diOS API Reference Manual.

0.1

Generated by Doxygen 1.7.2

Thu Jul 3 2014 02:17:39

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

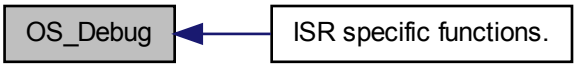
Here is a list of all documented files with brief descriptions:

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

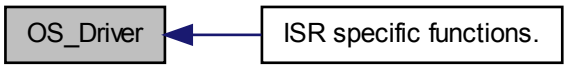
None.

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 indentifier.
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

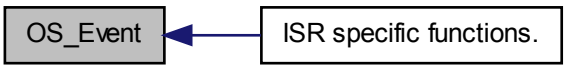
in	drv_name_p	Driver name.

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_1_p, OS_ListItem *item_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_l_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_l_p)

Remove item from the list.

Parameters

in	item_l_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

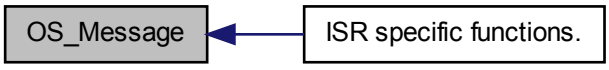
in	mem_type	Memory type.

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 trailling 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

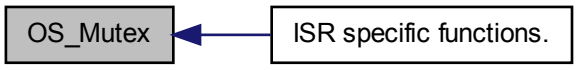
in	qhd	Receiver (task) queue handle.
in	msg_p	Message.
in	timeout	Message sending timeout.
in	priority	Message sending priority.

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

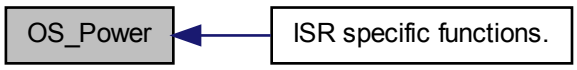
in	mhd	Mutex handle.

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

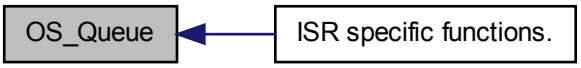
in	state	Power state.

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)

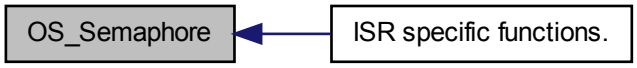
Get system service task standart input/output queue.

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_ShellClHandler (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)

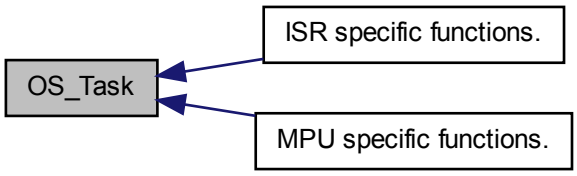
Get shell command prompt.

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)

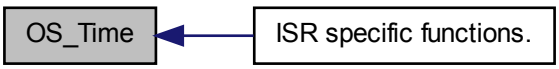
Get the system supervisor task standart input queue.

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.

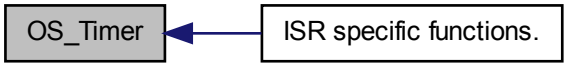
Note

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

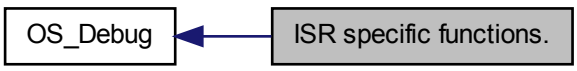
in	timer_hd	Timer handle.
in	timeout	Operation timeout.

Returns

Status.

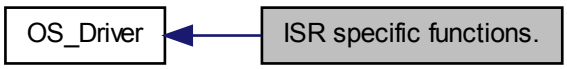
4.17 ISR specific functions.

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	dhd	Driver's handle.
in	request_id	Driver's request code indentifier.
in	args_p	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

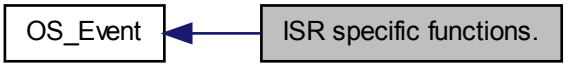
in	variable_name_p	Variable name.
in	variable_value_p	Variable value.

Returns

Status.

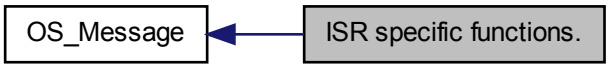
4.20 ISR specific functions.

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

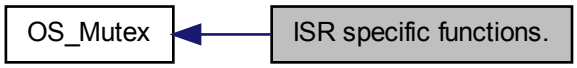
in	qhd	Receiver (task) queue handle.
in	msg_p	Message.
in	timeout	Message sending timeout.
in	priority	Message sending priority.

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

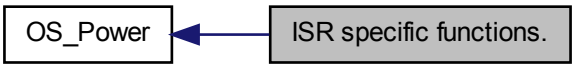
in	mhd	Mutex handle.

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

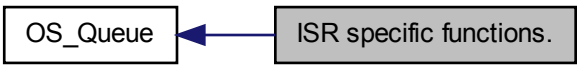
in	state	Power state.

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

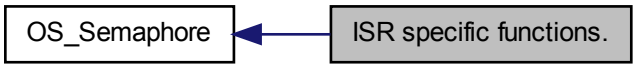
in	qhd	Receiver queue handle.
in	item_p	Item.
in	priority	Item sending priority.

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

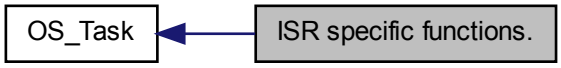
in	thd	Task handle.
in	qhd	Queue handle.
in	dir	I direction.

