# RTA-OSEK

Binding Manual: TMS470/TIMULTI

# Contact Details


**ETAS Group**

www.etasgroup.com

**Germany**

ETAS GmbH
Borsigstraße 14
70469 Stuttgart

Tel.:+49 (711) 8 96 61-102
Fax:+49 (711) 8 96 61-106

www.etas.de

**USA**

ETAS Inc.
3021 Miller Road
Ann Arbor, MI 48103

Tel.: +1 (888) ETAS INC
Fax: +1 (734) 997-94 49

www.etasinc.com

**Japan**

ETAS K.K.
Queen's Tower C-17F,
2-3-5, Minatomirai, Nishi-ku,
Yokohama, Kanagawa
220-6217 Japan

Tel.: +81 (45) 222-0900
Fax: +81 (45) 222-0956

www.etas.co.jp

**France**

ETAS S.A.S.
1, place des États-Unis
SILIC 307
94588 Rungis Cedex

Tel.: +33 (1) 56 70 00 50
Fax: +33 (1) 56 70 00 51

www.etas.fr

**Korea**

ETAS Korea Co. Ltd.
4F, 705 Bldg. 70-5
Yangjae-dong, Seocho-gu
Seoul 137-889, Korea

Tel.: +82 (2) 57 47-016
Fax: +82 (2) 57 47-120

www.etas.co.kr

**Great Britain**

ETAS UK Ltd.
Studio 3, Waterside Court
Third Avenue, Centrum 100
Burton-upon-Trent
Staffordshire DE14 2WQ

Tel.: +44 (0) 1283 - 54 65 12
Fax: +44 (0) 1283 - 54 87 67

www.etas-uk.net

# Copyright Notice



Version: RM00068-004



## Disclaimer

The information in this document is subject to change without notice and does not represent a commitment on any part of LiveDevices. While the information contained herein is assumed to be accurate, LiveDevices assumes no responsibility for any errors or omissions.

In no event shall LiveDevices, its employees, its contractors or the authors of this document be liable for special, direct, indirect, or consequential damage, losses, costs, charges, claims, demands, claim for lost profits, fees or expenses of any nature or kind.

## Trademarks

RTA-OSEK, RTA-TRACE and LiveDevices are trademarks of LiveDevices Ltd.

Windows and MS-DOS are trademarks of Microsoft Corp.

OSEK/VDX is a trademark of Siemens AG.

All other product names are trademarks or registered trademarks of their respective owners.

# Contents

# 1 About this Guide

This guide provides target-specific information for the TMS470/TIMULTI port of LiveDevices' RTA-OSEK. It supplements the more general information in the *RTA-OSEK User Guide*.

A port is defined as a specific target microcontroller/target toolchain pairing. This guide tells you about integration issues with your target toolchain and issues that you need to be aware of when using RTA-OSEK on your target hardware. Port specific parameters of implementation are also provided, giving the RAM and ROM requirements for each object in the RTA-OSEK Component and execution times for each API call to the RTA-OSEK Component.

## 1.1 Who Should Read this Guide?

The reader should have an understanding of real time embedded programming in an OSEK context. You should read this guide if you want to know low-level technical information to integrate the RTA-OSEK Component into your application.

## 1.2 Conventions

**Important:** Notes that appear like this contain important information that you need to be aware of. Make sure that you read them carefully and that you follow any instructions that you are given.

**Portability:** Notes that appear like this describe things that you will need to know if you want to write code that will work on any processor running the RTA-OSEK Component.

Program code, file names, C types and symbols, and RTA-OSEK API call names all appear in the `courier` typeface. When the name of an object is made available to the programmer the name also appears in the `courier` typeface, so, for example, a task named Task1 appears as a task handle called `Task1`.

# 2 Toolchain Issues

This chapter contains important details about RTA-OSEK and your toolchain. A port of the RTA-OSEK Component is specific to both the target hardware and a specific version of the compiler toolchain. You must make sure that you build your application with the supported toolchain.

If you are interested in using a different version of the same toolchain, please contact LiveDevices to confirm whether or not this is possible.

## 2.1 Compiler

The RTA-OSEK Component was built using the following compiler:

| | |
|---|---|
| Vendor | Texas Instruments Inc. |
| Compiler | cl470.exe |
| Version | 4.4.10.100 |

The compulsory compiler options for application code are shown in the following table:

| Option | Description |
|---|---|
| `-c` | Compile only, do not link |
| `--abi=eabi` | use ARM ABI |

The prohibited compiler options for application code are shown in the following table:

| Option | Description |
|---|---|
| `--abi=tiabi` | use TI ABI |

The C file that RTA-OSEK generates from your OIL configuration file is called `osekdefs.c`. This file defines configuration parameters for the RTA-OSEK Component when running your application.

The compulsory compiler options for `osekdefs.c` are shown in the following table:

| Option | Description |
|---|---|
| `-c` | Compile only, do not link |
| `--abi=eabi` | use ARM ABI |
| `-mt/--code_state=16` | Generate 16-bit code |
| `--symdebug:none` | No debug output |

The prohibited compiler options for `osekdefs.c` are shown in the following table:

| Option | Description |
|---|---|
| `--abi=tiabi` | use TI ABI |

### 2.1.1 Using the 16-Bit/32-Bit Assembler Instruction Set

The TI compiler can generate object code using either the TMS570 16-bit or 32-bit instruction set. The RTA-OSEK run-time libraries have been compiled using the 16-bit instruction set to reduce the amount of code memory used. The libraries support applications using either the 16-bit or the 32-bit instruction set and have been fully tested with both options.

If C files containing tasks (i.e. functions decorated with the OSEK standard `TASK()` macro), Category 2 ISRs (i.e. functions decorated with the OSEK standard `ISR()` macro), function callbacks and hooks are compiled using the 32-bit instruction set the veneers required to change from the 16-bit instruction set are automatically generated by the TI compiler.

Category 1 interrupts require user generated veneers if compiled using the 32-bit instruction set, please refer to section 3.1.10 for further details.

Please consult the TI compiler documentation for a full explanation of 16/32-bit inter-working and how to employ it in your application.

## 2.2 Assembler

The RTA-OSEK Component was built using the following assembler:

| | |
|---|---|
| Vendor | Texas Instruments Inc. |
| Assembler | cl470.exe |
| Version | 4.4.10.100 |

The compulsory assembler options for application code are shown in the following table:

| Option | Description |
|---|---|
| `-c` | Compile only, do not link |
| `--abi=eabi` | use ARM ABI |

The prohibited assembler options for application code are shown in the following table:

| Option | Description |
|---|---|
| `--abi=tiabi` | use TI ABI |

The assembly file that RTA-OSEK generates from your OIL configuration file is called `osgen.s`. This file defines configuration parameters for the RTA-OSEK Component when running your application.

The compulsory assembler options for `osgen.s` are shown in the following table:

| Option | Description |
|---|---|
| `-c` | Compile only, do not link |
| `--abi=eabi` | use ARM ABI |

The prohibited assembler options for `osgen.s` are shown in the following table:

| Option | Description |
|---|---|
| `--abi=tiabi` | use TI ABI |

## 2.3 Linker/Locator

The compulsory linker/locator options for an RTA-OSEK application are shown in the following table:

| Option | Description |
|---|---|
| `--abi=eabi` | use ARM ABI |

The prohibited linker/locator options for an RTA-OSEK application are shown in the following table:

| Option | Description |
|---|---|
| `--abi=tiabi` | use TI ABI |

In addition to the sections used by application code, the following RTA-OSEK sections must be located:

| Sections | ROM/RAM | Description |
|---|---|---|
| `os_pid` | ROM | RTA-OSEK read-only data |
| `os_pird` | ROM | RTA-OSEK initialization data |
| `os_reset_addr` | ROM | Reset vector if generated by RTA-OSEK |
| `os_vectbl` | ROM | Vector table if generated by RTA-OSEK |
| `os_pir` | RAM | RTA-OSEK initialized data |
| `os_pur` | RAM | RTA-OSEK uninitialized data |

### 2.3.1 Linker Command File

The RAM sections used by RTA-OSEK comprise of the `os_pur` and `os_pir` sections. These sections are initialized by the RTA-OSEK Component within the `StartOS()` API call (unlike standard C RAM variable sections, which are initialized in the application start-up code). These sections should be marked `NOLOAD` in the linker command file. This prevents the contents, relocation information and line number information being placed in the output module.

Please refer to the linker command file used in the example application where this is demonstrated.

## 2.4 Debugger

ORTI is the OSEK Run-Time Interface that is supported by RTA-OSEK. Support is provided for the debuggers in the following table. Further information about ORTI for RTA-OSEK can be found in the *RTA-OSEK ORTI Guide*.

| | |
|---|---|
| ORTI compatible debuggers | Lauterbach Trace32 |

The RTA-OSEK GUI outputs a file with the extension `.ort`. This file should be loaded into the debugger with the command `Task.ORTI <file>`. Note that this must be loaded after the executable (`.out`) file. Please refer to the debugger documentation for further details on its support for ORTI.

# 3 Target Hardware Issues

## 3.1 Interrupts

This section explains the implementation of RTA-OSEK's interrupt model for TMS470/TIMULTI. You can find out more about configuring interrupts for RTA-OSEK in the *RTA-OSEK User Guide*.

### 3.1.1 Interrupt Levels

In RTA-OSEK interrupts are allocated an Interrupt Priority Level (IPL). This is a processor independent abstraction of the interrupt priorities that are available on the target hardware. You can find out more about IPLs in the *RTA-OSEK User Guide*. The hardware interrupt controller is explained in the *Design Specification for TMS470PFS761*.

The following table shows how RTA-OSEK IPLs relate to interrupt priorities on the target hardware:

| IPL Value | Reqmask (Vim) | Vim Channel | Category | Vectors |
|---|---|---|---|---|
| 0 | 0xFFFFFFFFFFFFFFFF | - | - | - |
| 1 | 0x7FFFFFFFFFFFFFFF | 63 | 1/2 | 0x11C |
| 2 | 0x3FFFFFFFFFFFFFFF | 62 | 1/2 | 0x118 |
| 3 | 0x1FFFFFFFFFFFFFFF | 61 | 1/2 | 0x114 |
| … | … | … | … | … |
| 61 | 0x0000000000000007 | 3 | 1/2 | 0x2C |
| 62 | 0x0000000000000003 | 2 | 1/2 | 0x28 |
| 62 | 0xXXXXXXXXXXXXXXXX | Phantom (IRQ only) | 1/2 | 0x200 |
| 63 | 0xXXXXXXXXXXXXXXXX | 2 to 63 (FIQ only) | 1 | 0x11C to 0x28 |
| 63 | 0xXXXXXXXXXXXXXXXX | Phantom (FIQ only) | 1 | 0x200 |
| 63 | 0x0000000000000003 | 1 (FIQ only) | 1 | 0x24 |
| 63 | 0x0000000000000003 | 0 (FIQ only) | 1 | 0x20 |
| 64 | 0xXXXXXXXXXXXXXXXX | ARM Exceptions | 1 | 0x4, 0x8, 0xC, 0x10 |

### 3.1.2 Interrupt Vectors

For the allocation of Category 1 and Category 2 interrupt handlers to interrupt vectors on your target hardware, the following restrictions apply:

| Vector | Legality |
|---|---|
| CPU | Category 1 |
| VIM | Category 1 or 2 |

The valid base addresses for the vector table are:

| Base Address | Notes |
|---|---|
| 0xFFF82000 | VIM vector table |
| 0x0 | ARM CPU vector table |

### 3.1.3 Category 1 Handlers

Category 1 interrupt service routines (ISRs) must correctly handle the interrupt context themselves, without support from the operating system. The Texas Instruments Inc. C compiler can generate appropriate interrupt handling code for a C function decorated with the `void` function qualifier. You can find out more in your compiler documentation.

### 3.1.4 Category 2 Handlers

Category 2 ISRs are provided with a C function context by the RTA-OSEK Component, since the RTA-OSEK Component handles the interrupt context itself. The handlers are written using the OSEK OS standard `ISR()` macro, shown in Code Example 3:1.

```
#include "MyISR.h"
ISR(MyISR) {
  /* Handler routine */
}
```

**Code Example 3:1 - Category 2 ISR Interrupt Handler**

You must not insert a return from interrupt instruction in such a function. The return is handled automatically by the RTA-OSEK Component.

### 3.1.5 Vector Table Issues

When you configure your application with the RTA-OSEK GUI you can choose whether or not a vector table is generated within `osgen.s`.

Note that a generated vector table omits the reset vector entry. If you choose to provide your own vector table, it must contain an entry for each interrupt handler, including the Category 2 interrupt handlers in RTA-OSEK.

The following table shows the syntax for labels attached to RTA-OSEK Category 2 interrupt handlers (VVV represents the 3 hex digit, upper-case, zero-padded value of the vector location).

| Vector Location | Label |
|---|---|
| `0xVVV` | `os_wrapper_VVV` |
| eg : 0x030 | `os_wrapper_030` |

### 3.1.6 Vector Tables

The vector table used in the TMS470/TIMULTI implementation of RTA-OSEK concatenates the ARM CPU vector table with the 64 interrupt channels of the Vectored Interrupt Manager (VIM) (see Section 3.1.2). The 64 VIM channels have a separate vector table located within the VIM module which must be initialized during startup (see example application). The VIM interrupt channel assignment and priority selection must also be set up to the user's requirements during startup (CHANCTRL and FIRQPR registers)

### 3.1.7 Reset Vector

In the generated vector table, the Reset Vector at 0x0 is always attached to a user function `c_int00()`. The function is entered with a branch instruction (B) and so it must be located within 32 Mbytes of the reset vector.

The reset handler function should perform the start-up operation required for the TI C compiler and set up the stack pointers for the CPU modes used. An example of `c_int00()` is provided in the example application.

