

# diOS API Reference Manual.

0.1

Generated by Doxygen 1.7.2

Thu Jul 3 2014 02:17:39



# Contents

1	Module Index	1
1.1	Modules	1
2	Data Structure Index	3
2.1	Data Structures	3
3	File Index	5
3.1	File List	5
4	Module Documentation	7
4.1	OS_Debug	7
4.1.1	Define Documentation	8
4.1.1.1	OS_LOG_S	8
4.1.2	Function Documentation	8
4.1.2.1	OS_DebugDeInit	8
4.1.2.2	OS_DebugInit	9
4.1.2.3	OS_Log	9
4.1.2.4	OS_Trace	9
4.2	OS_Driver	10
4.2.1	Function Documentation	12
4.2.1.1	OS_DriverByNameGet	12
4.2.1.2	OS_DriverClose	12
4.2.1.3	OS_DriverConfigGet	12
4.2.1.4	OS_DriverCreate	12
4.2.1.5	OS_DriverDeInit	13
4.2.1.6	OS_DriverDelete	13
4.2.1.7	OS_DriverInit	13
4.2.1.8	OS_DriverIoCtl	13
4.2.1.9	OS_DriverNameGet	14
4.2.1.10	OS_DriverNextGet	14
4.2.1.11	OS_DriverOpen	14
4.2.1.12	OS_DriverParentGet	14
4.2.1.13	OS_DriverRead	15
4.2.1.14	OS_DriverStateNameGet	15
4.2.1.15	OS_DriverStateStateGet	15
4.2.1.16	OS_DriverStatsGet	16
4.2.1.17	OS_DriverWrite	16
4.3	OS_Environment	16
4.3.1	Function Documentation	17

4.3.1.1	OS_DriverStdIoGet	17
4.3.1.2	OS_LocaleGet	17
4.3.1.3	OS_LocaleSet	18
4.3.1.4	OS_LogLevelGet	18
4.3.1.5	OS_LogLevelSet	18
4.3.1.6	OS_PowerSet	18
4.3.1.7	OS_StdIoGet	18
4.3.1.8	OS_StdIoSet	19
4.4	OS_Event	19
4.4.1	Function Documentation	21
4.4.1.1	OS_EventCreate	21
4.4.1.2	OS_EventDelete	21
4.4.1.3	OS_EventItemCreate	21
4.4.1.4	OS_EventItemDelete	22
4.4.1.5	OS_EventItemLock	22
4.4.1.6	OS_EventItemOwnerAdd	22
4.4.1.7	OS_EventItemUnlock	22
4.4.1.8	OS_EventNextGet	23
4.4.1.9	OS_EventPeriodGet	23
4.4.1.10	OS_EventStateGet	23
4.4.1.11	OS_EventTimerGet	23
4.5	OS_List	24
4.5.1	Function Documentation	26
4.5.1.1	OS_ListAppend	26
4.5.1.2	OS_ListInit	26
4.5.1.3	OS_ListInsert	26
4.5.1.4	OS_ListItemByOwnerFind	26
4.5.1.5	OS_ListItemByValueFind	27
4.5.1.6	OS_ListItemCreate	27
4.5.1.7	OS_ListItemDelete	27
4.5.1.8	OS_ListItemInit	27
4.5.1.9	OS_ListItemsSwap	28
4.5.1.10	OS_ListRemove	28
4.6	OS_Memory	28
4.6.1	Function Documentation	30
4.6.1.1	OS_Free	30
4.6.1.2	OS_FreeEx	30
4.6.1.3	OS_Malloc	30
4.6.1.4	OS_MallocEx	30
4.6.1.5	OS_MemCacheFlush	31
4.6.1.6	OS_MemCpy32	31
4.6.1.7	OS_MemCpy8	31
4.6.1.8	OS_MemoryStatGet	32
4.6.1.9	OS_MemoryTypeHeapNextGet	32
4.7	OS_Message	32
4.7.1	Function Documentation	33
4.7.1.1	OS_MessageCreate	33
4.7.1.2	OS_MessageDelete	34
4.7.1.3	OS_MessageMulticastSend	34
4.7.1.4	OS_MessageReceive	34

4.7.1.5	OS_MessageSend	35
4.8	OS_Mutex	35
4.8.1	Function Documentation	36
4.8.1.1	OS_MutexCheck	36
4.8.1.2	OS_MutexCreate	37
4.8.1.3	OS_MutexDelete	37
4.8.1.4	OS_MutexLock	37
4.8.1.5	OS_MutexParentGet	37
4.8.1.6	OS_MutexRecursiveCheck	38
4.8.1.7	OS_MutexRecursiveCreate	38
4.8.1.8	OS_MutexRecursiveLock	38
4.8.1.9	OS_MutexRecursiveUnlock	38
4.8.1.10	OS_MutexUnlock	39
4.9	OS_Power	39
4.9.1	Function Documentation	40
4.9.1.1	OS_PowerInit	40
4.9.1.2	OS_PowerStateGet	40
4.9.1.3	OS_PowerStateNameGet	40
4.9.1.4	OS_PowerStateSet	40
4.10	OS_Queue	41
4.10.1	Function Documentation	42
4.10.1.1	OS_QueueConfigGet	42
4.10.1.2	OS_QueueCreate	43
4.10.1.3	OS_QueueDelete	43
4.10.1.4	OS_QueueFlush	43
4.10.1.5	OS_QueueItemsCountGet	43
4.10.1.6	OS_QueueNextGet	44
4.10.1.7	OS_QueueParentGet	44
4.10.1.8	OS_QueueReceive	44
4.10.1.9	OS_QueuesCountGet	44
4.10.1.10	OS_QueueSend	45
4.10.1.11	OS_QueueStatsGet	45
4.10.1.12	OS_QueueSvcStdInGet	45
4.11	OS_Semaphore	46
4.11.1	Function Documentation	47
4.11.1.1	OS_SemaphoreBinaryCreate	47
4.11.1.2	OS_SemaphoreCheck	47
4.11.1.3	OS_SemaphoreCountingCreate	47
4.11.1.4	OS_SemaphoreDelete	47
4.11.1.5	OS_SemaphoreLock	48
4.11.1.6	OS_SemaphoreUnlock	48
4.12	OS_Settings	48
4.12.1	Function Documentation	49
4.12.1.1	OS_SettingsDeInit	49
4.12.1.2	OS_SettingsDelete	49
4.12.1.3	OS_SettingsInit	50
4.12.1.4	OS_SettingsItemsRead	50
4.12.1.5	OS_SettingsItemsWrite	50
4.12.1.6	OS_SettingsRead	50
4.12.1.7	OS_SettingsWrite	51

4.13	OS_Shell	51
4.13.1	Function Documentation	52
4.13.1.1	OS_ShellArgumentsNumberCheck	52
4.13.1.2	OS_ShellCIHandler	53
4.13.1.3	OS_ShellCIs	53
4.13.1.4	OS_ShellCommandByNameGet	53
4.13.1.5	OS_ShellCommandCreate	53
4.13.1.6	OS_ShellCommandDelete	54
4.13.1.7	OS_ShellCommandExecute	54
4.13.1.8	OS_ShellCommandNextGet	54
4.13.1.9	OS_ShellInit	54
4.13.1.10	OS_ShellPromptGet	54
4.14	OS_Task	55
4.14.1	Function Documentation	58
4.14.1.1	OS_TaskAttrsGet	58
4.14.1.2	OS_TaskByNameGet	58
4.14.1.3	OS_TaskConfigGet	59
4.14.1.4	OS_TaskCreate	59
4.14.1.5	OS_TaskDelay	59
4.14.1.6	OS_TaskDelayUntil	60
4.14.1.7	OS_TaskDelete	60
4.14.1.8	OS_TaskHdByIdGet	60
4.14.1.9	OS_TaskHdGet	60
4.14.1.10	OS_TaskHdParentByHdGet	61
4.14.1.11	OS_TaskHdParentGet	61
4.14.1.12	OS_TaskIdGet	61
4.14.1.13	OS_TaskInit	61
4.14.1.14	OS_TaskMain	62
4.14.1.15	OS_TaskNameGet	62
4.14.1.16	OS_TaskNextGet	62
4.14.1.17	OS_TaskPower	62
4.14.1.18	OS_TaskPowerStateGet	63
4.14.1.19	OS_TaskPriorityGet	63
4.14.1.20	OS_TaskPrioritySet	63
4.14.1.21	OS_TaskResume	63
4.14.1.22	OS_TasksCountGet	64
4.14.1.23	OS_TasksStatsGet	64
4.14.1.24	OS_TaskStateGet	64
4.14.1.25	OS_TaskStateNameGet	64
4.14.1.26	OS_TaskStdIoGet	65
4.14.1.27	OS_TaskStorageGet	65
4.14.1.28	OS_TaskSuspend	65
4.14.1.29	OS_TaskSvcStdInGet	65
4.15	OS_Time	66
4.15.1	Function Documentation	68
4.15.1.1	OS_DateGet	68
4.15.1.2	OS_DateIsValid	68
4.15.1.3	OS_DateSet	68
4.15.1.4	OS_DateStringParse	69
4.15.1.5	OS_DateWeekDayGet	69

4.15.1.6	OS_TickCountGet	69
4.15.1.7	OS_TimeDayLightSavingsGet	70
4.15.1.8	OS_TimeDayLightSavingsSet	70
4.15.1.9	OS_TimeGet	70
4.15.1.10	OS_TimeIsValid	70
4.15.1.11	OS_TimeNameDayOfWeekGet	71
4.15.1.12	OS_TimeSet	71
4.15.1.13	OS_TimeStringParse	71
4.16	OS_Timer	72
4.16.1	Function Documentation	74
4.16.1.1	OS_TimerByIdGet	74
4.16.1.2	OS_TimerByNameGet	74
4.16.1.3	OS_TimerCreate	74
4.16.1.4	OS_TimerDelete	74
4.16.1.5	OS_TimerIdGet	75
4.16.1.6	OS_TimerIsActive	75
4.16.1.7	OS_TimerNameGet	75
4.16.1.8	OS_TimerNextGet	76
4.16.1.9	OS_TimerPeriodGet	76
4.16.1.10	OS_TimerPeriodSet	76
4.16.1.11	OS_TimerReset	77
4.16.1.12	OS_TimerStart	77
4.16.1.13	OS_TimerStatsGet	77
4.16.1.14	OS_TimerStop	77
4.17	ISR specific functions.	78
4.18	ISR specific functions.	78
4.18.1	Function Documentation	79
4.18.1.1	OS_ISR_DriverIoCtl	79
4.19	Environment variables user access functions.	79
4.19.1	Function Documentation	80
4.19.1.1	OS_EnvVariableDelete	80
4.19.1.2	OS_EnvVariableGet	80
4.19.1.3	OS_EnvVariableNextGet	80
4.19.1.4	OS_EnvVariableOwnerGet	81
4.19.1.5	OS_EnvVariableSet	81
4.20	ISR specific functions.	81
4.21	ISR specific functions.	82
4.21.1	Function Documentation	82
4.21.1.1	OS_ISR_MessageReceive	82
4.21.1.2	OS_ISR_MessageSend	82
4.22	ISR specific functions.	83
4.22.1	Function Documentation	83
4.22.1.1	OS_ISR_MutexCheck	83
4.22.1.2	OS_ISR_MutexLock	84
4.22.1.3	OS_ISR_MutexUnlock	84
4.23	ISR specific functions.	84
4.23.1	Function Documentation	85
4.23.1.1	OS_ISR_PowerStateSet	85
4.24	ISR specific functions.	85
4.24.1	Function Documentation	85

4.24.1.1	OS_ISR_QueueItemsCountGet	85
4.24.1.2	OS_ISR_QueueReceive	86
4.24.1.3	OS_ISR_QueueSend	86
4.25	ISR specific functions.	87
4.25.1	Function Documentation	87
4.25.1.1	OS_ISR_SemaphoreCheck	87
4.25.1.2	OS_ISR_SemaphoreLock	87
4.25.1.3	OS_ISR_SemaphoreUnlock	88
4.26	MPU specific functions.	88
4.27	ISR specific functions.	89
4.28	ISR specific functions.	89
4.28.1	Function Documentation	89
4.28.1.1	OS_ISR_TickCountGet	89
4.29	ISR specific functions.	90
4.29.1	Function Documentation	90
4.29.1.1	OS_ISR_TimerPeriodChange	90
4.29.1.2	OS_ISR_TimerReset	91
4.29.1.3	OS_ISR_TimerStart	91
4.29.1.4	OS_ISR_TimerStop	91
5	Data Structure Documentation	93
5.1	CommandDeviceDescription Struct Reference	93
5.1.1	Detailed Description	93
5.2	DeviceDescUnion Union Reference	94
5.2.1	Detailed Description	94
5.3	DeviceId Struct Reference	94
5.3.1	Detailed Description	94
5.4	DeviceRevision Struct Reference	95
5.4.1	Detailed Description	95
5.5	DeviceState Struct Reference	95
5.5.1	Detailed Description	96
5.6	HAL_DriverItf Struct Reference	96
5.6.1	Detailed Description	96
5.7	HAL_Env Struct Reference	97
5.7.1	Detailed Description	97
5.8	OS_DriverConfig Struct Reference	98
5.8.1	Detailed Description	98
5.9	OS_DriverStats Struct Reference	98
5.9.1	Detailed Description	99
5.10	OS_EventConfig Struct Reference	99
5.10.1	Detailed Description	99
5.11	OS_MemoryDesc Struct Reference	99
5.11.1	Detailed Description	100
5.12	OS_MemoryStat Struct Reference	100
5.12.1	Detailed Description	101
5.13	OS_Message Struct Reference	101
5.13.1	Detailed Description	101
5.14	OS_QueueConfig Struct Reference	101
5.14.1	Detailed Description	101
5.15	OS_QueueStats Struct Reference	102



5.15.1 Detailed Description . . . . .	102
5.16 OS_SettingsItem Struct Reference . . . . .	102
5.16.1 Detailed Description . . . . .	102
5.17 OS_ShellCommandConfig Struct Reference . . . . .	102
5.17.1 Detailed Description . . . . .	103
5.18 OS_TaskConfig Struct Reference . . . . .	103
5.18.1 Detailed Description . . . . .	103
5.19 OS_TimerConfig Struct Reference . . . . .	103
5.19.1 Detailed Description . . . . .	104
5.20 Packet Struct Reference . . . . .	104
5.20.1 Detailed Description . . . . .	104
5.21 ProtocolHeaderInfo Struct Reference . . . . .	104
5.21.1 Detailed Description . . . . .	104
5.22 ProtocolId Struct Reference . . . . .	105
5.22.1 Detailed Description . . . . .	105
5.23 RouteItem Struct Reference . . . . .	106
5.23.1 Detailed Description . . . . .	106
5.24 RouteListItem Struct Reference . . . . .	107
5.24.1 Detailed Description . . . . .	107
6 File Documentation . . . . .	109
6.1 crc32.c File Reference . . . . .	109
6.1.1 Detailed Description . . . . .	110
6.1.2 Function Documentation . . . . .	110
6.1.2.1 Crc32 . . . . .	110
6.1.2.2 Crc32Delta . . . . .	110
6.2 crc32.h File Reference . . . . .	111
6.2.1 Detailed Description . . . . .	112
6.2.2 Function Documentation . . . . .	112
6.2.2.1 Crc32 . . . . .	112
6.2.2.2 Crc32Delta . . . . .	113
6.3 crc8.c File Reference . . . . .	113
6.3.1 Detailed Description . . . . .	114
6.3.2 Function Documentation . . . . .	115
6.3.2.1 Crc8 . . . . .	115
6.3.2.2 Crc8Delta . . . . .	115
6.3.3 Variable Documentation . . . . .	116
6.3.3.1 crc_8_tbl . . . . .	116
6.4 crc8.h File Reference . . . . .	116
6.4.1 Detailed Description . . . . .	117
6.4.2 Function Documentation . . . . .	117
6.4.2.1 Crc8 . . . . .	117
6.4.2.2 Crc8Delta . . . . .	118
6.5 hal.h File Reference . . . . .	118
6.5.1 Detailed Description . . . . .	120
6.6 os_debug.h File Reference . . . . .	120
6.6.1 Detailed Description . . . . .	122
6.7 os_driver.h File Reference . . . . .	122
6.7.1 Detailed Description . . . . .	125
6.8 os_environment.h File Reference . . . . .	125

6.8.1	Detailed Description	127
6.9	os_event.h File Reference	127
6.9.1	Detailed Description	130
6.10	os_file_system.h File Reference	130
6.10.1	Detailed Description	131
6.11	os_list.h File Reference	131
6.11.1	Detailed Description	134
6.12	os_memory.h File Reference	134
6.12.1	Detailed Description	136
6.13	os_message.h File Reference	136
6.13.1	Detailed Description	137
6.14	os_mutex.h File Reference	137
6.14.1	Detailed Description	139
6.15	os_power.h File Reference	139
6.15.1	Detailed Description	141
6.16	os_queue.h File Reference	141
6.16.1	Detailed Description	144
6.17	os_semaphore.h File Reference	144
6.17.1	Detailed Description	146
6.18	os_settings.h File Reference	146
6.18.1	Detailed Description	148
6.19	os_shell.h File Reference	148
6.19.1	Detailed Description	150
6.20	os_task.h File Reference	151
6.20.1	Detailed Description	155
6.21	os_time.h File Reference	155
6.21.1	Detailed Description	158
6.22	os_timer.h File Reference	158
6.22.1	Detailed Description	161
6.23	protocol.h File Reference	162
6.23.1	Detailed Description	163
6.23.2	Enumeration Type Documentation	163
6.23.2.1	"@2	163
6.23.2.2	ProtocolCommand	164
6.23.3	Function Documentation	164
6.23.3.1	PACKED	164
6.24	status.h File Reference	164
6.24.1	Detailed Description	166
6.24.2	Define Documentation	166
6.24.2.1	IF_STATUS	166

# Chapter 1

## Module Index

### 1.1 Modules

Here is a list of all modules:

OS_Debug . . . . .	7
ISR specific functions. . . . .	78
OS_Driver . . . . .	10
ISR specific functions. . . . .	78
OS_Environment . . . . .	16
Environment variables user access functions. . . . .	79
OS_Event . . . . .	19
ISR specific functions. . . . .	81
OS_List . . . . .	24
OS_Memory . . . . .	28
OS_Message . . . . .	32
ISR specific functions. . . . .	82
OS_Mutex . . . . .	35
ISR specific functions. . . . .	83
OS_Power . . . . .	39
ISR specific functions. . . . .	84
OS_Queue . . . . .	41
ISR specific functions. . . . .	85
OS_Semaphore . . . . .	46
ISR specific functions. . . . .	87
OS_Settings . . . . .	48
OS_Shell . . . . .	51
OS_Task . . . . .	55
MPU specific functions. . . . .	88
ISR specific functions. . . . .	89
OS_Time . . . . .	66

ISR specific functions. . . . .	<a href="#">89</a>
OS_Timer . . . . .	<a href="#">72</a>
ISR specific functions. . . . .	<a href="#">90</a>

## Chapter 2

# Data Structure Index

### 2.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">CommandDeviceDescription</a> (Данные описания устройства ) . . . . .	93
<a href="#">DeviceDescUnion</a> (Дескриптор устройства ) . . . . .	94
<a href="#">DeviceId</a> . . . . .	94
<a href="#">DeviceRevision</a> . . . . .	95
<a href="#">DeviceState</a> (Полное состояние устройства ) . . . . .	95
<a href="#">HAL_DriverItf</a> . . . . .	96
<a href="#">HAL_Env</a> . . . . .	97
<a href="#">OS_DriverConfig</a> . . . . .	98
<a href="#">OS_DriverStats</a> . . . . .	98
<a href="#">OS_EventConfig</a> . . . . .	99
<a href="#">OS_MemoryDesc</a> (Memory description ) . . . . .	99
<a href="#">OS_MemoryStat</a> (Memory statistics ) . . . . .	100
<a href="#">OS_Message</a> . . . . .	101
<a href="#">OS_QueueConfig</a> . . . . .	101
<a href="#">OS_QueueStats</a> . . . . .	102
<a href="#">OS_SettingsItem</a> . . . . .	102
<a href="#">OS_ShellCommandConfig</a> . . . . .	102
<a href="#">OS_TaskConfig</a> . . . . .	103
<a href="#">OS_TimerConfig</a> . . . . .	103
<a href="#">Packet</a> (Пакет ) . . . . .	104
<a href="#">ProtocolHeaderInfo</a> . . . . .	104
<a href="#">ProtocolId</a> . . . . .	105
<a href="#">RouteItem</a> . . . . .	106
<a href="#">RouteListItem</a> . . . . .	107



## Chapter 3

# File Index

### 3.1 File List

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

<code>common.h</code>	??
<code>crc32.c</code> (CRC32)	109
<code>crc32.h</code> (CRC32)	111
<code>crc8.c</code> (CRC8)	113
<code>crc8.h</code> (CRC8)	116
<code>hal.h</code> (HAL)	118
<code>os_common.h</code>	??
<code>os_debug.h</code> (OS Debug)	120
<code>os_driver.h</code> (OS Driver)	122
<code>os_environment.h</code> (OS Environment)	125
<code>os_event.h</code> (OS Event)	127
<code>os_file_system.h</code> (OS File system)	130
<code>os_list.h</code> (OS List)	131
<code>os_memory.h</code> (OS Memory)	134
<code>os_message.h</code> (OS Message)	136
<code>os_mutex.h</code> (OS Mutex)	137
<code>os_network.h</code>	??
<code>os_power.h</code> (OS Power)	139
<code>os_queue.h</code> (OS Queue)	141
<code>os_semaphore.h</code> (OS Semaphore)	144
<code>os_settings.h</code> (OS Settings)	146
<code>os_shell.h</code> (OS Shell)	148
<code>os_signal.h</code>	??
<code>os_supervise.h</code>	??
<code>os_task.h</code> (OS Task)	151
<code>os_time.h</code> (OS Time)	155
<code>os_timer.h</code> (OS Timer)	158
<code>protocol.h</code> ()	162
<code>revision.h</code>	??

<a href="#">status.h</a> (Status codes ) . . . . .	<a href="#">164</a>
typedefs.h . . . . .	??
typedefs_app.h . . . . .	??

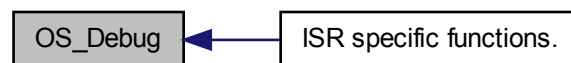


## Chapter 4

# Module Documentation

### 4.1 OS\_Debug

Collaboration diagram for OS\_Debug:



#### Modules

- [ISR specific functions.](#)

#### Defines

- `#define OS_ASSERT(a) D_ASSERT(a)`
- `#define OS_LOG(level,...) OS_Log(level, __VA_ARGS__)`
- `#define OS\_LOG\_S(level, status,...) OS_Log(level, StatusStringGet(status, MDL_STATUS_ITEMS))`  
Common status items array.
- `#define OS_TRACE(level, fmt_str_p,...) OS_Trace(level, fmt_str_p, __VA_ARGS__);`

## Typedefs

- typedef LogLevel [OS\\_LogLevel](#)  
Log level of tracing details.

## Functions

- Status [OS\\_DebugInit](#) (void)  
Init the debug module.
- Status [OS\\_DebugDeInit](#) (void)  
Deinit the debug module.
- void [OS\\_Log](#) (const [OS\\_LogLevel](#) level, ConstStrPtr format\_str\_p,...)  
  
Log the message.
- void [OS\\_Trace](#) (const [OS\\_LogLevel](#) level, ConstStrPtr format\_str\_p,...)  
  
Trace the message.

### 4.1.1 Define Documentation

- 4.1.1.1 `#define OS_LOG_S( level, status, ... ) OS_Log(level, StatusStringGet(status, MDL_STATUS_ITEMS))`

Common status items array.

Usage: Module using common one - do nothing. Module wants to use custom status items array: `undef MDL_STATUS_ITEMS define MDL_STATUS_ITEMS &status_items_app[0] // User status items array ----- extern const StatusItem status_items_app[];`

Status s = S\_CUSTOM; [OS\\_LOG\\_S\(D\\_DEBUG,s\); OS\\_LOG\\_S\(D\\_DEBUG,S\\_OK\);](#) Or you can directly point to the status items array: [OS\\_LOG\\_S\(D\\_WARNING,S\\_FS\\_UNMOUNTED, &status\\_fs\\_v\[0\]\);](#)

Definition at line 37 of file os\_debug.h.

### 4.1.2 Function Documentation

- 4.1.2.1 Status [OS\\_DebugDeInit](#) ( void )

Deinit the debug module.

### Returns

Status.

#### 4.1.2.2 Status OS\_DebugInit ( void )

Init the debug module.

### Returns

Status.

#### 4.1.2.3 void OS\_Log ( const OS\_LogLevel level, ConstStrPtr format\_str\_p, ... )

Log the message.

### Parameters

in	level	Level of details.
in	format_ - str_p	Format string pointer.

### Returns

None.

### Note

Writes log message to the STDOUT with the debug level and the task name.

#### 4.1.2.4 void OS\_Trace ( const OS\_LogLevel level, ConstStrPtr format\_str\_p, ... )

Trace the message.

### Parameters

in	level	Level of details.
in	format_ - str_p	Format string pointer.

### Returns

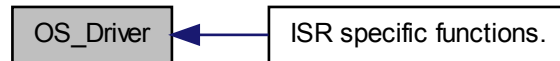
None.

### Note

Writes trace message to the STDOUT.

## 4.2 OS\_Driver

Collaboration diagram for OS\_Driver:



### Data Structures

- struct [OS\\_DriverStats](#)
- struct [OS\\_DriverConfig](#)

### Modules

- [ISR specific functions.](#)

### Typedefs

- typedef const void \* OS\_DriverHd
- typedef U8 OS\_DriverState

### Enumerations

- enum { OS\_DRV\_STATE\_UNDEF, OS\_DRV\_STATE\_IS\_INIT, OS\_DRV\_STATE\_IS\_OPEN, OS\_DRV\_STATE\_LAST = 7 }

### Functions

- Status [OS\\_DriverCreate](#) (const [OS\\_DriverConfig](#) \*cfg\_p, OS\_DriverHd \*dhd\_p)  
Create driver.
- Status [OS\\_DriverDelete](#) (const OS\_DriverHd dhd)  
Delete driver.
- Status [OS\\_DriverInit](#) (const OS\_DriverHd dhd)  
Init driver.

- Status [OS\\_DriverDeInit](#) (const OS\_DriverHd dhd)  
Deinit driver.
- Status [OS\\_DriverOpen](#) (const OS\_DriverHd dhd, void \*args\_p)  
Open driver.
- Status [OS\\_DriverClose](#) (const OS\_DriverHd dhd)  
Close driver.
- Status [OS\\_DriverRead](#) (const OS\_DriverHd dhd, void \*data\_in\_p, U32 size, void \*args\_p)  
Read data.
- Status [OS\\_DriverWrite](#) (const OS\_DriverHd dhd, void \*data\_out\_p, U32 size, void \*args\_p)  
Write data.
- Status [OS\\_DriverIoCtl](#) (const OS\_DriverHd dhd, const U32 request\_id, void \*args\_p)  
Input/Output control.
- ConstStrPtr [OS\\_DriverNameGet](#) (const OS\_DriverHd dhd)  
Get driver name.
- ConstStrPtr [OS\\_DriverStateNameGet](#) (const OS\_DriverState state)  
Get driver's state name.
- OS\_DriverHd [OS\\_DriverByNameGet](#) (ConstStrPtr name\_p)  
Get driver by it's name.
- OS\_DriverHd [OS\\_DriverNextGet](#) (const OS\_DriverHd dhd)  
Get the next driver.
- OS\_DriverState [OS\\_DriverStateStateGet](#) (const OS\_DriverHd dhd)  
Get driver state.
- Status [OS\\_DriverStatsGet](#) (const OS\_DriverHd dhd, [OS\\_DriverStats](#) \*stats\_p)  
Get driver stats.
- const [OS\\_DriverConfig](#) \* [OS\\_DriverConfigGet](#) (const OS\_DriverHd dhd)  
  
Get driver configuration.
- OS\_TaskHd [OS\\_DriverParentGet](#) (const OS\_DriverHd dhd)  
Get driver's parent.

### 4.2.1 Function Documentation

#### 4.2.1.1 OS\_DriverHd OS\_DriverByNameGet ( ConstStrPtr name\_p )

Get driver by it's name.

##### Parameters

in	name_p	Driver's name.
----	--------	----------------

##### Returns

Driver handle.

#### 4.2.1.2 Status OS\_DriverClose ( const OS\_DriverHd dhd )

Close driver.

##### Parameters

in	dhd	Driver's handle.
----	-----	------------------

##### Returns

Status.

#### 4.2.1.3 const OS\_DriverConfig\* OS\_DriverConfigGet ( const OS\_DriverHd dhd )

Get driver configuration.

##### Parameters

in	dhd	Driver's handle.
----	-----	------------------

##### Returns

Driver configuration.

#### 4.2.1.4 Status OS\_DriverCreate ( const OS\_DriverConfig \* cfg\_p, OS\_DriverHd \* dhd\_p )

Create driver.

##### Parameters

in	cfg_p	Driver's config.
out	dhd_p	Driver's handle.

## Returns

Status.

## 4.2.1.5 Status OS\_DriverDeInit ( const OS\_DriverHd dhd )

Deinit driver.

## Parameters

in	dhd	Driver's handle.
----	-----	------------------

## Returns

Status.

## 4.2.1.6 Status OS\_DriverDelete ( const OS\_DriverHd dhd )

Delete driver.