Returns

Queue handle.

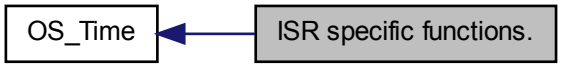
4.27 ISR specific functions.

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

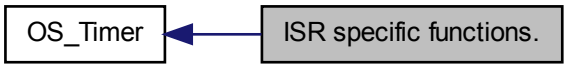
Tick count.

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

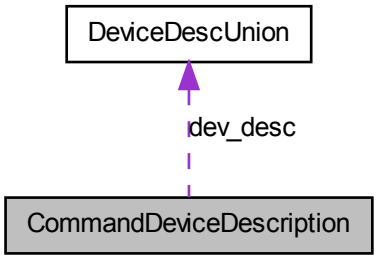
Data Structure Documentation

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 усройства.
- 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:

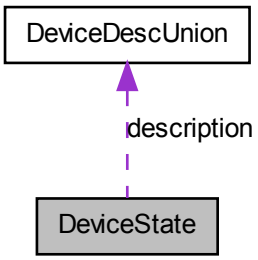
- typedefs_app.h

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

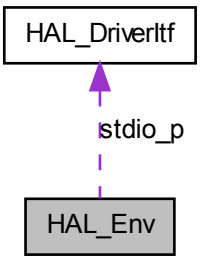
Definition at line 26 of file hal.h.

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

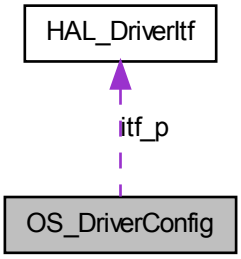
Definition at line 36 of file hal.h.

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

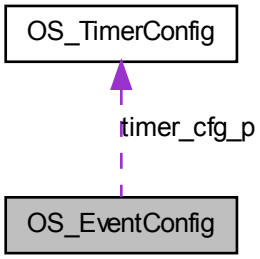
Definition at line 31 of file os_driver.h.

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:

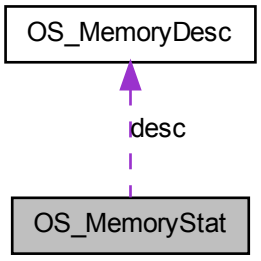
- os_memory.h

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

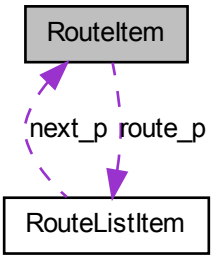
Definition at line 106 of file protocol.h.

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

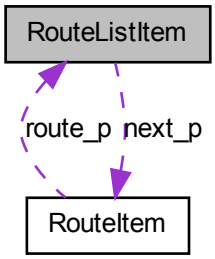
Definition at line 47 of file protocol.h.

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

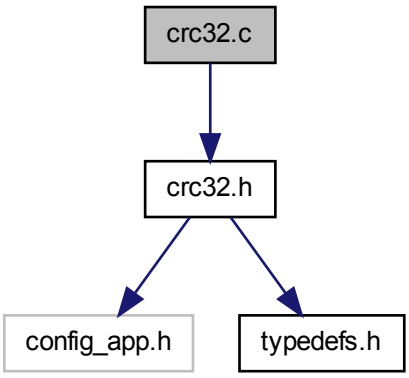
File Documentation

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 polynom.

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;
}
```

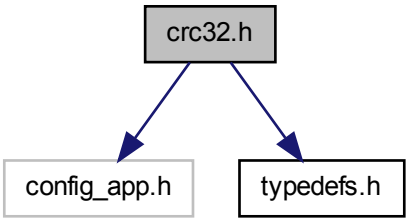
6.2 crc32.h File Reference

CRC32.

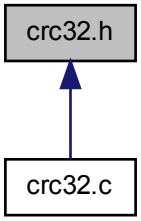
#include "config_app.h"

#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 polynom.

Returns

CRC32.

Definition at line 52 of file crc32.c.

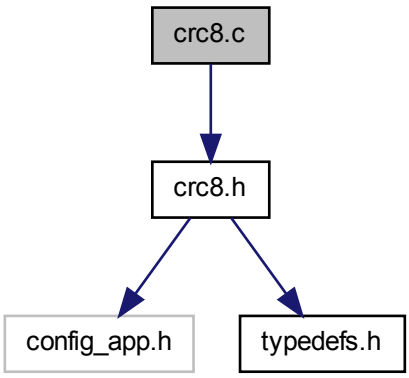
```
{
U32 crc_32 = init_poly;

   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 polynom.

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.

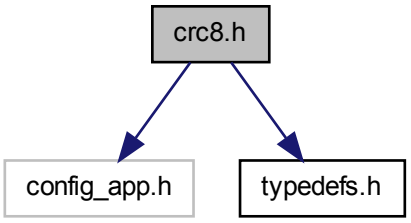
6.4 crc8.h File Reference

CRC8.

