MROS reference manual

**By Claus Andersen**

Current version: 0.01

[Version trace 3](#_Toc186043430)

[Manual versions 3](#_Toc186043431)

[Code versions 3](#_Toc186043432)

[Kernel API 4](#_Toc186043433)

[General 4](#_Toc186043434)

[Scheduler 4](#_Toc186043435)

[Task management 4](#_Toc186043436)

[Semaphores 5](#_Toc186043437)

[Message queues 5](#_Toc186043438)

[Mailboxes 5](#_Toc186043439)

[Memory management 5](#_Toc186043440)

[Data structures 6](#_Toc186043441)

[Kernel debugger and the Kernel Event Monitor 7](#_Toc186043442)

[Protocol 7](#_Toc186043443)

[File systems 8](#_Toc186043444)

[FileStore 8](#_Toc186043445)

[FAT 8](#_Toc186043446)

# Version trace

## Manual versions

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Notes** |
| 0.01 | 21-08-2007 | Manual created. |
| 0.01 | 21-12-2007 | A few updates / clean ups. |

## Code versions

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Notes** |
| 0.01 | xx-xx-2007 | Kernel (TaskManagement, Timer, Semaphores and MessageQueues) implemented. |
| 0.02 | xx-xx-2007 | The Peripherals group added. Currently supported peripherals are UART and SPI. |
| 0.03 | xx-xx-2007 | FileStore added. Only internal EEPROM is currently natively supported but user supplied low level media access functions may be used to access any media. |
| 0.04 | 26-10-2007 | EventRouter added. |
| 0.05 | 30-11-2007 | Menu module added. |
| 0.06 | 21-12-2007 | Dynamic memory management implemented (the MemoryManagement module). |

# Kernel API

## General

### taskerDelay

Execution is blocked for the specified number of system ticks.

Since tasker Kernel operates in a cooperative manner, this blocks the entire system. Use taskerSleep to only block the current task.

### taskerTicksPerSecond

Returns the number of ticks specified at compile time.

## Scheduler

### taskerInitiScheduler

Initializes the kernel scheduler. This MUST be done before using any of the other functions in the OS.

### taskerRunScheduler

Kicks of the kernel scheduler. Starts executing the tasks already created.

If no tasks have been created prior to calling taskerRunScheduler the system halts.

This function never returns.

## Task management

### taskerCreateTask

Adds the task to the internal task queue and includes the task in the scheduling.

Only one instance of a task can be created. Any attempt to recreate a task will be ignored.

### taskerDeleteTask

Removes the task from the task queue.

Attempts to delete a non existing task will be ignored.

If a task owning a semaphore is deleted the semaphore is lost!

### taskerSuspendTask

Suspends the specified task indefinitely.

### taskerSuspend

Suspends the current task indefinitely.

### taskerResumeTask

Resumes a previously suspended task.

### taskerSleepTask

Puts a task to sleep for the specified number of system ticks.

Calling sleep on a task that is already sleeping sets the sleep time to the highest of the two sleep times.

### taskerSleep

Puts the current task to sleep for the specified number of system ticks.

### taskerWakeTask

Wakes up a task that was previously put to sleep.

## Semaphores

### taskerSemaphoreWait

If the semaphore is already taken the task is blocked until the semaphore is signalled. Otherwise it is taken.

### taskerSemaphoreSignal

Signals the semaphore and unblocks the tasks that are blocked waiting for the semaphore.

Since all waiting tasks are unblocked there is no way to know which task gets the semaphore!

## Message queues

### taskerPostMessage

Post a message on the message queue and unblocks tasks waiting for messages on that queue.

All tasks that are blocked waiting for a message on the queue are unblocked and there is no way of knowing which task gets to the queue first to unqueue the element.

### taskerGetMessage

If a message has been posted to the queue it is unqueued. Otherwise the task is blocked until a message is posted to the queue.

## Mailboxes

### taskerPostMail

Not implemented yet.

### taskerGetMail

Not implemented yet.

## Memory management

### taskerAllocate

Not implemented yet.

### taskerDeallocate

Not implemented yet.

## Data structures

# Kernel debugger and the Kernel Event Monitor

## Protocol

Debugging event messages are sent from the kernel debugger to the PC application when calls to API functions are made. Event messages contain a time stamp, the event ID and a number of event specific arguments.

A message is sent as a stream of bytes. Each byte is divided into two bytes each con…

RLE

The fields are:

* \_ Start of event message
* <t> Time stamp for the event. An unsigned 32 bit integer (unsigned long) transferred as 8 hexadecimal digits.
* <e> Event message identifier. A single upper case letter in the range 'A' - 'Z'.
* <a> If the event message contains any arguments they are sent after the event message identifier. Numbers are transferred as hexadecimal digits.
* : End of event message

The following messages have been assigned:

* A Task x created at time stamp y
* B Task x deleted at time stamp y
* C Task x suspended at time stamp y
* D Task x resumed at time stamp y
* E Task x put to sleep at time stamp y
* F Task x awaken at time stamp y
* G Total execution time for task x
* H Task x just finished execution after running for y
* I
* J
* K
* L
* M
* N
* O
* P
* Q

# File systems

## FileStore

MROS contains a compact read only file system called FileStore. It is fully media configurable in that the low level functions for reading bytes from the physical media must be provided on initialization. This allows for user provided low level functions. Only support for the internal EEPROM is built at this time.

### API

## FAT