## Parameters

in	dhd	Driver's handle.
----	-----	------------------

## Returns

Status.

## 4.2.1.7 Status OS\_DriverInit ( const OS\_DriverHd dhd )

Init driver.

## Parameters

in	dhd	Driver's handle.
----	-----	------------------

## Returns

Status.

## 4.2.1.8 Status OS\_DriverIoCtl ( const OS\_DriverHd dhd, const U32 request\_id, void \* args\_p )

Input/Output control.

## Parameters

in	dhd	Driver's handle.
in	request_id	Driver's request code identifier.
in	args_p	Driver's specific input arguments (if presents).

Returns

Status.

4.2.1.9 ConstStrPtr OS\_DriverNameGet ( const OS\_DriverHd dhd )

Get driver name.

Parameters

in	dhd	Driver's handle.
----	-----	------------------

Returns

Name.

4.2.1.10 OS\_DriverHd OS\_DriverNextGet ( const OS\_DriverHd dhd )

Get the next driver.

Parameters

in	dhd	Driver's handle.
----	-----	------------------

Returns

Driver handle.

4.2.1.11 Status OS\_DriverOpen ( const OS\_DriverHd dhd, void \* args\_p )

Open driver.

Parameters

in	dhd	Driver's handle.
in	args_p	Driver's input arguments.

Returns

Status.

4.2.1.12 OS\_TaskHd OS\_DriverParentGet ( const OS\_DriverHd dhd )

Get driver's parent.

Parameters

in	dhd	Driver's handle.
----	-----	------------------



## Returns

Task handle.

4.2.1.13 Status OS\_DriverRead ( const OS\_DriverHd dhd, void \*  
data\_in\_p, U32 size, void \* args\_p )

Read data.

## Parameters

in	dhd	Driver's handle.
out	data_in_p	Data input buffer.
in	size	Data input buffer size.
in	args_p	Driver's specific input arguments (if presents).

## Returns

Status.

4.2.1.14 ConstStrPtr OS\_DriverStateNameGet ( const OS\_DriverState  
state )

Get driver's state name.

## Parameters

in	state	Driver's state.
----	-------	-----------------

## Returns

State name.

4.2.1.15 OS\_DriverState OS\_DriverStateStateGet ( const OS\_DriverHd  
dhd )

Get driver state.

## Parameters

in	dhd	Driver's handle.
----	-----	------------------

## Returns

Driver state.

4.2.1.16 Status OS\_DriverStatsGet ( const OS\_DriverHd dhd,  
OS\_DriverStats \* stats\_p )

Get driver stats.

Parameters

in	dhd	Driver's handle.
out	stats_p	Stats.

Returns

Status.

4.2.1.17 Status OS\_DriverWrite ( const OS\_DriverHd dhd, void \*  
data\_out\_p, U32 size, void \* args\_p )

Write data.

Parameters

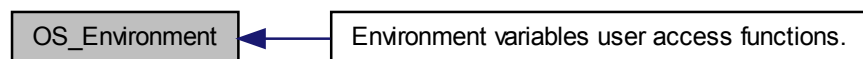
in	dhd	Driver's handle.
in	data_out_p	Data output buffer.
in	size	Data output buffer size.
in	args_p	Driver's specific input arguments (if presents).

Returns

Status.

## 4.3 OS\_Environment

Collaboration diagram for OS\_Environment:



Modules

- [Environment variables user access functions.](#)

## Defines

- `#define OS_ENV_POWER_STR "power"`

## Functions

- `OS_DriverHd OS_DriverStdIoGet (void)`  
Get system input/output driver.
- `Locale OS_LocaleGet (void)`  
Get current system locale.
- `Status OS_LocaleSet (ConstStrPtr locale_p)`  
Set the current system locale.
- `Status OS_PowerSet (ConstStrPtr power_p)`  
Set the current system power mode.
- `const HAL_DriverItf * OS_StdIoGet (void)`  
Get system input/output driver interface.
- `Status OS_StdIoSet (ConstStrPtr drv_name_p)`  
Set system input/output driver.
- `OS_LogLevel OS_LogLevelGet (void)`  
Get current log level of trace details.
- `Status OS_LogLevelSet (ConstStrPtr log_level_p)`  
Set current log level of trace details.

### 4.3.1 Function Documentation

#### 4.3.1.1 `OS_DriverHd OS_DriverStdIoGet ( void )`

Get system input/output driver.

##### Returns

Driver handle.

#### 4.3.1.2 `Locale OS_LocaleGet ( void )`

Get current system locale.

##### Returns

Locale.

#### 4.3.1.3 Status OS\_LocaleSet ( ConstStrPtr locale\_p )

Set the current system locale.

Parameters

in	locale_p	Locale.
----	----------	---------

Returns

Status.

#### 4.3.1.4 OS\_LogLevel OS\_LogLevelGet ( void )

Get current log level of trace details.

Returns

Log level.

#### 4.3.1.5 Status OS\_LogLevelSet ( ConstStrPtr log\_level\_p )

Set current log level of trace details.

Parameters

in	log_level_p	Log level name.
----	-------------	-----------------

Returns

Status.

#### 4.3.1.6 Status OS\_PowerSet ( ConstStrPtr power\_p )

Set the current system power mode.

Parameters

in	power_p	Power mode.
----	---------	-------------

Returns

Status.

#### 4.3.1.7 const HAL\_DriverItf\* OS\_StdIoGet ( void )

Get system input/output driver interface.

## Returns

Driver interface.

## 4.3.1.8 Status OS\_StdIoSet ( ConstStrPtr drv\_name\_p )

Set system input/output driver.

## Parameters

in	drv_ name_p	Driver name.
----	----------------	--------------

## Returns

Status.

## 4.4 OS\_Event

Collaboration diagram for OS\_Event:



## Data Structures

- struct [OS\\_EventConfig](#)

## Modules

- [ISR specific functions.](#)

## Typedefs

- typedef void \* OS\_EventHd
- typedef U8 OS\_EventState
- typedef OS\_StorageItem OS\_EventItem

## Enumerations

- enum { OS\_EVENT\_STATE\_UNDEF, OS\_EVENT\_STATE\_LAST }

## Functions

- Status [OS\\_EventCreate](#) (const [OS\\_EventConfig](#) \*cfg\_p, OS\_EventHd \*ehd\_p)  
Create an event.
- Status [OS\\_EventDelete](#) (const OS\_EventHd ehd, const TimeMs timeout)  
Delete the event.
- Status [OS\\_EventTimerGet](#) (const OS\_EventHd ehd, OS\_TimerHd \*timer\_hd\_p)  
Get the event timer.
- Status [OS\\_EventStateGet](#) (const OS\_EventHd ehd, OS\_EventState \*state\_p)  
Get the event state.
- Status [OS\\_EventPeriodGet](#) (const OS\_EventHd ehd, TimeMs \*period\_p)  
Get the event state.
- Status OS\_EventStatePeriodSet (const OS\_EventHd ehd, const TimeMs new\_period, const OS\_EventState new\_state, const TimeMs timeout)
- Status [OS\\_EventItemCreate](#) (const void \*data\_p, const U16 size, OS\_EventItem \*\*item\_pp)  
Create an event item.
- Status [OS\\_EventItemDelete](#) (OS\_EventItem \*item\_p)  
Delete the event item.
- Status [OS\\_EventItemOwnerAdd](#) (OS\_EventItem \*item\_p)  
Add event item owner.
- Status [OS\\_EventItemLock](#) (OS\_EventItem \*item\_p, const TimeMs timeout)  
Lock event item.
- Status [OS\\_EventItemUnlock](#) (OS\_EventItem \*item\_p)  
Unlock event item.
- OS\_EventItem \* OS\_EventItemGet (const OS\_EventHd ehd)

- OS\_EventItem \* OS\_EventItemByTimerIdGet (const OS\_TimerId timer\_id)
- OS\_EventItem \* OS\_EventItemByStateGet (const OS\_EventState state)
- OS\_EventHd [OS\\_EventNextGet](#) (const OS\_EventHd ehd)  
Get the next event.

#### 4.4.1 Function Documentation

4.4.1.1 Status OS\_EventCreate ( const OS\_EventConfig \* cfg\_p,  
OS\_EventHd \* ehd\_p )

Create an event.

Parameters

in	cfg_p	Event config.
out	ehd_p	Event handle.

Returns

Status.

4.4.1.2 Status OS\_EventDelete ( const OS\_EventHd ehd, const TimeMs  
timeout )

Delete the event.

Parameters

in	ehd	Event handle.
in	timeout	Operation timeout (ticks).

Returns

Status.

4.4.1.3 Status OS\_EventItemCreate ( const void \* data\_p, const U16  
size, OS\_EventItem \*\* item\_pp )

Create an event item.

Parameters

in	data_p	Item's data.
in	size	Items's data size.
out	item_pp	Item.

Returns

Status.

4.4.1.4 Status OS\_EventItemDelete ( OS\_EventItem \* item\_p )

Delete the event item.

Parameters

in	item_p	Item.
----	--------	-------

Returns

Status.

4.4.1.5 Status OS\_EventItemLock ( OS\_EventItem \* item\_p, const  
TimeMs timeout )

Lock event item.

Parameters

in	item_p	Item.
----	--------	-------

Returns

Status.

4.4.1.6 Status OS\_EventItemOwnerAdd ( OS\_EventItem \* item\_p )

Add event item owner.

Parameters

in	item_p	Item.
----	--------	-------

Returns

Status.

4.4.1.7 Status OS\_EventItemUnlock ( OS\_EventItem \* item\_p )

Unlock event item.

Parameters

in	item_p	Item.
----	--------	-------



## Returns

Status.

## 4.4.1.8 OS\_EventHd OS\_EventNextGet ( const OS\_EventHd ehd )

Get the next event.

## Parameters

in	ehd	Event handle.
----	-----	---------------

## Returns

Event handle.

## 4.4.1.9 Status OS\_EventPeriodGet ( const OS\_EventHd ehd, TimeMs \* period\_p )

Get the event state.

## Parameters

in	ehd	Event handle.
out	period_p	Period.

## Returns

Status.

## 4.4.1.10 Status OS\_EventStateGet ( const OS\_EventHd ehd, OS\_EventState \* state\_p )

Get the event state.

## Parameters

in	ehd	Event handle.
out	state_p	State.

## Returns

Status.

## 4.4.1.11 Status OS\_EventTimerGet ( const OS\_EventHd ehd, OS\_TimerHd \* timer\_hd\_p )

Get the event timer.

## Parameters

in	ehd	Event handle.
out	timer_ hd_p	Timer handle.

## Returns

Status.

## 4.5 OS\_List

## Defines

- #define OS\_LIST\_CURRENT\_LEN\_GET(OS\_ListP) listCURRENT\_  
LIST\_LENGTH(OS\_ListP)
- #define OS\_LIST\_IS\_EMPTY(OS\_ListP) listLIST\_IS\_EMPTY(OS\_  
ListP)
- #define OS\_LIST\_ITEM\_FIRST\_VALUE\_GET(OS\_ListP) listGET\_  
ITEM\_VALUE\_OF\_HEAD\_ENTRY(OS\_ListP)
- #define OS\_LIST\_ITEM\_VALUE\_GET(OS\_ListItemP) listGET\_LIST\_  
ITEM\_VALUE(OS\_ListItemP)
- #define OS\_LIST\_ITEM\_VALUE\_SET(OS\_ListItemP, OS\_Value) listSET\_  
LIST\_ITEM\_VALUE(OS\_ListItemP, OS\_Value)
- #define OS\_LIST\_ITEM\_OWNER\_GET(OS\_ListItemP) listGET\_LIST\_  
ITEM\_OWNER(OS\_ListItemP)
- #define OS\_LIST\_ITEM\_OWNER\_SET(OS\_ListItemP, OS\_Owner) listSET\_  
LIST\_ITEM\_OWNER(OS\_ListItemP, OS\_Owner)
- #define OS\_LIST\_ITEM\_PREVIOUS\_GET(OS\_ListItemP) ((OS\_ListItemP)-  
>pxPrevious)
- #define OS\_LIST\_ITEM\_NEXT\_GET(OS\_ListItemP) listGET\_NEXT(OS\_  
ListItemP)
- #define OS\_LIST\_ITEM\_FIRST\_GET(OS\_ListP) listGET\_HEAD\_  
ENTRY(OS\_ListP)
- #define OS\_LIST\_ITEM\_LAST\_GET(OS\_ListP) ((OS\_ListP)->xListEnd)
- #define OS\_LIST\_ITEM\_NEXT\_OWNER\_GET(OS\_Owner, OS\_ListP) listGET\_  
OWNER\_OF\_NEXT\_ENTRY(OS\_Owner, OS\_ListP)
- #define OS\_LIST\_ITEM\_FIRST\_OWNER\_GET(OS\_ListP) listGET\_  
OWNER\_OF\_HEAD\_ENTRY(OS\_ListP)
- #define OS\_LIST\_ITEM\_IS\_WITHIN(OS\_ListP, OS\_ListItemP) listIS\_  
CONTAINED\_WITHIN(OS\_ListP, OS\_ListItemP)
- #define OS\_LIST\_ITEM\_CONTAINER\_GET(OS\_ListItemP) listLIST\_  
ITEM\_CONTAINER(OS\_ListItemP)
- #define OS\_LIST\_IS\_INITIALISED(OS\_ListP) listLIST\_IS\_INITIALISED(OS\_  
ListP)

## Typedefs

- typedef List\_t OS\_List
- typedef ListItem\_t OS\_ListItem
- typedef MiniListItem\_t OS\_ListItemLight

## Functions

- void [OS\\_ListInit](#) (OS\_List \*list\_p)  
Initialise the list.
- OS\_ListItem \* [OS\\_ListItemCreate](#) (void)  
Create and initialize the list item.
- void [OS\\_ListItemDelete](#) (OS\_ListItem \*item\_l\_p)  
Remove and delete item from the list.
- void [OS\\_ListItemInit](#) (OS\_ListItem \*item\_l\_p)  
Initialise the list item.
- void [OS\\_ListInsert](#) (OS\_List \*list\_p, OS\_ListItem \*new\_item\_l\_p)  
Insert item to the list.
- void [OS\\_ListAppend](#) (OS\_List \*list\_p, OS\_ListItem \*new\_item\_l\_p)  
Append item to the list.
- U32 [OS\\_ListRemove](#) (OS\_ListItem \*item\_l\_p)  
Remove item from the list.
- OS\_ListItem \* [OS\\_ListItemByValueFind](#) (OS\_List \*list\_p, const OS\_Value value)  
Find list item by it's value.
- OS\_ListItem \* [OS\\_ListItemByOwnerFind](#) (OS\_List \*list\_p, const OS\_Owner owner)  
Find list item by it's owner.
- void [OS\\_ListItemsSwap](#) (OS\_ListItem \*item\_l\_p, OS\_ListItem \*item\_l\_2\_p)  
Swap list items.

### 4.5.1 Function Documentation

4.5.1.1 void OS\_ListAppend ( OS\_List \* list\_p, OS\_ListItem \* new\_item\_l\_p )

Append item to the list.

Parameters

in	list_p	List.
in	new_item_l_p	New list item.

Returns

None.

4.5.1.2 void OS\_ListInit ( OS\_List \* list\_p )

Initialise the list.

Parameters

in	list_p	List.
----	--------	-------

Returns

None.

4.5.1.3 void OS\_ListInsert ( OS\_List \* list\_p, OS\_ListItem \* new\_item\_l\_p )

Insert item to the list.

Parameters

in	list_p	List.
in	new_item_l_p	New list item.

Returns

None.

4.5.1.4 OS\_ListItem\* OS\_ListItemByOwnerFind ( OS\_List \* list\_p, const OS\_Owner owner )

Find list item by it's owner.

## Parameters

in	list_p	List.
in	owner	Owner.

## Returns

List item.

4.5.1.5 OS\_ListItem\* OS\_ListItemByValueFind ( OS\_List \* list\_p, const OS\_Value value )

Find list item by it's value.

## Parameters

in	list_p	List.
in	value	Value.

## Returns

List item.

4.5.1.6 OS\_ListItem\* OS\_ListItemCreate ( void )

Create and initialize the list item.

## Returns

List item.

4.5.1.7 void OS\_ListItemDelete ( OS\_ListItem \* item\_l\_p )

Remove and delete item from the list.

## Parameters

in	item_l_p	List item.
----	----------	------------

## Returns

None.

4.5.1.8 void OS\_ListItemInit ( OS\_ListItem \* item\_l\_p )

Initialise the list item.

## Parameters

in	item_1_p	List item.
----	----------	------------

## Returns

None.

4.5.1.9 void OS\_ListItemsSwap ( OS\_ListItem \* item\_1\_p, OS\_ListItem \* item\_2\_p )

Swap list items.

## Parameters

in	item_1_p	List item first.
in	item_2_p	List item second.

## Returns

None.

4.5.1.10 U32 OS\_ListRemove ( OS\_ListItem \* item\_1\_p )

Remove item from the list.

## Parameters

in	item_1_p	List item.
----	----------	------------

## Returns

The number of items that remain in the list.

## 4.6 OS\_\_Memory

### Data Structures

- struct [OS\\_\\_MemoryDesc](#)  
Memory description.
- struct [OS\\_\\_MemoryStat](#)  
Memory statistics.

### Typedefs

- typedef U32 OS\_\_MemoryType

## Enumerations

- enum {  
    OS\_MEM\_RAM\_INT\_SRAM, OS\_MEM\_RAM\_INT\_CCM, OS\_MEM\_RAM\_EXT\_SRAM, OS\_MEM\_LAST,  
    OS\_MEM\_UNDEF }  
    Memory type.

## Functions

- void \* [OS\\_Malloc](#) (const U32 size)  
    Common functions.
- void \* [OS\\_MallocEx](#) (const U32 size, const OS\_MemoryType mem\_type)  
    Allocate memory by type.
- void [OS\\_Free](#) (void \*addr\_p)  
    Free allocated memory.
- void [OS\\_FreeEx](#) (void \*addr\_p, const OS\_MemoryType mem\_type)  
    Free allocated memory by type.
- void [OS\\_MemCacheFlush](#) (void)  
    Flush memory caches.
- void [OS\\_MemCpy8](#) (void \*dst\_p, const void \*src\_p, SIZE size8)  
    Copy memory in bytes.
- void [OS\\_MemCpy32](#) (void \*dst\_p, const void \*src\_p, SIZE size32)  
    Copy memory in words.
- OS\_MemoryType [OS\\_MemoryTypeHeapNextGet](#) (const OS\_MemoryType mem\_type)  
    Get the next memory heap type.
- Status [OS\\_MemoryStatGet](#) (const OS\_MemoryType mem\_type, [OS\\_MemoryStat](#) \*mem\_stat\_p)  
    Get the memory heap usage statistics.

### 4.6.1 Function Documentation

#### 4.6.1.1 void OS\_Free ( void \* addr\_p )

Free allocated memory.

Parameters

in	addr_p	Memory address.
----	--------	-----------------

Returns

None.

#### 4.6.1.2 void OS\_FreeEx ( void \* addr\_p, const OS\_MemoryType mem\_type )

Free allocated memory by type.

Parameters

in	addr_p	Memory address.
in	mem_type	Memory type.

Returns

None.

#### 4.6.1.3 void\* OS\_Malloc ( const U32 size )

Common functions.

Allocate memory.

Parameters

in	size	Allocation size (in bytes).
----	------	-----------------------------

Returns

Memory pointer.

Tries to allocate memory in first memory pool of config.

#### 4.6.1.4 void\* OS\_MallocEx ( const U32 size, const OS\_MemoryType mem\_type )

Allocate memory by type.



## Parameters

in	size	Allocation size (in bytes).
in	mem_type	Memory type.

## Returns

Memory pointer.

## 4.6.1.5 void OS\_MemCacheFlush ( void )

Flush memory caches.

## Returns

None.

## 4.6.1.6 void OS\_MemCpy32 ( void \* dst\_p, const void \* src\_p, SIZE size32 )

Copy memory in words.

## Parameters

out	dst_p	Destination address.
in	src_p	Source address.
in	size32	Size to copy (words).

## Returns

None.

## 4.6.1.7 void OS\_MemCpy8 ( void \* dst\_p, const void \* src\_p, SIZE size8 )

Copy memory in bytes.

## Parameters

out	dst_p	Destination address.
in	src_p	Source address.
in	size8	Size to copy (bytes).

## Returns

None.

4.6.1.8 Status OS\_MemoryStatGet ( const OS\_MemoryType mem\_type,  
OS\_MemoryStat \* mem\_stat\_p )

Get the memory heap usage statistics.

Parameters

in	mem_type	Memory type.
out	mem_stat_p	Memory statistics.

Returns

Status.

4.6.1.9 OS\_MemoryType OS\_MemoryTypeHeapNextGet ( const  
OS\_MemoryType mem\_type )

Get the next memory heap type.

Parameters

in	mem_type	Memory type.
----	----------	--------------

Returns

Memory type.

## 4.7 OS\_Message

Collaboration diagram for OS\_Message:



Data Structures

- struct [OS\\_Message](#)

## Modules

- [ISR specific functions.](#)

## Typedefs

- typedef U16 OS\_MessageId

## Enumerations

- enum { OS\_MSG\_UNDEF, OS\_MSG\_BROADCAST, OS\_MSG\_CMD, OS\_MSG\_APP = 32 }

## Functions

- [OS\\_Message \\* OS\\_MessageCreate](#) (const OS\_MessageId id, const U16 size, const TimeMs timeout, const void \*data\_p)  
Create a message.
- void [OS\\_MessageDelete](#) (OS\_Message \*msg\_p)  
Delete the message.
- Status [OS\\_MessageSend](#) (const OS\_QueueHd qhd, const OS\_Message \*msg\_p, const TimeMs timeout, const OS\_MessagePrio priority)  
Send the message.
- Status [OS\\_MessageMulticastSend](#) (const OS\_QueueHd receivers\_qhd\_v[], const OS\_Message \*msg\_p, const TimeMs timeout, const OS\_MessagePrio priority)  
Send the multicast message.
- Status [OS\\_MessageReceive](#) (const OS\_QueueHd qhd, OS\_Message \*\*msg\_pp, const TimeMs timeout)  
Receive the message.

### 4.7.1 Function Documentation

- 4.7.1.1 OS\_Message\* OS\_MessageCreate ( const OS\_MessageId id, const U16 size, const TimeMs timeout, const void \* data\_p )

Create a message.

#### Parameters

in	id	Message id.
----	----	-------------

in	size	Message size.
in	timeout	Message creation timeout.
in	data_p	Message data.

Returns

Message.

4.7.1.2 void OS\_MessageDelete ( OS\_Message \* msg\_p )

Delete the message.

Parameters

in	msg_p	Message.
----	-------	----------

Returns

None.

4.7.1.3 Status OS\_MessageMulticastSend ( const OS\_QueueHd receivers\_qhd\_v[], const OS\_Message \* msg\_p, const TimeMs timeout, const OS\_MessagePrio priority )

Send the multicast message.

Parameters

in	qhd_v	Vector of receiver (tasks) queue handles with trailing OS_ - NULL;
in	msg_p	Message.
in	timeout	Message sending timeout.
in	priority	Message sending priority.

Returns

Status.

4.7.1.4 Status OS\_MessageReceive ( const OS\_QueueHd qhd, OS\_Message \*\* msg\_pp, const TimeMs timeout )

Receive the message.

Parameters

in	qhd	Receiver (task) queue handle.
out	msg_pp	Message.
in	timeout	Message receiving timeout.

## Returns

Status.

```
4.7.1.5 Status OS_MessageSend ( const OS_QueueHd qhd, const
    OS_Message * msg_p, const TimeMs timeout, const
    OS_MessagePrio priority )
```

Send the message.

## Parameters

in	qhd	Receiver (task) queue handle.
in	msg_p	Message.
in	timeout	Message sending timeout.
in	priority	Message sending priority.

## Returns

Status.

## 4.8 OS\_Mutex

Collaboration diagram for OS\_Mutex:



## Modules

- [ISR specific functions.](#)

## Typedefs

- typedef OS\_SemaphoreHd OS\_MutexHd
- typedef OS\_SemaphoreState OS\_MutexState

## Functions

- OS\_MutexHd [OS\\_MutexCreate](#) (void)

Create a mutex.

- OS\_MutexHd [OS\\_MutexRecursiveCreate](#) (void)  
Create a recursive mutex.
- void [OS\\_MutexDelete](#) (const OS\_MutexHd mhd)  
Delete the mutex.
- Status [OS\\_MutexLock](#) (const OS\_MutexHd mhd, const TimeMs timeout)  
Lock the mutex.
- Status [OS\\_MutexRecursiveLock](#) (const OS\_MutexHd mhd, const TimeMs timeout)  
Recursive lock the mutex.
- Status [OS\\_MutexUnlock](#) (const OS\_MutexHd mhd)  
Unlock the mutex.
- Status [OS\\_MutexRecursiveUnlock](#) (const OS\_MutexHd mhd)  
Recursive unlock the mutex.
- OS\_MutexState [OS\\_MutexCheck](#) (const OS\_MutexHd mhd)  
Check mutex state.
- OS\_MutexState [OS\\_MutexRecursiveCheck](#) (const OS\_MutexHd mhd)  
Check recursive mutex state.
- OS\_TaskHd [OS\\_MutexParentGet](#) (const OS\_MutexHd mhd)  
Get mutex parent.

#### 4.8.1 Function Documentation

##### 4.8.1.1 OS\_MutexState OS\_MutexCheck ( const OS\_MutexHd mhd )

Check mutex state.

Parameters

in	mhd	Mutex handle.
----	-----	---------------

Returns

Mutex state.

#### 4.8.1.2 OS\_MutexHd OS\_MutexCreate ( void )

Create a mutex.

Returns

Mutex handle.

#### 4.8.1.3 void OS\_MutexDelete ( const OS\_MutexHd mhd )

Delete the mutex.

Parameters

in	mhd	Mutex handle.
----	-----	---------------

Returns

None.

#### 4.8.1.4 Status OS\_MutexLock ( const OS\_MutexHd mhd, const TimeMs timeout )

Lock the mutex.

Parameters

in	mhd	Mutex handle.
in	timeout	Mutex locking timeout.

Returns

Status.

#### 4.8.1.5 OS\_TaskHd OS\_MutexParentGet ( const OS\_MutexHd mhd )

Get mutex parent.

Parameters

in	mhd	Mutex handle.
----	-----	---------------

Returns

Task handle.

#### 4.8.1.6 OS\_MutexState OS\_MutexRecursiveCheck ( const OS\_MutexHd mhd )

Check recursive mutex state.

Parameters

in	mhd	Mutex handle.
----	-----	---------------

Returns

Mutex state.

#### 4.8.1.7 OS\_MutexHd OS\_MutexRecursiveCreate ( void )

Create a recursive mutex.

Returns

Mutex handle.

#### 4.8.1.8 Status OS\_MutexRecursiveLock ( const OS\_MutexHd mhd, const TimeMs timeout )

Recursive lock the mutex.

Parameters

in	mhd	Mutex handle.
in	timeout	Mutex locking timeout.

Returns

Status.

#### 4.8.1.9 Status OS\_MutexRecursiveUnlock ( const OS\_MutexHd mhd )

Recursive unlock the mutex.

Parameters

in	mhd	Mutex handle.
----	-----	---------------

Returns

Status.



## 4.8.1.10 Status OS\_MutexUnlock ( const OS\_MutexHd mhd )

Unlock the mutex.

## Parameters

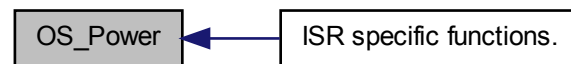
in	mhd	Mutex handle.
----	-----	---------------

## Returns

Status.

## 4.9 OS\_Power

Collaboration diagram for OS\_Power:



## Modules

- [ISR specific functions.](#)

## Typedefs

- typedef HAL\_PowerState OS\_PowerState
- typedef HAL\_PowerPrio OS\_PowerPrio

## Enumerations

- enum { OS\_PWR\_PRIO\_UNDEF, OS\_PWR\_PRIO\_DEFAULT = 1, OS\_PWR\_PRIO\_MAX = 255, OS\_PWR\_PRIO\_LAST = OS\_PWR\_PRIO\_MAX }

## Functions

- Status [OS\\_PowerInit](#) (void)  
Init power.

- OS\_PowerState [OS\\_PowerStateGet](#) (void)  
Get current system power state.
- Status [OS\\_PowerStateSet](#) (const OS\_PowerState state)  
Set current system power state.
- ConstStrPtr [OS\\_PowerStateNameGet](#) (const OS\_PowerState state)  
Get name of the current system power state.

#### 4.9.1 Function Documentation

##### 4.9.1.1 Status OS\_PowerInit ( void )

Init power.

Returns

Status.

##### 4.9.1.2 OS\_PowerState OS\_PowerStateGet ( void )

Get current system power state.

Returns

Power state.

##### 4.9.1.3 ConstStrPtr OS\_PowerStateNameGet ( const OS\_PowerState state )

Get name of the current system power state.

Parameters

in	state	Power state.
----	-------	--------------

Returns

Power state name.

##### 4.9.1.4 Status OS\_PowerStateSet ( const OS\_PowerState state )

Set current system power state.

## Parameters

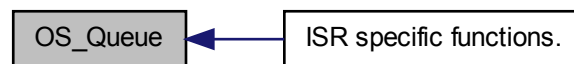
in	state	Power state.
----	-------	--------------

## Returns

Status.