### 3.1.8 Handling Multiple Interrupt Sources

The VIM allows multiple interrupt sources to be processed and handled by the CPU FIQ and IRQ interrupt lines. RTA-OSEK supports this using the direct hardware support offered by the VIM module to vector directly to the appropriate interrupt handling code.

### 3.1.9 Sharing Priorities

RTA-OSEK offers the ability for multiple interrupt sources to share the same IPL. Each interrupt source within the priority group must be adjacent. Arbitration within the group is performed in the normal way by the VIM hardware. Category 1 interrupts may be included provided there are no Category 2 interrupts with a higher IPL. **When grouping two channels or more together into a mutually exclusive group, care must be taken when selecting the priority for each interrupt channel, to ensure correct behavior of RTA-OSEK.** The configured priority of the lower priority channels within the group should be raised to the same IPL as the highest in the group. For example, to group channels 11, 9, and 8, they should all have their priority (IPL) set to 56. Also note that channel 10 could then only be used at priority 56 or not at all (adjacency rule).

The VIM channel priority registers should then be used to configure peripherals at the desired priority level by allocating them to the appropriate channel.

If priority sharing is not desired, **the configured priority should always be the lowest available IPL**. RTA-OSEK is not designed to allow the configured priority to be chosen arbitrarily by the user and would not function correctly.

### 3.1.10 IRQ Category 1 Interrupts

Category 1 interrupts configured on the IRQ line should be declared as a standard C function. RTA-OSEK outputs the following common code which is invoked when any IRQ interrupt occurs (Category 1 or 2). The VIM vector table should contain the address of the user interrupt handling function. The following code is output in `osgen.s` and is branched to when the IRQ is detected by the ARM CPU.

```
os_outer_wrapper:
  SUB   lr,lr,#4
  MRS   sp,CPSR
  ORR   sp,sp,#1
  MSR   CPSR,sp
  STMFD sp!,{r0-r5}
  ADD   r0,sp,#20
  MRS   r4,CPSR
  BIC   r4,r4,#1
  MSR   CPSR,r4
  STR   lr,[r0]
  MRS   r3,SPSR
  MSR   CPSR,sp
  STMFD sp!,{r3}
  STMFD sp!,{r5, r12, lr}
  LDR   lr,_os_wrapper_rtn_addr
  MOV   r0,#0
  LDR   pc,[r0,#-0x190]
```

When the user function returns, the following code if executed to return to the interruption point. This is also contained within the file `osgen.s`.

```
os_wrapper_return:
  LDMFD sp!,{r5,r12,lr}
  MOV   r0,sp
  ADD   sp,r0,#28
  BIC   r4,r4,#1
  MSR   CPSR,r4
  LDMFD r0!,{r1}
  MSR   SPSR_cxfs,r1
  LDMFD r0,{r0-r4,pc}^
```

It should be noted that Category 1 ISRs are entered using a load into the PC this places no restriction on the placement of ISR handles. All Category 1 ISRs will be entered in using the 32-bit instruction set. If an ISR uses the 16-bit instruction set then a veneer can be used to change the mode. An example of the veneer needed is shown below. Here the ISR hander `isr1_thumb()` is entered from the veneer isr1. The ISR entry function entered into the RTA-

OSEK GUI should be `isr1()`; its address will be located in the table loaded into the VIM vector table.

```
  .sect ".text:isr1"
  .align 4
  .clink
  .armfunc isr1
  .state32
  .def isr1
isr1:
  LDR   r12,isr1_addr
  BX    r12

isr1_addr:
  .ref  isr1_thumb
  .word isr1_thumb
```

### 3.1.11 Nesting Category 1 interrupts

When an IRQ Category 1 interrupt is entered the I bit of the CPSR is still set and so no other interrupts will be entered until the interrupt has been serviced. To allow safe nesting of Category 1 interrupts the VIM mask must be set to the correct value before the I bit is cleared. At the end of the Category 1 ISR the VIM mask value must also be returned to the state is was at before the mask was changed. RTA-OSEK provides the required mask values based upon the information entered in the GUI. These mask values are split into 32-bit values and the labels are based upon the address of the VIM vector used. In the following example interrupt `isr1` is attached to the VIM channel at 0x34.

```
void isr1(void)
{
  /* Enable IRQ interrupts and set the mask to block
all lower ISRs */
  osUInt32Type lo_mask=REQMASKCLR_LO;
  osUInt32Type hi_mask=REQMASKCLR_HI;
  REQMASKCLR_LO=OS_IMASK_LO_034;
  REQMASKCLR_HI=OS_IMASK_HI_034;
  os_clear_i_bit();

……
User code
……

  /* Restore the VIM mask to that at the start of
the ISR */
  os_set_i_bit();
  REQMASKCLR_LO=0xFFFFFFFFuL;
  REQMASKCLR_HI=0xFFFFFFFFuL;
  REQMASKSET_LO=lo_mask;
```

```
  REQMASKSET_HI=hi_mask;
}
```

### 3.1.12 FIQ Category 1 Interrupts

FIQ mode provides an increased number of shadow registers. Due to the differing user requirements for use of this interrupt mechanism, it is left to the user to implement their own handling code.

The IPL for all FIQ interrupts should be set to 63 regardless of the VIM channel. FIQ interrupts will always preempt IRQ interrupts and so have a higher priority.

RTA-OSEK uses a mask `os_fiq_mask` to ensure that channels in the VIM allocated to FIQ interrupts are never disabled within the OS. The inverse of this mask can be used to initialize the FIRQPR register (this is demonstrated in the example application).

### 3.1.13 Phantom interrupt support

A phantom interrupt occurs when an interrupt is triggered and the CPU cannot identify the source. This occurs when the IRQ or FIQ request signal is high but the contents of the VIM interrupt offset register (IRQIVEC/FIQIVEC) and interrupt request register (INTREQ) are zero.

RTA-OSEK supports a phantom interrupt handler to be attached to a VIM offset of 0. This can be entered as a result of an FIQ or IRQ interrupt. The IPL of the phantom interrupt should be either 62 for IRQ interrupts or 63 for FIQ interrupts.

## 3.2 CPU Operating Modes

All tasks and Category 2 ISRs must execute in supervisor (SVC) mode. When a Category 2 interrupt occurs, the RTA-OSEK Component switches from IRQ mode to SVC mode before processing the interrupt or utilizing any stack. This minimizes the worst-case stack requirements.

From reset, the processor runs on the SVC stack. The RTA-OSEK Component always expects to run on the SVC stack (i.e. in SVC mode), except when processing Category 1 interrupts. Category 1 interrupts can be configured to use the following CPU operating modes; SVC by using the SWI, FIQ, IRQ, Abort and Undefined.

## 3.3 Register Settings

The RTA-OSEK Component requires the following registers to be initialized before calling `StartOS()`.

| Register | Notes |
|---|---|
| FIRQPR | Initialize with the inverse of `os_fiq_mask` |
| VIM vector table | Initialize with the contents of `os_vim_vectors[]` |

The RTA-OSEK Component uses the following hardware registers. They should not be altered by user code.

| Registers Used | Notes |
|---|---|
| CPSR (CPU) | The CPU mode bits must not be changed |
| REQMASKSET (VIM) | The value of this register must not be changed |
| REQMASKCLR (VIM) | The value of this register must not be changed |

## 3.4 Stack Usage

### 3.4.1 Number of Stacks

A single stack is used. The first argument to `StackFaultHook` is always 0.

`osStackOffsetType` is a scalar, representing the number of bytes on the stack, with C type `unsigned long`.

### 3.4.2 Stack Usage within API Calls

The maximum stack usage within RTA-OSEK API calls, excluding calls to hooks and callbacks, is as follows:

#### Standard

API max usage (bytes): 56

#### Timing

API max usage (bytes): 56

#### Extended

API max usage (bytes): 72

To determine the correct stack usage for tasks that use other library code, you may need to contact the library vendor to find out more about call stack usage.

### 3.4.3 Stack Initialization for Different CPU Modes

The TMS570 CPU can use up to six different stacks, depending upon the number of operating modes that occur in an application.

RTA-OSEK uses only the supervisor (SVC) mode stack even when Category 2 interrupts occur in an application. FIQ mode will only be used in applications that use Category 1 FIQ interrupts and, therefore, operate outside of the scope of RTA-OSEK.

RTA-OSEK adheres to the ARM ABI requirements for stack discipline and the stack pointer is kept aligned on a 64-bit boundary.

All stack pointers must be initialized before use. It should be noted that it is the user's responsibility to ensure that each stack is large enough to avoid overrun.

**Important:** The initial SVC stack pointer (R13) value must be initialized to match the address of the symbol `_os_stack_top_svc`.

The example application demonstrates a suggested stack pointer initialization method. The start-up code, `init.asm`, declares the size of each stack section apart from the SVC stack with the lengths defined by `STACK_LEN_<mode>`. The size of the SVC stack can be defined using the `-stack` linker option. The linker command file, `linktms_flash.cmd`, locates the stack's sections in memory and defines labels at the top of each stack section, `_os_stack_top_<mode>`.

### 3.4.4 Stack Usage with 16-Bit/32-Bit Instructions

The RTA-OSEK runtime libraries can support applications written in either the 16-bit and 32-bit TMS570 instruction sets. The libraries are compiled using the 16-bit instruction set. Stack usage will change when applications use 32-bit instructions.

The 16-bit instruction set should be used to optimize stack usage. When the 32-bit instruction set is used, veneers are placed between functions compiled using the 16 and 32-bit instruction sets. Each 16 to 32-bit veneer uses and releases (before leaving the veneer) 4 bytes of stack. This stack should be taken into account during SVC mode stack calculations (i.e. if a task or ISR uses no stack the veneer requires 4 bytes in order to enter the task or ISR). 32 to 16-bit veneers do not use any stack.

### 3.4.5 Stack Usage in Tick_<Counter Name> ()

In applications that use alarms, a function `Tick_<counter name>` (where `<counter name>` is the associated counter that the alarm is attached to) is placed in `osekdefs.c`.

The amount of stack used in this function depends upon the level of compiler optimization and the number of alarms implemented. A nominal stack usage of 16 bytes has been attributed to this function, which is sufficient for up to 8 alarms. If more alarms are used within an application that uses ECC tasks, then the number of stack bytes used by `Tick_<counter name>()` should be reviewed. Any extra stack usage should be added to the idle task stack figure in the OIL configuration file.

# 4 Parameters of Implementation

This chapter provides detailed information on the functionality, performance and memory demands of the RTA-OSEK Component.

The RTA-OSEK Component is highly scalable. As a result, different figures will be obtained when your application uses different sets of features. These feature-sets give six classes of RTA-OSEK, depending on whether your application uses events, shared task priorities and/or multiple (queued) task activations. You should identify which class your application belongs to and then use the figures from the appropriate column in the table.

The following hardware was used to take the measurements in this chapter:

| **Processor** | **TMS470PSF761** |
|---|---|
| Clock speed (MHz) | 15 |
| Code memory | Internal FLASH |
| Read-only data memory | Internal FLASH |
| Read-write data memory | Internal RAM |

## 4.1 Functionality

The OSEK Operating System Specification specifies four conformance classes. These attributes apply to *systems* built with OSEK OS objects. The following table specifies the number of OSEK OS and COM objects supported per conformance class.

| **Configuration** | **Application Uses** | | | | | |
|---|---|---|---|---|---|---|
| **Events** | **No** | | | **Yes** | | |
| **Shared Task Priorities** | **No** | | **Yes** | **No** | | **Yes** |
| **Multiple Task Activations** | **No** | **Yes** | | **No** | **Yes** | |
| Maximum number of tasks | 32 | 32 | 32 | 32 | 32 | 32 |
| Maximum number of not suspended tasks | 32 | 32 | 32 | 32 | 32 | 32 |
| Maximum number of priorities | 32 | 32 | 32 | 32 | 32 | 32 |
| Number of tasks per priority (for BCC2 and ECC2) | n/a | 32 | 32 | n/a | 32 | 32 |
| Upper limit for number of basic task activations per task priority | 1 | 255 | 255 | 1 | 255 | 255 |
| Maximum number of events per task | 0 | 0 | 0 | 32 | 32 | 32 |
| Limits for the number of alarm objects (per system / per task) | not limited by RTA-OSEK | | | | | |
| Limits for the number of standard resources (per system) | 255 | 255 | 255 | 255 | 255 | 255 |
| Limits for the number of internal resources (per system) | not limited by RTA-OSEK | | | | | |
| Limits for the number of nested resources (per system / per task) | 255 | 255 | 255 | 255 | 255 | 255 |

| Configuration | Application Uses | | | | | |
|---|---|---|---|---|---|---|
| Events | No | | | Yes | | |
| Shared Task Priorities | No | | Yes | No | | Yes |
| Multiple Task Activations | No | Yes | | No | Yes | |
| Limits for the number of application modes | 4294967295 | | | | | |

## 4.2 Hardware Resources

### 4.2.1 ROM and RAM Overheads

The following tables give the ROM and RAM overheads for the RTA-OSEK Component (in bytes). The OSEK COM overheads are quoted separately. If you do not use messages, your application will not include this overhead for the parts of OSEK COM required to implement messaging.