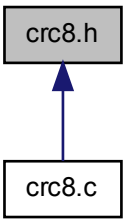
#include "config_app.h"

#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 polynom.

Returns

CRC8.

Definition at line 44 of file crc8.c.

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

6.5 hal.h File Reference

HAL.

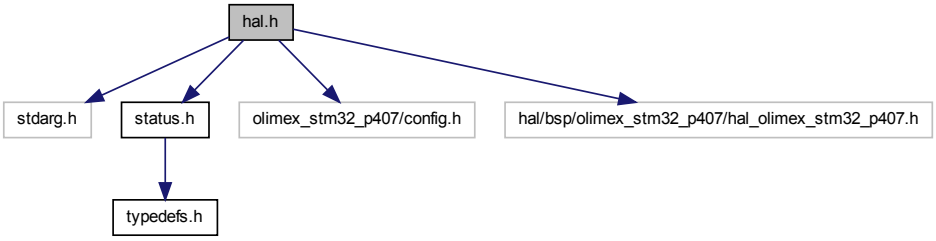
#include <stdarg.h>

#include "status.h"

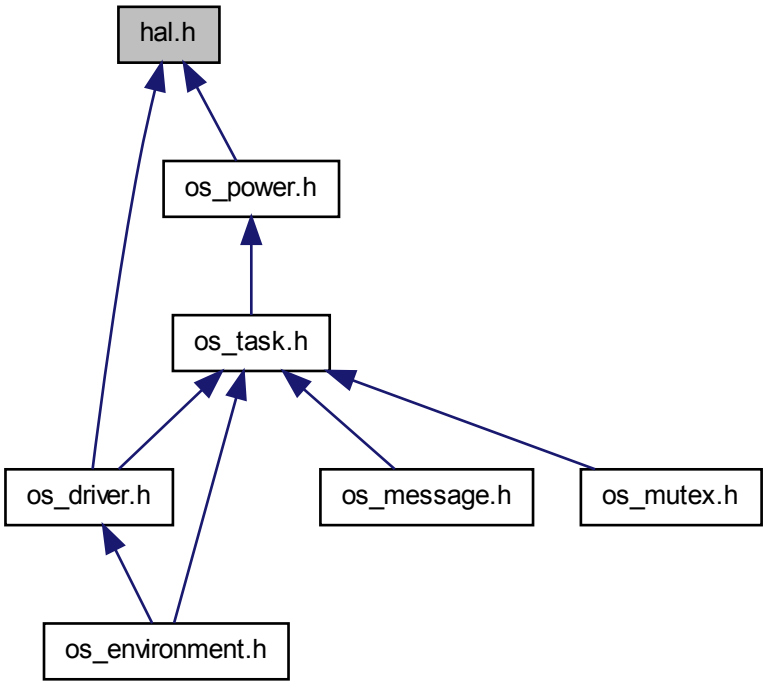
#include "olimex_stm32_p407/config.h"

#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

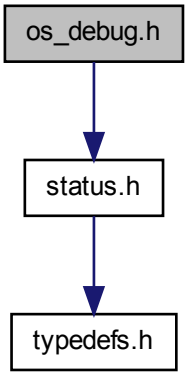
Definition in file hal.h.

6.6 os_debug.h File Reference

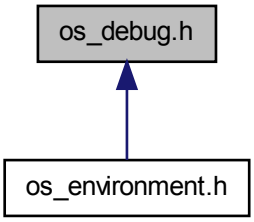
OS Debug.

#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.

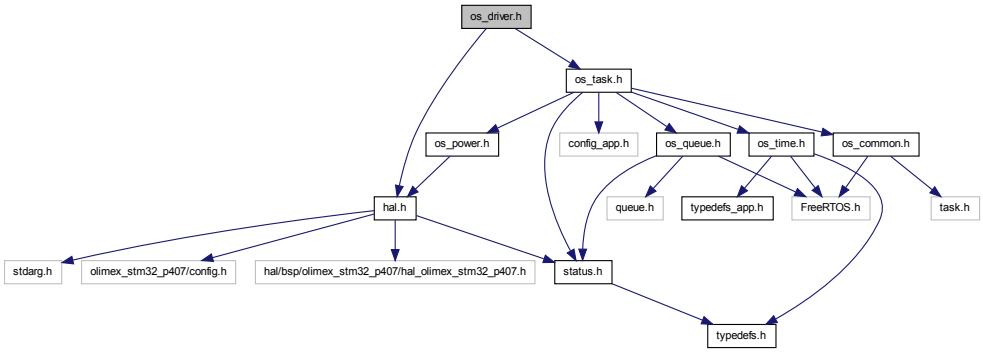
6.7 os_driver.h File Reference

OS Driver.

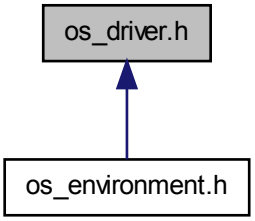
#include "hal.h"

#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.