## 4.10 OS\_Queue

Collaboration diagram for OS\_Queue:



## Data Structures

- struct [OS\\_QueueConfig](#)
- struct [OS\\_QueueStats](#)

## Modules

- [ISR specific functions.](#)

## Typedefs

- typedef void \* [OS\\_QueueHd](#)

## Functions

- Status [OS\\_QueueCreate](#) (const [OS\\_QueueConfig](#) \*cfg\_p, OS\_TaskHd parent\_thd, OS\_QueueHd \*qhd\_p)  
Create a queue.
- Status [OS\\_QueueDelete](#) (const OS\_QueueHd qhd)  
Delete the queue.
- Status [OS\\_QueueReceive](#) (const OS\_QueueHd qhd, void \*item\_p, const TimeMs timeout)

Receive the item.

- Status [OS\\_QueueSend](#) (const OS\_QueueHd qhd, const void \*item\_p, const TimeMs timeout, const OS\_MessagePrio priority)

Send the item.

- Status [OS\\_QueueFlush](#) (const OS\_QueueHd qhd)

Flush the queue.

- U32 [OS\\_QueueItemsCountGet](#) (const OS\_QueueHd qhd)

Get queue items count.

- Status [OS\\_QueueConfigGet](#) (const OS\_QueueHd qhd, [OS\\_QueueConfig](#) \*config\_p)

Get queue config.

- Status [OS\\_QueueStatsGet](#) (const OS\_QueueHd qhd, [OS\\_QueueStats](#) \*stats\_p)

Get queue statistics.

- OS\_TaskHd [OS\\_QueueParentGet](#) (const OS\_QueueHd qhd)

Get queue parent.

- OS\_QueueHd [OS\\_QueueSvcStdInGet](#) (void)

Get system service task standart input/output queue.

- U32 [OS\\_QueuesCountGet](#) (void)

Get system queues count.

- OS\_QueueHd [OS\\_QueueNextGet](#) (const OS\_QueueHd qhd)

Get the next queue.

#### 4.10.1 Function Documentation

##### 4.10.1.1 Status [OS\\_QueueConfigGet](#) ( const OS\_QueueHd qhd, [OS\\_QueueConfig](#) \* config\_p )

Get queue config.

Parameters

in	qhd	Queue handle.
out	config_p	Queue config.

Returns

Status.

4.10.1.2 Status OS\_QueueCreate ( const OS\_QueueConfig \* cfg\_p,  
OS\_TaskHd parent\_thd, OS\_QueueHd \* qhd\_p )

Create a queue.

Parameters

in	cfg_p	Queue config.
in	parent_thd	Parent task handle.
out	qhd_p	Queue handle.

Returns

Status.

4.10.1.3 Status OS\_QueueDelete ( const OS\_QueueHd qhd )

Delete the queue.

Parameters

in	qhd	Queue handle.
----	-----	---------------

Returns

Status.

4.10.1.4 Status OS\_QueueFlush ( const OS\_QueueHd qhd )

Flush the queue.

Parameters

in	qhd	Queue handle.
----	-----	---------------

Returns

Status.

4.10.1.5 U32 OS\_QueueItemsCountGet ( const OS\_QueueHd qhd )

Get queue items count.

Parameters

in	qhd	Queue handle.
----	-----	---------------

## Returns

Items count.

## 4.10.1.6 OS\_QueueHd OS\_QueueNextGet ( const OS\_QueueHd qhd )

Get the next queue.

## Parameters

in	qhd	Queue handle.
----	-----	---------------

## Returns

Queue handle.

## 4.10.1.7 OS\_TaskHd OS\_QueueParentGet ( const OS\_QueueHd qhd )

Get queue parent.

## Parameters

in	qhd	Queue handle.
----	-----	---------------

## Returns

Task handle.

## 4.10.1.8 Status OS\_QueueReceive ( const OS\_QueueHd qhd, void \* item\_p, const TimeMs timeout )

Receive the item.

## Parameters

in	qhd	Receiver queue handle.
out	item_p	Item.
in	timeout	Item receiving timeout.

## Returns

Status.

## 4.10.1.9 U32 OS\_QueuesCountGet ( void )

Get system queues count.

### Returns

Queues count.

```
4.10.1.10 Status OS_QueueSend ( const OS_QueueHd qhd, const void *  
    item_p, const TimeMs timeout, const OS_MessagePrio priority  
    )
```

Send the item.

### Parameters

in	qhd	Receiver queue handle.
in	item_p	Item.
in	timeout	Item sending timeout.
in	priority	Item sending priority.

### Returns

Status.

```
4.10.1.11 Status OS_QueueStatsGet ( const OS_QueueHd qhd,  
    OS_QueueStats * stats_p )
```

Get queue statistics.

### Parameters

in	qhd	Queue handle.
out	stats_p	Queue statistics.

### Returns

Status.

```
4.10.1.12 OS_QueueHd OS_QueueSvcStdInGet ( void )
```

Get system service task standart input/output queue.

### Returns

Queue handle.

## 4.11 OS\_Semaphore

Collaboration diagram for OS\_Semaphore:



### Modules

- [ISR specific functions.](#)

### Typedefs

- typedef MutexState OS\_SemaphoreState
- typedef SemaphoreHandle\_t OS\_SemaphoreHd

### Functions

- void [OS\\_SemaphoreBinaryCreate](#) (OS\_SemaphoreHd shd)  
Create a binary semaphore.
- OS\_SemaphoreHd [OS\\_SemaphoreCountingCreate](#) (const U32 count\_max, const U32 count\_init)  
Create a counting semaphore.
- void [OS\\_SemaphoreDelete](#) (const OS\_SemaphoreHd shd)  
Delete the semaphore.
- Status [OS\\_SemaphoreLock](#) (const OS\_SemaphoreHd shd, const TimeMs timeout)  
Lock the semaphore.
- Status [OS\\_SemaphoreUnlock](#) (const OS\_SemaphoreHd shd)  
Unlock the semaphore.
- OS\_SemaphoreState [OS\\_SemaphoreCheck](#) (const OS\_SemaphoreHd shd)  
Check semaphore state.



### 4.11.1 Function Documentation

#### 4.11.1.1 void OS\_SemaphoreBinaryCreate ( OS\_SemaphoreHd shd )

Create a binary semaphore.

##### Parameters

in	shd	Semaphore handle.
----	-----	-------------------

##### Returns

None.

#### 4.11.1.2 OS\_SemaphoreState OS\_SemaphoreCheck ( const OS\_SemaphoreHd shd )

Check semaphore state.

##### Parameters

in	shd	semaphore handle.
----	-----	-------------------

##### Returns

Semaphore state.

#### 4.11.1.3 OS\_SemaphoreHd OS\_SemaphoreCountingCreate ( const U32 count\_max, const U32 count\_init )

Create a counting semaphore.

##### Parameters

in	count_max	Counter maximum value.
in	count_init	Counter initial value.

##### Returns

Semaphore handle.

#### 4.11.1.4 void OS\_SemaphoreDelete ( const OS\_SemaphoreHd shd )

Delete the semaphore.

##### Parameters

in	shd	Semaphore handle.
----	-----	-------------------

## Returns

None.

4.11.1.5 Status OS\_SemaphoreLock ( const OS\_SemaphoreHd shd, const TimeMs timeout )

Lock the semaphore.

## Parameters

in	shd	Semaphore handle.
in	timeout	Semaphore locking timeout.

## Returns

Status.

4.11.1.6 Status OS\_SemaphoreUnlock ( const OS\_SemaphoreHd shd )

Unlock the semaphore.

## Parameters

in	shd	Semaphore handle.
----	-----	-------------------

## Returns

Status.

## 4.12 OS\_Settings

### Data Structures

- struct [OS\\_SettingsItem](#)

### Enumerations

- enum OS\_SettingsStatus { S\_SETT\_UNDEF = S\_MODULE, S\_SETT\_READ, S\_SETT\_WRITE }

### Functions

- Status [OS\\_SettingsInit](#) (void)  
Initialise the settings.

- Status [OS\\_SettingsDeInit](#) (void)  
Deinitialise the settings.
- Status [OS\\_SettingsDelete](#) (ConstStrPtr file\_path\_p, ConstStrPtr section\_p, ConstStrPtr key\_p)  
Delete settings item.
- Status [OS\\_SettingsRead](#) (ConstStrPtr file\_path\_p, ConstStrPtr section\_p, ConstStrPtr key\_p, StrPtr value\_p)  
Read settings item.
- Status [OS\\_SettingsWrite](#) (ConstStrPtr file\_path\_p, ConstStrPtr section\_p, ConstStrPtr key\_p, ConstStrPtr value\_p)  
Write settings item.
- Status [OS\\_SettingsItemsRead](#) (ConstStrPtr file\_path\_p, [OS\\_SettingsItem](#) items[])  
Read settings items.
- Status [OS\\_SettingsItemsWrite](#) (ConstStrPtr file\_path\_p, [OS\\_SettingsItem](#) items[])  
Write settings items.

### 4.12.1 Function Documentation

#### 4.12.1.1 Status OS\_SettingsDeInit ( void )

Deinitialise the settings.

Returns

Status.

#### 4.12.1.2 Status OS\_SettingsDelete ( ConstStrPtr file\_path\_p, ConstStrPtr section\_p, ConstStrPtr key\_p )

Delete settings item.

Parameters

in	file_path_p	Path to the settings file.
in	section_p	Settings section.
in	key_p	Key item.

Returns

Status.

#### 4.12.1.3 Status OS\_SettingsInit ( void )

Initialise the settings.

Returns

Status.

#### 4.12.1.4 Status OS\_SettingsItemsRead ( ConstStrPtr file\_path\_p, OS\_SettingsItem items[] )

Read settings items.

Parameters

in	file_path_p	Path to the settings file.
out	items[]	Items vector.

Returns

Status.

#### 4.12.1.5 Status OS\_SettingsItemsWrite ( ConstStrPtr file\_path\_p, OS\_SettingsItem items[] )

Write settings items.

Parameters

in	file_path_p	Path to the settings file.
in	items[]	Items vector.

Returns

Status.

#### 4.12.1.6 Status OS\_SettingsRead ( ConstStrPtr file\_path\_p, ConstStrPtr section\_p, ConstStrPtr key\_p, StrPtr value\_p )

Read settings item.

## Parameters

in	file_path_p	Path to the settings file.
in	key_p	Key item.
out	value_p	Key value.

## Returns

Status.

4.12.1.7 Status OS\_SettingsWrite ( ConstStrPtr file\_path\_p, ConstStrPtr section\_p, ConstStrPtr key\_p, ConstStrPtr value\_p )

Write settings item.

## Parameters

in	file_path_p	Path to the settings file.
in	section_p	Settings section.
in	key_p	Key item.
in	value_p	Key value.

## Returns

Status.

## 4.13 OS\_Shell

## Data Structures

- struct [OS\\_ShellCommandConfig](#)

## Defines

- #define SHELL\_COMMAND\_UNDEF OS\_NULL

## Typedefs

- typedef Status(\* OS\_ShellCommandHandler )(const U32 argc, ConstStrPtr argv[])
- typedef [OS\\_ShellCommandConfig](#) \* OS\_ShellCommandHd

## Enumerations

- enum OS\_ShellOptions { OS\_SHELL\_OPT\_UNDEF }

## Functions

- Status [OS\\_ShellInit](#) (void)  
Initialise shell.
- Status [OS\\_ShellCommandCreate](#) (const [OS\\_ShellCommandConfig](#) \*cmd\_cfg\_p)  
Create shell command.
- Status [OS\\_ShellCommandDelete](#) (ConstStrPtr name\_p)  
Delete shell command.
- Status [OS\\_ShellCommandExecute](#) (void)  
Execute current shell command.
- Status [OS\\_ShellArgumentsNumberCheck](#) (const [OS\\_ShellCommandHd](#) cmd\_hd, const U8 argc)  
Check shell command arguments number.
- [OS\\_ShellCommandHd](#) [OS\\_ShellCommandByNameGet](#) (ConstStrPtr name\_p)  
Get shell command by it's name.
- [OS\\_ShellCommandHd](#) [OS\\_ShellCommandNextGet](#) (const [OS\\_ShellCommandHd](#) cmd\_hd)  
Get the next shell command.
- ConstStrPtr [OS\\_ShellPromptGet](#) (void)  
Get shell command prompt.
- Status [OS\\_ShellCls](#) (void)  
Clear shell buffer.
- void [OS\\_ShellCIHandler](#) (const U8 c)  
Execute shell command line handler.

### 4.13.1 Function Documentation

#### 4.13.1.1 Status [OS\\_ShellArgumentsNumberCheck](#) ( const [OS\\_ShellCommandHd](#) cmd\_hd, const U8 argc )

Check shell command arguments number.

#### Parameters

in	cmd_hd	Shell command handler.
in	argc	Shell command arguments count.

#### Returns

Status.

4.13.1.2 void OS\_ShellClHandler ( const U8 c )

Execute shell command line handler.

#### Parameters

in	c	Command line input char.
----	---	--------------------------

#### Returns

None.

4.13.1.3 Status OS\_ShellCls ( void )

Clear shell buffer.

#### Returns

Status.

4.13.1.4 OS\_ShellCommandHd OS\_ShellCommandByNameGet ( ConstStrPtr name\_p )

Get shell command by it's name.

#### Parameters

in	name_p	Shell command name.
----	--------	---------------------

#### Returns

Shell command handler.

4.13.1.5 Status OS\_ShellCommandCreate ( const OS\_ShellCommandConfig \* cmd\_cfg\_p )

Create shell command.

#### Parameters

in	cmd_cfg_p	Shell command configuration.
----	-----------	------------------------------

Returns

Status.

#### 4.13.1.6 Status OS\_ShellCommandDelete ( ConstStrPtr name\_p )

Delete shell command.

Parameters

in	name_p	Shell command name.
----	--------	---------------------

Returns

Status.

#### 4.13.1.7 Status OS\_ShellCommandExecute ( void )

Execute current shell command.

Returns

Status.

#### 4.13.1.8 OS\_ShellCommandHd OS\_ShellCommandNextGet ( const OS\_ShellCommandHd cmd\_hd )

Get the next shell command.

Parameters

in	cmd_hd	Shell command handler.
----	--------	------------------------

Returns

Driver handler.

#### 4.13.1.9 Status OS\_ShellInit ( void )

Initialise shell.

Returns

Status.

#### 4.13.1.10 ConstStrPtr OS\_ShellPromptGet ( void )

Get shell command prompt.

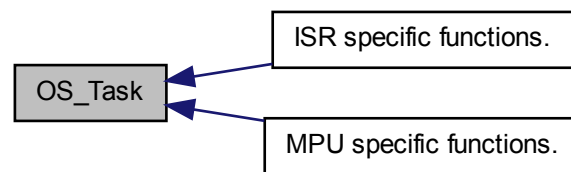


### Returns

Shell command prompt.

## 4.14 OS\_Task

Collaboration diagram for OS\_Task:



### Data Structures

- struct [OS\\_TaskConfig](#)

### Modules

- [MPU specific functions.](#)  
Set the task standart input/output queue.
- [ISR specific functions.](#)

### Defines

- #define [OS\\_THIS\\_TASK](#) `OS_NULL`  
Common declarations.
- #define `OS_STDIO_LEN` 4

### Typedefs

- typedef U8 `OS_StdIoDir`
- typedef U8 `OS_TaskAttrs`
- typedef U8 `OS_TaskState`
- typedef U8 `OS_TaskPrio`

- typedef OS\_Owner OS\_TaskHd
- typedef U8 OS\_TaskId
- typedef void OS\_TaskArgs
- typedef TaskStatus\_t OS\_TaskStats

## Enumerations

- enum { OS\_STDIO\_IN, OS\_STDIO\_OUT }
- enum { OS\_TASK\_ATTR\_UNDEF, OS\_TASK\_ATTR\_RECREATE, OS\_TASK\_ATTR\_LAST }
- enum {  
OS\_TASK\_STATE\_UNDEF, OS\_TASK\_STATE\_READY, OS\_TASK\_STATE\_RUN, OS\_TASK\_STATE\_BLOCK,  
OS\_TASK\_STATE\_SUSPEND, OS\_TASK\_STATE\_DELETED, OS\_TASK\_STATE\_LAST }
- enum {  
OS\_TASK\_PRIO\_UNDEF, OS\_TASK\_PRIO\_LOW = OS\_PRIORITY\_MIN, OS\_TASK\_PRIO\_BELOW\_NORMAL, OS\_TASK\_PRIO\_NORMAL,  
OS\_TASK\_PRIO\_ABOVE\_NORMAL, OS\_TASK\_PRIO\_HIGH, OS\_TASK\_PRIO\_REALTIME, OS\_TASK\_PRIO\_MAX = OS\_PRIORITY\_MAX - 1,  
OS\_TASK\_PRIO\_LAST = OS\_TASK\_PRIO\_MAX }

## Functions

- static Status [OS\\_TaskInit](#) (OS\_TaskArgs \*args\_p)  
Init task.
- static void [OS\\_TaskMain](#) (OS\_TaskArgs \*args\_p)  
Task main function.
- static Status [OS\\_TaskPower](#) (OS\_TaskArgs \*args\_p, const OS\_PowerState state)  
Task main function.
- Status [OS\\_TaskCreate](#) (const [OS\\_TaskConfig](#) \*cfg\_p, OS\_TaskHd \*thd\_p)  
Create a task.
- Status [OS\\_TaskDelete](#) (const OS\_TaskHd thd)  
Delete the task.
- void [OS\\_TaskDelay](#) (const TimeMs timeout)  
Delay the task.

- void [OS\\_TaskDelayUntil](#) (OS\_Tick \*tick\_last\_p, const TimeMs timeout)  
Delay the task until.
- void [OS\\_TaskSuspend](#) (const OS\_TaskHd thd)  
Suspend the task.
- void [OS\\_TaskResume](#) (const OS\_TaskHd thd)  
Resume the task.
- OS\_TaskId [OS\\_TaskIdGet](#) (const OS\_TaskHd thd)  
Get task id.
- OS\_TaskHd [OS\\_TaskHdGet](#) (void)  
Get current task handle.
- OS\_TaskHd [OS\\_TaskHdByIdGet](#) (const OS\_TaskId tid)  
Get task handle by it's id.
- OS\_TaskHd [OS\\_TaskHdParentGet](#) (void)  
Get current task parent's handle.
- OS\_TaskHd [OS\\_TaskHdParentByHdGet](#) (const OS\_TaskHd thd)  
Get parent's task by task handle.
- U32 [OS\\_TasksCountGet](#) (void)  
Get tasks count.
- U32 [OS\\_TasksStatsGet](#) (OS\_TaskStats \*stats\_p, const U32 stats\_count, U32 \*uptime\_p)  
Get tasks statistics.
- OS\_TaskState [OS\\_TaskStateGet](#) (const OS\_TaskHd thd)  
Get task state.
- ConstStrPtr [OS\\_TaskStateNameGet](#) (const OS\_TaskState state)  
Get task state name.
- ConstStrPtr [OS\\_TaskNameGet](#) (const OS\_TaskHd thd)  
Get task name.
- OS\_TaskAttrs [OS\\_TaskAttrsGet](#) (const OS\_TaskHd thd)  
Get task attributes.
- const [OS\\_TaskConfig](#) \* [OS\\_TaskConfigGet](#) (const OS\_TaskHd thd)  
Get task configuration.

- void \* [OS\\_TaskStorageGet](#) (const OS\_TaskHd thd)  
Get task storage.
- OS\_PowerState [OS\\_TaskPowerStateGet](#) (const OS\_TaskHd thd)  
Get task power state.
- OS\_TaskPrio [OS\\_TaskPriorityGet](#) (const OS\_TaskHd thd)  
Get task priority.
- Status [OS\\_TaskPrioritySet](#) (const OS\_TaskHd thd, const OS\_TaskPrio prio)  
Set task priority.
- OS\_TaskHd [OS\\_TaskByNameGet](#) (ConstStrPtr name\_p)  
Get task by it's name.
- OS\_TaskHd [OS\\_TaskNextGet](#) (const OS\_TaskHd thd)  
Get the next task.
- OS\_QueueHd [OS\\_TaskSvcStdInGet](#) (void)  
Get the system supervisor task standart input queue.
- OS\_QueueHd [OS\\_TaskStdIoGet](#) (const OS\_TaskHd thd, const OS\_StdIoDir dir)  
Get the task standart input/output queue.

#### 4.14.1 Function Documentation

##### 4.14.1.1 OS\_TaskAttrs OS\_TaskAttrsGet ( const OS\_TaskHd thd )

Get task attributes.

Parameters

in	thd	Task handle.
----	-----	--------------

Returns

Attributes.

##### 4.14.1.2 OS\_TaskHd OS\_TaskByNameGet ( ConstStrPtr name\_p )

Get task by it's name.

## Parameters

in	name_p	Task name.
----	--------	------------

## Returns

Task handle.

4.14.1.3 `const OS_TaskConfig* OS_TaskConfigGet ( const OS_TaskHd  
thd )`

Get task configuration.

## Parameters

in	thd	Task handle.
----	-----	--------------

## Returns

Task configuration.

4.14.1.4 `Status OS_TaskCreate ( const OS_TaskConfig * cfg_p,  
OS_TaskHd * thd_p )`

Create a task.

## Parameters

in	cfg_p	Task config.
out	thd_p	Task handle.

## Returns

Status.

4.14.1.5 `void OS_TaskDelay ( const TimeMs timeout )`

Delay the task.

## Parameters

in	timeout	Delay timeout.
----	---------	----------------

## Returns

None.

4.14.1.6 void OS\_TaskDelayUntil ( OS\_Tick \* tick\_last\_p, const TimeMs timeout )

Delay the task until.

Parameters

in	tick_last_p	Last time task unblocked system ticks.
in	timeout	Delay timeout.

Returns

None.

4.14.1.7 Status OS\_TaskDelete ( const OS\_TaskHd thd )

Delete the task.

Parameters

in	thd	Task handle.
----	-----	--------------

Returns

Status.

4.14.1.8 OS\_TaskHd OS\_TaskHdByIdGet ( const OS\_TaskId tid )

Get task handle by it's id.

Parameters

in	tid	Task id.
----	-----	----------

Returns

Task handle.

4.14.1.9 OS\_TaskHd OS\_TaskHdGet ( void )

Get current task handle.

Parameters

in	thd	Task handle.
----	-----	--------------

## Returns

Task handle.

4.14.1.10 OS\_TaskHd OS\_TaskHdParentByHdGet ( const OS\_TaskHd  
thd )

Get parent's task by task handle.

## Parameters

in	thd	Task handle.
----	-----	--------------

## Returns

Task handle.

4.14.1.11 OS\_TaskHd OS\_TaskHdParentGet ( void )

Get current task parent's handle.

## Returns

Task handle.

4.14.1.12 OS\_TaskId OS\_TaskIdGet ( const OS\_TaskHd thd )

Get task id.

## Parameters

in	thd	Task handle.
----	-----	--------------

## Returns

Task id.

4.14.1.13 static Status OS\_TaskInit ( OS\_TaskArgs \* args\_p ) [static]

Init task.

## Parameters

in	args_p	Task arguments.
----	--------	-----------------

## Returns

Status.

4.14.1.14 `static void OS_TaskMain ( OS_TaskArgs * args_p ) [static]`

Task main function.

Parameters

in	args_p	Task arguments.
----	--------	-----------------

Returns

None.

4.14.1.15 `ConstStrPtr OS_TaskNameGet ( const OS_TaskHd thd )`

Get task name.

Parameters

in	thd	Task handle.
----	-----	--------------

Returns

Name.

4.14.1.16 `OS_TaskHd OS_TaskNextGet ( const OS_TaskHd thd )`

Get the next task.

Parameters

in	thd	Task handle.
----	-----	--------------

Returns

Task handle.

4.14.1.17 `static Status OS_TaskPower ( OS_TaskArgs * args_p, const OS_PowerState state ) [static]`

Task main function.

Parameters

in	args_p	Task arguments.
in	state	Task new power state.

Returns

Status.



4.14.1.18 OS\_PowerState OS\_TaskPowerStateGet ( const OS\_TaskHd thd  
)

Get task power state.

Parameters

in	thd	Task handle.
----	-----	--------------

Returns

Task power state.

4.14.1.19 OS\_TaskPrio OS\_TaskPriorityGet ( const OS\_TaskHd thd )

Get task priority.

Parameters

in	thd	Task handle.
----	-----	--------------

Returns

Task priority.

4.14.1.20 Status OS\_TaskPrioritySet ( const OS\_TaskHd thd, const  
OS\_TaskPrio prio )

Set task priority.

Parameters

in	thd	Task handle.
in	prio	Priority.

Returns

Status.

4.14.1.21 void OS\_TaskResume ( const OS\_TaskHd thd )

Resume the task.

Parameters

in	thd	Task handle.
----	-----	--------------

## Returns

None.

## 4.14.1.22 U32 OS\_TasksCountGet ( void )

Get tasks count.

## Returns

Tasks count.

## 4.14.1.23 U32 OS\_TasksStatsGet ( OS\_TaskStats \* stats\_p, const U32 stats\_count, U32 \* uptime\_p )

Get tasks statistics.

## Parameters

out	stats_p	Task statistics.
in	stats_count	Task statistics count.
out	uptime_p	System uptime.

## Returns

Task statistics count that were populated.

## 4.14.1.24 OS\_TaskState OS\_TaskStateGet ( const OS\_TaskHd thd )

Get task state.

## Parameters

in	thd	Task handle.
----	-----	--------------

## Returns

Task state.

## 4.14.1.25 ConstStrPtr OS\_TaskStateNameGet ( const OS\_TaskState state )

Get task state name.

## Parameters

in	state	Task state.
----	-------	-------------

### Returns

Name.

4.14.1.26 OS\_QueueHd OS\_TaskStdIoGet ( const OS\_TaskHd thd, const OS\_StdIoDir dir )

Get the task standart input/output queue.

### Parameters

in	thd	Task handle.
in	dir	I direction.

### Returns

Queue handle.

4.14.1.27 void\* OS\_TaskStorageGet ( const OS\_TaskHd thd )

Get task storage.

### Parameters

in	thd	Task handle.
----	-----	--------------

### Returns

Task storage.

4.14.1.28 void OS\_TaskSuspend ( const OS\_TaskHd thd )

Suspend the task.

### Parameters

in	thd	Task handle.
----	-----	--------------

### Returns

None.

4.14.1.29 OS\_QueueHd OS\_TaskSvcStdInGet ( void )

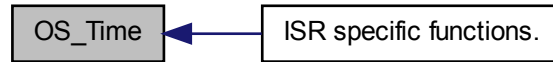
Get the system supervisor task standart input queue.

### Returns

Queue handle.

## 4.15 OS\_Time

Collaboration diagram for OS\_Time:



### Modules

- [ISR specific functions.](#)

### Defines

- `#define OS_TICKS_TO_MS(ticks) ((U32)((((ticks) * 1000UL) / configTICK_RATE_HZ))`  
Converts from RTOS ticks to milliseconds.

### Typedefs

- `typedef Time OS_DateTime`
- `typedef TickType_t OS_Tick`
- `typedef U32 TimeMs`
- `typedef U32 TimeS`

### Enumerations

- `enum OS_TimeWeekDay {`  
`OS_WEEK_DAY_UNDEF, OS_WEEK_DAY_MONDAY, OS_WEEK_DAY_TUESDAY, OS_WEEK_DAY_WEDNESDAY,`  
`OS_WEEK_DAY_THURSDAY, OS_WEEK_DAY_FRIDAY, OS_WEEK_DAY_SATURDAY, OS_WEEK_DAY_SUNDAY,`  
`OS_WEEK_DAY_LAST }`
- `enum OS_TimeFormat {`  
`OS_TIME_UNDEF, OS_TIME_GMT, OS_TIME_GMT_OFFSET, OS_TIME_LOCAL,`  
`OS_TIME_UPTIME, OS_TIME_LAST }`

- enum OS\_DateFormat { OS\_DATE\_UNDEF, OS\_DATE\_LAST }
- enum OS\_AlarmFormat { OS\_ALARM\_UNDEF, OS\_ALARM\_LAST }
- enum OS\_TimeDayLight { OS\_TIME\_DAYLIGHT\_UNDEF, OS\_TIME\_DAYLIGHT\_SUMMER, OS\_TIME\_DAYLIGHT\_WINTER, OS\_TIME\_DAYLIGHT\_LAST }

## Functions

- static U32 [OS\\_MS\\_TO\\_TICKS](#) (const TimeMs ms)  
Converts from milliseconds to RTOS ticks, value is always > 0.
- Status [OS\\_TimeGet](#) (const OS\_TimeFormat format, OS\_DateTime \*os\_time\_p)  
Get the current time.
- Status [OS\\_TimeSet](#) (const OS\_TimeFormat format, OS\_DateTime \*os\_time\_p)  
Set time.
- Status [OS\\_DateGet](#) (const OS\_DateFormat format, OS\_DateTime \*os\_date\_p)  
Get the current date.
- Status [OS\\_DateSet](#) (const OS\_DateFormat format, OS\_DateTime \*os\_date\_p)  
Set date.
- BL [OS\\_TimeIsValid](#) (const U8 hour, const U8 min, const U8 sec)  
Time validation.
- BL [OS\\_DateIsValid](#) (const U16 year, const U8 month, const U8 day)  
Date validation.
- OS\_TimeWeekDay [OS\\_DateWeekDayGet](#) (const U16 year, const U8 month, const U8 day)  
Get the day of the week.
- ConstStrPtr [OS\\_TimeNameDayOfWeekGet](#) (const OS\_TimeWeekDay week\_day, const Locale locale)  
Get the day of the week name.
- OS\_TimeDayLight [OS\\_TimeDayLightSavingsGet](#) (void)  
Get the current daylight savings.
- Status [OS\\_TimeDayLightSavingsSet](#) (const OS\_TimeDayLight savings)

Set the daylight savings.