#### Standard

| Configuration | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|
| Events | | No | | | Yes | | |
| Shared Task Priorities | | No | | Yes | No | | Yes |
| Multiple Task Activations | | No | Yes | | No | Yes | |
| OS overhead | RAM | 28 | 28 | 28 | 28 | 28 | 28 |
| | ROM | 176 | 176 | 176 | 176 | 176 | 176 |
| COM overhead | RAM | 8 | 8 | 8 | 8 | 8 | 8 |
| | ROM | 16 | 16 | 16 | 16 | 16 | 16 |

#### Timing

| Configuration | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|
| Events | | No | | | Yes | | |
| Shared Task Priorities | | No | | Yes | No | | Yes |
| Multiple Task Activations | | No | Yes | | No | Yes | |
| OS overhead | RAM | 48 | 48 | 48 | 48 | 48 | 48 |
| | ROM | 244 | 244 | 244 | 244 | 244 | 244 |
| COM overhead | RAM | 8 | 8 | 8 | 8 | 8 | 8 |
| | ROM | 16 | 16 | 16 | 16 | 16 | 16 |

## Extended

| Configuration | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|
| Events | | No | | | Yes | | |
| Shared Task Priorities | | No | | Yes | No | | Yes |
| Multiple Task Activations | | No | Yes | | No | Yes | |
| OS overhead | RAM | 74 | 74 | 74 | 74 | 74 | 74 |
| | ROM | 290 | 290 | 290 | 290 | 290 | 290 |
| COM overhead | RAM | 8 | 8 | 8 | 8 | 8 | 8 |
| | ROM | 16 | 16 | 16 | 16 | 16 | 16 |

### 4.2.2 ROM and RAM for OSEK OS Objects

In addition to the base OS overhead, detailed in Section 4.2.1, each OSEK OS object requires ROM and/or RAM. RTA-OSEK provides additional sub-task types for each task type in OSEK (basic and extended), determined by the offline configuration tools. They are as follows:

| OSEK Class | Termination | Arithmetic |
|---|---|---|
| BCC1 | Lightweight | Integer or Floating-Point |
| BCC1 | Heavyweight | Integer or Floating-Point |
| BCC2 | Light or Heavy | Integer or Floating-Point |
| ECC1 | Heavyweight | Integer |
| ECC1 | Heavyweight | Floating-Point |
| ECC2 | Heavyweight | Integer |
| ECC2 | Heavyweight | Floating-Point |

The following tables give the ROM and/or RAM requirements (in bytes) for each OS object in the RTA-OSEK Component. (Note that the OSEK COM class was set to CCCA for systems without events, CCCB for systems with events. A default message of size 10 bytes was used for both CCCA and CCCB. The CCCB message size includes queued messages.)

## Standard

| Configuration | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|
| Events | | No | | | Yes | | |
| Shared Task Priorities | | No | | Yes | No | | Yes |
| Multiple Task Activations | | No | Yes | | No | Yes | |
| BCC1 Lightweight task | RAM | 0 | 0 | 0 | 0 | 0 | 0 |
| | ROM | 44 | 44 | 44 | 44 | 44 | 44 |
| BCC1 Heavyweight task | RAM | 4 | 4 | 4 | 4 | 4 | 4 |
| | ROM | 48 | 48 | 48 | 48 | 48 | 48 |
| BCC2 task | RAM | n/a | 8 | 10 | n/a | 8 | 10 |
| | ROM | n/a | 52 | 60 | n/a | 52 | 60 |
| ECC1, Integer task | RAM | n/a | n/a | n/a | 56 | 56 | 56 |
| | ROM | n/a | n/a | n/a | 68 | 68 | 68 |
| ECC1, floating-point task | RAM | n/a | n/a | n/a | 58 | 58 | 58 |
| | ROM | n/a | n/a | n/a | 68 | 68 | 68 |
| ECC2, Integer task | RAM | n/a | n/a | n/a | n/a | n/a | 58 |
| | ROM | n/a | n/a | n/a | n/a | n/a | 76 |
| ECC2, floating-point task | RAM | n/a | n/a | n/a | n/a | n/a | 60 |
| | ROM | n/a | n/a | n/a | n/a | n/a | 76 |
| Category 2 ISR | RAM | 0 | 0 | 0 | 0 | 0 | 0 |
| | ROM | 120 | 120 | 120 | 120 | 120 | 120 |
| Category 2 ISR, floating-point | RAM | 1 | 1 | 1 | 1 | 1 | 1 |
| | ROM | 140 | 140 | 140 | 140 | 140 | 140 |
| Resource | RAM | 0 | 0 | 0 | 0 | 0 | 0 |
| | ROM | 20 | 20 | 20 | 20 | 20 | 20 |
| Internal resource | RAM | 0 | 0 | 0 | 0 | 0 | 0 |
| | ROM | 0 | 0 | 0 | 0 | 0 | 0 |
| Linked resource | RAM | 0 | 0 | 0 | 0 | 0 | 0 |
| | ROM | 20 | 20 | 20 | 20 | 20 | 20 |
| Alarm | RAM | 12 | 12 | 12 | 12 | 12 | 12 |
| | ROM | 36 | 36 | 36 | 36 | 36 | 36 |
| Counter | RAM | 4 | 4 | 4 | 4 | 4 | 4 |
| | ROM | 40 | 40 | 40 | 40 | 40 | 40 |
| Message | RAM | 11 | 11 | 11 | 51 | 51 | 51 |
| | ROM | 20 | 20 | 20 | 56 | 56 | 56 |
| Flag | RAM | 4 | 4 | 4 | 4 | 4 | 4 |
| | ROM | 4 | 4 | 4 | 4 | 4 | 4 |
| Message resource | RAM | 0 | 0 | 0 | 0 | 0 | 0 |
| | ROM | 20 | 20 | 20 | 20 | 20 | 20 |

| Configuration | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|
| Events | | No | | | Yes | | |
| Shared Task Priorities | | No | | Yes | No | | Yes |
| Multiple Task Activations | | No | Yes | | No | Yes | |
| Event | RAM | 0 | 0 | 0 | 0 | 0 | 0 |
| | ROM | 4 | 4 | 4 | 4 | 4 | 4 |
| Priority level | RAM | 0 | 0 | 6 | 0 | 6 | 6 |
| | ROM | 0 | 0 | 12 | 0 | 12 | 12 |
| Arrivalpoint (readonly) | RAM | 0 | 0 | 0 | 0 | 0 | 0 |
| | ROM | 12 | 12 | 12 | 12 | 12 | 12 |
| Arrivalpoint (writable) | RAM | 12 | 12 | 12 | 12 | 12 | 12 |
| | ROM | 12 | 12 | 12 | 12 | 12 | 12 |
| Schedule | RAM | 16 | 16 | 16 | 16 | 16 | 16 |
| | ROM | 36 | 36 | 36 | 36 | 36 | 36 |
| Taskset (readonly) | RAM | 0 | 0 | 0 | 0 | 0 | 0 |
| | ROM | 4 | 4 | 4 | 4 | 4 | 4 |
| Taskset (writable) | RAM | 4 | 4 | 4 | 4 | 4 | 4 |
| | ROM | 4 | 4 | 4 | 4 | 4 | 4 |

## Timing

| Configuration | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|
| Events | | No | | | Yes | | |
| Shared Task Priorities | | No | | Yes | No | | Yes |
| Multiple Task Activations | | No | Yes | | No | Yes | |
| BCC1 Lightweight task | RAM | 12 | 12 | 12 | 12 | 12 | 12 |
| | ROM | 52 | 52 | 52 | 52 | 52 | 52 |
| BCC1 Heavyweight task | RAM | 16 | 16 | 16 | 16 | 16 | 16 |
| | ROM | 56 | 56 | 56 | 56 | 56 | 56 |
| BCC2 task | RAM | n/a | 20 | 22 | n/a | 20 | 22 |
| | ROM | n/a | 60 | 68 | n/a | 60 | 68 |
| ECC1, Integer task | RAM | n/a | n/a | n/a | 68 | 68 | 68 |
| | ROM | n/a | n/a | n/a | 76 | 76 | 76 |
| ECC1, floating-point task | RAM | n/a | n/a | n/a | 70 | 70 | 70 |
| | ROM | n/a | n/a | n/a | 76 | 76 | 76 |
| ECC2, Integer task | RAM | n/a | n/a | n/a | n/a | n/a | 70 |
| | ROM | n/a | n/a | n/a | n/a | n/a | 84 |
| ECC2, floating-point task | RAM | n/a | n/a | n/a | n/a | n/a | 72 |
| | ROM | n/a | n/a | n/a | n/a | n/a | 84 |
| Category 2 ISR | RAM | 12 | 12 | 12 | 12 | 12 | 12 |

| Configuration | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|
| Events | | No | | | Yes | | |
| Shared Task Priorities | | No | | Yes | No | | Yes |
| Multiple Task Activations | | No | Yes | | No | Yes | |
| | ROM | 168 | 168 | 168 | 168 | 168 | 168 |
| Category 2 ISR, floating-point | RAM | 14 | 14 | 14 | 14 | 14 | 14 |
| | ROM | 176 | 176 | 176 | 176 | 176 | 176 |
| Resource | RAM | 0 | 0 | 0 | 0 | 0 | 0 |
| | ROM | 20 | 20 | 20 | 20 | 20 | 20 |
| Internal resource | RAM | 0 | 0 | 0 | 0 | 0 | 0 |
| | ROM | 0 | 0 | 0 | 0 | 0 | 0 |
| Linked resource | RAM | 0 | 0 | 0 | 0 | 0 | 0 |
| | ROM | 20 | 20 | 20 | 20 | 20 | 20 |
| Alarm | RAM | 12 | 12 | 12 | 12 | 12 | 12 |
| | ROM | 36 | 36 | 36 | 36 | 36 | 36 |
| Counter | RAM | 4 | 4 | 4 | 4 | 4 | 4 |
| | ROM | 40 | 40 | 40 | 40 | 40 | 40 |
| Message | RAM | 11 | 11 | 11 | 51 | 51 | 51 |
| | ROM | 20 | 20 | 20 | 56 | 56 | 56 |
| Flag | RAM | 4 | 4 | 4 | 4 | 4 | 4 |
| | ROM | 4 | 4 | 4 | 4 | 4 | 4 |
| Message resource | RAM | 0 | 0 | 0 | 0 | 0 | 0 |
| | ROM | 20 | 20 | 20 | 20 | 20 | 20 |
| Event | RAM | 0 | 0 | 0 | 0 | 0 | 0 |
| | ROM | 4 | 4 | 4 | 4 | 4 | 4 |
| Priority level | RAM | 0 | 0 | 6 | 0 | 6 | 6 |
| | ROM | 0 | 0 | 12 | 0 | 12 | 12 |
| Arrivalpoint (readonly) | RAM | 0 | 0 | 0 | 0 | 0 | 0 |
| | ROM | 12 | 12 | 12 | 12 | 12 | 12 |
| Arrivalpoint (writable) | RAM | 12 | 12 | 12 | 12 | 12 | 12 |
| | ROM | 12 | 12 | 12 | 12 | 12 | 12 |
| Schedule | RAM | 16 | 16 | 16 | 16 | 16 | 16 |
| | ROM | 36 | 36 | 36 | 36 | 36 | 36 |
| Taskset (readonly) | RAM | 0 | 0 | 0 | 0 | 0 | 0 |
| | ROM | 4 | 4 | 4 | 4 | 4 | 4 |
| Taskset (writable) | RAM | 4 | 4 | 4 | 4 | 4 | 4 |
| | ROM | 4 | 4 | 4 | 4 | 4 | 4 |

## Extended

| Configuration | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|
| Events | | No | | | Yes | | |
| Shared Task Priorities | | No | | Yes | No | | Yes |
| Multiple Task Activations | | No | Yes | | No | Yes | |
| BCC1 Lightweight task | RAM | 16 | 16 | 16 | 16 | 16 | 16 |
| | ROM | 64 | 64 | 64 | 64 | 64 | 64 |
| BCC1 Heavyweight task | RAM | 20 | 20 | 20 | 20 | 20 | 20 |
| | ROM | 64 | 64 | 64 | 64 | 64 | 64 |
| BCC2 task | RAM | n/a | 24 | 26 | n/a | 24 | 26 |
| | ROM | n/a | 68 | 76 | n/a | 68 | 76 |
| ECC1, Integer task | RAM | n/a | n/a | n/a | 72 | 72 | 72 |
| | ROM | n/a | n/a | n/a | 84 | 84 | 84 |
| ECC1, floating-point task | RAM | n/a | n/a | n/a | 74 | 74 | 74 |
| | ROM | n/a | n/a | n/a | 84 | 84 | 84 |
| ECC2, Integer task | RAM | n/a | n/a | n/a | n/a | n/a | 74 |
| | ROM | n/a | n/a | n/a | n/a | n/a | 92 |
| ECC2, floating-point task | RAM | n/a | n/a | n/a | n/a | n/a | 76 |
| | ROM | n/a | n/a | n/a | n/a | n/a | 92 |
| Category 2 ISR | RAM | 16 | 16 | 16 | 16 | 16 | 16 |
| | ROM | 180 | 180 | 180 | 180 | 180 | 180 |
| Category 2 ISR, floating-point | RAM | 18 | 18 | 18 | 18 | 18 | 18 |
| | ROM | 188 | 188 | 188 | 188 | 188 | 188 |
| Resource | RAM | 16 | 16 | 16 | 16 | 16 | 16 |
| | ROM | 28 | 28 | 28 | 28 | 28 | 28 |
| Internal resource | RAM | 0 | 0 | 0 | 0 | 0 | 0 |
| | ROM | 0 | 0 | 0 | 0 | 0 | 0 |
| Linked resource | RAM | 16 | 16 | 16 | 16 | 16 | 16 |
| | ROM | 28 | 28 | 28 | 28 | 28 | 28 |
| Alarm | RAM | 12 | 12 | 12 | 12 | 12 | 12 |
| | ROM | 40 | 40 | 40 | 40 | 40 | 40 |
| Counter | RAM | 4 | 4 | 4 | 4 | 4 | 4 |
| | ROM | 44 | 44 | 44 | 44 | 44 | 44 |
| Message | RAM | 11 | 11 | 11 | 51 | 51 | 51 |
| | ROM | 24 | 24 | 24 | 60 | 60 | 60 |
| Flag | RAM | 4 | 4 | 4 | 4 | 4 | 4 |
| | ROM | 4 | 4 | 4 | 4 | 4 | 4 |
| Message resource | RAM | 16 | 16 | 16 | 16 | 16 | 16 |
| | ROM | 28 | 28 | 28 | 28 | 28 | 28 |

| Configuration | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|
| Events | | No | | | Yes | | |
| Shared Task Priorities | | No | | Yes | No | | Yes |
| Multiple Task Activations | | No | Yes | | No | Yes | |
| Event | RAM | 0 | 0 | 0 | 0 | 0 | 0 |
| | ROM | 4 | 4 | 4 | 4 | 4 | 4 |
| Priority level | RAM | 0 | 0 | 6 | 0 | 6 | 6 |
| | ROM | 0 | 0 | 12 | 0 | 12 | 12 |
| Arrivalpoint (readonly) | RAM | 0 | 0 | 0 | 0 | 0 | 0 |
| | ROM | 20 | 20 | 20 | 20 | 20 | 20 |
| Arrivalpoint (writable) | RAM | 20 | 20 | 20 | 20 | 20 | 20 |
| | ROM | 20 | 20 | 20 | 20 | 20 | 20 |
| Schedule | RAM | 20 | 20 | 20 | 20 | 20 | 20 |
| | ROM | 44 | 44 | 44 | 44 | 44 | 44 |
| Taskset (readonly) | RAM | 0 | 0 | 0 | 0 | 0 | 0 |
| | ROM | 4 | 4 | 4 | 4 | 4 | 4 |
| Taskset (writable) | RAM | 4 | 4 | 4 | 4 | 4 | 4 |
| | ROM | 4 | 4 | 4 | 4 | 4 | 4 |

### 4.2.3 Size of Linkable Modules

The RTA-OSEK Component is demand linked. This means that each API call is placed into a separately linkable module. The following sections list the module sizes (in bytes) for each API call in the 3 RTA-OSEK build types (standard, timing, and extended).