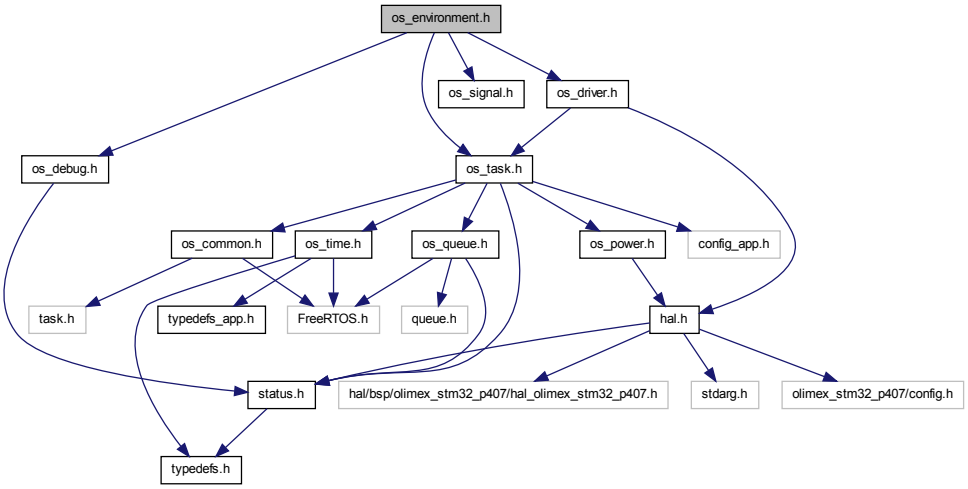
#include "os_debug.h"

#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.

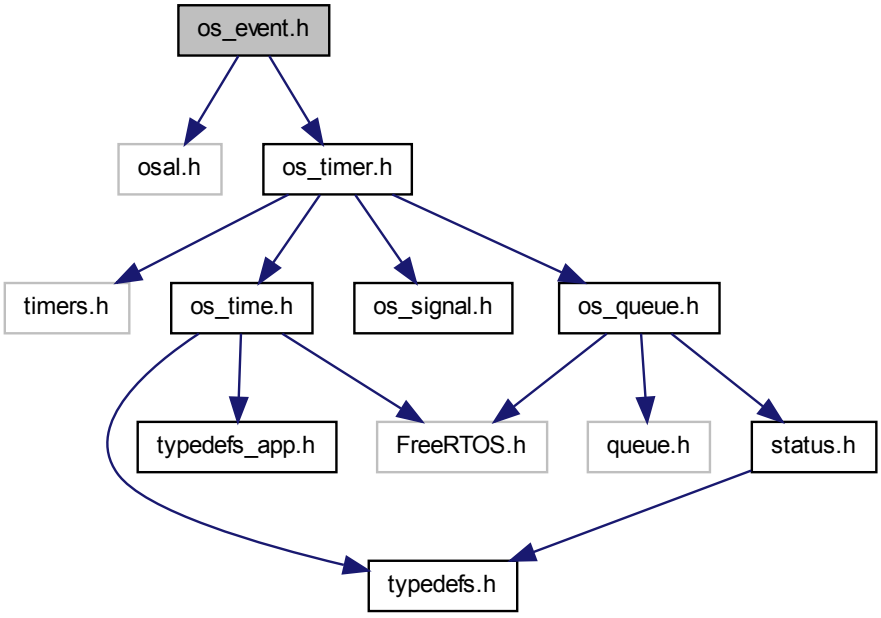
6.9 os_event.h File Reference

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

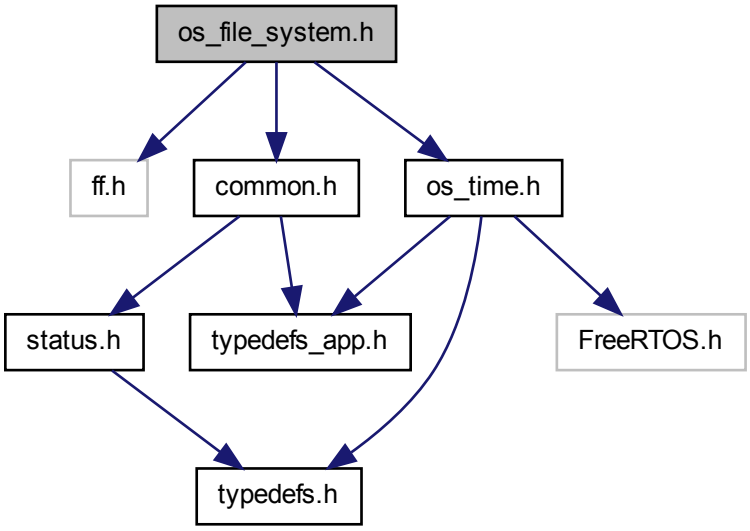
OS File system.

#include "ff.h"

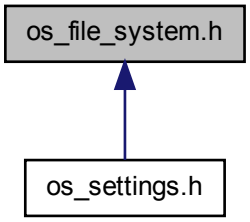
#include "common.h"

#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.

