUS, [2332](#)
- vbus_interfaces.h
 - L4VBUS_IFACE_SHIFT, [2335](#)
 - l4vbus_iface_type_t, [2335](#)
 - l4vbus_subinterface_supported, [2335](#)
- vbus_types.h
 - L4VBUS_DEVICE_F_CHILDREN, [2338](#)
 - l4vbus_device_flags_t, [2338](#)
 - L4VBUS_RESOURCE_BUS, [2339](#)
 - L4VBUS_RESOURCE_DMA_DOMAIN, [2339](#)
 - L4VBUS_RESOURCE_F_MEM_MMIO_READ, [2338](#)
 - L4VBUS_RESOURCE_F_MEM_MMIO_WRITE, [2338](#)
 - l4vbus_resource_flags_t, [2338](#)
 - L4VBUS_RESOURCE_GPIO, [2339](#)
 - L4VBUS_RESOURCE_INVALID, [2339](#)
 - L4VBUS_RESOURCE_IRQ, [2339](#)
 - L4VBUS_RESOURCE_MAX, [2339](#)
 - L4VBUS_RESOURCE_MEM, [2339](#)
 - L4VBUS_RESOURCE_PORT, [2339](#)
 - l4vbus_resource_type_t, [2339](#)
- vcon.h
 - L4_vcon_read_flags, [2217](#)
 - L4_VCON_READ_SIZE_MASK, [2217](#)
 - L4_VCON_READ_STAT_BREAK, [2217](#)
 - L4_VCON_READ_STAT_DONE, [2217](#)
- vCPU API, [523](#)
 - L4_VCPU_F_DEBUG_EXC, [525](#)
 - L4_VCPU_F_EXCEPTIONS, [525](#)
 - L4_VCPU_F_FPU_ENABLED, [525](#)
 - L4_VCPU_F_IRQ, [525](#)
 - L4_VCPU_F_PAGE_FAULTS, [525](#)
 - L4_VCPU_F_USER_MODE, [525](#)
 - L4_VCPU_OFFSET_EXT_INFOS, [525](#)
 - L4_VCPU_OFFSET_EXT_STATE, [525](#)
 - L4_VCPU_SF_IRQ_PENDING, [525](#)
 - L4_vcpu_state_flags, [525](#)
 - L4_vcpu_state_offset, [525](#)
 - L4_vcpu_sticky_flags, [525](#)
- vCPU Support Library, [526](#)
 - l4vcpu_irq_disable, [527](#)
 - l4vcpu_irq_disable_save, [527](#)
 - l4vcpu_irq_enable, [528](#)
 - l4vcpu_irq_restore, [529](#)
 - l4vcpu_is_irq_entry, [530](#)
 - l4vcpu_is_page_fault_entry, [531](#)
 - l4vcpu_print_state, [531](#)
 - l4vcpu_wait_for_event, [532](#)
- vcpu_control
 - L4::Thread, [1111](#)
- vcpu_control_ext
 - L4::Thread, [1112](#)
- vcpu_resume_commit
 - L4::Thread, [1113](#)
- vcpu_resume_start
 - L4::Thread, [1113](#)
- Video API, [497](#)
 - F_l4re_video_goos_auto_refresh, [499](#)
 - F_l4re_video_goos_dynamic_buffers, [499](#)
 - F_l4re_video_goos_dynamic_views, [499](#)
 - F_l4re_video_goos_pointer, [499](#)
 - F_l4re_video_view_above, [499](#)
 - F_l4re_video_view_dyn_allocated, [499](#)
 - F_l4re_video_view_flags_mask, [499](#)
 - F_l4re_video_view_none, [499](#)
 - F_l4re_video_view_set_background, [499](#)
 - F_l4re_video_view_set_buffer, [499](#)
 - F_l4re_video_view_set_buffer_offset, [499](#)
 - F_l4re_video_view_set_bytes_per_line, [499](#)
 - F_l4re_video_view_set_flags, [499](#)
 - F_l4re_video_view_set_pixel, [499](#)
 - F_l4re_video_view_set_position, [499](#)
 - l4re_video_goos_create_buffer, [499](#)
 - l4re_video_goos_create_view, [500](#)
 - l4re_video_goos_delete_buffer, [500](#)
 - l4re_video_goos_delete_view, [502](#)
 - l4re_video_goos_get_static_buffer, [502](#)
 - l4re_video_goos_get_view, [502](#)
 - l4re_video_goos_info, [503](#)
 - l4re_video_goos_info_flags_t, [499](#)
 - l4re_video_goos_refresh, [503](#)
 - l4re_video_view_get_info, [504](#)
 - l4re_video_view_info_flags_t, [499](#)
 - l4re_video_view_refresh, [504](#)
 - l4re_video_view_set_info, [504](#)
 - l4re_video_view_set_viewport, [505](#)
 - l4re_video_view_stack, [505](#)
 - l4re_video_view_t, [498](#)
- view
 - L4Re::Video::Goos, [1423](#)
- view_info
 - L4Re::Util::Video::Goos_svr, [1375](#)
- Virtual Console, [506](#)
 - L4_VCON_ECHO, [508](#)
 - l4_vcon_get_attr, [509](#)
 - l4_vcon_get_attr_u, [509](#)
 - L4_vcon_i_flags, [507](#)
 - L4_VCON_ICANON, [508](#)
 - L4_VCON_ICRNL, [508](#)
 - L4_VCON_IGNCR, [508](#)
 - L4_VCON_INLCR, [508](#)
 - L4_vcon_l_flags, [508](#)
 - L4_vcon_o_flags, [508](#)
 - L4_VCON_OCRNL, [508](#)
 - L4_VCON_ONLCR, [508](#)
 - L4_VCON_ONLRET, [508](#)
 - l4_vcon_read, [510](#)
 - L4_VCON_READ_SIZE, [509](#)
 - l4_vcon_read_u, [511](#)
 - l4_vcon_read_with_flags, [512](#)
 - l4_vcon_send, [513](#)

x86/l4/util/cpu.h, [1782](#), [1785](#)
x86/l4/util/idt.h, [1694](#), [1695](#)
x86/l4/util/irq.h, [1871](#), [1872](#)
x86/l4/util/l4_macros.h, [1788](#)
x86/l4/util/mbi_argv.h, [1791](#), [1792](#)
x86/l4/util/perform.h, [1701](#), [1702](#)
x86/l4/util/port_io.h, [1754](#), [1761](#)
x86/l4/util/rdtsc.h, [1719](#), [1726](#)
x86/l4/util/spin.h, [1730](#), [1731](#)
x86/l4/util/util.h, [1734](#), [1736](#)
x86/l4f/l4/sys/ipc.h, [2129](#), [2130](#)
x86/l4f/l4/sys/segment.h, [1745](#), [1746](#)
x86/l4f/l4/util/port_io.h, [1751](#), [1753](#)