In some cases there are multiple variants of particular API calls. This is because the offline configuration of RTA-OSEK can determine when optimized versions of the API calls can be used. The smallest and fastest call will be selected. In these cases, modules sizes are given for each variant under the particular configuration of the RTA-OSEK Component for which the call is valid.

The call variants are as follows:

| Variant | Description |
|---|---|
| 1i | Idle task is only ECC task. |
| CCCA | OSEK COM class. |
| CCCB | OSEK COM class. |
| CLEx | Resource tests in Extended OS Status. |
| fp | ECC task uses floating-point. |
| H | Used for heavyweight termination only. |

| Variant | Description |
|---|---|
| Hook | Pre- and Post- Task hooks are used. |
| KL | API is called from OS level. |
| KL1i | API is called from OS level, idle task is only ECC task. |
| KL2 | Activated taskset has one BCC2 task. |
| LExt | Used for lightweight termination in Extended Status. |
| ServiceID | `ErrorHook` uses `GetServiceID`, but does not use `GetServiceParameters`. |
| Parameters | `ErrorHook` uses `GetServiceID` and `GetServiceParameters`. |
| NoHook | Pre- and/or Post- Task hooks are not used. |
| NS | No context switch is possible. |
| NS1i | No context switch is possible, idle task is only ECC task. |
| NS2 | Activated taskset has one BCC2 task. |
| NSH | Chain from heavyweight task, not to higher priority. |
| NSL | Chain from lightweight task, not to higher priority. |
| Shared | Resource is used by tasks and ISRs. |
| SW | A context switch is made if required. |
| SW2 | Activated taskset has one BCC2 task. |
| SWH | Chain from heavyweight task to possibly higher priority. |
| SWL | Chain from lightweight task to possibly higher priority. |
| Task | Resource is used only by tasks. |

## Standard

| Configuration | | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|---|
| Events | | | No | | | Yes | | |
| Shared Task Priorities | | | No | | Yes | No | | Yes |
| Multiple Task Activations | | | No | Yes | | No | Yes | |
| Service name | Variant | Notes | | | | | | |
| ActivateTask | SW | 1 | 120 | 158 | 192 | 130 | 168 | 222 |
| | NS | | 88 | 126 | 160 | 98 | 136 | 186 |
| | KL | 2 | 48 | 84 | 118 | 58 | 94 | 144 |

| Configuration | | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|---|
| Events | | | No | | | Yes | | |
| Shared Task Priorities | | | No | | Yes | No | | Yes |
| Multiple Task Activations | | | No | Yes | | No | Yes | |
| TerminateTask | LExt | 3 | n/a | n/a | n/a | n/a | n/a | n/a |
| | H | 5 | 26 | 26 | 26 | 26 | 26 | 26 |
| ChainTask | SWL | 1, 8 | 92 | 128 | 158 | 102 | 138 | 184 |
| | SWH | 1, 9 | 126 | 166 | 196 | 136 | 176 | 226 |
| | NSL | 8 | 92 | 128 | 158 | 102 | 138 | 184 |
| | NSH | 9 | 114 | 154 | 184 | 124 | 164 | 214 |
| Schedule | | | 86 | 86 | 104 | 86 | 86 | 104 |
| GetTaskID | | | 28 | 28 | 28 | 28 | 28 | 28 |
| GetTaskState | | | 82 | 82 | 82 | 98 | 98 | 98 |
| EnableAllInterrupts | | | 20 | 20 | 20 | 20 | 20 | 20 |
| DisableAllInterrupts | | | 32 | 32 | 32 | 32 | 32 | 32 |
| ResumeAllInterrupts | | | 34 | 34 | 34 | 34 | 34 | 34 |
| SuspendAllInterrupts | | | 50 | 50 | 50 | 50 | 50 | 50 |
| ResumeOSInterrupts | | | 34 | 34 | 34 | 34 | 34 | 34 |
| SuspendOSInterrupts | | | 64 | 64 | 64 | 64 | 64 | 64 |
| GetResource | Task | 7 | 30 | 30 | 34 | 30 | 30 | 34 |
| | Combined | 6 | 98 | 98 | 98 | 98 | 98 | 98 |
| | CLEx | 3 | n/a | n/a | n/a | n/a | n/a | n/a |
| ReleaseResource | Task | 7 | 82 | 82 | 82 | 82 | 82 | 82 |
| | Combined | 6 | 156 | 156 | 156 | 156 | 156 | 156 |
| | CLEx | 3 | n/a | n/a | n/a | n/a | n/a | n/a |
| SetEvent | SW | 1 | n/a | n/a | n/a | 124 | 124 | 194 |
| | NS | | n/a | n/a | n/a | 90 | 90 | 160 |
| | NS1i | 10 | n/a | n/a | n/a | 56 | n/a | n/a |
| | KL | 2 | n/a | n/a | n/a | 58 | 58 | 128 |
| | KL1i | 2, 10 | n/a | n/a | n/a | 20 | n/a | n/a |
| ClearEvent | | | n/a | n/a | n/a | 52 | 52 | 52 |
| GetEvent | | | n/a | n/a | n/a | 10 | 10 | 10 |
| WaitEvent | <default> | | n/a | n/a | n/a | 178 | 178 | 320 |
| | fp | 11 | n/a | n/a | n/a | 200 | 200 | 368 |
| | 1i | 10 | n/a | n/a | n/a | 18 | n/a | n/a |
| GetAlarmBase | | | 50 | 50 | 50 | 50 | 50 | 50 |
| GetAlarm | | | 82 | 82 | 82 | 82 | 82 | 82 |
| SetRelAlarm | | | 110 | 110 | 110 | 110 | 110 | 110 |
| SetAbsAlarm | | | 112 | 112 | 112 | 112 | 112 | 112 |
| CancelAlarm | | | 70 | 70 | 70 | 70 | 70 | 70 |
| InitCounter | | | 56 | 56 | 56 | 56 | 56 | 56 |

| Configuration | | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|---|
| Events | | | No | | | Yes | | |
| Shared Task Priorities | | | No | | Yes | No | | Yes |
| Multiple Task Activations | | | No | Yes | | No | Yes | |
| GetCounterValue | | | 68 | 68 | 68 | 68 | 68 | 68 |
| osek_tick_alarm | <default> | | 70 | 70 | 70 | 70 | 70 | 70 |
| | KL | 2 | 34 | 34 | 34 | 34 | 34 | 34 |
| osek_incr_counter | | | 26 | 26 | 26 | 26 | 26 | 26 |
| GetActiveApplicationMode | | 30 | n/a | n/a | n/a | n/a | n/a | n/a |
| StartOS | | | 130 | 130 | 130 | 130 | 130 | 130 |
| ShutdownOS | NoHook | 12 | 32 | 32 | 32 | 32 | 32 | 32 |
| | Hook | 13 | 40 | 40 | 40 | 40 | 40 | 40 |
| InitCOM | | | 4 | 4 | 4 | 4 | 4 | 4 |
| CloseCOM | | | 4 | 4 | 4 | 4 | 4 | 4 |
| StartCOM | | | 22 | 22 | 22 | 22 | 22 | 22 |
| StopCOM | | | 20 | 20 | 20 | 20 | 20 | 20 |
| ReadFlag | | 30 | n/a | n/a | n/a | n/a | n/a | n/a |
| ResetFlag | | 30 | n/a | n/a | n/a | n/a | n/a | n/a |
| ReceiveMessage | CCCA | 14 | 56 | 56 | 56 | 148 | 148 | 148 |
| | CCCB | 15 | 148 | 148 | 148 | 148 | 148 | 148 |
| GetMessageResource | | | 40 | 40 | 40 | 40 | 40 | 40 |
| ReleaseMessageResource | | | 38 | 38 | 38 | 38 | 38 | 38 |
| GetMessageStatus | | | 38 | 38 | 38 | 38 | 38 | 38 |
| SendMessage | SW CCCA | 1, 14 | 82 | 82 | 82 | 182 | 182 | 182 |
| | SW CCCB | 1, 15 | 170 | 170 | 170 | 182 | 182 | 182 |
| | NS CCCA | 14 | 82 | 82 | 82 | 182 | 182 | 182 |
| | NS CCCB | 15 | 170 | 170 | 170 | 182 | 182 | 182 |
| | KL CCCA | 2, 14 | 50 | 50 | 50 | 152 | 152 | 152 |
| | KL CCCB | 2, 15 | 140 | 140 | 140 | 152 | 152 | 152 |
| main_dispatch | NoHook | 12 | 114 | 114 | 144 | 114 | 114 | 144 |
| | Hook | 13 | 150 | 150 | 182 | 150 | 150 | 182 |
| sub_dispatch | B1LF | 19 | 36 | 36 | 36 | 36 | 36 | 36 |
| | B1HI | 20 | 86 | 86 | 86 | 86 | 86 | 86 |
| | B1HF | 21 | 94 | 94 | 94 | 94 | 94 | 94 |
| | B2LI | 22 | n/a | 74 | 92 | n/a | 74 | 92 |
| | B2LF | 23 | n/a | 82 | 100 | n/a | 82 | 100 |
| | B2HI | 24 | n/a | 132 | 180 | n/a | 132 | 180 |
| | B2HF | 25 | n/a | 140 | 188 | n/a | 140 | 188 |
| | E1HI | 26 | n/a | n/a | n/a | 262 | 262 | 310 |
| | E1HF | 27 | n/a | n/a | n/a | 270 | 270 | 318 |
| | E2HI | 28 | n/a | n/a | n/a | n/a | n/a | 310 |

| Configuration | | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|---|
| Events | | | No | | | Yes | | |
| Shared Task Priorities | | | No | | Yes | No | | Yes |
| Multiple Task Activations | | | No | Yes | | No | Yes | |
| | E2HF | 29 | n/a | n/a | n/a | n/a | n/a | 318 |
| ErrorHook support | | 16 | 36 | 36 | 36 | 36 | 36 | 36 |
| | ServiceID | 17 | 44 | 44 | 44 | 44 | 44 | 44 |
| | Parameters | 18 | 56 | 56 | 56 | 56 | 56 | 56 |
| validity_checks | | 3 | n/a | n/a | n/a | n/a | n/a | n/a |
| Timing_dispatch | | 4 | n/a | n/a | n/a | n/a | n/a | n/a |
| Timing_termination | | 4 | n/a | n/a | n/a | n/a | n/a | n/a |
| ActivateTaskset | SW | 1 | 118 | 218 | 256 | 132 | 246 | 310 |
| | NS | | 86 | 182 | 220 | 100 | 210 | 274 |
| | KL | 2 | 46 | 142 | 178 | 60 | 168 | 234 |
| ChainTaskset | SWL | 1, 8 | 98 | 190 | 224 | 108 | 222 | 268 |
| | SWH | 1, 9 | 132 | 242 | 276 | 140 | 272 | 324 |
| | NSL | 8 | 98 | 190 | 224 | 108 | 222 | 268 |
| | NSH | 9 | 120 | 230 | 264 | 128 | 262 | 312 |
| GetTasksetRef | | | 8 | 8 | 8 | 8 | 8 | 8 |
| MergeTaskset | | | 50 | 50 | 50 | 50 | 50 | 50 |
| AssignTaskset | | | 8 | 8 | 8 | 8 | 8 | 8 |
| RemoveTaskset | | | 50 | 50 | 50 | 50 | 50 | 50 |
| TestSubTaskset | | | 60 | 60 | 60 | 60 | 60 | 60 |
| TestEquivalentTaskset | | | 58 | 58 | 58 | 58 | 58 | 58 |
| TickSchedule | SW | 1 | 172 | 144 | 144 | 144 | 144 | 144 |
| | NS | | 132 | 104 | 104 | 104 | 104 | 104 |
| | KL | 2 | 104 | 72 | 72 | 72 | 72 | 72 |
| AdvanceSchedule | SW | 1 | 164 | 136 | 136 | 136 | 136 | 136 |
| | NS | | 124 | 96 | 96 | 96 | 96 | 96 |
| | KL | 2 | 96 | 68 | 68 | 68 | 68 | 68 |
| StartSchedule | | | 78 | 78 | 78 | 78 | 78 | 78 |
| StopSchedule | | | 64 | 64 | 64 | 64 | 64 | 64 |
| GetScheduleStatus | | | 92 | 92 | 92 | 92 | 92 | 92 |
| GetScheduleValue | | | 70 | 70 | 70 | 70 | 70 | 70 |
| GetScheduleNext | | | 10 | 10 | 10 | 10 | 10 | 10 |
| SetScheduleNext | | | 8 | 8 | 8 | 8 | 8 | 8 |
| GetArrivalpointDelay | | | 8 | 8 | 8 | 8 | 8 | 8 |
| SetArrivalpointDelay | | | 6 | 6 | 6 | 6 | 6 | 6 |
| GetArrivalpointTasksetRef | | | 6 | 6 | 6 | 6 | 6 | 6 |
| GetArrivalpointNext | | | 8 | 8 | 8 | 8 | 8 | 8 |
| SetArrivalpointNext | | | 6 | 6 | 6 | 6 | 6 | 6 |