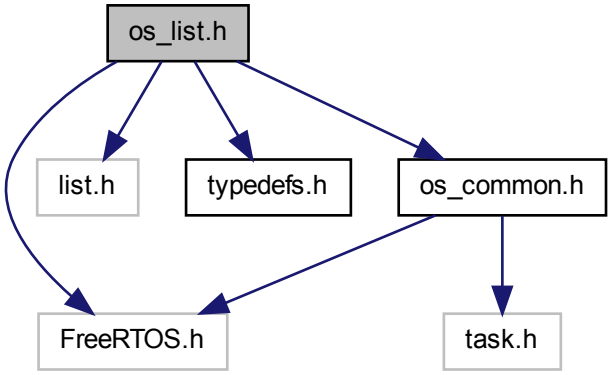
#include "FreeRTOS.h"

#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_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_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.

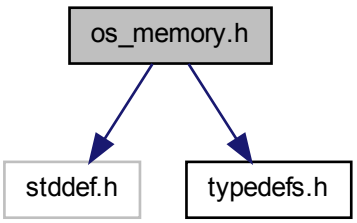
6.12 os_memory.h File Reference

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

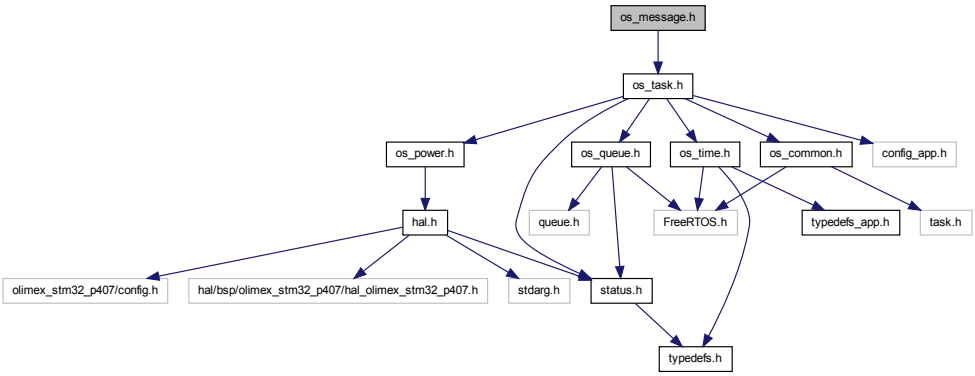
Definition in file os_memory.h.

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.

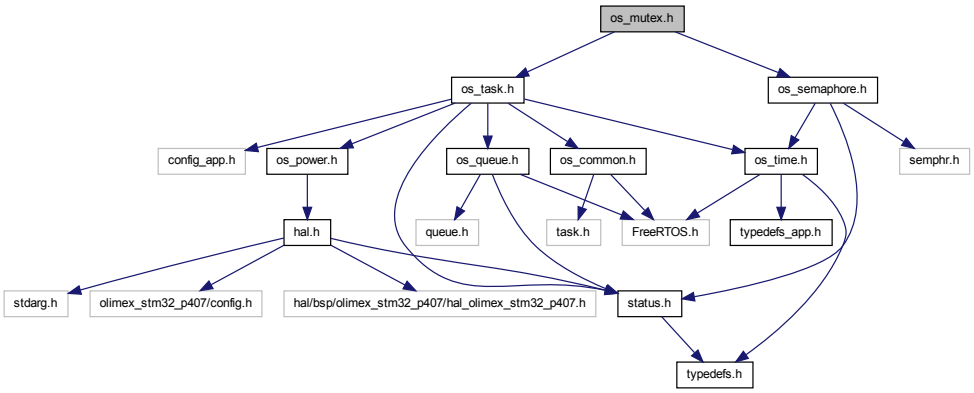
6.14 os_mutex.h File Reference

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

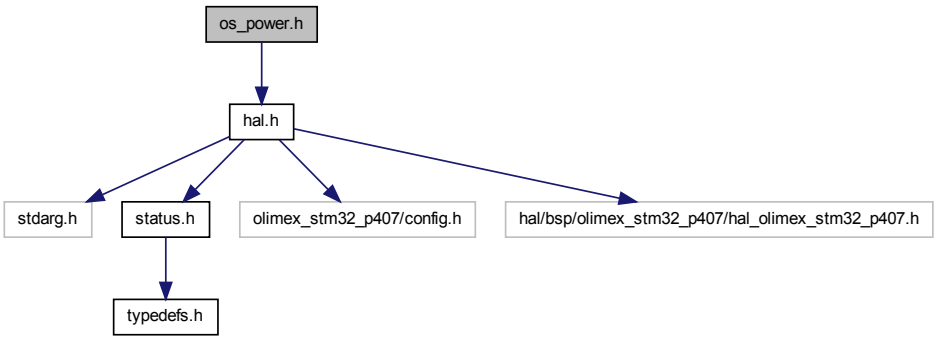
Definition in file os_mutex.h.