- OS\_Tick [OS\\_TickCountGet](#) (void)  
Get tick count.
- OS\_DateTime [OS\\_TimeStringParse](#) (ConstStrPtr time\_p)  
Parse the time string.
- OS\_DateTime [OS\\_DateStringParse](#) (ConstStrPtr date\_p)  
Parse the date string.

#### 4.15.1 Function Documentation

4.15.1.1 Status [OS\\_DateGet](#) ( const OS\_DateFormat format,  
OS\_DateTime \* os\_date\_p )

Get the current date.

Parameters

in	format	Date format.
out	os_date_p	Date data.

Returns

Status.

4.15.1.2 BL [OS\\_DateIsValid](#) ( const U16 year, const U8 month, const U8  
day )

Date validation.

Parameters

in	year	Year.
in	month	Month.
in	day	Day.

Returns

Bool.

4.15.1.3 Status [OS\\_DateSet](#) ( const OS\_DateFormat format,  
OS\_DateTime \* os\_date\_p )

Set date.

## Parameters

in	format	Date format.
in	os_date_p	Date data.

## Returns

Status.

## 4.15.1.4 OS\_DateTime OS\_DateStringParse ( ConstStrPtr date\_p )

Parse the date string.

## Parameters

in	date_p	String of date.
----	--------	-----------------

## Returns

Date.

## Note

String delimiter format depends on the current locale settings.

Example string for the EN locale: MM/DD/YYYY

## 4.15.1.5 OS\_TimeWeekDay OS\_DateWeekDayGet ( const U16 year, const U8 month, const U8 day )

Get the day of the week.

## Parameters

in	year	Year.
in	month	Month.
in	day	Day.

## Returns

Day of the week.

## 4.15.1.6 OS\_Tick OS\_TickCountGet ( void )

Get tick count.

## Returns

Tick count.

## Note

Get the current tick count since system start.

## 4.15.1.7 OS\_TimeDayLight OS\_TimeDayLightSavingsGet ( void )

Get the current daylight savings.

## Returns

Daylight savings.

## 4.15.1.8 Status OS\_TimeDayLightSavingsSet ( const OS\_TimeDayLight savings )

Set the daylight savings.

## Parameters

in	savings	Daylight savings.
----	---------	-------------------

## Returns

Status.

## 4.15.1.9 Status OS\_TimeGet ( const OS\_TimeFormat format, OS\_DateTime \* os\_time\_p )

Get the current time.

## Parameters

in	format	Time format.
out	os_time_p	Time data.

## Returns

Status.

## 4.15.1.10 BL OS\_TimeIsValid ( const U8 hour, const U8 min, const U8 sec )

Time validation.

## Parameters

in	hour	Hour.
in	min	Minute.
in	sec	Second.



### Returns

Bool.

### Warning

Currently only for 24H mode!

4.15.1.11 ConstStrPtr OS\_TimeNameDayOfWeekGet ( const  
OS\_TimeWeekDay week\_day, const Locale locale )

Get the day of the week name.

### Parameters

in	week_day	Day of week.
in	locale	Locale.

### Returns

Day of the week string.

4.15.1.12 Status OS\_TimeSet ( const OS\_TimeFormat format,  
OS\_DateTime \* os\_time\_p )

Set time.

### Parameters

in	format	Time format.
in	os_time_p	Time data.

### Returns

Status.

4.15.1.13 OS\_DateTime OS\_TimeStringParse ( ConstStrPtr time\_p )

Parse the time string.

### Parameters

in	time_p	String of time.
----	--------	-----------------

### Returns

Time.

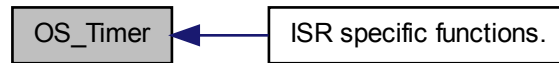
## Note

String delimiter format depends on the current locale settings.

Example string for the EN locale: HH:MM:SS

## 4.16 OS\_Timer

Collaboration diagram for OS\_Timer:



### Data Structures

- struct [OS\\_TimerConfig](#)

### Modules

- [ISR specific functions.](#)

### Typedefs

- typedef TimerHandle\_t OS\_TimerHd
- typedef OS\_SignalData OS\_TimerId
- typedef struct [OS\\_TimerConfig](#) OS\_TimerStats

### Enumerations

- enum { OS\_TIM\_ID\_UNDEF, OS\_TIM\_ID\_APP = 0x01, OS\_TIM\_ID\_LAST }
- enum OS\_TimerOptions { OS\_TIM\_OPT\_UNDEF, OS\_TIM\_OPT\_PERIODIC, OS\_TIM\_OPT\_EVENT }

### Functions

- Status [OS\\_TimerCreate](#) (const [OS\\_TimerConfig](#) \*cfg\_p, OS\_TimerHd \*timer\_hd\_p)

Create a timer.

- Status [OS\\_TimerDelete](#) (const OS\_TimerHd timer\_hd, const TimeMs timeout)

Delete the timer.

- Status [OS\\_TimerReset](#) (const OS\_TimerHd timer\_hd, const TimeMs timeout)

Reset the timer.

- Status [OS\\_TimerStart](#) (const OS\_TimerHd timer\_hd, const TimeMs timeout)

Start the timer.

- Status [OS\\_TimerStop](#) (const OS\_TimerHd timer\_hd, const TimeMs timeout)

Stop the timer.

- Status [OS\\_TimerPeriodGet](#) (const OS\_TimerHd timer\_hd, TimeMs \*period\_p)

Get the timer period.

- Status [OS\\_TimerPeriodSet](#) (const OS\_TimerHd timer\_hd, const TimeMs new\_period, const TimeMs timeout)

Set the timer period.

- BL [OS\\_TimerIsActive](#) (const OS\_TimerHd timer\_hd)

Get the timer slot.

- OS\_TimerId [OS\\_TimerIdGet](#) (const OS\_TimerHd timer\_hd)

Get the timer's id.

- OS\_TimerHd [OS\\_TimerByIdGet](#) (const OS\_TimerId timer\_id)

Get the timer by id.

- ConstStrPtr [OS\\_TimerNameGet](#) (const OS\_TimerHd timer\_hd)

Get timer name.

- OS\_TimerHd [OS\\_TimerByNameGet](#) (ConstStrPtr name\_p)

Get the timer by its name.

- Status [OS\\_TimerStatsGet](#) (const OS\_TimerHd timer\_hd, [OS\\_TimerStats](#) \*stats\_p)

Get timer statistics.

- OS\_TimerHd [OS\\_TimerNextGet](#) (const OS\_TimerHd timer\_hd)

Get the next timer.

### 4.16.1 Function Documentation

#### 4.16.1.1 OS\_TimerHd OS\_TimerByIdGet ( const OS\_TimerId timer\_id )

Get the timer by id.

##### Parameters

in	timer_id	Timer id.
----	----------	-----------

##### Returns

Timer handle.

#### 4.16.1.2 OS\_TimerHd OS\_TimerByNameGet ( ConstStrPtr name\_p )

Get the timer by its name.

##### Parameters

in	name_p	Timer's name.
----	--------	---------------

##### Returns

Timer handle.

#### 4.16.1.3 Status OS\_TimerCreate ( const OS\_TimerConfig \* cfg\_p, OS\_TimerHd \* timer\_hd\_p )

Create a timer.

##### Parameters

in	cfg_p	Timer config.
out	timer_hd_p	Timer handle.

##### Returns

Status.

#### 4.16.1.4 Status OS\_TimerDelete ( const OS\_TimerHd timer\_hd, const TimeMs timeout )

Delete the timer.

##### Parameters

in	timer_hd	Timer handle.
in	timeout	Operation timeout.

### Returns

Status.

#### 4.16.1.5 OS\_TimerId OS\_TimerIdGet ( const OS\_TimerHd timer\_hd )

Get the timer's id.

### Parameters

in	timer_hd	Timer handle.
----	----------	---------------

### Returns

Timer id.

#### 4.16.1.6 BL OS\_TimerIsActive ( const OS\_TimerHd timer\_hd )

Get the timer slot.

### Parameters

in	timer_hd	Timer handle.
out	period_p	Timer slot.

### Returns

Status. Set the timer slot.

### Parameters

in	timer_hd	Timer handle.
in	new_slot	Timer new slot.
in	timeout	Operation timeout.

### Returns

Status. Check the timer is active.

### Parameters

in	timer_hd	Timer handle.
----	----------	---------------

### Returns

Bool.

#### 4.16.1.7 ConstStrPtr OS\_TimerNameGet ( const OS\_TimerHd timer\_hd )

Get timer name.

## Parameters

in	timer_hd	Timer handle.
----	----------	---------------

## Returns

Name.

4.16.1.8 OS\_TimerHd OS\_TimerNextGet ( const OS\_TimerHd timer\_hd  
)

Get the next timer.

## Parameters

in	timer_hd	Timer handle.
----	----------	---------------

## Returns

Timer handle.

4.16.1.9 Status OS\_TimerPeriodGet ( const OS\_TimerHd timer\_hd,  
TimeMs \* period\_p )

Get the timer period.

## Parameters

in	timer_hd	Timer handle.
out	period_p	Timer period.

## Returns

Status.

4.16.1.10 Status OS\_TimerPeriodSet ( const OS\_TimerHd timer\_hd,  
const TimeMs new\_period, const TimeMs timeout )

Set the timer period.

## Parameters

in	timer_hd	Timer handle.
in	new_ period	Timer new period.
in	timeout	Operation timeout.

## Returns

Status.

4.16.1.11 Status OS\_TimerReset ( const OS\_TimerHd timer\_hd, const TimeMs timeout )

Reset the timer.

Parameters

in	timer_hd	Timer handle.
in	timeout	Operation timeout.

Returns

Status.

4.16.1.12 Status OS\_TimerStart ( const OS\_TimerHd timer\_hd, const TimeMs timeout )

Start the timer.

Parameters

in	timer_hd	Timer handle.
in	timeout	Operation timeout.

Returns

Status.

4.16.1.13 Status OS\_TimerStatsGet ( const OS\_TimerHd timer\_hd, OS\_TimerStats \* stats\_p )

Get timer statistics.

Parameters

in	timer_hd	Timer handle.
out	stats_p	Queue statistics.

Returns

Status.

4.16.1.14 Status OS\_TimerStop ( const OS\_TimerHd timer\_hd, const TimeMs timeout )

Stop the timer.

Parameters

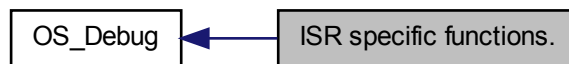
in	timer_hd	Timer handle.
in	timeout	Operation timeout.

Returns

Status.

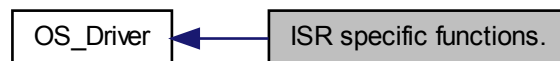
#### 4.17 ISR specific functions.

Collaboration diagram for ISR specific functions.:



#### 4.18 ISR specific functions.

Collaboration diagram for ISR specific functions.:



Functions

- Status [OS\\_ISR\\_DriverIoCtl](#) (const OS\_DriverHd dhd, const U32 request\_id, void \*args\_p)  
Input/Output control.



### 4.18.1 Function Documentation

4.18.1.1 Status `OS_ISR_DriverIoCtl` ( const `OS_DriverHd` `dhd`, const `U32` `request_id`, void \* `args_p` )

Input/Output control.

Parameters

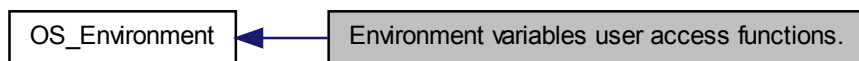
in	<code>dhd</code>	Driver's handle.
in	<code>request_id</code>	Driver's request code identifier.
in	<code>args_p</code>	Driver's specific input arguments (if presents).

Returns

Status.

## 4.19 Environment variables user access functions.

Collaboration diagram for Environment variables user access functions.:



Functions

- `OS_TaskHd` [OS\\_EnvVariableOwnerGet](#) (ConstStrPtr `variable_name_p`)  
Get the variable owner.
- ConstStrPtr [OS\\_EnvVariableGet](#) (ConstStrPtr `variable_name_p`)  
Get the environment variable value.
- Status [OS\\_EnvVariableSet](#) (ConstStrPtr `variable_name_p`, ConstStrPtr `variable_value_p`)  
Set the environment variable value.
- Status [OS\\_EnvVariableDelete](#) (ConstStrPtr `variable_name_p`)  
Delete the environment variable.

- ConstStrPtr [OS\\_EnvVariableNextGet](#) (ConstStrPtr variable\_name\_p)

Get the next environment variable.

#### 4.19.1 Function Documentation

##### 4.19.1.1 Status OS\_EnvVariableDelete ( ConstStrPtr variable\_name\_p )

Delete the environment variable.

Parameters

in	variable_ name_p	Variable name.
----	---------------------	----------------

Returns

Status.

##### 4.19.1.2 ConstStrPtr OS\_EnvVariableGet ( ConstStrPtr variable\_name\_p )

Get the environment variable value.

Parameters

in	variable_ name_p	Variable name.
----	---------------------	----------------

Returns

Value.

##### 4.19.1.3 ConstStrPtr OS\_EnvVariableNextGet ( ConstStrPtr variable\_name\_p )

Get the next environment variable.

Parameters

in	variable_ name_p	Variable name.
----	---------------------	----------------

Returns

Variable name.

4.19.1.4 OS\_TaskHd OS\_EnvVariableOwnerGet ( ConstStrPtr  
variable\_name\_p )

Get the variable owner.

#### Warning

Please, do not use environment variables in time critical parts of code. If it possible cache these values locally.

#### Parameters

in	variable_- name_p	Variable name.
----	----------------------	----------------

#### Returns

Task handle.

4.19.1.5 Status OS\_EnvVariableSet ( ConstStrPtr variable\_name\_p,  
ConstStrPtr variable\_value\_p )

Set the environment variable value.

#### Parameters

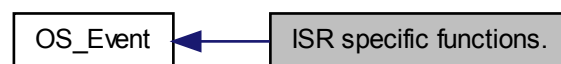
in	variable_- name_p	Variable name.
in	variable_- value_p	Variable value.

#### Returns

Status.

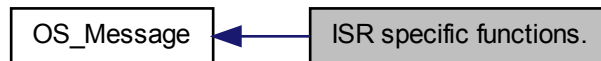
## 4.20 ISR specific functions.

Collaboration diagram for ISR specific functions.:



## 4.21 ISR specific functions.

Collaboration diagram for ISR specific functions.:



### Functions

- Status [OS\\_ISR\\_MessageSend](#) (const OS\_QueueHd qhd, const [OS\\_Message](#) \*msg\_p, const OS\_MessagePrio priority)  
Send the message.
- Status [OS\\_ISR\\_MessageReceive](#) (const OS\_QueueHd qhd, [OS\\_Message](#) \*\*msg\_pp)  
Receive the message.

### 4.21.1 Function Documentation

4.21.1.1 Status [OS\\_ISR\\_MessageReceive](#) ( const OS\_QueueHd qhd,  
OS\_Message \*\* msg\_pp )

Receive the message.

#### Parameters

in	qhd	Receiver (task) queue handle.
out	msg_pp	Message.
in	timeout	Message receiving timeout.

#### Returns

Status.

4.21.1.2 Status [OS\\_ISR\\_MessageSend](#) ( const OS\_QueueHd qhd, const  
OS\_Message \* msg\_p, const OS\_MessagePrio priority )

Send the message.

## Parameters

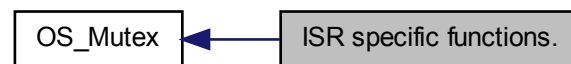
in	qhd	Receiver (task) queue handle.
in	msg_p	Message.
in	timeout	Message sending timeout.
in	priority	Message sending priority.

## Returns

Status.

## 4.22 ISR specific functions.

Collaboration diagram for ISR specific functions.:



## Functions

- Status [OS\\_ISR\\_MutexLock](#) (const OS\_MutexHd mhd)  
Lock the mutex.
- Status [OS\\_ISR\\_MutexUnlock](#) (const OS\_MutexHd mhd)  
Unlock the mutex.
- OS\_MutexState [OS\\_ISR\\_MutexCheck](#) (const OS\_MutexHd mhd)  
Check mutex state.

## 4.22.1 Function Documentation

4.22.1.1 OS\_MutexState OS\_ISR\_MutexCheck ( const OS\_MutexHd mhd )

Check mutex state.

## Parameters

in	mhd	Mutex handle.
----	-----	---------------

Returns

Mutex state.

4.22.1.2 Status OS\_ISR\_MutexLock ( const OS\_MutexHd mhd )

Lock the mutex.

Parameters

in	mhd	Mutex handle.
----	-----	---------------

Returns

Status.

4.22.1.3 Status OS\_ISR\_MutexUnlock ( const OS\_MutexHd mhd )

Unlock the mutex.

Parameters

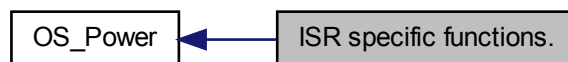
in	mhd	Mutex handle.
----	-----	---------------

Returns

Status.

## 4.23 ISR specific functions.

Collaboration diagram for ISR specific functions.:



Functions

- Status [OS\\_ISR\\_PowerStateSet](#) (const OS\_PowerState state)  
Set current system power state.

### 4.23.1 Function Documentation

#### 4.23.1.1 Status OS\_ISR\_PowerStateSet ( const OS\_PowerState state )

Set current system power state.

Parameters

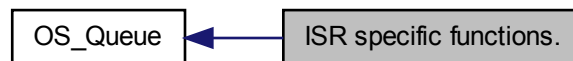
in	state	Power state.
----	-------	--------------

Returns

Status.

## 4.24 ISR specific functions.

Collaboration diagram for ISR specific functions.:



### Functions

- Status [OS\\_ISR\\_QueueReceive](#) (const OS\_QueueHd qhd, void \*item\_  
p)  
Receive the item.
- Status [OS\\_ISR\\_QueueSend](#) (const OS\_QueueHd qhd, const void \*item\_  
p, const OS\_MessagePrio priority)  
Send the item.
- U32 [OS\\_ISR\\_QueueItemsCountGet](#) (const OS\_QueueHd qhd)  
Get queue items count.

### 4.24.1 Function Documentation

#### 4.24.1.1 U32 OS\_ISR\_QueueItemsCountGet ( const OS\_QueueHd qhd )

Get queue items count.

## Parameters

in	qhd	Queue handle.
----	-----	---------------

## Returns

Items count.

4.24.1.2 Status OS\_ISR\_QueueReceive ( const OS\_QueueHd qhd, void \* item\_p )

Receive the item.

## Parameters

in	qhd	Receiver queue handle.
out	item_p	Item.

## Returns

Status.

0 - OK

1 - OK, needs to context switch (reading from queue unblock the task waiting for room in this one).

< 0 - error Status.

4.24.1.3 Status OS\_ISR\_QueueSend ( const OS\_QueueHd qhd, const void \* item\_p, const OS\_MessagePrio priority )

Send the item.

## Parameters

in	qhd	Receiver queue handle.
in	item_p	Item.
in	priority	Item sending priority.

## Returns

Status.

0 - OK

1 - OK, needs to context switch (reading from queue unblock the task waiting for room in this one).

< 0 - error Status.



## 4.25 ISR specific functions.

Collaboration diagram for ISR specific functions.:



### Functions

- Status [OS\\_ISR\\_SemaphoreLock](#) (const OS\_SemaphoreHd shd)  
Lock the semaphore.
- Status [OS\\_ISR\\_SemaphoreUnlock](#) (const OS\_SemaphoreHd shd)  
Unlock the semaphore.
- OS\_SemaphoreState [OS\\_ISR\\_SemaphoreCheck](#) (const OS\_SemaphoreHd shd)  
Check semaphore state.

### 4.25.1 Function Documentation

#### 4.25.1.1 OS\_SemaphoreState OS\_ISR\_SemaphoreCheck ( const OS\_SemaphoreHd shd )

Check semaphore state.

##### Parameters

in	shd	semaphore handle.
----	-----	-------------------

##### Returns

Semaphore state.

#### 4.25.1.2 Status OS\_ISR\_SemaphoreLock ( const OS\_SemaphoreHd shd )

Lock the semaphore.

## Parameters

in	shd	Semaphore handle.
----	-----	-------------------

## Returns

Status.

4.25.1.3 Status OS\_ISR\_SemaphoreUnlock ( const OS\_SemaphoreHd shd  
)

Unlock the semaphore.

## Parameters

in	shd	Semaphore handle.
----	-----	-------------------

## Returns

Status.

## 4.26 MPU specific functions.

Set the task standart input/output queue.

Collaboration diagram for MPU specific functions.:



Set the task standart input/output queue.

## Parameters

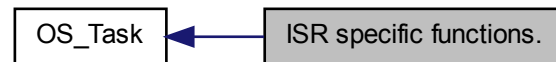
in	thd	Task handle.
in	qhd	Queue handle.
in	dir	I direction.

## Returns

Queue handle.

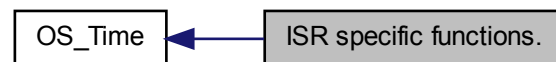
## 4.27 ISR specific functions.

Collaboration diagram for ISR specific functions.:



## 4.28 ISR specific functions.

Collaboration diagram for ISR specific functions.:



### Functions

- OS\_Tick [OS\\_ISR\\_TickCountGet](#) (void)  
Get tick count.

#### 4.28.1 Function Documentation

##### 4.28.1.1 OS\_Tick OS\_ISR\_TickCountGet ( void )

Get tick count.

Returns

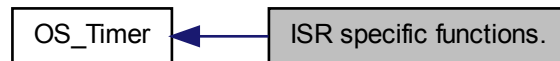
Tick count.

Note

Get the current tick count since system uptime.

## 4.29 ISR specific functions.

Collaboration diagram for ISR specific functions.:



### Functions

- Status [OS\\_ISR\\_TimerReset](#) (const OS\_TimerHd timer\_hd)  
Reset the timer.
- Status [OS\\_ISR\\_TimerStart](#) (const OS\_TimerHd timer\_hd)  
Start the timer.
- Status [OS\\_ISR\\_TimerStop](#) (const OS\_TimerHd timer\_hd)  
Stop the timer.
- Status [OS\\_ISR\\_TimerPeriodChange](#) (const OS\_TimerHd timer\_hd, const TimeMs new\_period)  
Change the timer period.

### 4.29.1 Function Documentation

4.29.1.1 Status [OS\\_ISR\\_TimerPeriodChange](#) ( const OS\_TimerHd timer\_hd, const TimeMs new\_period )

Change the timer period.

#### Parameters

in	timer_hd	Timer handle.
in	new_ period	Timer new period.

#### Returns

Status.

## 4.29.1.2 Status OS\_ISR\_TimerReset ( const OS\_TimerHd timer\_hd )

Reset the timer.

Parameters

in	timer_hd	Timer handle.
----	----------	---------------

Returns

Status.

## 4.29.1.3 Status OS\_ISR\_TimerStart ( const OS\_TimerHd timer\_hd )

Start the timer.

Parameters

in	timer_hd	Timer handle.
----	----------	---------------

Returns

Status.

## 4.29.1.4 Status OS\_ISR\_TimerStop ( const OS\_TimerHd timer\_hd )

Stop the timer.

Parameters

in	timer_hd	Timer handle.
----	----------	---------------

Returns

Status.



## Chapter 5

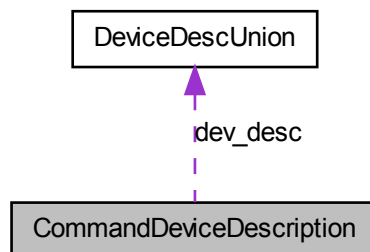
# Data Structure Documentation

### 5.1 CommandDeviceDescription Struct Reference

Данные описания устройства.

```
#include <protocol.h>
```

Collaboration diagram for CommandDeviceDescription:



#### Data Fields

- [DeviceDescUnion](#) dev\_desc

#### 5.1.1 Detailed Description

Данные описания устройства.

Definition at line 121 of file protocol.h.

The documentation for this struct was generated from the following file:

- [protocol.h](#)

## 5.2 DeviceDescUnion Union Reference

Дескриптор устройства.

```
#include <typedefs_app.h>
```

### Data Fields

- U8 data [64]
- DeviceDesc device\_description

### 5.2.1 Detailed Description

Дескриптор устройства. Объединение для возможности изменения размера структуры описания устройства, но не более значения CMD\_PACKET\_SIZE\_MAX!

Definition at line 66 of file typedefs\_app.h.

The documentation for this union was generated from the following file:

- typedefs\_app.h

## 5.3 DeviceId Struct Reference

### Data Fields

- U8 data [DEV\_ID\_SIZE]

### 5.3.1 Detailed Description

Definition at line 27 of file typedefs\_app.h.

The documentation for this struct was generated from the following file:

- typedefs\_app.h



## 5.4 DeviceRevision Struct Reference

### Data Fields

- U16 [maj](#)  
Старшая версия.
- U16 [min](#)  
Младшая версия.
- U16 [id](#)  
Идентификатор устройства.
- U16 [padding](#)  
(Резерв/выравнивание).

### 5.4.1 Detailed Description

Definition at line 31 of file typedefs\_app.h.

The documentation for this struct was generated from the following file:

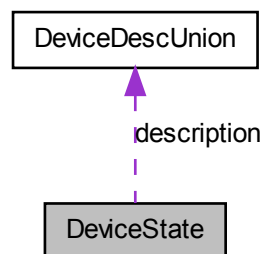
- typedefs\_app.h

## 5.5 DeviceState Struct Reference

Полное состояние устройства.

```
#include <typedefs_app.h>
```

Collaboration diagram for DeviceState:



## Data Fields

- ConnectState [connection](#)  
Состояние соединения устройства.
- [DeviceDescUnion](#) description  
Описание устройства.

### 5.5.1 Detailed Description

Полное состояние устройства.

Definition at line 72 of file typedefs\_app.h.

The documentation for this struct was generated from the following file:

- typedefs\_app.h

## 5.6 HAL\_DriverItf Struct Reference

### Data Fields

- Status(\* Init )(void)
- Status(\* DeInit )(void)
- Status(\* Open )(void \*args\_p)
- Status(\* Close )(void)
- Status(\* Read )(U8 \*data\_in\_p, U32 size, void \*args\_p)
- Status(\* Write )(U8 \*data\_out\_p, U32 size, void \*args\_p)
- Status(\* IoCtl )(const U32 request\_id, void \*args\_p)

### 5.6.1 Detailed Description

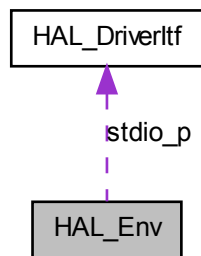
Definition at line 26 of file hal.h.

The documentation for this struct was generated from the following file:

- [hal.h](#)

## 5.7 HAL\_Env Struct Reference

Collaboration diagram for HAL\_Env:



### Data Fields

- Locale locale
- HAL\_PowerState power
- const [HAL\\_DriverItf](#) \* stdio\_p
- LogLevel log\_level

#### 5.7.1 Detailed Description

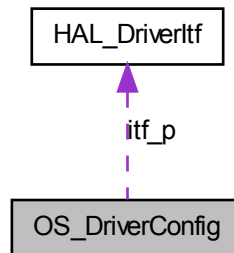
Definition at line 36 of file `hal.h`.

The documentation for this struct was generated from the following file:

- [hal.h](#)

## 5.8 OS\_DriverConfig Struct Reference

Collaboration diagram for OS\_DriverConfig:



### Data Fields

- ConstStr name [OS\_DRIVER\_NAME\_LEN]
- [HAL\\_DriverItf](#) \* itf\_p
- OS\_PowerPrio prio\_power

#### 5.8.1 Detailed Description

Definition at line 41 of file `os_driver.h`.

The documentation for this struct was generated from the following file:

- [os\\_driver.h](#)

## 5.9 OS\_DriverStats Struct Reference

### Data Fields

- OS\_DriverState state
- OS\_PowerState power
- U16 owners
- U32 sended
- U32 received
- U32 errors\_cnt
- Status status\_last

### 5.9.1 Detailed Description

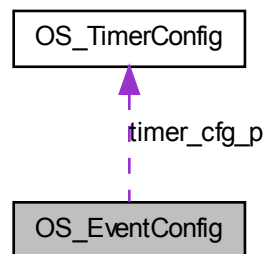
Definition at line 31 of file os\_driver.h.

The documentation for this struct was generated from the following file:

- [os\\_driver.h](#)

## 5.10 OS\_EventConfig Struct Reference

Collaboration diagram for OS\_EventConfig:



### Data Fields

- const [OS\\_TimerConfig](#) \* timer\_cfg\_p
- OS\_EventItem \* item\_p
- OS\_EventState state

### 5.10.1 Detailed Description

Definition at line 31 of file os\_event.h.

The documentation for this struct was generated from the following file:

- [os\\_event.h](#)

## 5.11 OS\_MemoryDesc Struct Reference

Memory description.

```
#include <os_memory.h>
```

## Data Fields

- U32 addr
- U32 size
- U32 block\_size
- OS\_MemoryType type
- ConstStrPtr name\_p

### 5.11.1 Detailed Description

Memory description.

Definition at line 32 of file os\_memory.h.