| Configuration | | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|---|
| Events | | | No | | | Yes | | |
| Shared Task Priorities | | | No | | Yes | No | | Yes |
| Multiple Task Activations | | | No | Yes | | No | Yes | |
| TestArrivalpointWritable | | | 30 | 30 | 30 | 30 | 30 | 30 |
| GetExecutionTime | | | 4 | 4 | 4 | 4 | 4 | 4 |
| GetLargestExecutionTime | | | 6 | 6 | 6 | 6 | 6 | 6 |
| ResetLargestExecutionTime | | | 4 | 4 | 4 | 4 | 4 | 4 |
| GetStackOffset | | | 26 | 26 | 26 | 26 | 26 | 26 |
| Floating point support | | | 32 | 32 | 32 | 32 | 32 | 32 |
| Interrupt support | | | 82 | 82 | 82 | 82 | 82 | 82 |
| Utility functions | | | 226 | 226 | 226 | 226 | 226 | 226 |
| Utility functions | | | 76 | 76 | 76 | 76 | 76 | 76 |
| Utility functions | | | 40 | 40 | 40 | 40 | 40 | 40 |
| Utility functions | | | 8 | 8 | 8 | 8 | 8 | 8 |

## Timing

| Configuration | | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|---|
| Events | | | No | | | Yes | | |
| Shared Task Priorities | | | No | | Yes | No | | Yes |
| Multiple Task Activations | | | No | Yes | | No | Yes | |
| Service name | Variant | Notes | | | | | | |
| ActivateTask | SW | 1 | 120 | 158 | 192 | 130 | 168 | 222 |
| | NS | | 88 | 126 | 160 | 98 | 136 | 186 |
| | KL | 2 | 48 | 84 | 118 | 58 | 94 | 144 |
| TerminateTask | LExt | 3 | n/a | n/a | n/a | n/a | n/a | n/a |
| | H | 5 | 26 | 26 | 26 | 26 | 26 | 26 |
| ChainTask | SWL | 1, 8 | 92 | 128 | 158 | 102 | 138 | 184 |
| | SWH | 1, 9 | 126 | 166 | 196 | 136 | 176 | 226 |
| | NSL | 8 | 92 | 128 | 158 | 102 | 138 | 184 |
| | NSH | 9 | 114 | 154 | 184 | 124 | 164 | 214 |
| Schedule | | | 106 | 106 | 124 | 106 | 106 | 124 |
| GetTaskID | | | 28 | 28 | 28 | 28 | 28 | 28 |
| GetTaskState | | | 82 | 82 | 82 | 98 | 98 | 98 |
| EnableAllInterrupts | | | 20 | 20 | 20 | 20 | 20 | 20 |
| DisableAllInterrupts | | | 32 | 32 | 32 | 32 | 32 | 32 |
| ResumeAllInterrupts | | | 34 | 34 | 34 | 34 | 34 | 34 |
| SuspendAllInterrupts | | | 50 | 50 | 50 | 50 | 50 | 50 |
| ResumeOSInterrupts | | | 34 | 34 | 34 | 34 | 34 | 34 |

| Configuration | | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|---|
| Events | | | No | | | Yes | | |
| Shared Task Priorities | | | No | | Yes | No | | Yes |
| Multiple Task Activations | | | No | Yes | | No | Yes | |
| SuspendOSInterrupts | | | 64 | 64 | 64 | 64 | 64 | 64 |
| GetResource | Task | 7 | 30 | 30 | 34 | 30 | 30 | 34 |
| | Combined | 6 | 98 | 98 | 98 | 98 | 98 | 98 |
| | CLEx | 3 | n/a | n/a | n/a | n/a | n/a | n/a |
| ReleaseResource | Task | 7 | 102 | 102 | 102 | 102 | 102 | 102 |
| | Combined | 6 | 192 | 192 | 192 | 192 | 192 | 192 |
| | CLEx | 3 | n/a | n/a | n/a | n/a | n/a | n/a |
| SetEvent | SW | 1 | n/a | n/a | n/a | 124 | 124 | 194 |
| | NS | | n/a | n/a | n/a | 90 | 90 | 160 |
| | NS1i | 10 | n/a | n/a | n/a | 56 | n/a | n/a |
| | KL | 2 | n/a | n/a | n/a | 58 | 58 | 128 |
| | KL1i | 2, 10 | n/a | n/a | n/a | 20 | n/a | n/a |
| ClearEvent | | | n/a | n/a | n/a | 52 | 52 | 52 |
| GetEvent | | | n/a | n/a | n/a | 10 | 10 | 10 |
| WaitEvent | <default> | | n/a | n/a | n/a | 236 | 236 | 378 |
| | fp | 11 | n/a | n/a | n/a | 258 | 258 | 426 |
| | 1i | 10 | n/a | n/a | n/a | 84 | n/a | n/a |
| GetAlarmBase | | | 50 | 50 | 50 | 50 | 50 | 50 |
| GetAlarm | | | 82 | 82 | 82 | 82 | 82 | 82 |
| SetRelAlarm | | | 110 | 110 | 110 | 110 | 110 | 110 |
| SetAbsAlarm | | | 112 | 112 | 112 | 112 | 112 | 112 |
| CancelAlarm | | | 70 | 70 | 70 | 70 | 70 | 70 |
| InitCounter | | | 56 | 56 | 56 | 56 | 56 | 56 |
| GetCounterValue | | | 68 | 68 | 68 | 68 | 68 | 68 |
| osek_tick_alarm | <default> | | 70 | 70 | 70 | 70 | 70 | 70 |
| | KL | 2 | 34 | 34 | 34 | 34 | 34 | 34 |
| osek_incr_counter | | | 26 | 26 | 26 | 26 | 26 | 26 |
| GetActiveApplicationMode | | 30 | n/a | n/a | n/a | n/a | n/a | n/a |
| StartOS | | | 176 | 176 | 176 | 176 | 176 | 176 |
| ShutdownOS | NoHook | 12 | 32 | 32 | 32 | 32 | 32 | 32 |
| | Hook | 13 | 40 | 40 | 40 | 40 | 40 | 40 |
| InitCOM | | | 4 | 4 | 4 | 4 | 4 | 4 |
| CloseCOM | | | 4 | 4 | 4 | 4 | 4 | 4 |
| StartCOM | | | 22 | 22 | 22 | 22 | 22 | 22 |
| StopCOM | | | 20 | 20 | 20 | 20 | 20 | 20 |
| ReadFlag | | 30 | n/a | n/a | n/a | n/a | n/a | n/a |
| ResetFlag | | 30 | n/a | n/a | n/a | n/a | n/a | n/a |

| Configuration | | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|---|
| Events | | | No | | | Yes | | |
| Shared Task Priorities | | | No | | Yes | No | | Yes |
| Multiple Task Activations | | | No | Yes | | No | Yes | |
| ReceiveMessage | CCCA | 14 | 56 | 56 | 56 | 148 | 148 | 148 |
| | CCCB | 15 | 148 | 148 | 148 | 148 | 148 | 148 |
| GetMessageResource | | | 40 | 40 | 40 | 40 | 40 | 40 |
| ReleaseMessageResource | | | 38 | 38 | 38 | 38 | 38 | 38 |
| GetMessageStatus | | | 38 | 38 | 38 | 38 | 38 | 38 |
| SendMessage | SW CCCA | 1, 14 | 82 | 82 | 82 | 182 | 182 | 182 |
| | SW CCCB | 1, 15 | 170 | 170 | 170 | 182 | 182 | 182 |
| | NS CCCA | 14 | 82 | 82 | 82 | 182 | 182 | 182 |
| | NS CCCB | 15 | 170 | 170 | 170 | 182 | 182 | 182 |
| | KL CCCA | 2, 14 | 50 | 50 | 50 | 152 | 152 | 152 |
| | KL CCCB | 2, 15 | 140 | 140 | 140 | 152 | 152 | 152 |
| main_dispatch | NoHook | 12 | 160 | 160 | 190 | 160 | 160 | 190 |
| | Hook | 13 | 198 | 198 | 228 | 198 | 198 | 228 |
| sub_dispatch | B1LF | 19 | 20 | 20 | 20 | 20 | 20 | 20 |
| | B1HI | 20 | 82 | 82 | 82 | 82 | 82 | 82 |
| | B1HF | 21 | 90 | 90 | 90 | 90 | 90 | 90 |
| | B2LI | 22 | n/a | 52 | 70 | n/a | 52 | 70 |
| | B2LF | 23 | n/a | 60 | 78 | n/a | 60 | 78 |
| | B2HI | 24 | n/a | 108 | 156 | n/a | 108 | 156 |
| | B2HF | 25 | n/a | 116 | 164 | n/a | 116 | 164 |
| | E1HI | 26 | n/a | n/a | n/a | 284 | 284 | 328 |
| | E1HF | 27 | n/a | n/a | n/a | 292 | 292 | 336 |
| | E2HI | 28 | n/a | n/a | n/a | n/a | n/a | 328 |
| | E2HF | 29 | n/a | n/a | n/a | n/a | n/a | 336 |
| ErrorHook support | | 16 | 36 | 36 | 36 | 36 | 36 | 36 |
| | ServiceID | 17 | 44 | 44 | 44 | 44 | 44 | 44 |
| | Parameters | 18 | 56 | 56 | 56 | 56 | 56 | 56 |
| validity_checks | | 3 | n/a | n/a | n/a | n/a | n/a | n/a |
| Timing_dispatch | | 4 | 76 | 76 | 76 | 76 | 76 | 76 |
| Timing_termination | | 4 | 72 | 72 | 72 | 72 | 72 | 72 |
| ActivateTaskset | SW | 1 | 118 | 218 | 256 | 132 | 246 | 310 |
| | NS | | 86 | 182 | 220 | 100 | 210 | 274 |
| | KL | 2 | 46 | 142 | 178 | 60 | 168 | 234 |
| ChainTaskset | SWL | 1, 8 | 98 | 190 | 224 | 108 | 222 | 268 |
| | SWH | 1, 9 | 132 | 242 | 276 | 140 | 272 | 324 |
| | NSL | 8 | 98 | 190 | 224 | 108 | 222 | 268 |
| | NSH | 9 | 120 | 230 | 264 | 128 | 262 | 312 |

| Configuration | | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|---|
| Events | | | No | | | Yes | | |
| Shared Task Priorities | | | No | | Yes | No | | Yes |
| Multiple Task Activations | | | No | Yes | | No | Yes | |
| GetTasksetRef | | | 8 | 8 | 8 | 8 | 8 | 8 |
| MergeTaskset | | | 50 | 50 | 50 | 50 | 50 | 50 |
| AssignTaskset | | | 8 | 8 | 8 | 8 | 8 | 8 |
| RemoveTaskset | | | 50 | 50 | 50 | 50 | 50 | 50 |
| TestSubTaskset | | | 60 | 60 | 60 | 60 | 60 | 60 |
| TestEquivalentTaskset | | | 58 | 58 | 58 | 58 | 58 | 58 |
| TickSchedule | SW | 1 | 172 | 144 | 144 | 144 | 144 | 144 |
| | NS | | 132 | 104 | 104 | 104 | 104 | 104 |
| | KL | 2 | 104 | 72 | 72 | 72 | 72 | 72 |
| AdvanceSchedule | SW | 1 | 164 | 136 | 136 | 136 | 136 | 136 |
| | NS | | 124 | 96 | 96 | 96 | 96 | 96 |
| | KL | 2 | 96 | 68 | 68 | 68 | 68 | 68 |
| StartSchedule | | | 78 | 78 | 78 | 78 | 78 | 78 |
| StopSchedule | | | 64 | 64 | 64 | 64 | 64 | 64 |
| GetScheduleStatus | | | 92 | 92 | 92 | 92 | 92 | 92 |
| GetScheduleValue | | | 70 | 70 | 70 | 70 | 70 | 70 |
| GetScheduleNext | | | 10 | 10 | 10 | 10 | 10 | 10 |
| SetScheduleNext | | | 8 | 8 | 8 | 8 | 8 | 8 |
| GetArrivalpointDelay | | | 8 | 8 | 8 | 8 | 8 | 8 |
| SetArrivalpointDelay | | | 6 | 6 | 6 | 6 | 6 | 6 |
| GetArrivalpointTasksetRef | | | 6 | 6 | 6 | 6 | 6 | 6 |
| GetArrivalpointNext | | | 8 | 8 | 8 | 8 | 8 | 8 |
| SetArrivalpointNext | | | 6 | 6 | 6 | 6 | 6 | 6 |
| TestArrivalpointWritable | | | 30 | 30 | 30 | 30 | 30 | 30 |
| GetExecutionTime | | | 98 | 98 | 98 | 98 | 98 | 98 |
| GetLargestExecutionTime | | | 12 | 12 | 12 | 12 | 12 | 12 |
| ResetLargestExecutionTime | | | 12 | 12 | 12 | 12 | 12 | 12 |
| GetStackOffset | | | 26 | 26 | 26 | 26 | 26 | 26 |
| Floating point support | | | 32 | 32 | 32 | 32 | 32 | 32 |
| Interrupt support | | | 208 | 208 | 208 | 208 | 208 | 208 |
| Utility functions | | | 226 | 226 | 226 | 226 | 226 | 226 |
| Utility functions | | | 76 | 76 | 76 | 76 | 76 | 76 |
| Utility functions | | | 40 | 40 | 40 | 40 | 40 | 40 |
| Utility functions | | | 8 | 8 | 8 | 8 | 8 | 8 |

