# ROSA User API proposal

Lamija Hasanagić
Zhaneta Nene
Tin Vidović

November 2019

# Contents


**1 Kernel control** **3**

1.1 Brief API list . . . 3
1.2 General functions . . . 3
1.2.1 ROSA_selectMode . . . 3
1.2.2 ROSA_initialize . . . 3
1.3 Scheduler . . . 3
1.3.1 ROSA_startScheduler . . . 3
1.3.2 ROSA_endScheduler . . . 4
1.3.3 ROSA_suspendScheduler . . . 4
1.3.4 ROSA_resumeScheduler . . . 4
1.4 Tasks . . . 4
1.4.1 ROSA_tcbCreateTask . . . 4
1.4.2 ROSA_tcbInstallTask . . . 4
1.4.3 ROSA_tcbUninstallTask . . . 4
1.4.4 ROSA_tcbDeleteTask . . . 5
1.4.5 ROSA_tcbGetTaskName . . . 5
1.4.6 ROSA_tcbGetTaskTCB . . . 5
1.4.7 ROSA_tcbGetTaskPriority . . . 5
1.4.8 ROSA_tcbSetTaskPriority . . . 5
1.5 Clock . . . 5
1.5.1 ROSA_delay . . . 5
1.5.2 ROSA_delayUntil . . . 5
1.5.3 ROSA_getTickCount . . . 6
1.6 Semaphores . . . 6
1.6.1 ROSA_semaphoreCreateBinary . . . 6
1.6.2 ROSA_semaphoreDeleteBinary . . . 6
1.6.3 ROSA_semaphoreLock . . . 6
1.6.4 ROSA_semaphoreUnlock . . . 6

# 1 Kernel control

## 1.1 Brief API list

- void ROSA_selectMode(ROSAmode mode);
- void ROSA_initialize(void);
- void ROSA_startScheduler(void);
- void ROSA_endScheduler(void);
- void ROSA_suspendScheduler(void);
- void ROSA_resumeScheduler(void);
- void ROSA_tcbCreateTask(tcb * TCB, char *id, void *taskFunction, int *stack, int stackSize, ROSApriority taskPriority);
- void ROSA_tcbInstallTask(tcb * TCB);
- void ROSA_tcbUninstallTask(tcb * TCB);
- void ROSA_tcbDeleteTask(tcb * TCB);
- char * ROSA_tcbGetTaskName(tcb * TCB);
- tcb * ROSA_tcbGetTaskTCB(const char * taskName);
- void ROSA_tcbGetTaskPriority(tcb * TCB);
- void ROSA_tcbSetTaskPriority(tcb * TCB, ROSApriority newTaskPriority);
- void ROSA_delay(const ROSAtick ticksToDelay);
- void ROSA_delayUntil(ROSAtick * previousWakeTime, const ROSAtick timeToIncrement);
- volatile ROSA_getTickCount(void);
- ROSAsemaphoreHandle ROSA_semaphoreCreateBinary(void);
- void ROSA_semaphoreDeleteBinary(ROSAsemaphoreHandle semaphore);
- ROSAbool ROSA_semaphoreLock(ROSAsemaphoreHandle semaphore);
- ROSAbool ROSA_semaphoreUnlock(ROSAsemaphoreHandle semaphore);

## 1.2 General functions

### 1.2.1 ROSA_selectMode

Prototype: void ROSA_selectMode(ROSAmode mode)
Description: Select between the legacy and the expanded ROSA operating mode.
Parameters: ROSAmode mode - The desired operating mode
Return Value: Nothing.

### 1.2.2 ROSA_initialize

Prototype: void ROSA_initialize(void);
Description: Initialize the ROSA kernel in expanded operating mode.
Parameters: None.
Return Value: Nothing.

## 1.3 Scheduler

### 1.3.1 ROSA_startScheduler

Prototype: void ROSA_startScheduler(void)
Description: Start execution of the installed TCB's using the fixed-priority preemptive scheduler.
Parameters: None.
Return Value: Nothing.

### 1.3.2 ROSA_endScheduler

Prototype: void ROSA_endScheduler(void)
Description: Stops system ticks and deletes all created and installed tasks thereby ending any multitasking. The execution returns to the point when ROSA_startScheduler was called.
Parameters: None.
Return Value: Nothing.

### 1.3.3 ROSA_suspendScheduler

Prototype: void ROSA_suspendScheduler(void)
Description: Suspends the scheduler without disabling interrupts. Context switches will not occur until ROSA_resumeScheduler is called but the system ticks that occur are maintaned.
Parameters: None.
Return Value: Nothing.

### 1.3.4 ROSA_resumeScheduler

Prototype: ROSAbool ROSA_resumeScheduler(void)
Description: Resumes the scheduler after it had been suspended.
Parameters: None.
Return Value: ROSAtrue if resuming the scheduler caused a context switch, ROSAfalse otherwise.

## 1.4 Tasks

### 1.4.1 ROSA_tcbCreateTask

Prototype: ROSAbool ROSA_tcbCreate(tcb * TCB, char *id, void *taskFunction, int *stack, int stackSize, ROSApriority taskPriority)
Description: Create a new TCB entry according to the given parameters.
Parameters:

- tcb *TCB - A pointer to the TCB block to be created.
- char *id - A identification for the TCB block of length NAMESIZE (default NAMESIZE = 4)
- void *taskFunc - A pointer to the function which are to be executed by the task.
- int *stack - A pointer to the task stack area.
- int stackSize - The maximum allowed stack for this task.
- ROSApriority taskPriority - the priority of the task, awarded to it during creation (Higher number = Higher priority).