The documentation for this struct was generated from the following file:

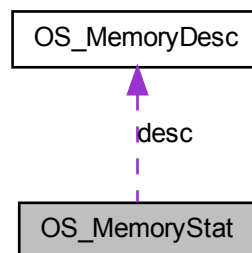
- [os\\_memory.h](#)

## 5.12 OS\_MemoryStat Struct Reference

Memory statistics.

`#include <os_memory.h>`

Collaboration diagram for OS\_MemoryStat:



## Data Fields

- [OS\\_MemoryDesc](#) desc
- U32 used
- U32 free

### 5.12.1 Detailed Description

Memory statistics.

Definition at line 41 of file `os_memory.h`.

The documentation for this struct was generated from the following file:

- [os\\_memory.h](#)

## 5.13 OS\_Message Struct Reference

### Data Fields

- OS\_TaskHd src
- OS\_MessageId id
- U16 size
- U8 data [0]

### 5.13.1 Detailed Description

Definition at line 28 of file `os_message.h`.

The documentation for this struct was generated from the following file:

- [os\\_message.h](#)

## 5.14 OS\_QueueConfig Struct Reference

### Data Fields

- U16 len
- U16 item\_size
- Direction dir

### 5.14.1 Detailed Description

Definition at line 24 of file `os_queue.h`.

The documentation for this struct was generated from the following file:

- [os\\_queue.h](#)

## 5.15 OS\_QueueStats Struct Reference

### Data Fields

- U32 sended
- U32 received

### 5.15.1 Detailed Description

Definition at line 30 of file `os_queue.h`.

The documentation for this struct was generated from the following file:

- [os\\_queue.h](#)

## 5.16 OS\_SettingsItem Struct Reference

### Data Fields

- ConstStrPtr section\_p
- ConstStrPtr key\_p
- Str value [OS\_SETTINGS\_VALUE\_LEN]

### 5.16.1 Detailed Description

Definition at line 30 of file `os_settings.h`.

The documentation for this struct was generated from the following file:

- [os\\_settings.h](#)

## 5.17 OS\_ShellCommandConfig Struct Reference

### Data Fields

- ConstStrPtr command
- OS\_ShellCommandHandler handler
- U8 args\_min
- U8 args\_max
- OS\_ShellOptions options



### 5.17.1 Detailed Description

Definition at line 32 of file `os_shell.h`.

The documentation for this struct was generated from the following file:

- [os\\_shell.h](#)

## 5.18 OS\_TaskConfig Struct Reference

### Data Fields

- ConstStr name [OS\_TASK\_NAME\_LEN]
- void(\* func\_main )(void \*)
- Status(\* func\_power )(void \*, const OS\_PowerState)
- void \* args\_p
- OS\_TaskAttrs attrs
- OS\_TaskPrio prio\_init
- OS\_PowerPrio prio\_power
- U8 timeout
- U16 stack\_size
- U8 stdin\_len
- U8 stdout\_len

### 5.18.1 Detailed Description

Definition at line 78 of file `os_task.h`.

The documentation for this struct was generated from the following file:

- [os\\_task.h](#)

## 5.19 OS\_TimerConfig Struct Reference

### Data Fields

- ConstStrPtr name\_p
- OS\_QueueHd slot
- TimeMs period
- OS\_TimerId id
- OS\_TimerOptions options

### 5.19.1 Detailed Description

Definition at line 39 of file `os_timer.h`.

The documentation for this struct was generated from the following file:

- [os\\_timer.h](#)

## 5.20 Packet Struct Reference

Пакет.

```
#include <protocol.h>
```

### Data Fields

- U8 \* [data\\_p](#)  
Указатель на данные.
- U16 [len](#)  
Размер данных.

### 5.20.1 Detailed Description

Пакет.

Definition at line 33 of file `protocol.h`.

The documentation for this struct was generated from the following file:

- [protocol.h](#)

## 5.21 ProtocolHeaderInfo Struct Reference

### Data Fields

- ProtocolIdInfo `id`
- U8 `payload [0]`

### 5.21.1 Detailed Description

Definition at line 80 of file `protocol.h`.

The documentation for this struct was generated from the following file:

- [protocol.h](#)

## 5.22 ProtocolId Struct Reference

### Data Fields

- U32 [ver](#): 2  
protocol version
- U32 [typ](#): 1  
frame type (info = 1, data = 0)
- U32 [seq](#): 1  
sequence number
- U32 [ack](#): 1  
acknowledge
- U32 [nak](#): 1  
no acknowledge
- U32 [dnt](#): 1  
do not transmit (receiver sends to transmitter info packet(PROTO\_INFO\_ID\_RECEIVE\_READY) when ready to receive again)
- U32 [lsf](#): 1  
last fragment (one/last fragment = 1, next fragment = 0)
- U32 [dst](#): 12  
destination address
- U32 [src](#): 12  
source address

### 5.22.1 Detailed Description

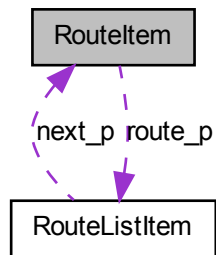
Definition at line 106 of file protocol.h.

The documentation for this struct was generated from the following file:

- [protocol.h](#)

## 5.23 RouteItem Struct Reference

Collaboration diagram for RouteItem:



### Data Fields

- OS\_Driver \* if\_p
- [RouteListItem](#) \* route\_p

#### 5.23.1 Detailed Description

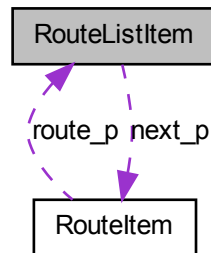
Definition at line 47 of file protocol.h.

The documentation for this struct was generated from the following file:

- [protocol.h](#)

## 5.24 RouteListItem Struct Reference

Collaboration diagram for RouteListItem:



### Data Fields

- struct [RouteItem](#) \* next\_p
- RouteAddr dst

#### 5.24.1 Detailed Description

Definition at line 42 of file protocol.h.

The documentation for this struct was generated from the following file:

- [protocol.h](#)



## Chapter 6

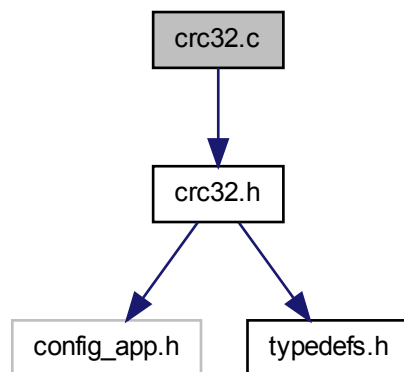
# File Documentation

### 6.1 crc32.c File Reference

CRC32.

```
#include "crc32.h"
```

Include dependency graph for `crc32.c`:



### Functions

- U32 [Crc32](#) (U8 \*data\_p, U32 size)  
CRC32 computation.

- U32 [Crc32Delta](#) (U8 \*data\_p, U32 size, U32 init\_poly)  
CRC32 delta computation.

## Variables

- static const U32 crc\_32\_tbl [256]

### 6.1.1 Detailed Description

CRC32.

Author

A. Filyanov

Definition in file [crc32.c](#).

### 6.1.2 Function Documentation

#### 6.1.2.1 U32 Crc32 ( U8 \* data\_p, U32 size )

CRC32 computation.

Parameters

in	data_p	Input data.
in	size	Input data size.

Returns

CRC32.

Definition at line 46 of file [crc32.c](#).

```
{
    return (Crc32Delta(data_p, size, CRC32_POLYNOMIAL) ^ CRC32_POLYNOMIAL);
}
```

#### 6.1.2.2 U32 Crc32Delta ( U8 \* data\_p, U32 size, U32 init\_poly )

CRC32 delta computation.

Parameters

in	data_p	Input data.
in	size	Input data size.
in	init_poly	Input polynomial.



Returns

CRC32.

Definition at line 52 of file crc32.c.

```
{
U32 crc_32 = init_poly;

while (0 != size) {
    crc_32 = crc_32_tbl[(crc_32 ^ *data_p) & 0xFF] ^ (crc_32 >> 8);
    ++data_p;
    --size;
}
return crc_32;
}
```

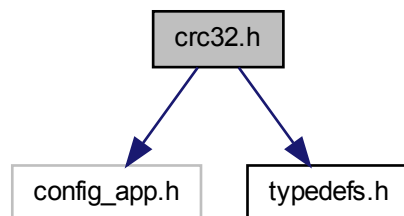
## 6.2 crc32.h File Reference

CRC32.

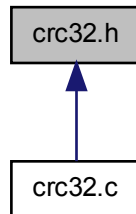
```
#include "config_app.h"
```

```
#include "typedefs.h"
```

Include dependency graph for crc32.h:



This graph shows which files directly or indirectly include this file:



## Defines

- `#define CRC32_POLYNOMIAL 0xFFFFFFFFU`

## Functions

- U32 [Crc32](#) (U8 \*data\_p, U32 size)  
CRC32 computation.
- U32 [Crc32Delta](#) (U8 \*data\_p, U32 size, U32 init\_poly)  
CRC32 delta computation.

### 6.2.1 Detailed Description

CRC32.

Author

A. Filyanov

Definition in file [crc32.h](#).

### 6.2.2 Function Documentation

#### 6.2.2.1 U32 Crc32 ( U8 \* data\_p, U32 size )

CRC32 computation.

Parameters

in	data_p	Input data.
in	size	Input data size.

Returns

CRC32.

Definition at line 46 of file crc32.c.

```
{
    return (Crc32Delta(data_p, size, CRC32_POLYNOMIAL) ^ CRC32_POLYNOMIAL);
}
```

6.2.2.2 U32 Crc32Delta ( U8 \* data\_p, U32 size, U32 init\_poly )

CRC32 delta computation.

Parameters

in	data_p	Input data.
in	size	Input data size.
in	init_poly	Input polynomial.

Returns

CRC32.

Definition at line 52 of file crc32.c.

```
{
    U32 crc_32 = init_poly;

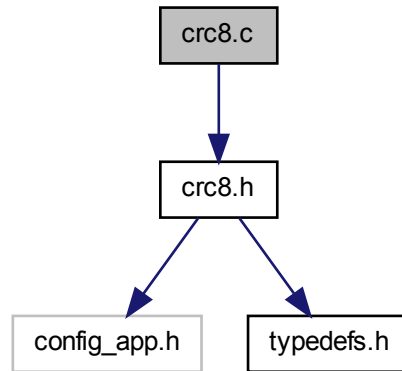
    while (0 != size) {
        crc_32 = crc_32_tbl[(crc_32 ^ *data_p) & 0xFF] ^ (crc_32 >> 8);
        ++data_p;
        --size;
    }
    return crc_32;
}
```

## 6.3 crc8.c File Reference

CRC8.

```
#include "crc8.h"
```

Include dependency graph for `crc8.c`:



## Functions

- U8 [Crc8](#) (U8 \*data\_p, U16 size)  
CRC8 computation.
- U8 [Crc8Delta](#) (const U8 value, const U8 init\_poly)  
CRC8 delta computation.

## Variables

- static const U8 `crc_8_tbl` [256]

### 6.3.1 Detailed Description

CRC8.

Author

A. Filyanov

Definition in file [crc8.c](#).

### 6.3.2 Function Documentation

#### 6.3.2.1 U8 Crc8 ( U8 \* data\_p, U16 size )

CRC8 computation.

Parameters

in	data_p	Input data.
in	size	Input data size.

Returns

CRC8.

Definition at line 30 of file crc8.c.

```
{
    U8 crc_8 = 0;

    while (0 != size) {
        crc_8 = crc_8_tbl[crc_8 ^ *data_p];
        ++data_p;
        --size;
    }

    return crc_8;
}
```

#### 6.3.2.2 U8 Crc8Delta ( const U8 value, const U8 init\_poly )

CRC8 delta computation.

Parameters

in	value	Input value.
in	init_poly	Input polynomial.

Returns

CRC8.

Definition at line 44 of file crc8.c.

```
{
    return crc_8_tbl[init_poly ^ value];
}
```

### 6.3.3 Variable Documentation

#### 6.3.3.1 `const U8 crc_8_tbl[256]` [static]

Initial value:

```
{
  0, 94,188,226, 97, 63,221,131,194,156,126, 32,163,253, 31, 65,
  157,195, 33,127,252,162, 64, 30, 95, 1,227,189, 62, 96,130,220,
  35,125,159,193, 66, 28,254,160,225,191, 93, 3,128,222, 60, 98,
  190,224, 2, 92,223,129, 99, 61,124, 34,192,158, 29, 67,161,255,
  70, 24,250,164, 39,121,155,197,132,218, 56,102,229,187, 89, 7,
  219,133,103, 57,186,228, 6, 88, 25, 71,165,251,120, 38,196,154,
  101, 59,217,135, 4, 90,184,230,167,249, 27, 69,198,152,122, 36,
  248,166, 68, 26,153,199, 37,123, 58,100,134,216, 91, 5,231,185,
  140,210, 48,110,237,179, 81, 15, 78, 16,242,172, 47,113,147,205,
  17, 79,173,243,112, 46,204,146,211,141,111, 49,178,236, 14, 80,
  175,241, 19, 77,206,144,114, 44,109, 51,209,143, 12, 82,176,238,
  50,108,142,208, 83, 13,239,177,240,174, 76, 18,145,207, 45,115,
  202,148,118, 40,171,245, 23, 73, 8, 86,180,234,105, 55,213,139,
  87, 9,235,181, 54,104,138,212,149,203, 41,119,244,170, 72, 22,
  233,183, 85, 11,136,214, 52,106, 43,117,151,201, 74, 20,246,168,
  116, 42,200,150, 21, 75,169,247,182,232, 10, 84,215,137,107, 53
}
```

Definition at line 9 of file `crc8.c`.

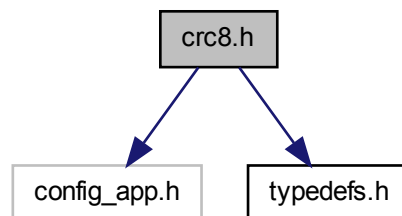
## 6.4 `crc8.h` File Reference

CRC8.

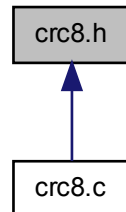
```
#include "config_app.h"
```

```
#include "typedefs.h"
```

Include dependency graph for `crc8.h`:



This graph shows which files directly or indirectly include this file:



## Functions

- U8 [Crc8](#) (U8 \*data\_p, U16 size)  
CRC8 computation.
- U8 [Crc8Delta](#) (const U8 value, const U8 init\_poly)  
CRC8 delta computation.

### 6.4.1 Detailed Description

CRC8.

Author

A. Filyanov

Definition in file [crc8.h](#).

### 6.4.2 Function Documentation

#### 6.4.2.1 U8 Crc8 ( U8 \* data\_p, U16 size )

CRC8 computation.

Parameters

in	data_p	Input data.
in	size	Input data size.

Returns

CRC8.

Definition at line 30 of file crc8.c.

```
{
U8 crc_8 = 0;

while (0 != size) {
    crc_8 = crc_8_tbl[crc_8 ^ *data_p];
    ++data_p;
    --size;
}

return crc_8;
}
```

#### 6.4.2.2 U8 Crc8Delta ( const U8 value, const U8 init\_poly )

CRC8 delta computation.

Parameters

in	value	Input value.
in	init_poly	Input polynomial.

Returns

CRC8.

Definition at line 44 of file crc8.c.

```
{
    return crc_8_tbl[init_poly ^ value];
}
```

## 6.5 hal.h File Reference

HAL.

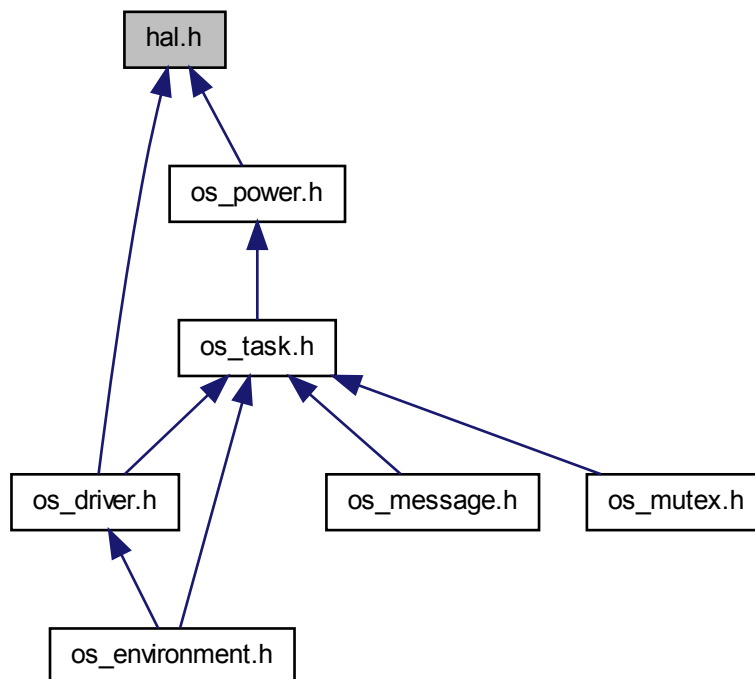
```
#include <stdarg.h>
#include "status.h"
#include "olimex_stm32_p407/config.h"
#include "hal/bsp/olimex_stm32_p407/hal_olimex_stm32_p407.h"
```



Include dependency graph for hal.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [HAL\\_DriverItf](#)
- struct [HAL\\_Env](#)

## Typedefs

- typedef U8 HAL\_PowerState
- typedef U8 HAL\_PowerPrio
- typedef void(\* HAL\_IrqCallbackFunc )(void)

## Enumerations

- enum { DRV\_REQ\_STD\_UNDEF = 64, DRV\_REQ\_STD\_SYNC, DRV\_REQ\_STD\_POWER, DRV\_REQ\_STD\_LAST }

### 6.5.1 Detailed Description

HAL.

Author

A. Filyanov

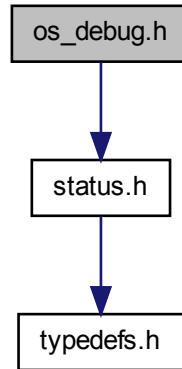
Definition in file [hal.h](#).

## 6.6 os\_debug.h File Reference

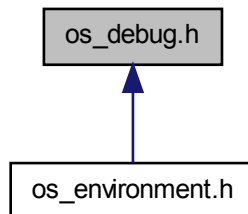
OS Debug.

```
#include "status.h"
```

Include dependency graph for os\_debug.h:



This graph shows which files directly or indirectly include this file:



## Defines

- #define OS\_ASSERT(a) D\_ASSERT(a)
- #define OS\_LOG(level,...) OS\_Log(level, \_\_VA\_ARGS\_\_)
- #define OS\_LOG\_S(level, status,...) OS\_Log(level, StatusStringGet(status, MDL\_STATUS\_ITEMS))  
Common status items array.

- `#define OS_TRACE(level, fmt_str_p,...) OS_Trace(level, fmt_str_p, __VA_ARGS__);`

## Typedefs

- typedef LogLevel [OS\\_LogLevel](#)  
Log level of tracing details.

## Functions

- Status [OS\\_DebugInit](#) (void)  
Init the debug module.
- Status [OS\\_DebugDeInit](#) (void)  
Deinit the debug module.
- void [OS\\_Log](#) (const [OS\\_LogLevel](#) level, ConstStrPtr format\_str\_p,...)  
  
Log the message.
- void [OS\\_Trace](#) (const [OS\\_LogLevel](#) level, ConstStrPtr format\_str\_p,...)  
  
Trace the message.

### 6.6.1 Detailed Description

OS Debug.

Author

A. Filyanov

Definition in file [os\\_debug.h](#).

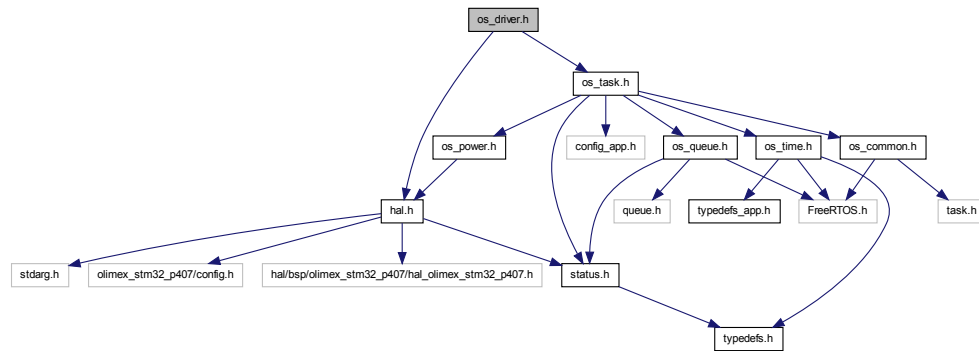
## 6.7 os\_driver.h File Reference

OS Driver.

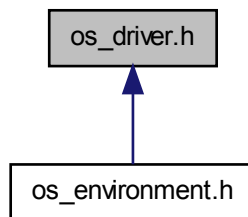
```
#include "hal.h"
```

```
#include "os_task.h"
```

Include dependency graph for os\_driver.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [OS\\_DriverStats](#)
- struct [OS\\_DriverConfig](#)

## Typedefs

- typedef const void \* [OS\\_DriverHd](#)
- typedef U8 [OS\\_DriverState](#)

## Enumerations

- enum { OS\_DRV\_STATE\_UNDEF, OS\_DRV\_STATE\_IS\_INIT, OS\_DRV\_STATE\_IS\_OPEN, OS\_DRV\_STATE\_LAST = 7 }

## Functions

- Status [OS\\_DriverCreate](#) (const [OS\\_DriverConfig](#) \*cfg\_p, OS\_DriverHd \*dhd\_p)  
Create driver.
- Status [OS\\_DriverDelete](#) (const OS\_DriverHd dhd)  
Delete driver.
- Status [OS\\_DriverInit](#) (const OS\_DriverHd dhd)  
Init driver.
- Status [OS\\_DriverDeInit](#) (const OS\_DriverHd dhd)  
Deinit driver.
- Status [OS\\_DriverOpen](#) (const OS\_DriverHd dhd, void \*args\_p)  
Open driver.
- Status [OS\\_DriverClose](#) (const OS\_DriverHd dhd)  
Close driver.
- Status [OS\\_DriverRead](#) (const OS\_DriverHd dhd, void \*data\_in\_p, U32 size, void \*args\_p)  
Read data.
- Status [OS\\_DriverWrite](#) (const OS\_DriverHd dhd, void \*data\_out\_p, U32 size, void \*args\_p)  
Write data.
- Status [OS\\_DriverIoCtl](#) (const OS\_DriverHd dhd, const U32 request\_id, void \*args\_p)  
Input/Output control.
- ConstStrPtr [OS\\_DriverNameGet](#) (const OS\_DriverHd dhd)  
Get driver name.
- ConstStrPtr [OS\\_DriverStateNameGet](#) (const OS\_DriverState state)  
Get driver's state name.
- OS\_DriverHd [OS\\_DriverByNameGet](#) (ConstStrPtr name\_p)  
Get driver by it's name.

- OS\_DriverHd [OS\\_DriverNextGet](#) (const OS\_DriverHd dhd)  
Get the next driver.
- OS\_DriverState [OS\\_DriverStateStateGet](#) (const OS\_DriverHd dhd)  
Get driver state.
- Status [OS\\_DriverStatsGet](#) (const OS\_DriverHd dhd, [OS\\_DriverStats](#) \*stats\_p)  
Get driver stats.
- const [OS\\_DriverConfig](#) \* [OS\\_DriverConfigGet](#) (const OS\_DriverHd dhd)  
  
Get driver configuration.
- OS\_TaskHd [OS\\_DriverParentGet](#) (const OS\_DriverHd dhd)  
Get driver's parent.
- Status [OS\\_ISR\\_DriverIoCtl](#) (const OS\_DriverHd dhd, const U32 request\_id, void \*args\_p)  
Input/Output control.

### 6.7.1 Detailed Description

OS Driver.

Author

A. Filyanov

Definition in file [os\\_driver.h](#).

## 6.8 os\_\_environment.h File Reference

OS Environment.

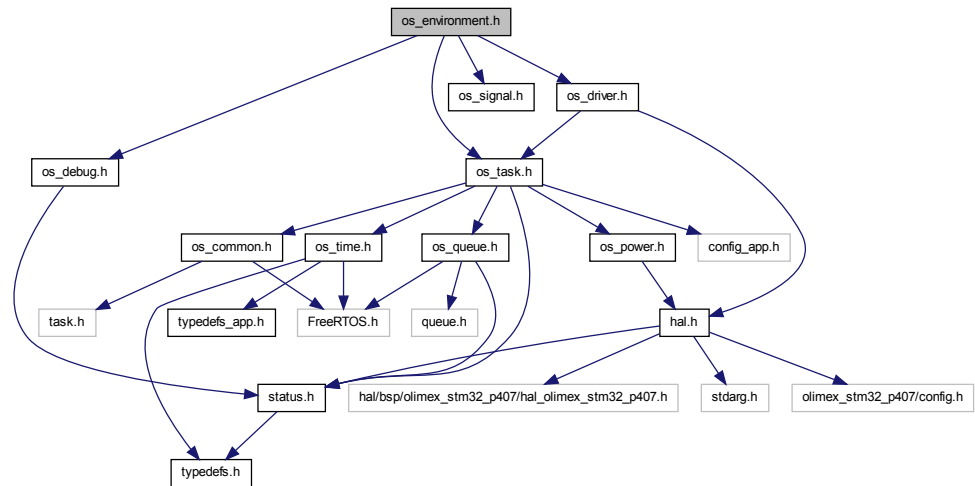
```
#include "os_debug.h"
```

```
#include "os_task.h"
```

```
#include "os_signal.h"
```

```
#include "os_driver.h"
```

Include dependency graph for `os_environment.h`:



## Defines

- `#define OS_ENV_POWER_STR "power"`

## Functions

- `OS_DriverHd OS_DriverStdIoGet (void)`  
Get system input/output driver.
- `Locale OS_LocaleGet (void)`  
Get current system locale.
- `Status OS_LocaleSet (ConstStrPtr locale_p)`  
Set the current system locale.
- `Status OS_PowerSet (ConstStrPtr power_p)`  
Set the current system power mode.
- `const HAL_DriverItf * OS_StdIoGet (void)`  
Get system input/output driver interface.
- `Status OS_StdIoSet (ConstStrPtr drv_name_p)`  
Set system input/output driver.
- `OS_LogLevel OS_LogLevelGet (void)`



Get current log level of trace details.

- Status [OS\\_LogLevelSet](#) (ConstStrPtr log\_level\_p)

Set current log level of trace details.

- OS\_TaskHd [OS\\_EnvVariableOwnerGet](#) (ConstStrPtr variable\_name\_p)

Get the variable owner.

- ConstStrPtr [OS\\_EnvVariableGet](#) (ConstStrPtr variable\_name\_p)

Get the environment variable value.

- Status [OS\\_EnvVariableSet](#) (ConstStrPtr variable\_name\_p, ConstStrPtr variable\_value\_p)

Set the environment variable value.

- Status [OS\\_EnvVariableDelete](#) (ConstStrPtr variable\_name\_p)

Delete the environment variable.

- ConstStrPtr [OS\\_EnvVariableNextGet](#) (ConstStrPtr variable\_name\_p)

Get the next environment variable.

### 6.8.1 Detailed Description

OS Environment.

Author

A. Filyanov

Definition in file [os\\_environment.h](#).

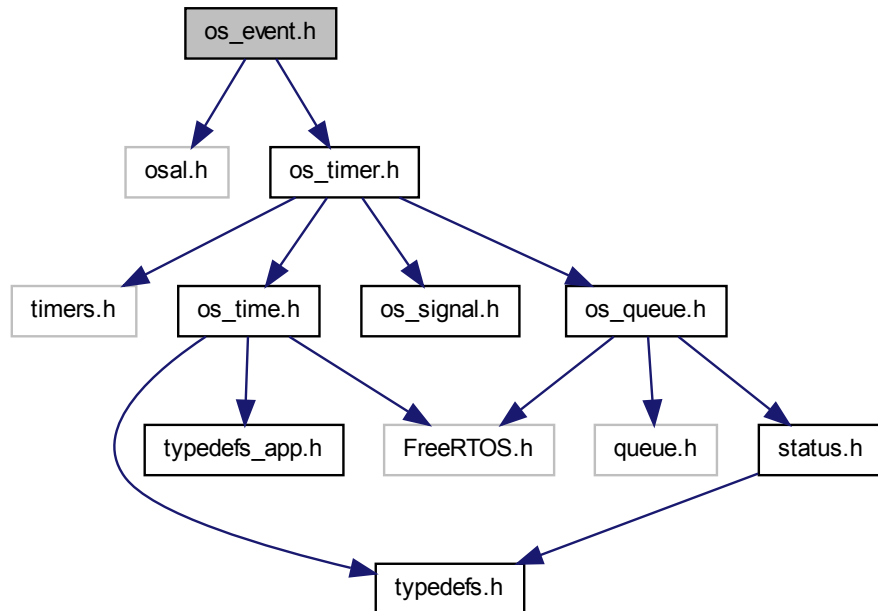
## 6.9 os\_event.h File Reference

OS Event.

```
#include "osal.h"
```

```
#include "os_timer.h"
```

Include dependency graph for `os_event.h`:



## Data Structures

- struct [OS\\_EventConfig](#)

## Typedefs

- typedef void \* OS\_EventHd
- typedef U8 OS\_EventState
- typedef OS\_StorageItem OS\_EventItem

## Enumerations

- enum { OS\_EVENT\_STATE\_UNDEF, OS\_EVENT\_STATE\_LAST }

## Functions

- Status [OS\\_EventCreate](#) (const [OS\\_EventConfig](#) \*cfg\_p, OS\_EventHd \*ehd\_p)

Create an event.