## Extended

| Configuration | | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|---|
| Events | | | No | | | Yes | | |
| Shared Task Priorities | | | No | | Yes | No | | Yes |
| Multiple Task Activations | | | No | Yes | | No | Yes | |
| Service name | Variant | Notes | | | | | | |
| ActivateTask | SW | 1 | 214 | 256 | 286 | 228 | 266 | 312 |
| | NS | | 246 | 286 | 316 | 260 | 296 | 348 |
| | KL | 2 | 130 | 170 | 200 | 144 | 180 | 232 |
| TerminateTask | LExt | 3 | 114 | 114 | 114 | 114 | 114 | 114 |
| | H | 5 | 134 | 134 | 134 | 134 | 134 | 134 |
| ChainTask | SWL | 1, 8 | 248 | 286 | 314 | 260 | 296 | 344 |
| | SWH | 1, 9 | 288 | 326 | 354 | 300 | 336 | 384 |
| | NSL | 8 | 306 | 344 | 372 | 318 | 354 | 400 |
| | NSH | 9 | 342 | 382 | 410 | 354 | 392 | 442 |
| Schedule | | | 216 | 216 | 234 | 216 | 216 | 234 |
| GetTaskID | | | 42 | 42 | 42 | 42 | 42 | 42 |
| GetTaskState | | | 202 | 202 | 202 | 208 | 208 | 208 |
| EnableAllInterrupts | | | 34 | 34 | 34 | 34 | 34 | 34 |
| DisableAllInterrupts | | | 46 | 46 | 46 | 46 | 46 | 46 |
| ResumeAllInterrupts | | | 70 | 70 | 70 | 70 | 70 | 70 |
| SuspendAllInterrupts | | | 64 | 64 | 64 | 64 | 64 | 64 |
| ResumeOSInterrupts | | | 70 | 70 | 70 | 70 | 70 | 70 |
| SuspendOSInterrupts | | | 78 | 78 | 78 | 78 | 78 | 78 |
| GetResource | Task | 7 | 334 | 334 | 314 | 334 | 334 | 314 |
| | Combined | 6 | 302 | 302 | 302 | 302 | 302 | 302 |
| | CLEx | 3 | 300 | 300 | 300 | 300 | 300 | 300 |
| ReleaseResource | Task | 7 | 304 | 304 | 304 | 304 | 304 | 304 |
| | Combined | 6 | 408 | 408 | 408 | 408 | 408 | 408 |
| | CLEx | 3 | 264 | 264 | 264 | 264 | 264 | 264 |
| SetEvent | SW | 1 | n/a | n/a | n/a | 256 | 256 | 326 |
| | NS | | n/a | n/a | n/a | 302 | 302 | 372 |
| | NS1i | 10 | n/a | n/a | n/a | 176 | n/a | n/a |
| | KL | 2 | n/a | n/a | n/a | 182 | 182 | 252 |
| | KL1i | 2, 10 | n/a | n/a | n/a | 132 | n/a | n/a |
| ClearEvent | | | n/a | n/a | n/a | 116 | 116 | 116 |
| GetEvent | | | n/a | n/a | n/a | 128 | 128 | 128 |
| WaitEvent | <default> | | n/a | n/a | n/a | 330 | 330 | 470 |
| | fp | 11 | n/a | n/a | n/a | 356 | 356 | 508 |

| Configuration | | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|---|
| Events | | | No | | | Yes | | |
| Shared Task Priorities | | | No | | Yes | No | | Yes |
| Multiple Task Activations | | | No | Yes | | No | Yes | |
| | 1i | 10 | n/a | n/a | n/a | 180 | n/a | n/a |
| GetAlarmBase | | | 138 | 138 | 138 | 138 | 138 | 138 |
| GetAlarm | | | 148 | 148 | 148 | 148 | 148 | 148 |
| SetRelAlarm | | | 200 | 200 | 200 | 200 | 200 | 200 |
| SetAbsAlarm | | | 208 | 208 | 208 | 208 | 208 | 208 |
| CancelAlarm | | | 132 | 132 | 132 | 132 | 132 | 132 |
| InitCounter | | | 184 | 184 | 184 | 184 | 184 | 184 |
| GetCounterValue | | | 156 | 156 | 156 | 156 | 156 | 156 |
| osek_tick_alarm | <default> | | 100 | 100 | 100 | 100 | 100 | 100 |
| | KL | 2 | 34 | 34 | 34 | 34 | 34 | 34 |
| osek_incr_counter | | | 26 | 26 | 26 | 26 | 26 | 26 |
| GetActiveApplicationMode | | 30 | n/a | n/a | n/a | n/a | n/a | n/a |
| StartOS | | | 190 | 190 | 190 | 190 | 190 | 190 |
| ShutdownOS | NoHook | 12 | 42 | 42 | 42 | 42 | 42 | 42 |
| | Hook | 13 | 50 | 50 | 50 | 50 | 50 | 50 |
| InitCOM | | | 4 | 4 | 4 | 4 | 4 | 4 |
| CloseCOM | | | 4 | 4 | 4 | 4 | 4 | 4 |
| StartCOM | | | 36 | 36 | 36 | 36 | 36 | 36 |
| StopCOM | | | 42 | 42 | 42 | 42 | 42 | 42 |
| ReadFlag | | | 26 | 26 | 26 | 26 | 26 | 26 |
| ResetFlag | | | 28 | 28 | 28 | 28 | 28 | 28 |
| ReceiveMessage | CCCA | 14 | 140 | 140 | 140 | 226 | 226 | 226 |
| | CCCB | 15 | 226 | 226 | 226 | 226 | 226 | 226 |
| GetMessageResource | | | 76 | 76 | 76 | 76 | 76 | 76 |
| ReleaseMessageResource | | | 76 | 76 | 76 | 76 | 76 | 76 |
| GetMessageStatus | | | 78 | 78 | 78 | 78 | 78 | 78 |
| SendMessage | SW CCCA | 1, 14 | 170 | 170 | 170 | 270 | 270 | 270 |
| | SW CCCB | 1, 15 | 258 | 258 | 258 | 270 | 270 | 270 |
| | NS CCCA | 14 | 170 | 170 | 170 | 270 | 270 | 270 |
| | NS CCCB | 15 | 258 | 258 | 258 | 270 | 270 | 270 |
| | KL CCCA | 2, 14 | 130 | 130 | 130 | 228 | 228 | 228 |
| | KL CCCB | 2, 15 | 216 | 216 | 216 | 228 | 228 | 228 |
| main_dispatch | NoHook | 12 | 160 | 160 | 190 | 160 | 160 | 190 |
| | Hook | 13 | 198 | 198 | 228 | 198 | 198 | 228 |
| sub_dispatch | B1LF | 19 | 20 | 20 | 20 | 20 | 20 | 20 |
| | B1HI | 20 | 82 | 82 | 82 | 82 | 82 | 82 |
| | B1HF | 21 | 90 | 90 | 90 | 90 | 90 | 90 |

| Configuration | | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|---|
| Events | | | No | | | Yes | | |
| Shared Task Priorities | | | No | | Yes | No | | Yes |
| Multiple Task Activations | | | No | Yes | | No | Yes | |
| | B2LI | 22 | n/a | 52 | 70 | n/a | 52 | 70 |
| | B2LF | 23 | n/a | 60 | 78 | n/a | 60 | 78 |
| | B2HI | 24 | n/a | 108 | 156 | n/a | 108 | 156 |
| | B2HF | 25 | n/a | 116 | 164 | n/a | 116 | 164 |
| | E1HI | 26 | n/a | n/a | n/a | 284 | 284 | 328 |
| | E1HF | 27 | n/a | n/a | n/a | 292 | 292 | 336 |
| | E2HI | 28 | n/a | n/a | n/a | n/a | n/a | 328 |
| | E2HF | 29 | n/a | n/a | n/a | n/a | n/a | 336 |
| ErrorHook support | | 16 | 108 | 108 | 108 | 108 | 108 | 108 |
| | ServiceID | 17 | 120 | 120 | 120 | 120 | 120 | 120 |
| | Parameters | 18 | 150 | 150 | 150 | 150 | 150 | 150 |
| validity_checks | | 3 | 26 | 26 | 26 | 26 | 26 | 26 |
| Timing_dispatch | | 4 | 76 | 76 | 76 | 76 | 76 | 76 |
| Timing_termination | | 4 | 72 | 72 | 72 | 72 | 72 | 72 |
| ActivateTaskset | SW | 1 | 268 | 332 | 358 | 290 | 362 | 410 |
| | NS | | 308 | 360 | 394 | 330 | 384 | 442 |
| | KL | 2 | 186 | 236 | 272 | 208 | 262 | 320 |
| ChainTaskset | SWL | 1, 8 | 312 | 386 | 416 | 334 | 412 | 458 |
| | SWH | 1, 9 | 366 | 436 | 472 | 388 | 464 | 510 |
| | NSL | 8 | 366 | 450 | 480 | 396 | 476 | 522 |
| | NSH | 9 | 406 | 488 | 520 | 432 | 514 | 558 |
| GetTasksetRef | | | 96 | 96 | 96 | 96 | 96 | 96 |
| MergeTaskset | | | 196 | 196 | 196 | 196 | 196 | 196 |
| AssignTaskset | | | 118 | 118 | 118 | 118 | 118 | 118 |
| RemoveTaskset | | | 196 | 196 | 196 | 196 | 196 | 196 |
| TestSubTaskset | | | 204 | 204 | 204 | 204 | 204 | 204 |
| TestEquivalentTaskset | | | 202 | 202 | 202 | 202 | 202 | 202 |
| TickSchedule | SW | 1 | 316 | 236 | 236 | 236 | 236 | 236 |
| | NS | | 352 | 292 | 292 | 292 | 292 | 292 |
| | KL | 2 | 230 | 162 | 162 | 162 | 162 | 162 |
| AdvanceSchedule | SW | 1 | 310 | 244 | 244 | 244 | 244 | 244 |
| | NS | | 356 | 298 | 298 | 298 | 298 | 298 |
| | KL | 2 | 228 | 170 | 170 | 170 | 170 | 170 |
| StartSchedule | | | 192 | 192 | 192 | 192 | 192 | 192 |
| StopSchedule | | | 148 | 148 | 148 | 148 | 148 | 148 |
| GetScheduleStatus | | | 184 | 184 | 184 | 184 | 184 | 184 |
| GetScheduleValue | | | 158 | 158 | 158 | 158 | 158 | 158 |

| Configuration | | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|---|
| Events | | | No | | | Yes | | |
| Shared Task Priorities | | | No | | Yes | No | | Yes |
| Multiple Task Activations | | | No | Yes | | No | Yes | |
| GetScheduleNext | | | 66 | 66 | 66 | 66 | 66 | 66 |
| SetScheduleNext | | | 128 | 128 | 128 | 128 | 128 | 128 |
| GetArrivalpointDelay | | | 92 | 92 | 92 | 92 | 92 | 92 |
| SetArrivalpointDelay | | | 106 | 106 | 106 | 106 | 106 | 106 |
| GetArrivalpointTasksetRef | | | 90 | 90 | 90 | 90 | 90 | 90 |
| GetArrivalpointNext | | | 92 | 92 | 92 | 92 | 92 | 92 |
| SetArrivalpointNext | | | 126 | 126 | 126 | 126 | 126 | 126 |
| TestArrivalpointWritable | | | 102 | 102 | 102 | 102 | 102 | 102 |
| GetExecutionTime | | | 142 | 142 | 142 | 142 | 142 | 142 |
| GetLargestExecutionTime | | | 84 | 84 | 84 | 84 | 84 | 84 |
| ResetLargestExecutionTime | | | 80 | 80 | 80 | 80 | 80 | 80 |
| GetStackOffset | | | 26 | 26 | 26 | 26 | 26 | 26 |
| Floating point support | | | 32 | 32 | 32 | 32 | 32 | 32 |
| Interrupt support | | | 208 | 208 | 208 | 208 | 208 | 208 |
| Utility functions | | | 226 | 226 | 226 | 226 | 226 | 226 |
| Utility functions | | | 76 | 76 | 76 | 76 | 76 | 76 |
| Utility functions | | | 40 | 40 | 40 | 40 | 40 | 40 |
| Utility functions | | | 8 | 8 | 8 | 8 | 8 | 8 |

## Notes

| Number | Note |
|---|---|
| 1 | Linked only if upward activations are allowed |
| 2 | Linked only if API is called within ISR |
| 3 | Present only in Extended OS status |
| 4 | Present only in Timing or Extended OS status |
| 5 | Linked only if there are heavyweight tasks in the system |
| 6 | Linked only if Resource is used by both tasks and ISRs |
| 7 | Linked only if Resource is used only by tasks |
| 8 | Linked only if Chaining task is Lightweight |
| 9 | Linked only if Chaining task is Heavyweight |
| 10 | Linked only if Idle task is the only extended task in the system |
| 11 | Linked only if calling Extended task uses floating-point |
| 12 | Linked only if neither Pre- nor Post-TaskHook is used |
| 13 | Linked only if Pre- or Post-TaskHook is used |
| 14 | Linked only if there are no flags, message queues, or message |

| Number | Note |
|---|---|
| | resources in the system, and COM status is not requested. |
| 15 | Linked only if there are any flags, message queues, or message resources in the system, or COM status is requested. |
| 16 | Linked only if `USEGETSERVICEID = FALSE` and `USEPARAMETERACCESS = FALSE` |
| 17 | Linked only if `USEGETSERVICEID = TRUE` and `USEPARAMETERACCESS = FALSE` |
| 18 | Linked only if `USEGETSERVICEID = TRUE` and `USEPARAMETERACCESS = TRUE` |
| 19 | Linked only for basic, single-activation, lightweight, floating-point tasks |
| 20 | Linked only for basic, single-activation, heavyweight, integer tasks |
| 21 | Linked only for basic, single-activation, heavyweight, floating-point tasks |
| 22 | Linked only for basic, multiple-activation, lightweight, integer tasks |
| 23 | Linked only for basic, multiple-activation, lightweight, floating-point tasks |
| 24 | Linked only for basic, multiple-activation, heavyweight, integer tasks |
| 25 | Linked only for basic, multiple-activation, heavyweight, floating-point tasks |
| 26 | Linked only for extended, unique priority, integer tasks |
| 27 | Linked only for extended, unique priority, floating-point tasks |
| 28 | Linked only for extended, shared priority, integer tasks |
| 29 | Linked only for extended, shared priority, floating-point tasks |
| 30 | Implemented as a macro, so no code is linked |
| 31 | Not required on some targets |

### 4.2.4 Reserved Hardware Resources

Timer units, interrupts, traps and other hardware resources are not reserved by RTA-OSEK.