6.15 os_power.h File Reference

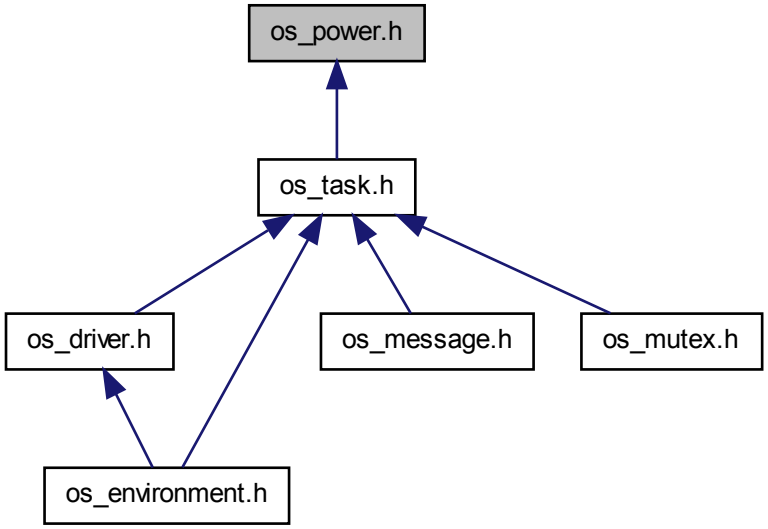
OS Power.

#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

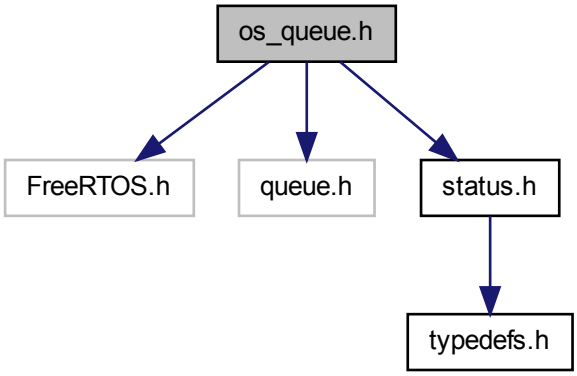
OS Queue.

#include "FreeRTOS.h"

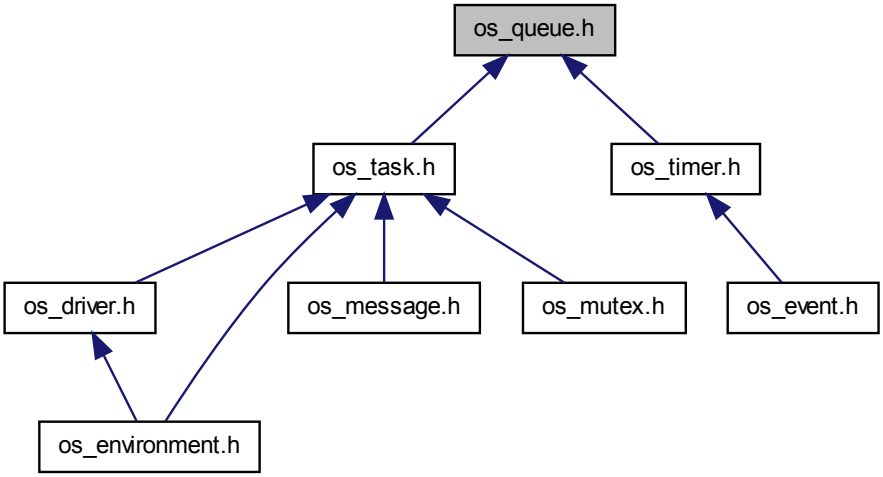
#include "queue.h"

#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_QueuesCountGet (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

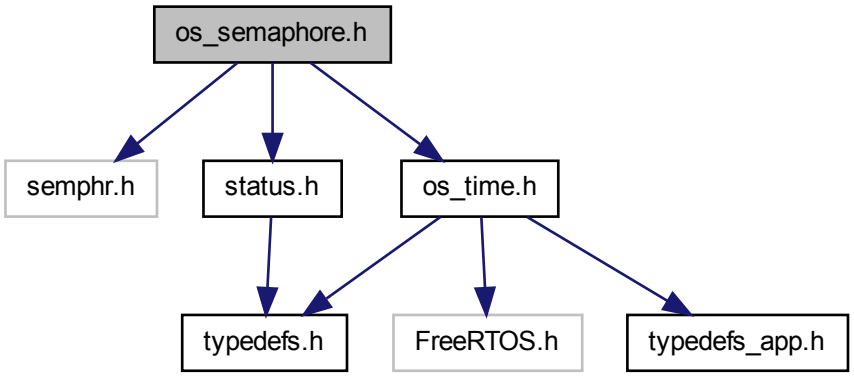
OS Semaphore.