- Status [OS\\_EventDelete](#) (const OS\_EventHd ehd, const TimeMs timeout)

Delete the event.

- Status [OS\\_EventTimerGet](#) (const OS\_EventHd ehd, OS\_TimerHd \*timer\_hd\_p)

Get the event timer.

- Status [OS\\_EventStateGet](#) (const OS\_EventHd ehd, OS\_EventState \*state\_p)

Get the event state.

- Status [OS\\_EventPeriodGet](#) (const OS\_EventHd ehd, TimeMs \*period\_p)

Get the event state.

- Status [OS\\_EventStatePeriodSet](#) (const OS\_EventHd ehd, const TimeMs new\_period, const OS\_EventState new\_state, const TimeMs timeout)

- Status [OS\\_EventItemCreate](#) (const void \*data\_p, const U16 size, OS\_EventItem \*\*item\_pp)

Create an event item.

- Status [OS\\_EventItemDelete](#) (OS\_EventItem \*item\_p)

Delete the event item.

- Status [OS\\_EventItemOwnerAdd](#) (OS\_EventItem \*item\_p)

Add event item owner.

- Status [OS\\_EventItemLock](#) (OS\_EventItem \*item\_p, const TimeMs timeout)

Lock event item.

- Status [OS\\_EventItemUnlock](#) (OS\_EventItem \*item\_p)

Unlock event item.

- OS\_EventItem \* [OS\\_EventItemGet](#) (const OS\_EventHd ehd)

- OS\_EventItem \* [OS\\_EventItemByTimerIdGet](#) (const OS\_TimerId timer\_id)

- OS\_EventItem \* [OS\\_EventItemByStateGet](#) (const OS\_EventState state)

- OS\_EventHd [OS\\_EventNextGet](#) (const OS\_EventHd ehd)

Get the next event.

### 6.9.1 Detailed Description

OS Event.

Author

A. Filyanov

Definition in file [os\\_event.h](#).

## 6.10 os\_file\_system.h File Reference

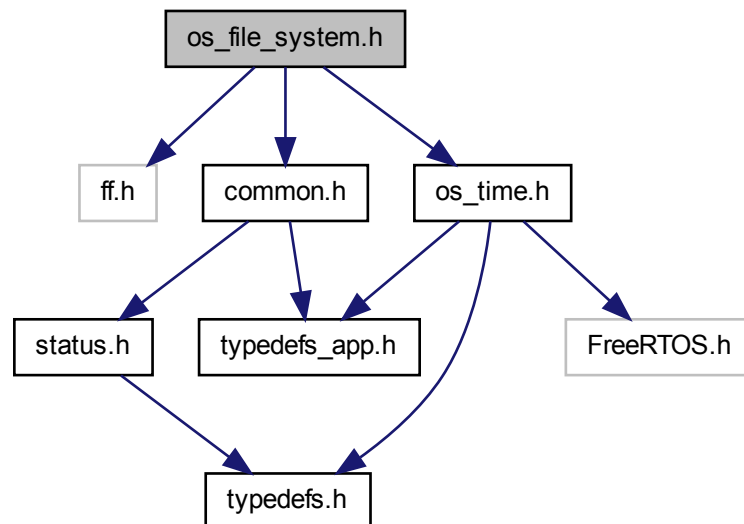
OS File system.

```
#include "ff.h"
```

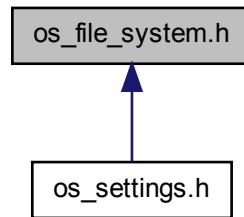
```
#include "common.h"
```

```
#include "os_time.h"
```

Include dependency graph for os\_file\_system.h:



This graph shows which files directly or indirectly include this file:



### 6.10.1 Detailed Description

OS File system.

Author

A. Filyanov

Definition in file [os\\_file\\_system.h](#).

## 6.11 os\_list.h File Reference

OS List.

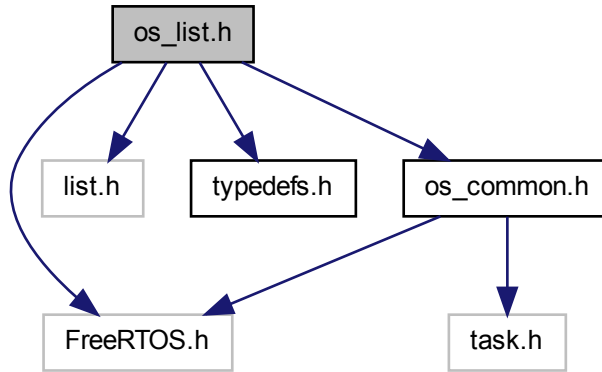
```
#include "FreeRTOS.h"
```

```
#include "list.h"
```

```
#include "typedefs.h"
```

```
#include "os_common.h"
```

Include dependency graph for os\_list.h:



## Defines

- `#define OS_LIST_CURRENT_LEN_GET(OS_ListP) listCURRENT_LENGTH(OS_ListP)`
- `#define OS_LIST_IS_EMPTY(OS_ListP) listLIST_IS_EMPTY(OS_ListP)`
- `#define OS_LIST_ITEM_FIRST_VALUE_GET(OS_ListP) listGET_ITEM_VALUE_OF_HEAD_ENTRY(OS_ListP)`
- `#define OS_LIST_ITEM_VALUE_GET(OS_ListItemP) listGET_LIST_ITEM_VALUE(OS_ListItemP)`
- `#define OS_LIST_ITEM_VALUE_SET(OS_ListItemP, OS_Value) listSET_LIST_ITEM_VALUE(OS_ListItemP, OS_Value)`
- `#define OS_LIST_ITEM_OWNER_GET(OS_ListItemP) listGET_LIST_ITEM_OWNER(OS_ListItemP)`
- `#define OS_LIST_ITEM_OWNER_SET(OS_ListItemP, OS_Owner) listSET_LIST_ITEM_OWNER(OS_ListItemP, OS_Owner)`
- `#define OS_LIST_ITEM_PREVIOUS_GET(OS_ListItemP) ((OS_ListItemP)->pxPrevious)`
- `#define OS_LIST_ITEM_NEXT_GET(OS_ListItemP) listGET_NEXT(OS_ListItemP)`
- `#define OS_LIST_ITEM_FIRST_GET(OS_ListP) listGET_HEAD_ENTRY(OS_ListP)`
- `#define OS_LIST_ITEM_LAST_GET(OS_ListP) ((OS_ListP)->xListEnd)`
- `#define OS_LIST_ITEM_NEXT_OWNER_GET(OS_Owner, OS_ListP) listGET_OWNER_OF_NEXT_ENTRY(OS_Owner, OS_ListP)`

- `#define OS_LIST_ITEM_FIRST_OWNER_GET(OS_ListP) listGET_OWNER_OF_HEAD_ENTRY(OS_ListP)`
- `#define OS_LIST_ITEM_IS_WITHIN(OS_ListP, OS_ListItemP) listIS_CONTAINED_WITHIN(OS_ListP, OS_ListItemP)`
- `#define OS_LIST_ITEM_CONTAINER_GET(OS_ListItemP) listLIST_ITEM_CONTAINER(OS_ListItemP)`
- `#define OS_LIST_IS_INITIALISED(OS_ListP) listLIST_IS_INITIALISED(OS_ListP)`

## Typedefs

- `typedef List_t OS_List`
- `typedef ListItem_t OS_ListItem`
- `typedef MiniListItem_t OS_ListItemLight`

## Functions

- void [OS\\_ListInit](#) (OS\_List \*list\_p)  
Initialise the list.
- OS\_ListItem \* [OS\\_ListItemCreate](#) (void)  
Create and initialize the list item.
- void [OS\\_ListItemDelete](#) (OS\_ListItem \*item\_l\_p)  
Remove and delete item from the list.
- void [OS\\_ListItemInit](#) (OS\_ListItem \*item\_l\_p)  
Initialise the list item.
- void [OS\\_ListInsert](#) (OS\_List \*list\_p, OS\_ListItem \*new\_item\_l\_p)  
Insert item to the list.
- void [OS\\_ListAppend](#) (OS\_List \*list\_p, OS\_ListItem \*new\_item\_l\_p)  
Append item to the list.
- U32 [OS\\_ListRemove](#) (OS\_ListItem \*item\_l\_p)  
Remove item from the list.
- OS\_ListItem \* [OS\\_ListItemByValueFind](#) (OS\_List \*list\_p, const OS\_Value value)  
Find list item by it's value.
- OS\_ListItem \* [OS\\_ListItemByOwnerFind](#) (OS\_List \*list\_p, const OS\_Owner owner)  
Find list item by it's owner.

- void [OS\\_ListItemsSwap](#) (OS\_ListItem \*item\_1\_p, OS\_ListItem \*item\_2\_p)  
Swap list items.

### 6.11.1 Detailed Description

OS List.

Author

A. Filyanov

Definition in file [os\\_list.h](#).

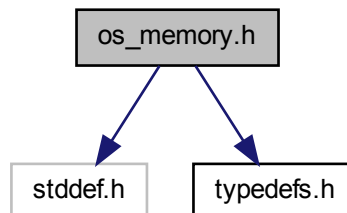
## 6.12 os\_memory.h File Reference

OS Memory.

```
#include <stddef.h>
```

```
#include "typedefs.h"
```

Include dependency graph for os\_memory.h:



### Data Structures

- struct [OS\\_MemoryDesc](#)  
Memory description.
- struct [OS\\_MemoryStat](#)  
Memory statistics.



## Typedefs

- typedef U32 OS\_MemoryType

## Enumerations

- enum {  
OS\_MEM\_RAM\_INT\_SRAM, OS\_MEM\_RAM\_INT\_CCM, OS\_MEM\_RAM\_EXT\_SRAM, OS\_MEM\_LAST,  
OS\_MEM\_UNDEF }  
Memory type.

## Functions

- void \* [OS\\_Malloc](#) (const U32 size)  
Common functions.
- void \* [OS\\_MallocEx](#) (const U32 size, const OS\_MemoryType mem\_type)  
Allocate memory by type.
- void [OS\\_Free](#) (void \*addr\_p)  
Free allocated memory.
- void [OS\\_FreeEx](#) (void \*addr\_p, const OS\_MemoryType mem\_type)  
Free allocated memory by type.
- void [OS\\_MemCacheFlush](#) (void)  
Flush memory caches.
- void [OS\\_MemCpy8](#) (void \*dst\_p, const void \*src\_p, SIZE size8)  
Copy memory in bytes.
- void [OS\\_MemCpy32](#) (void \*dst\_p, const void \*src\_p, SIZE size32)  
Copy memory in words.
- OS\_MemoryType [OS\\_MemoryTypeHeapNextGet](#) (const OS\_MemoryType mem\_type)  
Get the next memory heap type.
- Status [OS\\_MemoryStatGet](#) (const OS\_MemoryType mem\_type, [OS\\_MemoryStat](#) \*mem\_stat\_p)  
Get the memory heap usage statistics.

### 6.12.1 Detailed Description

OS Memory.

Author

A. Filyanov

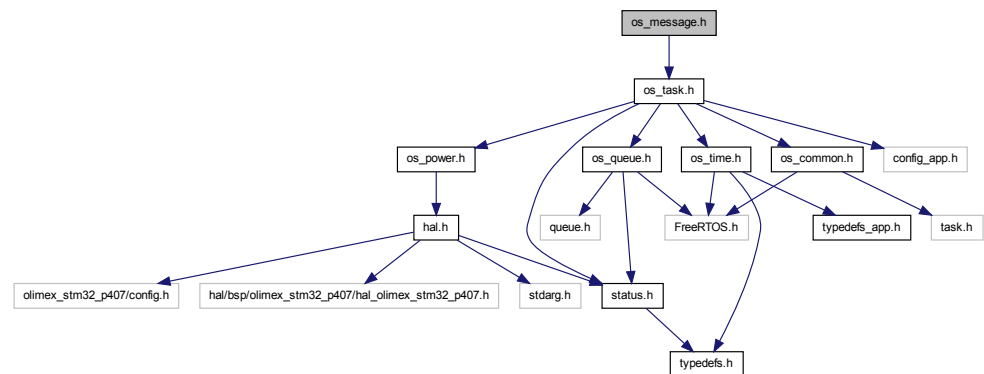
Definition in file [os\\_memory.h](#).

### 6.13 os\_message.h File Reference

OS Message.

```
#include "os_task.h"
```

Include dependency graph for os\_message.h:



### Data Structures

- struct [OS\\_Message](#)

### Typedefs

- typedef U16 OS\_MessageId

### Enumerations

- enum { OS\_MSG\_UNDEF, OS\_MSG\_BROADCAST, OS\_MSG\_CMD, OS\_MSG\_APP = 32 }

## Functions

- `OS_Message * OS_MessageCreate` (const OS\_MessageId id, const U16 size, const TimeMs timeout, const void \*data\_p)  
Create a message.
- void `OS_MessageDelete` (OS\_Message \*msg\_p)  
Delete the message.
- Status `OS_MessageSend` (const OS\_QueueHd qhd, const OS\_Message \*msg\_p, const TimeMs timeout, const OS\_MessagePrio priority)  
Send the message.
- Status `OS_MessageMulticastSend` (const OS\_QueueHd receivers\_qhd\_v[], const OS\_Message \*msg\_p, const TimeMs timeout, const OS\_MessagePrio priority)  
Send the multicast message.
- Status `OS_MessageReceive` (const OS\_QueueHd qhd, OS\_Message \*\*msg\_pp, const TimeMs timeout)  
Receive the message.
- Status `OS_ISR_MessageSend` (const OS\_QueueHd qhd, const OS\_Message \*msg\_p, const OS\_MessagePrio priority)  
Send the message.
- Status `OS_ISR_MessageReceive` (const OS\_QueueHd qhd, OS\_Message \*\*msg\_pp)  
Receive the message.

### 6.13.1 Detailed Description

OS Message.

Author

A. Filyanov

Definition in file [os\\_\\_message.h](#).

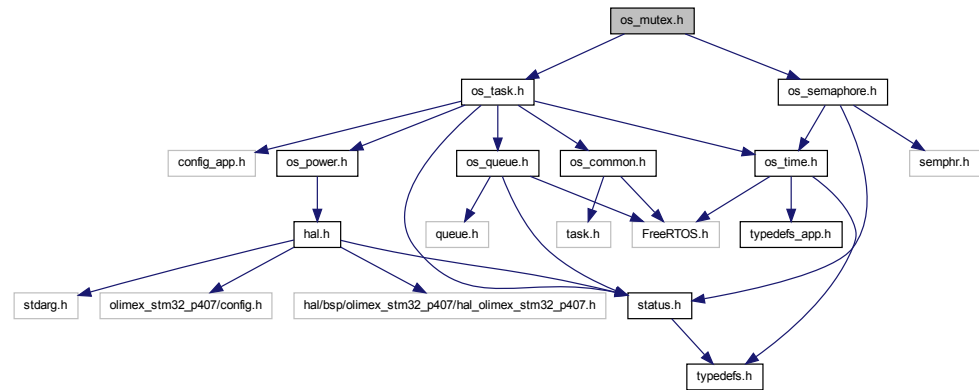
## 6.14 os\_\_mutex.h File Reference

OS Mutex.

```
#include "os__task.h"
```

```
#include "os__semaphore.h"
```

Include dependency graph for `os_mutex.h`:



## Typedefs

- typedef OS\_SemaphoreHd OS\_MutexHd
- typedef OS\_SemaphoreState OS\_MutexState

## Functions

- OS\_MutexHd [OS\\_MutexCreate](#) (void)  
Create a mutex.
- OS\_MutexHd [OS\\_MutexRecursiveCreate](#) (void)  
Create a recursive mutex.
- void [OS\\_MutexDelete](#) (const OS\_MutexHd mhd)  
Delete the mutex.
- Status [OS\\_MutexLock](#) (const OS\_MutexHd mhd, const TimeMs timeout)  
Lock the mutex.
- Status [OS\\_MutexRecursiveLock](#) (const OS\_MutexHd mhd, const TimeMs timeout)  
Recursive lock the mutex.
- Status [OS\\_MutexUnlock](#) (const OS\_MutexHd mhd)  
Unlock the mutex.
- Status [OS\\_MutexRecursiveUnlock](#) (const OS\_MutexHd mhd)

Recursive unlock the mutex.

- OS\_MutexState [OS\\_MutexCheck](#) (const OS\_MutexHd mhd)

Check mutex state.

- OS\_MutexState [OS\\_MutexRecursiveCheck](#) (const OS\_MutexHd mhd)

Check recursive mutex state.

- OS\_TaskHd [OS\\_MutexParentGet](#) (const OS\_MutexHd mhd)

Get mutex parent.

- Status [OS\\_ISR\\_MutexLock](#) (const OS\_MutexHd mhd)

Lock the mutex.

- Status [OS\\_ISR\\_MutexUnlock](#) (const OS\_MutexHd mhd)

Unlock the mutex.

- OS\_MutexState [OS\\_ISR\\_MutexCheck](#) (const OS\_MutexHd mhd)

Check mutex state.

### 6.14.1 Detailed Description

OS Mutex.

Author

A. Filyanov

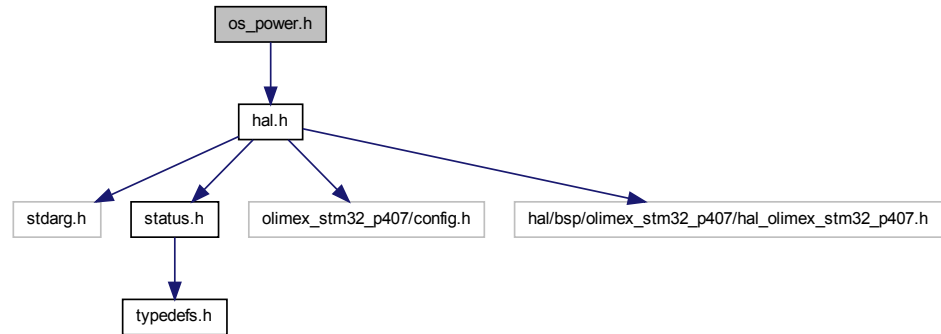
Definition in file [os\\_mutex.h](#).

## 6.15 os\_power.h File Reference

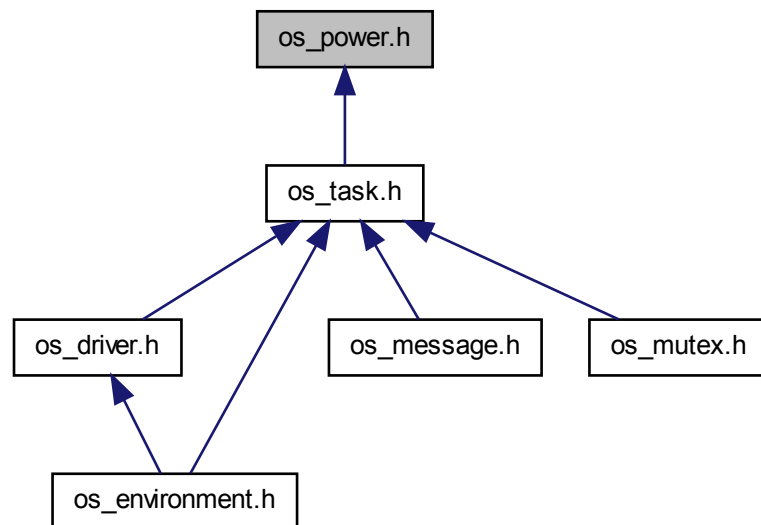
OS Power.

#include "hal.h"

Include dependency graph for `os_power.h`:



This graph shows which files directly or indirectly include this file:



## Typedefs

- typedef HAL\_PowerState OS\_PowerState

- typedef HAL\_PowerPrio OS\_PowerPrio

## Enumerations

- enum { OS\_PWR\_PRIO\_UNDEF, OS\_PWR\_PRIO\_DEFAULT = 1, OS\_PWR\_PRIO\_MAX = 255, OS\_PWR\_PRIO\_LAST = OS\_PWR\_PRIO\_MAX }

## Functions

- Status [OS\\_PowerInit](#) (void)  
Init power.
- OS\_PowerState [OS\\_PowerStateGet](#) (void)  
Get current system power state.
- Status [OS\\_PowerStateSet](#) (const OS\_PowerState state)  
Set current system power state.
- ConstStrPtr [OS\\_PowerStateNameGet](#) (const OS\_PowerState state)  
Get name of the current system power state.
- Status [OS\\_ISR\\_PowerStateSet](#) (const OS\_PowerState state)  
Set current system power state.

### 6.15.1 Detailed Description

OS Power.

Author

A. Filyanov

Definition in file [os\\_power.h](#).

## 6.16 os\_queue.h File Reference

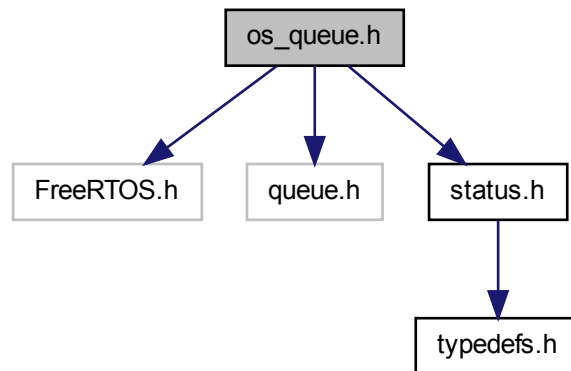
OS Queue.

```
#include "FreeRTOS.h"
```

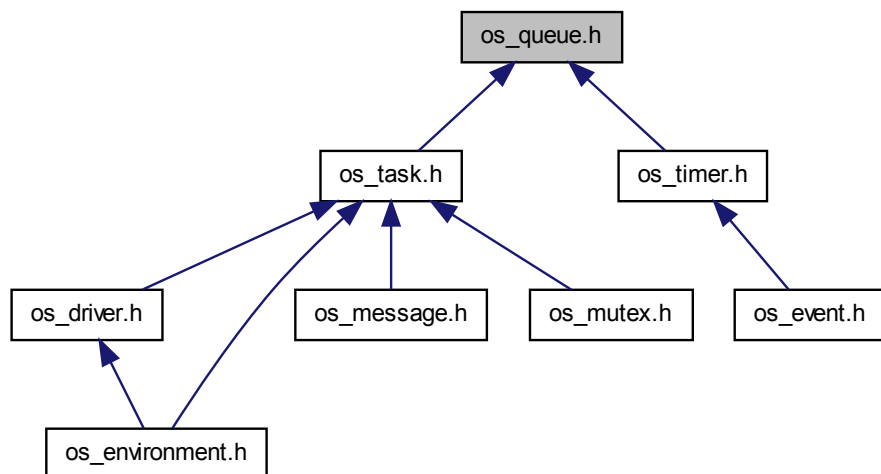
```
#include "queue.h"
```

```
#include "status.h"
```

Include dependency graph for `os_queue.h`:



This graph shows which files directly or indirectly include this file:





## Data Structures

- struct [OS\\_QueueConfig](#)
- struct [OS\\_QueueStats](#)

## Typedefs

- typedef void \* [OS\\_QueueHd](#)

## Functions

- Status [OS\\_QueueCreate](#) (const [OS\\_QueueConfig](#) \*cfg\_p, [OS\\_TaskHd](#) parent\_thd, [OS\\_QueueHd](#) \*qhd\_p)  
Create a queue.
- Status [OS\\_QueueDelete](#) (const [OS\\_QueueHd](#) qhd)  
Delete the queue.
- Status [OS\\_QueueReceive](#) (const [OS\\_QueueHd](#) qhd, void \*item\_p, const [TimeMs](#) timeout)  
Receive the item.
- Status [OS\\_QueueSend](#) (const [OS\\_QueueHd](#) qhd, const void \*item\_p, const [TimeMs](#) timeout, const [OS\\_MessagePrio](#) priority)  
Send the item.
- Status [OS\\_QueueFlush](#) (const [OS\\_QueueHd](#) qhd)  
Flush the queue.
- U32 [OS\\_QueueItemsCountGet](#) (const [OS\\_QueueHd](#) qhd)  
Get queue items count.
- Status [OS\\_QueueConfigGet](#) (const [OS\\_QueueHd](#) qhd, [OS\\_QueueConfig](#) \*config\_p)  
Get queue config.
- Status [OS\\_QueueStatsGet](#) (const [OS\\_QueueHd](#) qhd, [OS\\_QueueStats](#) \*stats\_p)  
Get queue statistics.
- [OS\\_TaskHd](#) [OS\\_QueueParentGet](#) (const [OS\\_QueueHd](#) qhd)  
Get queue parent.
- [OS\\_QueueHd](#) [OS\\_QueueSvcStdInGet](#) (void)  
Get system service task standart input/output queue.

- U32 [OS\\_QueueCountGet](#) (void)  
Get system queues count.
- OS\_QueueHd [OS\\_QueueNextGet](#) (const OS\_QueueHd qhd)  
Get the next queue.
- Status [OS\\_ISR\\_QueueReceive](#) (const OS\_QueueHd qhd, void \*item\_p)  
Receive the item.
- Status [OS\\_ISR\\_QueueSend](#) (const OS\_QueueHd qhd, const void \*item\_p, const OS\_MessagePrio priority)  
Send the item.
- U32 [OS\\_ISR\\_QueueItemsCountGet](#) (const OS\_QueueHd qhd)  
Get queue items count.

### 6.16.1 Detailed Description

OS Queue.

Author

A. Filyanov

Definition in file [os\\_queue.h](#).

## 6.17 os\_semaphore.h File Reference

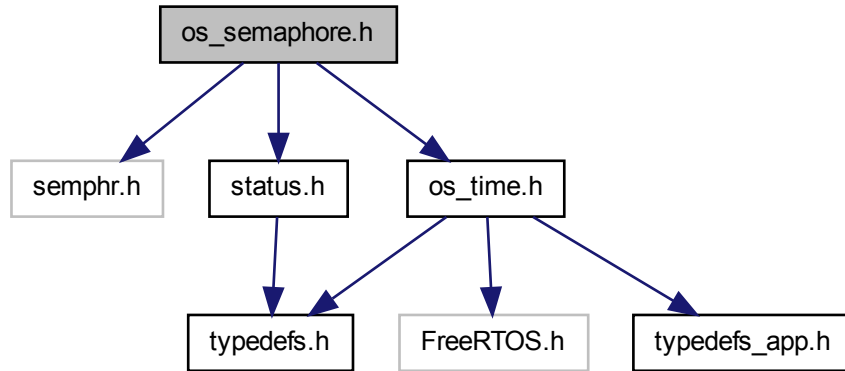
OS Semaphore.

```
#include "semphr.h"
```

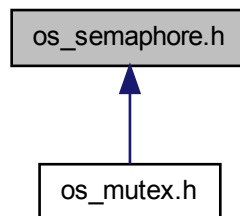
```
#include "status.h"
```

```
#include "os_time.h"
```

Include dependency graph for os\_semaphore.h:



This graph shows which files directly or indirectly include this file:



## Typedefs

- typedef MutexState OS\_SemaphoreState
- typedef SemaphoreHandle\_t OS\_SemaphoreHd

## Functions

- void [OS\\_SemaphoreBinaryCreate](#) (OS\_SemaphoreHd shd)

Create a binary semaphore.

- OS\_SemaphoreHd [OS\\_SemaphoreCountingCreate](#) (const U32 count\_max, const U32 count\_init)

Create a counting semaphore.

- void [OS\\_SemaphoreDelete](#) (const OS\_SemaphoreHd shd)

Delete the semaphore.

- Status [OS\\_SemaphoreLock](#) (const OS\_SemaphoreHd shd, const TimeMs timeout)

Lock the semaphore.

- Status [OS\\_SemaphoreUnlock](#) (const OS\_SemaphoreHd shd)

Unlock the semaphore.

- OS\_SemaphoreState [OS\\_SemaphoreCheck](#) (const OS\_SemaphoreHd shd)

Check semaphore state.

- Status [OS\\_ISR\\_SemaphoreLock](#) (const OS\_SemaphoreHd shd)

Lock the semaphore.

- Status [OS\\_ISR\\_SemaphoreUnlock](#) (const OS\_SemaphoreHd shd)

Unlock the semaphore.

- OS\_SemaphoreState [OS\\_ISR\\_SemaphoreCheck](#) (const OS\_SemaphoreHd shd)

Check semaphore state.

### 6.17.1 Detailed Description

OS Semaphore.

Author

A. Filyanov

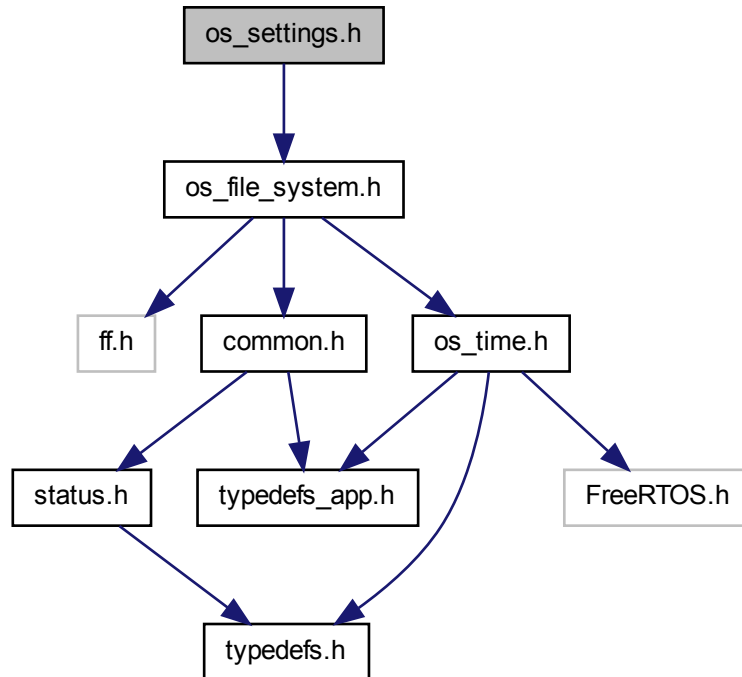
Definition in file [os\\_semaphore.h](#).