Return Value: ROSAtrue if task was created successfully, ROSAfalse otherwise.

### 1.4.2 ROSA_tcbInstallTask

Prototype: ROSAbool ROSA_tcbInstall(tcb * TCB)
Description: Install a new TCB entry into the TCBLIST of the ROSA kernel.
Parameters: tcb *TCB - A pointer to the TCB to install into the kernel.
Return Value: ROSAtrue if task was installed successfully, ROSAfalse otherwise.

### 1.4.3 ROSA_tcbUninstallTask

Prototype: void ROSA_tcbUninstall(tcb * TCB);
Description: Uninstall the TCB entry pointed to by the parameter from the TCBLIST of the ROSA kernel.
Parameters: tcb *TCB - A pointer to the TCB to be uninstalled.
Return Value: Nothing.

#### 1.4.4 ROSA_tcbDeleteTask

Prototype: void ROSA_tcbDelete(tcb * TCB);
Description: Delete the TCB entry specified by the parameter.
Parameters: tcb *TCB - A pointer to the TCB to be deleted.
Return Value: Nothing.

#### 1.4.5 ROSA_tcbGetTaskName

Prototype: char * ROSA_tcbGetTaskName(tcb * TCB)
Description: Returns the name of the task pointed to by the parameter.
Parameters: tcb *TCB - A pointer to the TCB whose name is to be looked up.
Return Value: A pointer to the task name, which is a standard NULL terminated C string.

#### 1.4.6 ROSA_tcbGetTaskTCB

Prototype: tcb * ROSA_tcbGetTaskTCB(const char * taskName)
Description: Returns a pointer to the TCB of the task whose name matches the provided parameter.
Parameters: const char * taskName - A standard NULL terminated C string representing the name of the task which is to be looked up.
Return Value: A pointer to the task TCB, or NULL if the task could not be found.

#### 1.4.7 ROSA_tcbGetTaskPriority

Prototype: ROSApriority ROSA_tcbGetTaskPriority(tcb * TCB)
Description: Returns the priority of the task pointed to by the parameter.
Parameters: tcb *TCB - A pointer to the TCB whose priority is to be looked up.
Return Value: The ROSApriority of the task being looked up , or NULL if the task could not be found.

#### 1.4.8 ROSA_tcbSetTaskPriority

Prototype: void ROSA_tcbSetTaskPriority(tcb * TCB, ROSApriority newTaskPriority)
Description: Sets the priority of the task pointed to by the parameter TCB to the parameter newTaskPriority.
Parameters:

- tcb *TCB - A pointer to the TCB whose priority is to be changed.
- ROSApriority newTaskPriority - The desired new priority of the task.

Return Value: Nothing.

### 1.5 Clock

#### 1.5.1 ROSA_delay

Prototype: void ROSA_delay(const ROSAtick ticksToDelay)
Description: Create a relative delay, i.e. suspend task for an amount of ticks specified by the parameter from the tick when the function was called.
Parameters: const ROSAtick ticksToDelay - The amount of ticks to suspend after function call.
Return Value: Nothing.

#### 1.5.2 ROSA_delayUntil

Prototype: void ROSA_delayUntil(ROSAtick * previousWakeTime, const ROSAtick timeToIncrement)
Description: Create an absolute delay, i.e. suspend task until a specific system clock tick independent of when the function was called.
Parameters:

- ROSAtick * previousWakeTime - Pointer to a ROSAtick variable holding the previous Wake Up Time of the task. This value is automatically corrected within the function.
- const ROSAtick timeToIncrement - The amount of ticks from the previous wake time the task will be suspended.

Return Value: Nothing.

### 1.5.3 ROSA_getTickCount

Prototype: volatile ROSA_getTickCount(void)
Description: Returns the amount of system clock ticks that have passed since the starting of the scheduler.
Parameters: None.
Return Value: The number of ticks that have passed since ROSA_startScheduler was called in a volatile variable to prevent wrong values being read due to compiler optimizations.

## 1.6 Semaphores

### 1.6.1 ROSA_semaphoreCreateBinary

Prototype: ROSAsemaphoreHandle ROSA_semaphoreCreateBinary(void)
Description: Create a binary semaphore and return a handle to it.
Parameters: None.
Return Value: The handle to the binary semaphore.

### 1.6.2 ROSA_semaphoreDeleteBinary

Prototype: void ROSA_semaphoreDeleteBinary(ROSAsemaphoreHandle semaphore)
Description: Deletes the binary semaphore whose handle is provided as the parameter.
Parameters: ROSAsemaphoreHandle semaphore - a handle to the binary semaphore to be deleted.
Return Value: Nothing.

### 1.6.3 ROSA_semaphoreLock

Prototype: ROSAbool ROSA_semaphoreLock(ROSAsemaphoreHandle semaphore)
Description: Attempts to Lock the binary semaphore whose handle is provided as the parameter.
Parameters: ROSAsemaphoreHandle semaphore - a handle to the binary semaphore to be locked.
Return Value: ROSAbool - ROSAtrue if the semaphore was locked successfully, ROSAfalse otherwise.

### 1.6.4 ROSA_semaphoreUnlock

Prototype: ROSAbool ROSA_semaphoreUnlock(ROSAsemaphoreHandle semaphore)
Description: Attempts to Unlock the binary semaphore whose handle is provided as the parameter.
Parameters: ROSAsemaphoreHandle semaphore - a handle to the binary semaphore to be unlocked.
Return Value: ROSAbool - ROSAtrue if the semaphore was unlocked successfully, ROSAfalse otherwise.