## 4.3 Performance

The collection of performance data for the TMS470/TIMULTI port of the RTA-OSEK Component was achieved using a timer running two times slower than the CPU clock speed. The figures in this section, therefore, have an uncertainty level of up to two CPU cycles. The actual times are between 0 and two cycles shorter than those reported in the remainder of this section.

### 4.3.1 Execution Times for RTA-OSEK API Calls

The following tables give the execution time (in CPU cycles) for each API call. (Note that: (1) the OSEK COM class was set to CCCA for systems without events and to CCCB for systems with events; (2) `ShutdownOS()` enters an

infinite loop; the execution time for `ShutdownOS()` reported below is the time up to the point at which `ShutdownOS()` calls `ShutdownHook()`).

## Standard

| Configuration | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|
| Events | | No | | | Yes | | |
| Shared Task Priorities | | No | | Yes | No | | Yes |
| Multiple Task Activations | | No | Yes | | No | Yes | |
| Service | Variant | | | | | | |
| ActivateTask | SW | 378 | 434 | 498 | 388 | 418 | 510 |
| | NS | 342 | 396 | 460 | 352 | 382 | 470 |
| | KL | 76 | 130 | 194 | 86 | 116 | 196 |
| TerminateTask | LExt | 0 | 0 | 0 | 0 | 0 | 0 |
| | H | 396 | 398 | 410 | 398 | 398 | 414 |
| ChainTask | SWL | 688 | 750 | 854 | 788 | 818 | 940 |
| | SWH | 870 | 936 | 1042 | 970 | 998 | 1126 |
| | NSL | 688 | 750 | 854 | 788 | 818 | 940 |
| | NSH | 858 | 924 | 1030 | 958 | 986 | 1114 |
| Schedule | SW | 350 | 352 | 370 | 350 | 350 | 370 |
| GetTaskID | | 54 | 54 | 54 | 54 | 54 | 54 |
| GetTaskState | | 368 | 368 | 368 | 382 | 382 | 380 |
| EnableAllInterrupts | | 142 | 142 | 142 | 142 | 142 | 140 |
| DisableAllInterrupts | | 186 | 186 | 186 | 186 | 186 | 188 |
| ResumeAllInterrupts | | 158 | 158 | 158 | 158 | 158 | 156 |
| SuspendAllInterrupts | | 208 | 208 | 208 | 208 | 208 | 210 |
| ResumeOSInterrupts | | 158 | 158 | 158 | 158 | 158 | 156 |
| SuspendOSInterrupts | | 208 | 208 | 208 | 208 | 208 | 210 |
| GetResource | Task | 72 | 72 | 76 | 72 | 72 | 78 |
| | Combined | 288 | 288 | 288 | 288 | 288 | 288 |
| | CLEx | n/a | n/a | n/a | n/a | n/a | n/a |
| ReleaseResource | Task | 358 | 358 | 358 | 358 | 358 | 358 |
| | Combined | 648 | 648 | 648 | 648 | 648 | 648 |
| | CLEx | n/a | n/a | n/a | n/a | n/a | n/a |
| SetEvent | SW | n/a | n/a | n/a | 390 | 388 | 390 |
| | NS | n/a | n/a | n/a | 392 | 392 | 392 |
| | KL | n/a | n/a | n/a | 122 | 122 | 122 |
| ClearEvent | | n/a | n/a | n/a | 294 | 294 | 294 |
| GetEvent | | n/a | n/a | n/a | 44 | 42 | 44 |
| WaitEvent | <default> | n/a | n/a | n/a | 1024 | 1024 | 1104 |

| Configuration | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|
| Events | | No | | | Yes | | |
| Shared Task Priorities | | No | | Yes | No | | Yes |
| Multiple Task Activations | | No | Yes | | No | Yes | |
| | fp | n/a | n/a | n/a | 1038 | 1036 | 1118 |
| GetAlarmBase | | 336 | 336 | 336 | 338 | 338 | 336 |
| GetAlarm | | 366 | 364 | 366 | 366 | 364 | 366 |
| SetRelAlarm | | 388 | 388 | 388 | 388 | 388 | 388 |
| SetAbsAlarm | | 376 | 378 | 376 | 376 | 378 | 376 |
| CancelAlarm | | 330 | 330 | 330 | 332 | 332 | 330 |
| InitCounter | | 334 | 334 | 334 | 334 | 334 | 334 |
| GetCounterValue | | 340 | 340 | 340 | 342 | 342 | 340 |
| osek_tick_alarm | <default> | 358 | 358 | 358 | 358 | 358 | 358 |
| | KL | 72 | 72 | 72 | 72 | 72 | 72 |
| osek_incr_counter | | 26 | 26 | 26 | 26 | 26 | 26 |
| GetActiveApplicationMode | | 20 | 20 | 20 | 20 | 20 | 20 |
| StartOS | | 1992 | 1992 | 1992 | 1992 | 1992 | 1992 |
| ShutdownOS | NoHook | n/a | n/a | n/a | n/a | n/a | n/a |
| | Hook | 206 | 206 | 206 | 206 | 206 | 206 |
| InitCOM | | 18 | 18 | 18 | 18 | 18 | 20 |
| CloseCOM | | 18 | 20 | 18 | 18 | 20 | 18 |
| StartCOM | | 66 | 66 | 66 | 184 | 184 | 184 |
| StopCOM | | 34 | 34 | 34 | 34 | 34 | 34 |
| ReadFlag | | n/a | n/a | n/a | 32 | 32 | 32 |
| ResetFlag | | n/a | n/a | n/a | 24 | 26 | 24 |
| ReceiveMessage | | 330 | 330 | 330 | 650 | 648 | 648 |
| GetMessageResource | | n/a | n/a | n/a | 202 | 202 | 202 |
| ReleaseMessageResource | | n/a | n/a | n/a | 464 | 462 | 462 |
| GetMessageStatus | | n/a | n/a | n/a | 74 | 72 | 74 |
| SendMessage | SW | 718 | 774 | 838 | 1050 | 1080 | 1174 |
| | NS | 682 | 736 | 800 | 1014 | 1044 | 1132 |
| | KL | 146 | 202 | 266 | 482 | 510 | 594 |
| ActivateTaskset | SW | 368 | 1762 | 1834 | 382 | 1760 | 1874 |
| | NS | 332 | 1716 | 1788 | 346 | 1716 | 1828 |
| | KL | 66 | 1452 | 1516 | 80 | 1452 | 1566 |
| | SW2 | 368 | 1762 | 1834 | 382 | 1760 | 1874 |
| | NS2 | 332 | 1716 | 1790 | 346 | 1716 | 1828 |
| | KL2 | 66 | 1452 | 1518 | 80 | 1452 | 1566 |
| ChainTaskset | SWL | 704 | 2092 | 2208 | 804 | 2180 | 2388 |
| | SWH | 872 | 2282 | 2396 | 970 | 2358 | 2568 |

| Configuration | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|
| Events | | No | | | Yes | | |
| Shared Task Priorities | | No | | Yes | No | | Yes |
| Multiple Task Activations | | No | Yes | | No | Yes | |
| | NSL | 704 | 2092 | 2208 | 804 | 2180 | 2390 |
| | NSH | 860 | 2270 | 2384 | 958 | 2348 | 2556 |
| GetTasksetRef | | 40 | 42 | 42 | 40 | 40 | 42 |
| MergeTaskset | | 324 | 326 | 326 | 324 | 324 | 326 |
| AssignTaskset | | 36 | 36 | 36 | 36 | 36 | 36 |
| RemoveTaskset | | 326 | 324 | 324 | 326 | 326 | 324 |
| TestSubTaskset | | 344 | 342 | 342 | 344 | 344 | 342 |
| TestEquivalentTaskset | | 342 | 340 | 340 | 342 | 342 | 340 |
| TickSchedule | SW | 464 | 1896 | 1962 | 526 | 1930 | 2038 |
| | NS | 420 | 1850 | 1916 | 478 | 1882 | 1994 |
| | KL | 148 | 1578 | 1646 | 208 | 1612 | 1722 |
| | SW2 | 466 | 1894 | 1962 | 526 | 1896 | 2010 |
| | NS2 | 420 | 1848 | 1916 | 480 | 1850 | 1966 |
| | KL2 | 150 | 1578 | 1644 | 208 | 1578 | 1694 |
| AdvanceSchedule | SW | 442 | 1864 | 1930 | 494 | 1896 | 2006 |
| | NS | 388 | 1818 | 1884 | 446 | 1852 | 1962 |
| | KL | 122 | 1550 | 1618 | 180 | 1586 | 1696 |
| | SW2 | 442 | 1864 | 1930 | 492 | 1864 | 1980 |
| | NS2 | 388 | 1818 | 1884 | 446 | 1818 | 1932 |
| | KL2 | 122 | 1550 | 1618 | 182 | 1552 | 1666 |
| StartSchedule | | 374 | 374 | 374 | 374 | 374 | 374 |
| StopSchedule | | 348 | 348 | 348 | 348 | 350 | 348 |
| GetScheduleStatus | | 366 | 366 | 366 | 366 | 368 | 366 |
| GetScheduleValue | | 358 | 358 | 358 | 358 | 356 | 358 |
| GetScheduleNext | | 42 | 42 | 42 | 42 | 40 | 42 |
| SetScheduleNext | | 42 | 42 | 42 | 42 | 40 | 42 |
| GetArrivalpointDelay | | 38 | 38 | 38 | 38 | 36 | 38 |
| SetArrivalpointDelay | | 36 | 36 | 36 | 36 | 36 | 36 |
| GetArrivalpointTasksetRef | | 32 | 32 | 32 | 32 | 32 | 32 |
| GetArrivalpointNext | | 36 | 36 | 36 | 36 | 36 | 36 |
| SetArrivalpointNext | | 36 | 36 | 36 | 36 | 36 | 36 |
| TestArrivalpointWritable | | 54 | 54 | 54 | 54 | 54 | 54 |
| GetExecutionTime | | 18 | 18 | 18 | 18 | 20 | 18 |
| GetLargestExecutionTime | | 34 | 34 | 34 | 34 | 34 | 34 |
| ResetLargestExecutionTime | | 30 | 30 | 30 | 30 | 28 | 30 |
| GetStackOffset | | 60 | 60 | 60 | 60 | 62 | 60 |

## Timing

| Configuration | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|
| Events | | No | | | Yes | | |
| Shared Task Priorities | | No | | Yes | No | | Yes |
| Multiple Task Activations | | No | Yes | | No | Yes | |
| Service | Variant | | | | | | |
| ActivateTask | SW | 376 | 432 | 494 | 388 | 416 | 510 |
| | NS | 338 | 394 | 458 | 350 | 378 | 468 |
| | KL | 74 | 128 | 192 | 86 | 114 | 196 |
| TerminateTask | LExt | 0 | 0 | 0 | 0 | 0 | 0 |
| | H | 904 | 904 | 918 | 904 | 904 | 922 |
| ChainTask | SWL | 1246 | 1300 | 1406 | 1344 | 1372 | 1498 |
| | SWH | 1416 | 1476 | 1582 | 1514 | 1540 | 1672 |
| | NSL | 1244 | 1300 | 1406 | 1342 | 1374 | 1496 |
| | NSH | 1398 | 1460 | 1566 | 1496 | 1526 | 1654 |
| Schedule | SW | 350 | 348 | 368 | 348 | 350 | 368 |
| GetTaskID | | 52 | 52 | 52 | 52 | 52 | 52 |
| GetTaskState | | 366 | 366 | 366 | 380 | 378 | 378 |
| EnableAllInterrupts | | 138 | 140 | 138 | 138 | 140 | 140 |
| DisableAllInterrupts | | 186 | 184 | 186 | 186 | 184 | 184 |
| ResumeAllInterrupts | | 154 | 156 | 154 | 154 | 156 | 156 |
| SuspendAllInterrupts | | 208 | 206 | 208 | 208 | 206 | 206 |
| ResumeOSInterrupts | | 154 | 156 | 154 | 154 | 156 | 156 |
| SuspendOSInterrupts | | 208 | 206 | 208 | 208 | 206 | 206 |
| GetResource | Task | 72 | 70 | 76 | 72 | 70 | 74 |
| | Combined | 286 | 286 | 286 | 286 | 286 | 286 |
| | CLEx | n/a | n/a | n/a | n/a | n/a | n/a |
| ReleaseResource | Task | 356 | 356 | 356 | 356 | 356 | 356 |
| | Combined | 650 | 652 | 650 | 650 | 652 | 652 |
| | CLEx | n/a | n/a | n/a | n/a | n/a | n/a |
| SetEvent | SW | n/a | n/a | n/a | 386 | 386 | 388 |
| | NS | n/a | n/a | n/a | 390 | 390 | 390 |
| | KL | n/a | n/a | n/a | 120 | 120 | 120 |
| ClearEvent | | n/a | n/a | n/a | 292 | 292 | 292 |
| GetEvent | | n/a | n/a | n/a | 42 | 42 | 40 |
| WaitEvent | <default> | n/a | n/a | n/a | 1424 | 1422 | 1502 |
| | fp | n/a | n/a | n/a | 1436 | 1436 | 1516 |
| GetAlarmBase | | 336 | 334 | 336 | 336 | 334 | 334 |
| GetAlarm | | 364 | 364 | 362 | 362 | 362 | 364 |