#include "semphr.h"

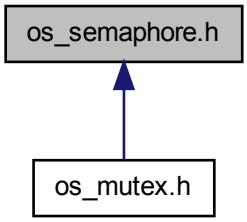
#include "status.h"

#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

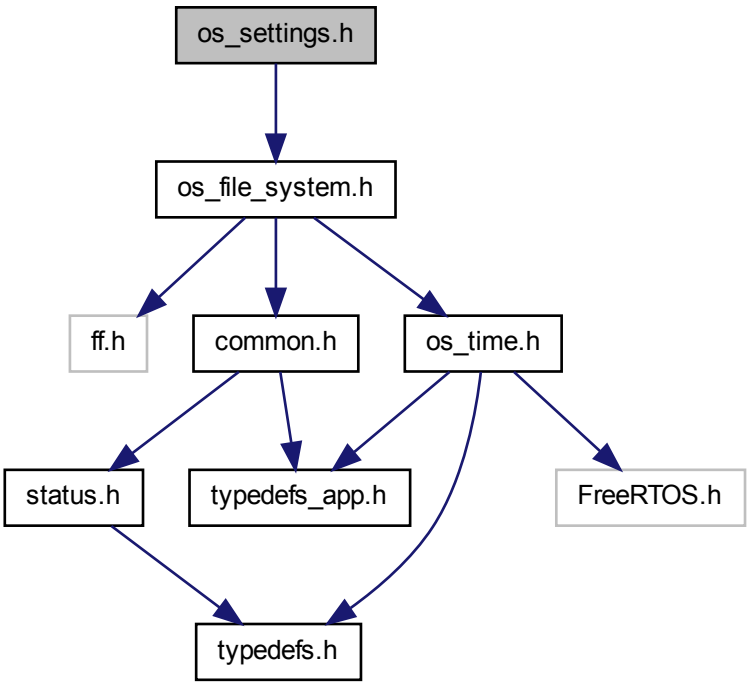
Definition in file os_semaphore.h.

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.

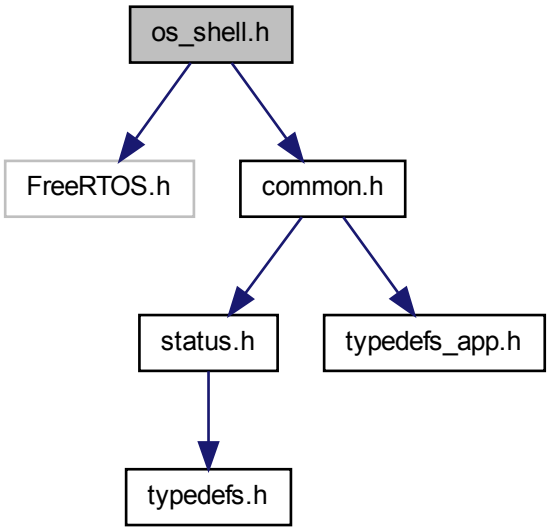
6.19 os_shell.h File Reference

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"

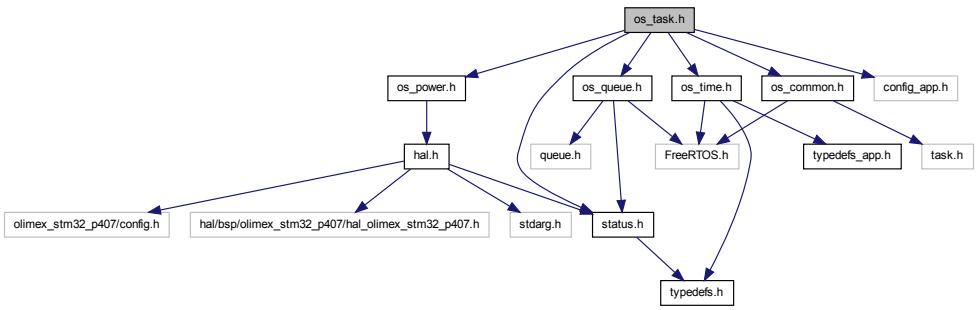
#include "os_common.h"

#include "os_power.h"

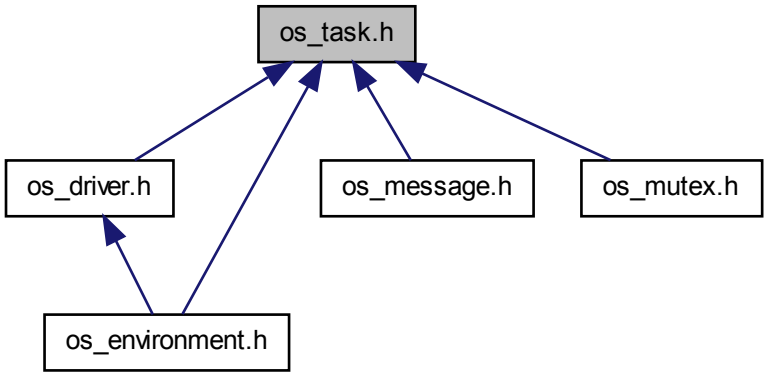
#include "os_time.h"

#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

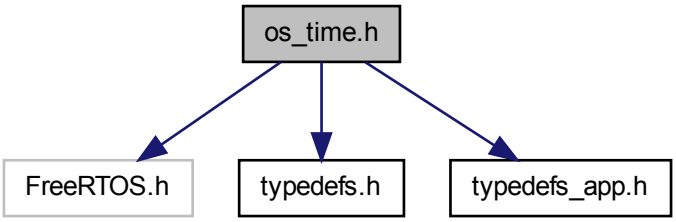
OS Time.