## 6.18 os\_settings.h File Reference

OS Settings.

```
#include "os_file_system.h"
```

Include dependency graph for os\_settings.h:



## Data Structures

- struct [OS\\_SettingsItem](#)

## Enumerations

- enum `OS_SettingsStatus` { `S_SETT_UNDEF` = `S_MODULE`, `S_SETT_READ`, `S_SETT_WRITE` }

## Functions

- Status [OS\\_SettingsInit](#) (void)  
Initialise the settings.
- Status [OS\\_SettingsDeInit](#) (void)

Deinitialise the settings.

- Status [OS\\_SettingsDelete](#) (ConstStrPtr file\_path\_p, ConstStrPtr section\_p, ConstStrPtr key\_p)

Delete settings item.

- Status [OS\\_SettingsRead](#) (ConstStrPtr file\_path\_p, ConstStrPtr section\_p, ConstStrPtr key\_p, StrPtr value\_p)

Read settings item.

- Status [OS\\_SettingsWrite](#) (ConstStrPtr file\_path\_p, ConstStrPtr section\_p, ConstStrPtr key\_p, ConstStrPtr value\_p)

Write settings item.

- Status [OS\\_SettingsItemsRead](#) (ConstStrPtr file\_path\_p, [OS\\_SettingsItem](#) items[])

Read settings items.

- Status [OS\\_SettingsItemsWrite](#) (ConstStrPtr file\_path\_p, [OS\\_SettingsItem](#) items[])

Write settings items.

### 6.18.1 Detailed Description

OS Settings.

Author

A. Filyanov

Definition in file [os\\_settings.h](#).

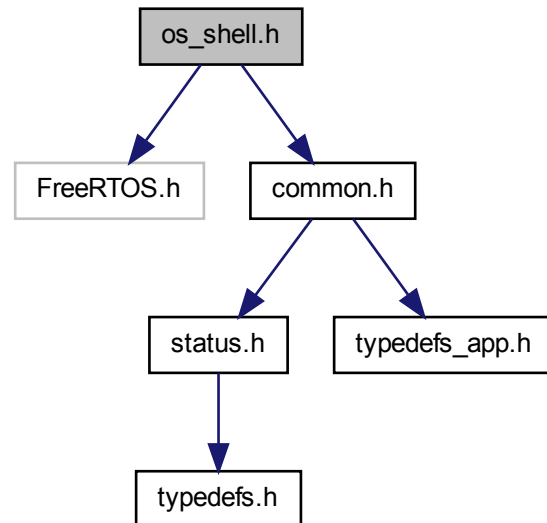
## 6.19 os\_shell.h File Reference

OS Shell.

```
#include "FreeRTOS.h"
```

```
#include "common.h"
```

Include dependency graph for os\_shell.h:



## Data Structures

- struct [OS\\_ShellCommandConfig](#)

## Defines

- `#define SHELL_COMMAND_UNDEF OS_NULL`

## Typedefs

- typedef Status(\* OS\_ShellCommandHandler )(const U32 argc, ConstStrPtr argv[])
- typedef [OS\\_ShellCommandConfig](#) \* OS\_ShellCommandHd

## Enumerations

- enum OS\_ShellOptions { OS\_SHELL\_OPT\_UNDEF }

## Functions

- Status [OS\\_ShellInit](#) (void)  
Initialise shell.
- Status [OS\\_ShellCommandCreate](#) (const [OS\\_ShellCommandConfig](#) \*cmd\_cfg\_p)  
Create shell command.
- Status [OS\\_ShellCommandDelete](#) (ConstStrPtr name\_p)  
Delete shell command.
- Status [OS\\_ShellCommandExecute](#) (void)  
Execute current shell command.
- Status [OS\\_ShellArgumentsNumberCheck](#) (const [OS\\_ShellCommandHd](#) cmd\_hd, const U8 argc)  
Check shell command arguments number.
- [OS\\_ShellCommandHd](#) [OS\\_ShellCommandByNameGet](#) (ConstStrPtr name\_p)  
Get shell command by it's name.
- [OS\\_ShellCommandHd](#) [OS\\_ShellCommandNextGet](#) (const [OS\\_ShellCommandHd](#) cmd\_hd)  
Get the next shell command.
- ConstStrPtr [OS\\_ShellPromptGet](#) (void)  
Get shell command prompt.
- Status [OS\\_ShellCls](#) (void)  
Clear shell buffer.
- void [OS\\_ShellClHandler](#) (const U8 c)  
Execute shell command line handler.

### 6.19.1 Detailed Description

OS Shell.

Author

A. Filyanov

Warning

Functions are only can be called from single instance of the shell task!  
No thread safe, no stdio driver protection!



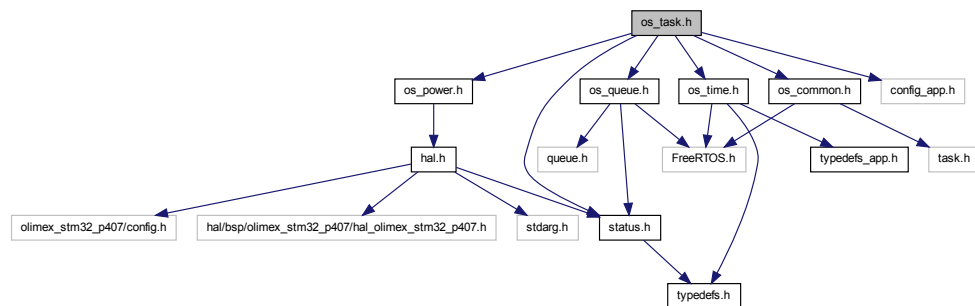
Definition in file [os\\_shell.h](#).

## 6.20 os\_task.h File Reference

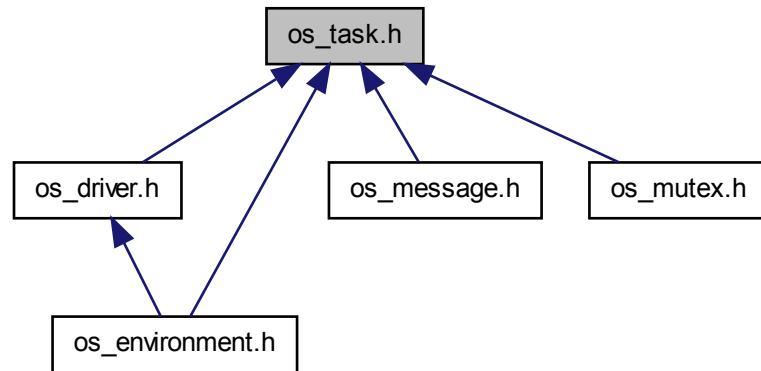
OS Task.

```
#include "status.h"
#include "config_app.h"
#include "os_common.h"
#include "os_power.h"
#include "os_time.h"
#include "os_queue.h"
```

Include dependency graph for os\_task.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [OS\\_TaskConfig](#)

## Defines

- `#define OS_THIS_TASK OS_NULL`  
Common declarations.
- `#define OS_STDIO_LEN 4`

## Typedefs

- `typedef U8 OS_StdIoDir`
- `typedef U8 OS_TaskAttrs`
- `typedef U8 OS_TaskState`
- `typedef U8 OS_TaskPrio`
- `typedef OS_Owner OS_TaskHd`
- `typedef U8 OS_TaskId`
- `typedef void OS_TaskArgs`
- `typedef TaskStatus_t OS_TaskStats`

## Enumerations

- enum { OS\_STDIO\_IN, OS\_STDIO\_OUT }
- enum { OS\_TASK\_ATTR\_UNDEF, OS\_TASK\_ATTR\_RECREATE, OS\_TASK\_ATTR\_LAST }
- enum {  
OS\_TASK\_STATE\_UNDEF, OS\_TASK\_STATE\_READY, OS\_TASK\_STATE\_RUN, OS\_TASK\_STATE\_BLOCK,  
OS\_TASK\_STATE\_SUSPEND, OS\_TASK\_STATE\_DELETED, OS\_TASK\_STATE\_LAST }
- enum {  
OS\_TASK\_PRIO\_UNDEF, OS\_TASK\_PRIO\_LOW = OS\_PRIORITY\_MIN, OS\_TASK\_PRIO\_BELOW\_NORMAL, OS\_TASK\_PRIO\_NORMAL,  
OS\_TASK\_PRIO\_ABOVE\_NORMAL, OS\_TASK\_PRIO\_HIGH, OS\_TASK\_PRIO\_REALTIME, OS\_TASK\_PRIO\_MAX = OS\_PRIORITY\_MAX - 1,  
OS\_TASK\_PRIO\_LAST = OS\_TASK\_PRIO\_MAX }

## Functions

- static Status [OS\\_TaskInit](#) (OS\_TaskArgs \*args\_p)  
Init task.
- static void [OS\\_TaskMain](#) (OS\_TaskArgs \*args\_p)  
Task main function.
- static Status [OS\\_TaskPower](#) (OS\_TaskArgs \*args\_p, const OS\_PowerState state)  
Task main function.
- Status [OS\\_TaskCreate](#) (const [OS\\_TaskConfig](#) \*cfg\_p, OS\_TaskHd \*thd\_p)  
Create a task.
- Status [OS\\_TaskDelete](#) (const OS\_TaskHd thd)  
Delete the task.
- void [OS\\_TaskDelay](#) (const TimeMs timeout)  
Delay the task.
- void [OS\\_TaskDelayUntil](#) (OS\_Tick \*tick\_last\_p, const TimeMs timeout)  
Delay the task until.
- void [OS\\_TaskSuspend](#) (const OS\_TaskHd thd)  
Suspend the task.

- void [OS\\_TaskResume](#) (const OS\_TaskHd thd)  
Resume the task.
- OS\_TaskId [OS\\_TaskIdGet](#) (const OS\_TaskHd thd)  
Get task id.
- OS\_TaskHd [OS\\_TaskHdGet](#) (void)  
Get current task handle.
- OS\_TaskHd [OS\\_TaskHdByIdGet](#) (const OS\_TaskId tid)  
Get task handle by it's id.
- OS\_TaskHd [OS\\_TaskHdParentGet](#) (void)  
Get current task parent's handle.
- OS\_TaskHd [OS\\_TaskHdParentByHdGet](#) (const OS\_TaskHd thd)  
Get parent's task by task handle.
- U32 [OS\\_TasksCountGet](#) (void)  
Get tasks count.
- U32 [OS\\_TasksStatsGet](#) (OS\_TaskStats \*stats\_p, const U32 stats\_count, U32 \*uptime\_p)  
Get tasks statistics.
- OS\_TaskState [OS\\_TaskStateGet](#) (const OS\_TaskHd thd)  
Get task state.
- ConstStrPtr [OS\\_TaskStateNameGet](#) (const OS\_TaskState state)  
Get task state name.
- ConstStrPtr [OS\\_TaskNameGet](#) (const OS\_TaskHd thd)  
Get task name.
- OS\_TaskAttrs [OS\\_TaskAttrsGet](#) (const OS\_TaskHd thd)  
Get task attributes.
- const [OS\\_TaskConfig](#) \* [OS\\_TaskConfigGet](#) (const OS\_TaskHd thd)  
Get task configuration.
- void \* [OS\\_TaskStorageGet](#) (const OS\_TaskHd thd)  
Get task storage.
- OS\_PowerState [OS\\_TaskPowerStateGet](#) (const OS\_TaskHd thd)  
Get task power state.

- OS\_TaskPrio [OS\\_TaskPriorityGet](#) (const OS\_TaskHd thd)  
Get task priority.
- Status [OS\\_TaskPrioritySet](#) (const OS\_TaskHd thd, const OS\_TaskPrio prio)  
Set task priority.
- OS\_TaskHd [OS\\_TaskByNameGet](#) (ConstStrPtr name\_p)  
Get task by it's name.
- OS\_TaskHd [OS\\_TaskNextGet](#) (const OS\_TaskHd thd)  
Get the next task.
- OS\_QueueHd [OS\\_TaskSvcStdInGet](#) (void)  
Get the system supervisor task standart input queue.
- OS\_QueueHd [OS\\_TaskStdIoGet](#) (const OS\_TaskHd thd, const OS\_StdIoDir dir)  
Get the task standart input/output queue.

### 6.20.1 Detailed Description

OS Task.

Author

A. Filyanov

Definition in file [os\\_task.h](#).

## 6.21 os\_time.h File Reference

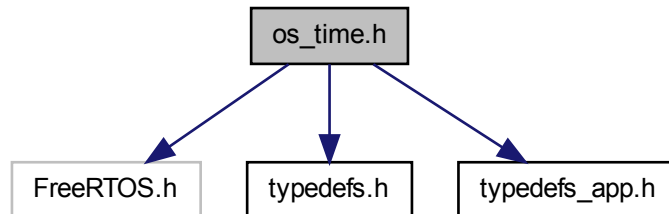
OS Time.

```
#include "FreeRTOS.h"
```

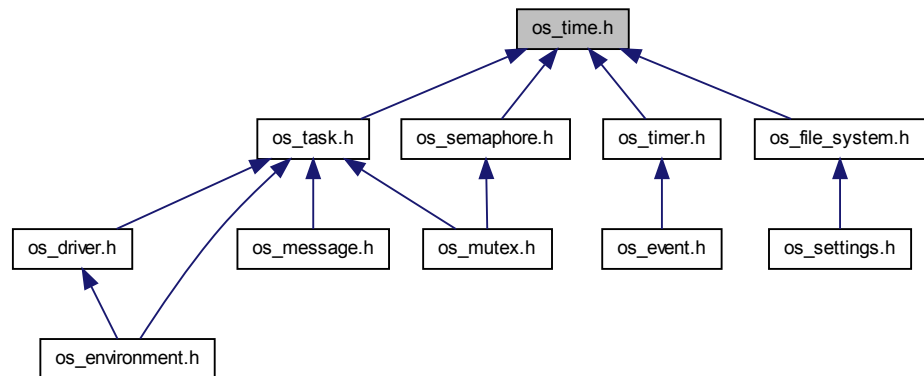
```
#include "typedefs.h"
```

```
#include "typedefs_app.h"
```

Include dependency graph for `os_time.h`:



This graph shows which files directly or indirectly include this file:



## Defines

- `#define OS_TICKS_TO_MS(ticks) (((U32)((((ticks) * 1000UL) / configTICK_RATE_HZ)))`  
Converts from RTOS ticks to milliseconds.

## Typedefs

- `typedef Time OS_DateTime`

- typedef TickType\_t OS\_Tick
- typedef U32 TimeMs
- typedef U32 TimeS

## Enumerations

- enum OS\_TimeWeekDay {  
OS\_WEEK\_DAY\_UNDEF, OS\_WEEK\_DAY\_MONDAY, OS\_WEEK\_DAY\_TUESDAY, OS\_WEEK\_DAY\_WEDNESDAY,  
OS\_WEEK\_DAY\_THURSDAY, OS\_WEEK\_DAY\_FRIDAY, OS\_WEEK\_DAY\_SATURDAY, OS\_WEEK\_DAY\_SUNDAY,  
OS\_WEEK\_DAY\_LAST }
- enum OS\_TimeFormat {  
OS\_TIME\_UNDEF, OS\_TIME\_GMT, OS\_TIME\_GMT\_OFFSET, OS\_TIME\_LOCAL,  
OS\_TIME\_UPTIME, OS\_TIME\_LAST }
- enum OS\_DateFormat { OS\_DATE\_UNDEF, OS\_DATE\_LAST }
- enum OS\_AlarmFormat { OS\_ALARM\_UNDEF, OS\_ALARM\_LAST }
- enum OS\_TimeDayLight { OS\_TIME\_DAYLIGHT\_UNDEF, OS\_TIME\_DAYLIGHT\_SUMMER, OS\_TIME\_DAYLIGHT\_WINTER, OS\_TIME\_DAYLIGHT\_LAST }

## Functions

- static U32 [OS\\_MS\\_TO\\_TICKS](#) (const TimeMs ms)  
Converts from milliseconds to RTOS ticks, value is always > 0.
- Status [OS\\_TimeGet](#) (const OS\_TimeFormat format, OS\_DateTime \*os\_time\_p)  
Get the current time.
- Status [OS\\_TimeSet](#) (const OS\_TimeFormat format, OS\_DateTime \*os\_time\_p)  
Set time.
- Status [OS\\_DateGet](#) (const OS\_DateFormat format, OS\_DateTime \*os\_date\_p)  
Get the current date.
- Status [OS\\_DateSet](#) (const OS\_DateFormat format, OS\_DateTime \*os\_date\_p)  
Set date.
- BL [OS\\_TimeIsValid](#) (const U8 hour, const U8 min, const U8 sec)

Time validation.

- BL [OS\\_DateIsValid](#) (const U16 year, const U8 month, const U8 day)  
Date validation.
- OS\_TimeWeekDay [OS\\_DateWeekDayGet](#) (const U16 year, const U8 month, const U8 day)  
Get the day of the week.
- ConstStrPtr [OS\\_TimeNameDayOfWeekGet](#) (const OS\_TimeWeekDay week\_day, const Locale locale)  
Get the day of the week name.
- OS\_TimeDayLight [OS\\_TimeDayLightSavingsGet](#) (void)  
Get the current daylight savings.
- Status [OS\\_TimeDayLightSavingsSet](#) (const OS\_TimeDayLight savings)  
  
Set the daylight savings.
- OS\_Tick [OS\\_TickCountGet](#) (void)  
Get tick count.
- OS\_DateTime [OS\\_TimeStringParse](#) (ConstStrPtr time\_p)  
Parse the time string.
- OS\_DateTime [OS\\_DateStringParse](#) (ConstStrPtr date\_p)  
Parse the date string.
- OS\_Tick [OS\\_ISR\\_TickCountGet](#) (void)  
Get tick count.

### 6.21.1 Detailed Description

OS Time.

Author

A. Filyanov

Definition in file [os\\_time.h](#).

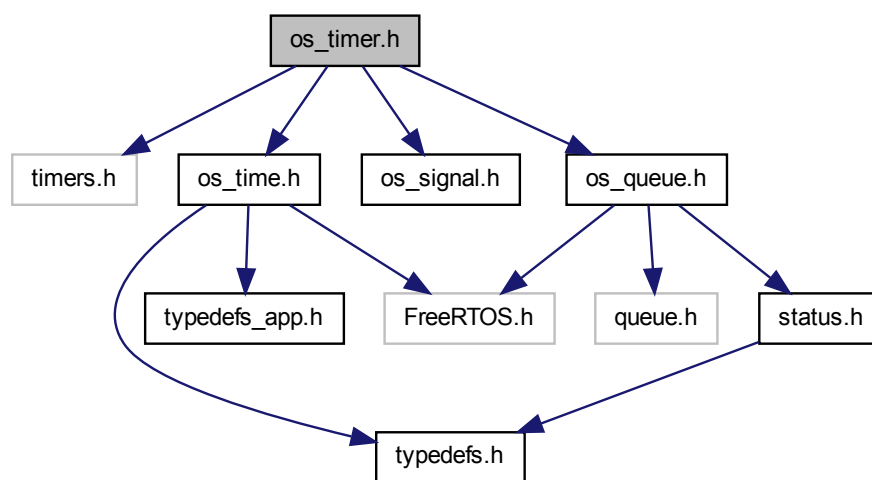
## 6.22 os\_timer.h File Reference

OS Timer.

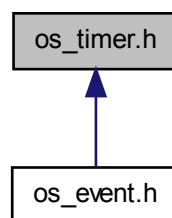


```
#include "timers.h"  
#include "os_time.h"  
#include "os_signal.h"  
#include "os_queue.h"
```

Include dependency graph for os\_timer.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [OS\\_TimerConfig](#)

## Typedefs

- typedef TimerHandle\_t OS\_TimerHd
- typedef OS\_SignalData OS\_TimerId
- typedef struct [OS\\_TimerConfig](#) OS\_TimerStats

## Enumerations

- enum { OS\_TIM\_ID\_UNDEF, OS\_TIM\_ID\_APP = 0x01, OS\_TIM\_ID\_LAST }
- enum OS\_TimerOptions { OS\_TIM\_OPT\_UNDEF, OS\_TIM\_OPT\_PERIODIC, OS\_TIM\_OPT\_EVENT }

## Functions

- Status [OS\\_TimerCreate](#) (const [OS\\_TimerConfig](#) \*cfg\_p, OS\_TimerHd \*timer\_hd\_p)  
Create a timer.
- Status [OS\\_TimerDelete](#) (const OS\_TimerHd timer\_hd, const TimeMs timeout)  
Delete the timer.
- Status [OS\\_TimerReset](#) (const OS\_TimerHd timer\_hd, const TimeMs timeout)  
Reset the timer.
- Status [OS\\_TimerStart](#) (const OS\_TimerHd timer\_hd, const TimeMs timeout)  
Start the timer.
- Status [OS\\_TimerStop](#) (const OS\_TimerHd timer\_hd, const TimeMs timeout)  
Stop the timer.
- Status [OS\\_TimerPeriodGet](#) (const OS\_TimerHd timer\_hd, TimeMs \*period\_p)  
Get the timer period.
- Status [OS\\_TimerPeriodSet](#) (const OS\_TimerHd timer\_hd, const TimeMs new\_period, const TimeMs timeout)  
Set the timer period.

- BL [OS\\_TimerIsActive](#) (const OS\_TimerHd timer\_hd)  
Get the timer slot.
- OS\_TimerId [OS\\_TimerIdGet](#) (const OS\_TimerHd timer\_hd)  
Get the timer's id.
- OS\_TimerHd [OS\\_TimerByIdGet](#) (const OS\_TimerId timer\_id)  
Get the timer by id.
- ConstStrPtr [OS\\_TimerNameGet](#) (const OS\_TimerHd timer\_hd)  
Get timer name.
- OS\_TimerHd [OS\\_TimerByNameGet](#) (ConstStrPtr name\_p)  
Get the timer by its name.
- Status [OS\\_TimerStatsGet](#) (const OS\_TimerHd timer\_hd, [OS\\_TimerStats](#) \*stats\_p)  
Get timer statistics.
- OS\_TimerHd [OS\\_TimerNextGet](#) (const OS\_TimerHd timer\_hd)  
Get the next timer.
- Status [OS\\_ISR\\_TimerReset](#) (const OS\_TimerHd timer\_hd)  
Reset the timer.
- Status [OS\\_ISR\\_TimerStart](#) (const OS\_TimerHd timer\_hd)  
Start the timer.
- Status [OS\\_ISR\\_TimerStop](#) (const OS\_TimerHd timer\_hd)  
Stop the timer.
- Status [OS\\_ISR\\_TimerPeriodChange](#) (const OS\_TimerHd timer\_hd, const TimeMs new\_period)  
Change the timer period.

### 6.22.1 Detailed Description

OS Timer.

Author

A. Filyanov

Definition in file [os\\_timer.h](#).

## 6.23 protocol.h File Reference

### Data Structures

- struct [Packet](#)  
Пакет.
- struct [RouteListItem](#)
- struct [RouteItem](#)
- struct [ProtocolHeaderInfo](#)
- struct [ProtocolId](#)
- struct [CommandDeviceDescription](#)  
Данные описания устройства.

### Defines

- `#define PACK_VAL_PROTO 1`
- `#define PROTO_PACKET_SIZE_MAX 64`
- `#define PROTO_ID_LEN 4`

### Typedefs

- `typedef int OS_Driver`
- `typedef U16 RouteAddr`
- `typedef U8 CommandId`
- `typedef U8 PortApp`
- `typedef U8 ProtocolIdInfo`

### Enumerations

- enum [ProtocolCommand](#) {  
CMD\_NONE, CMD\_ACKN, CMD\_NACK, CMD\_PING,  
CMD\_ECHO, CMD\_TEST, CMD\_LAST = 64 }  
Команды.
- enum { CMD\_ID\_NONE, CMD\_ID\_LAST }
- enum {  
PROTO\_ID\_INFO\_NONE, PROTO\_ID\_INFO\_PING, PROTO\_ID\_INFO\_ECHO,  
PROTO\_ID\_INFO\_HELLO,  
PROTO\_ID\_INFO\_RECEIVE\_READY, PROTO\_ID\_INFO\_DISCONNECT,  
PROTO\_ID\_INFO\_LAST }

- enum {  
    [PROTO\\_ID\\_VER](#) = 0x00, PROTO\_ID\_TYP = 0x00, PROTO\_ID\_-  
    SEQ = 0x00, PROTO\_ID\_ACK = 0x00,  
    PROTO\_ID\_NAK = 0x00, PROTO\_ID\_DNT = 0x00, PROTO\_ID\_-  
    LSF = 0x00 }

Маски атрибутов командного пакета.

## Functions

- [PACKED](#) (PACK\_VAL\_PROTO, typedef struct {U8 id[PROTO\_ID\_-  
    LEN];U8 payload[0];}ProtocolHeaderFrame)

Заголовок командного пакета.

### 6.23.1 Detailed Description

Author

A. Filyanov

Definition in file [protocol.h](#).

### 6.23.2 Enumeration Type Documentation

#### 6.23.2.1 anonymous enum

Маски атрибутов командного пакета.

Enumerator:

PROTO\_ID\_VER Protocol version.

Definition at line 86 of file protocol.h.

```
{  
    PROTO_ID_VER = 0x00,  
    PROTO_ID_TYP = 0x00,  
    PROTO_ID_SEQ = 0x00,  
    PROTO_ID_ACK = 0x00,  
    PROTO_ID_NAK = 0x00,  
    PROTO_ID_DNT = 0x00,  
    PROTO_ID_LSF = 0x00  
};
```

### 6.23.2.2 enum ProtocolCommand

Команды.

Все команды из списка обязаны иметь поддержку в виде обработчиков с обеих сторон(Host<->Target).

Definition at line 21 of file protocol.h.

```
{
    CMD_NONE,
    CMD_ACKN,
    CMD_NACK,
    CMD_PING,
    CMD_ECHO,
    CMD_TEST,
    CMD_LAST = 64
} ProtocolCommand;
```

### 6.23.3 Function Documentation

#### 6.23.3.1 PACKED ( PACK\_VAL\_PROTO )

Заголовок командного пакета.

Дескриптор устройства.

Дескриптор DSPMB. (Плата управления).

Версия ПО.

Применяется для всех исходящих командных пакетов.

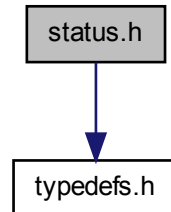
Детальное описание.

## 6.24 status.h File Reference

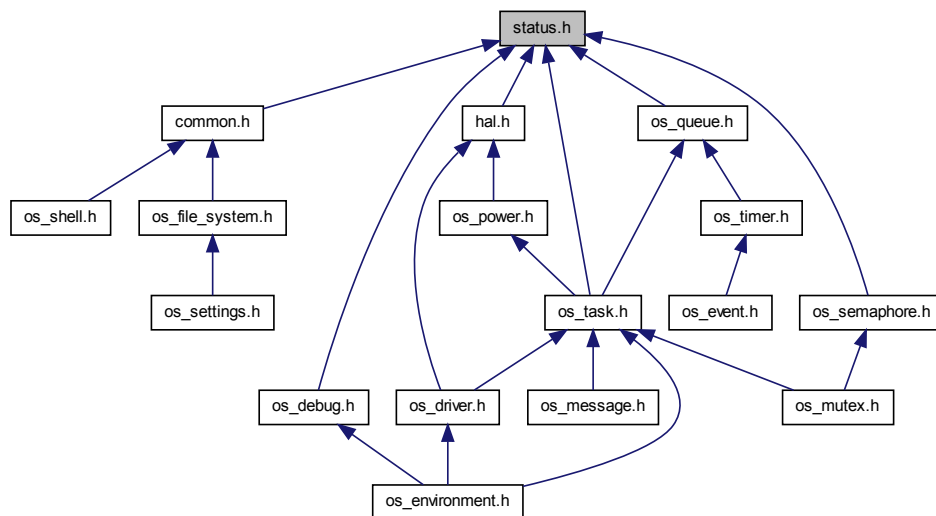
Status codes.

```
#include "typedefs.h"
```

Include dependency graph for status.h:



This graph shows which files directly or indirectly include this file:



## Defines

- `#define STATUS_MAX 256`
- `#define IS_STATUS(s) ((Status)(s) != (Status)S_OK)`  
Status checking macro.
- `#define IF_STATUS(s) if (IS_STATUS(s))`

Overloaded status checking macro.