#include "FreeRTOS.h"

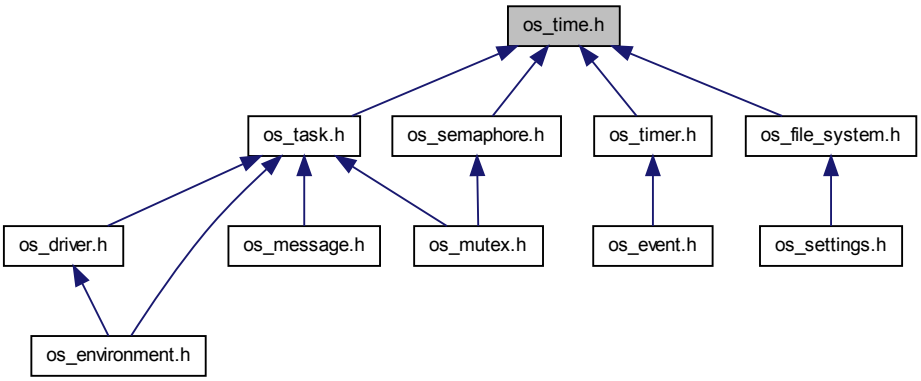
#include "typedefs.h"

#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.

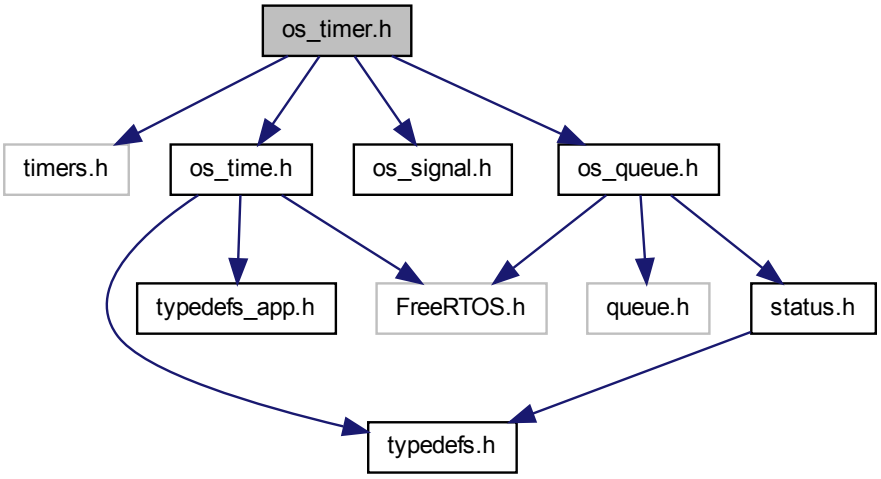
#include "timers.h"

#include "os_time.h"

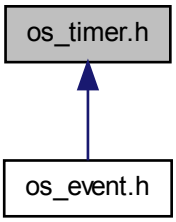
#include "os_signal.h"

#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 аттрибутов 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 аттрибутов 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.

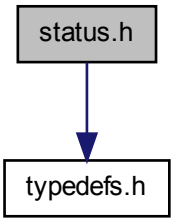
Детальное описание.

6.24 status.h File Reference

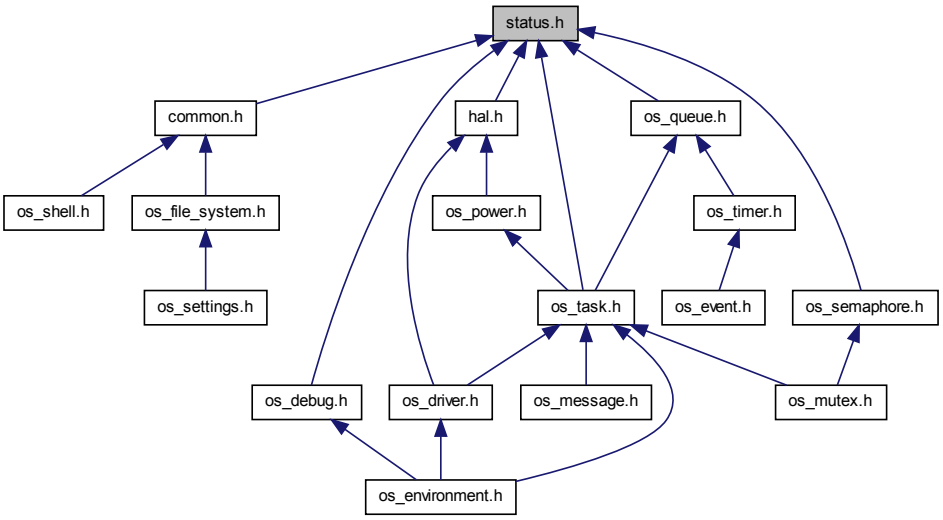
Status codes.

#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