- `#define IF_STATUS_OK(s) if (!IS_STATUS_(s))`
- `#define S_COMMON (-64)`
- `#define S_MODULE (-128)`

## Enumerations

- enum `StatusType` {  
`S_OK = S_COMMON, S_STOP, S_ABORT, S_RESUME,`  
`S_HARDWARE_FAULT, S_TIMEOUT, S_NO_MEMORY, S_OPEN,`  
`S_ISNT_OPENED, S_BUSY, S_UNSUPPORTED, S_INVALID_OPERATION,`  
`S_OVERFLOW, S_INVALID_VALUE, S_INVALID_TASK, S_INVALID_STATE,`  
`S_INVALID_STATE_FSM, S_INVALID_ARGS_NUMBER, S_INVALID_REF,`  
`S_UNDEF_PARAMETER,`  
`S_UNDEF_FUNCTION, S_UNDEF_QUEUE, S_UNDEF_EVENT, S_UNDEF_DEVICE,`  
`S_UNDEF_DRV, S_UNDEF_CMD, S_UNDEF_MSG, S_UNDEF_SIG,`  
`S_UNDEF_STATE, S_UNDEF_TIMER, S_UNDEF_REQ_ID, S_CRC_MISMATCH,`  
`S_SIZE_MISMATCH, S_INIT, S_ISNT_INITED, S_APP_MODULE,`  
`S_LAST }`  
 Status definitions.

### 6.24.1 Detailed Description

Status codes.

Author

A. Filyanov

Definition in file [status.h](#).

### 6.24.2 Define Documentation

#### 6.24.2.1 `#define IF_STATUS( s ) if (IS_STATUS_(s))`

Overloaded status checking macro.

Usage: `Status s = S_MDL; IF_STATUS(s) { D_LOG_S(D_WARNING, s); return s; }`

Definition at line 69 of file `status.h`.



# Index

CommandDeviceDescription, [93](#)

Crc32

[crc32.c](#), [110](#)

[crc32.h](#), [112](#)

[crc32.c](#), [109](#)

[Crc32](#), [110](#)

[Crc32Delta](#), [110](#)

[crc32.h](#), [111](#)

[Crc32](#), [112](#)

[Crc32Delta](#), [113](#)

[Crc32Delta](#)

[crc32.c](#), [110](#)

[crc32.h](#), [113](#)

[Crc8](#)

[crc8.c](#), [115](#)

[crc8.h](#), [117](#)

[crc8.c](#), [113](#)

[Crc8](#), [115](#)

[Crc8Delta](#), [115](#)

[crc\\_8\\_tbl](#), [116](#)

[crc8.h](#), [116](#)

[Crc8](#), [117](#)

[Crc8Delta](#), [118](#)

[Crc8Delta](#)

[crc8.c](#), [115](#)

[crc8.h](#), [118](#)

[crc\\_8\\_tbl](#)

[crc8.c](#), [116](#)

[DeviceDescUnion](#), [94](#)

[DeviceId](#), [94](#)

[DeviceRevision](#), [95](#)

[DeviceState](#), [95](#)

Environment variables user access functions., [79](#)

[hal.h](#), [118](#)

[HAL\\_DriverItf](#), [96](#)

[HAL\\_Env](#), [97](#)

[IF\\_STATUS](#)

[status.h](#), [166](#)

ISR specific functions., [78](#), [81–85](#), [87](#), [89](#), [90](#)

MPU specific functions., [88](#)

[OS\\_DateGet](#)

[OS\\_Time](#), [68](#)

[OS\\_DateIsValid](#)

[OS\\_Time](#), [68](#)

[OS\\_DateSet](#)

[OS\\_Time](#), [68](#)

[OS\\_DateStringParse](#)

[OS\\_Time](#), [69](#)

[OS\\_DateWeekDayGet](#)

[OS\\_Time](#), [69](#)

[OS\\_Debug](#), [7](#)

[OS\\_DebugDeInit](#), [8](#)

[OS\\_DebugInit](#), [9](#)

[OS\\_Log](#), [9](#)

[OS\\_LOG\\_S](#), [8](#)

[OS\\_Trace](#), [9](#)

[os\\_debug.h](#), [120](#)

[OS\\_DebugDeInit](#)

[OS\\_Debug](#), [8](#)

[OS\\_DebugInit](#)

[OS\\_Debug](#), [9](#)

[OS\\_Driver](#), [10](#)

[OS\\_DriverByNameGet](#), [12](#)

[OS\\_DriverClose](#), [12](#)

[OS\\_DriverConfigGet](#), [12](#)

[OS\\_DriverCreate](#), [12](#)

[OS\\_DriverDeInit](#), [13](#)

[OS\\_DriverDelete](#), [13](#)

[OS\\_DriverInit](#), [13](#)

[OS\\_DriverIoCtl](#), [13](#)

[OS\\_DriverNameGet](#), [14](#)

[OS\\_DriverNextGet](#), [14](#)

[OS\\_DriverOpen](#), [14](#)

[OS\\_DriverParentGet](#), [14](#)

[OS\\_DriverRead](#), [15](#)

- OS\_DriverStateNameGet, [15](#)
- OS\_DriverStateStateGet, [15](#)
- OS\_DriverStatsGet, [15](#)
- OS\_DriverWrite, [16](#)
- os\_driver.h, [122](#)
- OS\_DriverByNameGet
  - OS\_Driver, [12](#)
- OS\_DriverClose
  - OS\_Driver, [12](#)
- OS\_DriverConfig, [98](#)
- OS\_DriverConfigGet
  - OS\_Driver, [12](#)
- OS\_DriverCreate
  - OS\_Driver, [12](#)
- OS\_DriverDeInit
  - OS\_Driver, [13](#)
- OS\_DriverDelete
  - OS\_Driver, [13](#)
- OS\_DriverInit
  - OS\_Driver, [13](#)
- OS\_DriverIoCtl
  - OS\_Driver, [13](#)
- OS\_DriverNameGet
  - OS\_Driver, [14](#)
- OS\_DriverNextGet
  - OS\_Driver, [14](#)
- OS\_DriverOpen
  - OS\_Driver, [14](#)
- OS\_DriverParentGet
  - OS\_Driver, [14](#)
- OS\_DriverRead
  - OS\_Driver, [15](#)
- OS\_DriverStateNameGet
  - OS\_Driver, [15](#)
- OS\_DriverStateStateGet
  - OS\_Driver, [15](#)
- OS\_DriverStats, [98](#)
- OS\_DriverStatsGet
  - OS\_Driver, [15](#)
- OS\_DriverStdIoGet
  - OS\_Environment, [17](#)
- OS\_DriverWrite
  - OS\_Driver, [16](#)
- OS\_Environment, [16](#)
  - OS\_DriverStdIoGet, [17](#)
  - OS\_LocaleGet, [17](#)
  - OS\_LocaleSet, [17](#)
  - OS\_LogLevelGet, [18](#)
  - OS\_LogLevelSet, [18](#)
  - OS\_PowerSet, [18](#)
  - OS\_StdIoGet, [18](#)
  - OS\_StdIoSet, [19](#)
- os\_environment.h, [125](#)
- OS\_EnvironmentUser
  - OS\_EnvVariableDelete, [80](#)
  - OS\_EnvVariableGet, [80](#)
  - OS\_EnvVariableNextGet, [80](#)
  - OS\_EnvVariableOwnerGet, [80](#)
  - OS\_EnvVariableSet, [81](#)
- OS\_EnvVariableDelete
  - OS\_EnvironmentUser, [80](#)
- OS\_EnvVariableGet
  - OS\_EnvironmentUser, [80](#)
- OS\_EnvVariableNextGet
  - OS\_EnvironmentUser, [80](#)
- OS\_EnvVariableOwnerGet
  - OS\_EnvironmentUser, [80](#)
- OS\_EnvVariableSet
  - OS\_EnvironmentUser, [81](#)
- OS\_Event, [19](#)
  - OS\_EventCreate, [21](#)
  - OS\_EventDelete, [21](#)
  - OS\_EventItemCreate, [21](#)
  - OS\_EventItemDelete, [22](#)
  - OS\_EventItemLock, [22](#)
  - OS\_EventItemOwnerAdd, [22](#)
  - OS\_EventItemUnlock, [22](#)
  - OS\_EventNextGet, [23](#)
  - OS\_EventPeriodGet, [23](#)
  - OS\_EventStateGet, [23](#)
  - OS\_EventTimerGet, [23](#)
- os\_event.h, [127](#)
- OS\_EventConfig, [99](#)
- OS\_EventCreate
  - OS\_Event, [21](#)
- OS\_EventDelete
  - OS\_Event, [21](#)
- OS\_EventItemCreate
  - OS\_Event, [21](#)
- OS\_EventItemDelete
  - OS\_Event, [22](#)
- OS\_EventItemLock
  - OS\_Event, [22](#)
- OS\_EventItemOwnerAdd
  - OS\_Event, [22](#)
- OS\_EventItemUnlock
  - OS\_Event, [22](#)
- OS\_EventNextGet
  - OS\_Event, [23](#)
- OS\_EventPeriodGet

- OS\_Event, [23](#)
- OS\_EventStateGet
  - OS\_Event, [23](#)
- OS\_EventTimerGet
  - OS\_Event, [23](#)
- os\_file\_system.h, [130](#)
- OS\_Free
  - OS\_Memory, [30](#)
- OS\_FreeEx
  - OS\_Memory, [30](#)
- OS\_ISR\_Driver
  - OS\_ISR\_DriverIoCtl, [79](#)
- OS\_ISR\_DriverIoCtl
  - OS\_ISR\_Driver, [79](#)
- OS\_ISR\_Message
  - OS\_ISR\_MessageReceive, [82](#)
  - OS\_ISR\_MessageSend, [82](#)
- OS\_ISR\_MessageReceive
  - OS\_ISR\_Message, [82](#)
- OS\_ISR\_MessageSend
  - OS\_ISR\_Message, [82](#)
- OS\_ISR\_Mutex
  - OS\_ISR\_MutexCheck, [83](#)
  - OS\_ISR\_MutexLock, [84](#)
  - OS\_ISR\_MutexUnlock, [84](#)
- OS\_ISR\_MutexCheck
  - OS\_ISR\_Mutex, [83](#)
- OS\_ISR\_MutexLock
  - OS\_ISR\_Mutex, [84](#)
- OS\_ISR\_MutexUnlock
  - OS\_ISR\_Mutex, [84](#)
- OS\_ISR\_Power
  - OS\_ISR\_PowerStateSet, [85](#)
- OS\_ISR\_PowerStateSet
  - OS\_ISR\_Power, [85](#)
- OS\_ISR\_Queue
  - OS\_ISR\_QueueItemsCountGet, [85](#)
  - OS\_ISR\_QueueReceive, [86](#)
  - OS\_ISR\_QueueSend, [86](#)
- OS\_ISR\_QueueItemsCountGet
  - OS\_ISR\_Queue, [85](#)
- OS\_ISR\_QueueReceive
  - OS\_ISR\_Queue, [86](#)
- OS\_ISR\_QueueSend
  - OS\_ISR\_Queue, [86](#)
- OS\_ISR\_Semaphore
  - OS\_ISR\_SemaphoreCheck, [87](#)
  - OS\_ISR\_SemaphoreLock, [87](#)
  - OS\_ISR\_SemaphoreUnlock, [88](#)
- OS\_ISR\_SemaphoreCheck
  - OS\_ISR\_Semaphore, [87](#)
- OS\_ISR\_SemaphoreLock
  - OS\_ISR\_Semaphore, [87](#)
- OS\_ISR\_SemaphoreUnlock
  - OS\_ISR\_Semaphore, [88](#)
- OS\_ISR\_TickCountGet
  - OS\_ISR\_Time, [89](#)
- OS\_ISR\_Time
  - OS\_ISR\_TickCountGet, [89](#)
- OS\_ISR\_Timer
  - OS\_ISR\_TimerPeriodChange, [90](#)
  - OS\_ISR\_TimerReset, [90](#)
  - OS\_ISR\_TimerStart, [91](#)
  - OS\_ISR\_TimerStop, [91](#)
- OS\_ISR\_TimerPeriodChange
  - OS\_ISR\_Timer, [90](#)
- OS\_ISR\_TimerReset
  - OS\_ISR\_Timer, [90](#)
- OS\_ISR\_TimerStart
  - OS\_ISR\_Timer, [91](#)
- OS\_ISR\_TimerStop
  - OS\_ISR\_Timer, [91](#)
- OS\_List, [24](#)
  - OS\_ListAppend, [26](#)
  - OS\_ListInit, [26](#)
  - OS\_ListInsert, [26](#)
  - OS\_ListItemByOwnerFind, [26](#)
  - OS\_ListItemByValueFind, [27](#)
  - OS\_ListItemCreate, [27](#)
  - OS\_ListItemDelete, [27](#)
  - OS\_ListItemInit, [27](#)
  - OS\_ListItemsSwap, [28](#)
  - OS\_ListRemove, [28](#)
- os\_list.h, [131](#)
- OS\_ListAppend
  - OS\_List, [26](#)
- OS\_ListInit
  - OS\_List, [26](#)
- OS\_ListInsert
  - OS\_List, [26](#)
- OS\_ListItemByOwnerFind
  - OS\_List, [26](#)
- OS\_ListItemByValueFind
  - OS\_List, [27](#)
- OS\_ListItemCreate
  - OS\_List, [27](#)
- OS\_ListItemDelete
  - OS\_List, [27](#)
- OS\_ListItemInit
  - OS\_List, [27](#)

- 
- OS\_ListItemsSwap
    - OS\_List, [28](#)
  - OS\_ListRemove
    - OS\_List, [28](#)
  - OS\_LocaleGet
    - OS\_Environment, [17](#)
  - OS\_LocaleSet
    - OS\_Environment, [17](#)
  - OS\_Log
    - OS\_Debug, [9](#)
  - OS\_LOG\_S
    - OS\_Debug, [8](#)
  - OS\_LogLevelGet
    - OS\_Environment, [18](#)
  - OS\_LogLevelSet
    - OS\_Environment, [18](#)
  - OS\_Malloc
    - OS\_Memory, [30](#)
  - OS\_MallocEx
    - OS\_Memory, [30](#)
  - OS\_MemCacheFlush
    - OS\_Memory, [31](#)
  - OS\_MemCpy32
    - OS\_Memory, [31](#)
  - OS\_MemCpy8
    - OS\_Memory, [31](#)
  - OS\_Memory, [28](#)
    - OS\_Free, [30](#)
    - OS\_FreeEx, [30](#)
    - OS\_Malloc, [30](#)
    - OS\_MallocEx, [30](#)
    - OS\_MemCacheFlush, [31](#)
    - OS\_MemCpy32, [31](#)
    - OS\_MemCpy8, [31](#)
    - OS\_MemoryStatGet, [31](#)
    - OS\_MemoryTypeHeapNextGet, [32](#)
  - os\_memory.h, [134](#)
  - OS\_MemoryDesc, [99](#)
  - OS\_MemoryStat, [100](#)
  - OS\_MemoryStatGet
    - OS\_Memory, [31](#)
  - OS\_MemoryTypeHeapNextGet
    - OS\_Memory, [32](#)
  - OS\_Message, [32](#), [101](#)
    - OS\_MessageCreate, [33](#)
    - OS\_MessageDelete, [34](#)
    - OS\_MessageMulticastSend, [34](#)
    - OS\_MessageReceive, [34](#)
    - OS\_MessageSend, [35](#)
  - os\_message.h, [136](#)
  - OS\_MessageCreate
    - OS\_Message, [33](#)
  - OS\_MessageDelete
    - OS\_Message, [34](#)
  - OS\_MessageMulticastSend
    - OS\_Message, [34](#)
  - OS\_MessageReceive
    - OS\_Message, [34](#)
  - OS\_MessageSend
    - OS\_Message, [35](#)
  - OS\_Mutex, [35](#)
    - OS\_MutexCheck, [36](#)
    - OS\_MutexCreate, [36](#)
    - OS\_MutexDelete, [37](#)
    - OS\_MutexLock, [37](#)
    - OS\_MutexParentGet, [37](#)
    - OS\_MutexRecursiveCheck, [37](#)
    - OS\_MutexRecursiveCreate, [38](#)
    - OS\_MutexRecursiveLock, [38](#)
    - OS\_MutexRecursiveUnlock, [38](#)
    - OS\_MutexUnlock, [38](#)
  - os\_mutex.h, [137](#)
  - OS\_MutexCheck
    - OS\_Mutex, [36](#)
  - OS\_MutexCreate
    - OS\_Mutex, [36](#)
  - OS\_MutexDelete
    - OS\_Mutex, [37](#)
  - OS\_MutexLock
    - OS\_Mutex, [37](#)
  - OS\_MutexParentGet
    - OS\_Mutex, [37](#)
  - OS\_MutexRecursiveCheck
    - OS\_Mutex, [37](#)
  - OS\_MutexRecursiveCreate
    - OS\_Mutex, [38](#)
  - OS\_MutexRecursiveLock
    - OS\_Mutex, [38](#)
  - OS\_MutexRecursiveUnlock
    - OS\_Mutex, [38](#)
  - OS\_MutexUnlock
    - OS\_Mutex, [38](#)
  - OS\_Power, [39](#)
    - OS\_PowerInit, [40](#)
    - OS\_PowerStateGet, [40](#)
    - OS\_PowerStateNameGet, [40](#)
    - OS\_PowerStateSet, [40](#)
  - os\_power.h, [139](#)
  - OS\_PowerInit
    - OS\_Power, [40](#)
-

- OS\_PowerSet
  - OS\_Environment, [18](#)
- OS\_PowerStateGet
  - OS\_Power, [40](#)
- OS\_PowerStateNameGet
  - OS\_Power, [40](#)
- OS\_PowerStateSet
  - OS\_Power, [40](#)
- OS\_Queue, [41](#)
  - OS\_QueueConfigGet, [42](#)
  - OS\_QueueCreate, [43](#)
  - OS\_QueueDelete, [43](#)
  - OS\_QueueFlush, [43](#)
  - OS\_QueueItemsCountGet, [43](#)
  - OS\_QueueNextGet, [44](#)
  - OS\_QueueParentGet, [44](#)
  - OS\_QueueReceive, [44](#)
  - OS\_QueuesCountGet, [44](#)
  - OS\_QueueSend, [45](#)
  - OS\_QueueStatsGet, [45](#)
  - OS\_QueueSvcStdInGet, [45](#)
- os\_queue.h, [141](#)
- OS\_QueueConfig, [101](#)
- OS\_QueueConfigGet
  - OS\_Queue, [42](#)
- OS\_QueueCreate
  - OS\_Queue, [43](#)
- OS\_QueueDelete
  - OS\_Queue, [43](#)
- OS\_QueueFlush
  - OS\_Queue, [43](#)
- OS\_QueueItemsCountGet
  - OS\_Queue, [43](#)
- OS\_QueueNextGet
  - OS\_Queue, [44](#)
- OS\_QueueParentGet
  - OS\_Queue, [44](#)
- OS\_QueueReceive
  - OS\_Queue, [44](#)
- OS\_QueuesCountGet
  - OS\_Queue, [44](#)
- OS\_QueueSend
  - OS\_Queue, [45](#)
- OS\_QueueStats, [102](#)
- OS\_QueueStatsGet
  - OS\_Queue, [45](#)
- OS\_QueueSvcStdInGet
  - OS\_Queue, [45](#)
- OS\_Semaphore, [46](#)
  - OS\_SemaphoreBinaryCreate, [47](#)
  - OS\_SemaphoreCheck, [47](#)
  - OS\_SemaphoreCountingCreate, [47](#)
  - OS\_SemaphoreDelete, [47](#)
  - OS\_SemaphoreLock, [48](#)
  - OS\_SemaphoreUnlock, [48](#)
- os\_semaphore.h, [144](#)
- OS\_SemaphoreBinaryCreate
  - OS\_Semaphore, [47](#)
- OS\_SemaphoreCheck
  - OS\_Semaphore, [47](#)
- OS\_SemaphoreCountingCreate
  - OS\_Semaphore, [47](#)
- OS\_SemaphoreDelete
  - OS\_Semaphore, [47](#)
- OS\_SemaphoreLock
  - OS\_Semaphore, [48](#)
- OS\_SemaphoreUnlock
  - OS\_Semaphore, [48](#)
- OS\_Settings, [48](#)
  - OS\_SettingsDeInit, [49](#)
  - OS\_SettingsDelete, [49](#)
  - OS\_SettingsInit, [50](#)
  - OS\_SettingsItemsRead, [50](#)
  - OS\_SettingsItemsWrite, [50](#)
  - OS\_SettingsRead, [50](#)
  - OS\_SettingsWrite, [51](#)
- os\_settings.h, [146](#)
- OS\_SettingsDeInit
  - OS\_Settings, [49](#)
- OS\_SettingsDelete
  - OS\_Settings, [49](#)
- OS\_SettingsInit
  - OS\_Settings, [50](#)
- OS\_SettingsItem, [102](#)
- OS\_SettingsItemsRead
  - OS\_Settings, [50](#)
- OS\_SettingsItemsWrite
  - OS\_Settings, [50](#)
- OS\_SettingsRead
  - OS\_Settings, [50](#)
- OS\_SettingsWrite
  - OS\_Settings, [51](#)
- OS\_Shell, [51](#)
  - OS\_ShellArgumentsNumberCheck, [52](#)
  - OS\_ShellClHandler, [53](#)
  - OS\_ShellCls, [53](#)
  - OS\_ShellCommandByNameGet, [53](#)
  - OS\_ShellCommandCreate, [53](#)
  - OS\_ShellCommandDelete, [54](#)

- OS\_ShellCommandExecute, [54](#)
- OS\_ShellCommandNextGet, [54](#)
- OS\_ShellInit, [54](#)
- OS\_ShellPromptGet, [54](#)
- os\_shell.h, [148](#)
- OS\_ShellArgumentsNumberCheck
  - OS\_Shell, [52](#)
- OS\_ShellCllHandler
  - OS\_Shell, [53](#)
- OS\_ShellCls
  - OS\_Shell, [53](#)
- OS\_ShellCommandByNameGet
  - OS\_Shell, [53](#)
- OS\_ShellCommandConfig, [102](#)
- OS\_ShellCommandCreate
  - OS\_Shell, [53](#)
- OS\_ShellCommandDelete
  - OS\_Shell, [54](#)
- OS\_ShellCommandExecute
  - OS\_Shell, [54](#)
- OS\_ShellCommandNextGet
  - OS\_Shell, [54](#)
- OS\_ShellInit
  - OS\_Shell, [54](#)
- OS\_ShellPromptGet
  - OS\_Shell, [54](#)
- OS\_StdIoGet
  - OS\_Environment, [18](#)
- OS\_StdIoSet
  - OS\_Environment, [19](#)
- OS\_Task, [55](#)
  - OS\_TaskAttrsGet, [58](#)
  - OS\_TaskByNameGet, [58](#)
  - OS\_TaskConfigGet, [59](#)
  - OS\_TaskCreate, [59](#)
  - OS\_TaskDelay, [59](#)
  - OS\_TaskDelayUntil, [59](#)
  - OS\_TaskDelete, [60](#)
  - OS\_TaskHdByIdGet, [60](#)
  - OS\_TaskHdGet, [60](#)
  - OS\_TaskHdParentByIdGet, [61](#)
  - OS\_TaskHdParentGet, [61](#)
  - OS\_TaskIdGet, [61](#)
  - OS\_TaskInit, [61](#)
  - OS\_TaskMain, [61](#)
  - OS\_TaskNameGet, [62](#)
  - OS\_TaskNextGet, [62](#)
  - OS\_TaskPower, [62](#)
  - OS\_TaskPowerStateGet, [62](#)
  - OS\_TaskPriorityGet, [63](#)
- OS\_TaskPrioritySet, [63](#)
- OS\_TaskResume, [63](#)
- OS\_TasksCountGet, [64](#)
- OS\_TasksStatsGet, [64](#)
- OS\_TaskStateGet, [64](#)
- OS\_TaskStateNameGet, [64](#)
- OS\_TaskStdIoGet, [65](#)
- OS\_TaskStorageGet, [65](#)
- OS\_TaskSuspend, [65](#)
- OS\_TaskSvcStdInGet, [65](#)
- os\_task.h, [151](#)
- OS\_TaskAttrsGet
  - OS\_Task, [58](#)
- OS\_TaskByNameGet
  - OS\_Task, [58](#)
- OS\_TaskConfig, [103](#)
- OS\_TaskConfigGet
  - OS\_Task, [59](#)
- OS\_TaskCreate
  - OS\_Task, [59](#)
- OS\_TaskDelay
  - OS\_Task, [59](#)
- OS\_TaskDelayUntil
  - OS\_Task, [59](#)
- OS\_TaskDelete
  - OS\_Task, [60](#)
- OS\_TaskHdByIdGet
  - OS\_Task, [60](#)
- OS\_TaskHdGet
  - OS\_Task, [60](#)
- OS\_TaskHdParentByIdGet
  - OS\_Task, [61](#)
- OS\_TaskHdParentGet
  - OS\_Task, [61](#)
- OS\_TaskIdGet
  - OS\_Task, [61](#)
- OS\_TaskInit
  - OS\_Task, [61](#)
- OS\_TaskMain
  - OS\_Task, [61](#)
- OS\_TaskNameGet
  - OS\_Task, [62](#)
- OS\_TaskNextGet
  - OS\_Task, [62](#)
- OS\_TaskPower
  - OS\_Task, [62](#)
- OS\_TaskPowerStateGet
  - OS\_Task, [62](#)
- OS\_TaskPriorityGet
  - OS\_Task, [63](#)

- 
- OS\_TaskPrioritySet
    - OS\_Task, [63](#)
  - OS\_TaskResume
    - OS\_Task, [63](#)
  - OS\_TasksCountGet
    - OS\_Task, [64](#)
  - OS\_TasksStatsGet
    - OS\_Task, [64](#)
  - OS\_TaskStateGet
    - OS\_Task, [64](#)
  - OS\_TaskStateNameGet
    - OS\_Task, [64](#)
  - OS\_TaskStdIoGet
    - OS\_Task, [65](#)
  - OS\_TaskStorageGet
    - OS\_Task, [65](#)
  - OS\_TaskSuspend
    - OS\_Task, [65](#)
  - OS\_TaskSvcStdInGet
    - OS\_Task, [65](#)
  - OS\_TickCountGet
    - OS\_Time, [69](#)
  - OS\_Time, [66](#)
    - OS\_DateGet, [68](#)
    - OS\_DateIsValid, [68](#)
    - OS\_DateSet, [68](#)
    - OS\_DateStringParse, [69](#)
    - OS\_DateWeekDayGet, [69](#)
    - OS\_TickCountGet, [69](#)
    - OS\_TimeDayLightSavingsGet, [70](#)
    - OS\_TimeDayLightSavingsSet, [70](#)
    - OS\_TimeGet, [70](#)
    - OS\_TimeIsValid, [70](#)
    - OS\_TimeNameDayOfWeekGet, [71](#)
    - OS\_TimeSet, [71](#)
    - OS\_TimeStringParse, [71](#)
  - os\_time.h, [155](#)
  - OS\_TimeDayLightSavingsGet
    - OS\_Time, [70](#)
  - OS\_TimeDayLightSavingsSet
    - OS\_Time, [70](#)
  - OS\_TimeGet
    - OS\_Time, [70](#)
  - OS\_TimeIsValid
    - OS\_Time, [70](#)
  - OS\_TimeNameDayOfWeekGet
    - OS\_Time, [71](#)
  - OS\_Timer, [72](#)
    - OS\_TimerByIdGet, [74](#)
    - OS\_TimerByNameGet, [74](#)
  - OS\_TimerCreate, [74](#)
  - OS\_TimerDelete, [74](#)
  - OS\_TimerIdGet, [75](#)
  - OS\_TimerIsActive, [75](#)
  - OS\_TimerNameGet, [75](#)
  - OS\_TimerNextGet, [76](#)
  - OS\_TimerPeriodGet, [76](#)
  - OS\_TimerPeriodSet, [76](#)
  - OS\_TimerReset, [77](#)
  - OS\_TimerStart, [77](#)
  - OS\_TimerStatsGet, [77](#)
  - OS\_TimerStop, [77](#)
  - os\_timer.h, [158](#)
  - OS\_TimerByIdGet
    - OS\_Timer, [74](#)
  - OS\_TimerByNameGet
    - OS\_Timer, [74](#)
  - OS\_TimerConfig, [103](#)
  - OS\_TimerCreate
    - OS\_Timer, [74](#)
  - OS\_TimerDelete
    - OS\_Timer, [74](#)
  - OS\_TimerIdGet
    - OS\_Timer, [75](#)
  - OS\_TimerIsActive
    - OS\_Timer, [75](#)
  - OS\_TimerNameGet
    - OS\_Timer, [75](#)
  - OS\_TimerNextGet
    - OS\_Timer, [76](#)
  - OS\_TimerPeriodGet
    - OS\_Timer, [76](#)
  - OS\_TimerPeriodSet
    - OS\_Timer, [76](#)
  - OS\_TimerReset
    - OS\_Timer, [77](#)
  - OS\_TimerStart
    - OS\_Timer, [77](#)
  - OS\_TimerStatsGet
    - OS\_Timer, [77](#)
  - OS\_TimerStop
    - OS\_Timer, [77](#)
  - OS\_TimeSet
    - OS\_Time, [71](#)
  - OS\_TimeStringParse
    - OS\_Time, [71](#)
  - OS\_Trace
    - OS\_Debug, [9](#)
  - PACKED
-

- protocol.h, [164](#)
- Packet, [104](#)
- PROTO\_ID\_VER
  - protocol.h, [163](#)
- protocol.h, [162](#)
  - PACKED, [164](#)
  - PROTO\_ID\_VER, [163](#)
  - ProtocolCommand, [163](#)
- ProtocolCommand
  - protocol.h, [163](#)
- ProtocolHeaderInfo, [104](#)
- ProtocolId, [105](#)
  
- RouteItem, [106](#)
- RouteListItem, [107](#)
  
- status.h, [164](#)
  - IF\_STATUS, [166](#)