| Configuration | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|
| Events | | No | | | Yes | | |
| Shared Task Priorities | | No | | Yes | No | | Yes |
| Multiple Task Activations | | No | Yes | | No | Yes | |
| SetRelAlarm | | 386 | 386 | 386 | 386 | 386 | 386 |
| SetAbsAlarm | | 374 | 374 | 376 | 376 | 376 | 374 |
| CancelAlarm | | 330 | 328 | 330 | 330 | 328 | 328 |
| InitCounter | | 332 | 332 | 332 | 332 | 332 | 332 |
| GetCounterValue | | 340 | 338 | 340 | 340 | 338 | 338 |
| osek_tick_alarm | <default> | 356 | 356 | 356 | 356 | 356 | 356 |
| | KL | 70 | 70 | 70 | 70 | 70 | 70 |
| osek_incr_counter | | 24 | 24 | 24 | 24 | 24 | 24 |
| GetActiveApplicationMode | | 18 | 18 | 18 | 18 | 18 | 18 |
| StartOS | | 4588 | 4588 | 4588 | 4588 | 4588 | 4588 |
| ShutdownOS | NoHook | n/a | n/a | n/a | n/a | n/a | n/a |
| | Hook | 204 | 204 | 204 | 204 | 204 | 204 |
| InitCOM | | 16 | 18 | 16 | 18 | 16 | 16 |
| CloseCOM | | 18 | 18 | 16 | 16 | 16 | 18 |
| StartCOM | | 64 | 62 | 64 | 182 | 182 | 182 |
| StopCOM | | 32 | 32 | 32 | 32 | 32 | 32 |
| ReadFlag | | n/a | n/a | n/a | 30 | 30 | 30 |
| ResetFlag | | n/a | n/a | n/a | 22 | 22 | 24 |
| ReceiveMessage | | 328 | 328 | 328 | 646 | 646 | 648 |
| GetMessageResource | | n/a | n/a | n/a | 200 | 198 | 198 |
| ReleaseMessageResource | | n/a | n/a | n/a | 466 | 466 | 466 |
| GetMessageStatus | | n/a | n/a | n/a | 72 | 72 | 70 |
| SendMessage | SW | 716 | 770 | 836 | 1050 | 1078 | 1170 |
| | NS | 678 | 734 | 798 | 1012 | 1040 | 1130 |
| | KL | 146 | 198 | 264 | 482 | 508 | 590 |
| ActivateTaskset | SW | 366 | 1760 | 1832 | 382 | 1758 | 1872 |
| | NS | 330 | 1714 | 1788 | 344 | 1712 | 1826 |
| | KL | 64 | 1448 | 1516 | 80 | 1450 | 1564 |
| | SW2 | 366 | 1760 | 1832 | 382 | 1758 | 1872 |
| | NS2 | 328 | 1714 | 1788 | 344 | 1712 | 1828 |
| | KL2 | 64 | 1448 | 1516 | 80 | 1450 | 1564 |
| ChainTaskset | SWL | 1262 | 2644 | 2760 | 1358 | 2736 | 2946 |
| | SWH | 1418 | 2820 | 2936 | 1512 | 2902 | 3112 |
| | NSL | 1260 | 2644 | 2760 | 1360 | 2736 | 2946 |
| | NSH | 1400 | 2804 | 2920 | 1498 | 2888 | 3096 |
| GetTasksetRef | | 38 | 38 | 40 | 40 | 38 | 38 |

| Configuration | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|
| Events | | No | | | Yes | | |
| Shared Task Priorities | | No | | Yes | No | | Yes |
| Multiple Task Activations | | No | Yes | | No | Yes | |
| MergeTaskset | | 322 | 322 | 324 | 324 | 322 | 322 |
| AssignTaskset | | 34 | 34 | 34 | 34 | 34 | 34 |
| RemoveTaskset | | 324 | 324 | 322 | 322 | 324 | 324 |
| TestSubTaskset | | 342 | 342 | 340 | 340 | 342 | 342 |
| TestEquivalentTaskset | | 340 | 340 | 338 | 338 | 340 | 340 |
| TickSchedule | SW | 462 | 1892 | 1960 | 522 | 1928 | 2036 |
| | NS | 418 | 1846 | 1914 | 476 | 1880 | 1990 |
| | KL | 146 | 1576 | 1642 | 206 | 1610 | 1720 |
| | SW2 | 464 | 1892 | 1960 | 522 | 1894 | 2008 |
| | NS2 | 418 | 1848 | 1914 | 478 | 1848 | 1964 |
| | KL2 | 148 | 1576 | 1644 | 206 | 1578 | 1692 |
| AdvanceSchedule | SW | 442 | 1862 | 1928 | 490 | 1896 | 2006 |
| | NS | 386 | 1816 | 1882 | 444 | 1848 | 1960 |
| | KL | 120 | 1548 | 1616 | 180 | 1582 | 1692 |
| | SW2 | 442 | 1860 | 1928 | 492 | 1862 | 1976 |
| | NS2 | 386 | 1816 | 1882 | 444 | 1816 | 1932 |
| | KL2 | 120 | 1550 | 1616 | 178 | 1550 | 1666 |
| StartSchedule | | 372 | 372 | 372 | 372 | 372 | 372 |
| StopSchedule | | 346 | 348 | 346 | 348 | 348 | 348 |
| GetScheduleStatus | | 364 | 366 | 364 | 366 | 366 | 366 |
| GetScheduleValue | | 356 | 354 | 356 | 354 | 354 | 354 |
| GetScheduleNext | | 40 | 38 | 40 | 38 | 38 | 38 |
| SetScheduleNext | | 40 | 38 | 40 | 38 | 38 | 38 |
| GetArrivalpointDelay | | 36 | 34 | 36 | 34 | 34 | 34 |
| SetArrivalpointDelay | | 34 | 34 | 34 | 34 | 34 | 34 |
| GetArrivalpointTasksetRef | | 30 | 30 | 30 | 30 | 30 | 30 |
| GetArrivalpointNext | | 34 | 34 | 34 | 34 | 34 | 34 |
| SetArrivalpointNext | | 34 | 34 | 34 | 34 | 34 | 34 |
| TestArrivalpointWritable | | 52 | 52 | 52 | 52 | 52 | 52 |
| GetExecutionTime | | 388 | 388 | 388 | 388 | 388 | 388 |
| GetLargestExecutionTime | | 46 | 46 | 46 | 46 | 46 | 46 |
| ResetLargestExecutionTime | | 42 | 42 | 42 | 42 | 42 | 42 |
| GetStackOffset | | 58 | 60 | 58 | 60 | 60 | 60 |

## Extended

| Configuration | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|
| Events | | No | | | Yes | | |
| Shared Task Priorities | | No | | Yes | No | | Yes |
| Multiple Task Activations | | No | Yes | | No | Yes | |
| Service | Variant | | | | | | |
| ActivateTask | SW | 706 | 766 | 826 | 722 | 752 | 848 |
| | NS | 760 | 812 | 872 | 774 | 800 | 888 |
| | KL | 410 | 464 | 524 | 424 | 450 | 538 |
| TerminateTask | LExt | 1090 | 1090 | 1104 | 1092 | 1092 | 1108 |
| | H | 1120 | 1120 | 1134 | 1118 | 1118 | 1138 |
| ChainTask | SWL | 1778 | 1832 | 1936 | 1878 | 1898 | 2034 |
| | SWH | 1942 | 1996 | 2100 | 2040 | 2064 | 2198 |
| | NSL | 1856 | 1908 | 2012 | 1954 | 1976 | 2104 |
| | NSH | 2002 | 2060 | 2164 | 2102 | 2126 | 2262 |
| Schedule | SW | 470 | 470 | 488 | 470 | 470 | 488 |
| GetTaskID | | 68 | 68 | 68 | 70 | 70 | 70 |
| GetTaskState | | 732 | 734 | 734 | 736 | 738 | 736 |
| EnableAllInterrupts | | 156 | 156 | 156 | 156 | 156 | 156 |
| DisableAllInterrupts | | 202 | 202 | 202 | 202 | 202 | 202 |
| ResumeAllInterrupts | | 186 | 186 | 186 | 186 | 186 | 186 |
| SuspendAllInterrupts | | 228 | 228 | 228 | 228 | 228 | 228 |
| ResumeOSInterrupts | | 186 | 186 | 186 | 186 | 186 | 186 |
| SuspendOSInterrupts | | 228 | 228 | 228 | 228 | 228 | 228 |
| GetResource | Task | 986 | 986 | 730 | 1054 | 1056 | 800 |
| | Combined | 724 | 724 | 722 | 792 | 792 | 792 |
| | CLEx | 748 | 746 | 748 | 816 | 816 | 816 |
| ReleaseResource | Task | 712 | 712 | 714 | 782 | 782 | 782 |
| | Combined | 962 | 962 | 962 | 1032 | 1030 | 1030 |
| | CLEx | 686 | 686 | 686 | 754 | 754 | 756 |
| SetEvent | SW | n/a | n/a | n/a | 750 | 750 | 750 |
| | NS | n/a | n/a | n/a | 822 | 822 | 822 |
| | KL | n/a | n/a | n/a | 488 | 488 | 488 |
| ClearEvent | | n/a | n/a | n/a | 392 | 392 | 392 |
| GetEvent | | n/a | n/a | n/a | 390 | 390 | 390 |
| WaitEvent | <default> | n/a | n/a | n/a | 1658 | 1658 | 1696 |
| | fp | n/a | n/a | n/a | 1678 | 1678 | 1750 |
| GetAlarmBase | | 590 | 590 | 592 | 590 | 592 | 592 |
| GetAlarm | | 622 | 620 | 622 | 620 | 620 | 622 |

| Configuration | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|
| Events | | No | | | Yes | | |
| Shared Task Priorities | | No | | Yes | No | | Yes |
| Multiple Task Activations | | No | Yes | | No | Yes | |
| SetRelAlarm | | 672 | 672 | 674 | 672 | 674 | 674 |
| SetAbsAlarm | | 648 | 650 | 648 | 650 | 650 | 648 |
| CancelAlarm | | 582 | 582 | 582 | 582 | 582 | 582 |
| InitCounter | | 766 | 768 | 766 | 768 | 768 | 766 |
| GetCounterValue | | 572 | 572 | 572 | 572 | 572 | 572 |
| osek_tick_alarm | <default> | 422 | 422 | 422 | 422 | 422 | 422 |
| | KL | 72 | 72 | 72 | 72 | 72 | 72 |
| osek_incr_counter | | 26 | 26 | 26 | 26 | 26 | 26 |
| GetActiveApplicationMode | | 20 | 20 | 20 | 20 | 20 | 20 |
| StartOS | | 4766 | 4766 | 4766 | 4766 | 4766 | 4766 |
| ShutdownOS | NoHook | n/a | n/a | n/a | n/a | n/a | n/a |
| | Hook | 216 | 216 | 216 | 216 | 216 | 216 |
| InitCOM | | 18 | 18 | 20 | 18 | 20 | 20 |
| CloseCOM | | 18 | 20 | 18 | 20 | 20 | 18 |
| StartCOM | | 94 | 94 | 94 | 208 | 208 | 208 |
| StopCOM | | 64 | 62 | 64 | 62 | 62 | 64 |
| ReadFlag | | n/a | n/a | n/a | 62 | 60 | 60 |
| ResetFlag | | n/a | n/a | n/a | 54 | 54 | 54 |
| ReceiveMessage | | 534 | 532 | 534 | 846 | 846 | 844 |
| GetMessageResource | | n/a | n/a | n/a | 982 | 980 | 980 |
| ReleaseMessageResource | | n/a | n/a | n/a | 1020 | 1020 | 1020 |
| GetMessageStatus | | n/a | n/a | n/a | 206 | 206 | 204 |
| SendMessage | SW | 1240 | 1298 | 1360 | 1576 | 1608 | 1704 |
| | NS | 1292 | 1346 | 1406 | 1630 | 1654 | 1744 |
| | KL | 682 | 738 | 796 | 1010 | 1036 | 1122 |
| ActivateTaskset | SW | 1340 | 1898 | 1958 | 1364 | 1904 | 1992 |
| | NS | 1230 | 1932 | 1998 | 1250 | 1932 | 2040 |
| | KL | 870 | 1576 | 1648 | 900 | 1580 | 1692 |
| | SW2 | 1342 | 1898 | 1960 | 1364 | 1904 | 1992 |
| | NS2 | 1230 | 1932 | 1998 | 1250 | 1932 | 2038 |
| | KL2 | 868 | 1576 | 1646 | 900 | 1582 | 1690 |
| ChainTaskset | SWL | 2422 | 3046 | 3158 | 2534 | 3068 | 3214 |
| | SWH | 2582 | 3204 | 3328 | 2690 | 3230 | 3442 |
| | NSL | 2488 | 3070 | 3180 | 2598 | 3154 | 3300 |
| | NSH | 2634 | 3218 | 3334 | 2748 | 3302 | 3448 |
| GetTasksetRef | | 352 | 352 | 352 | 352 | 352 | 352 |

| Configuration | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|
| Events | | No | | | Yes | | |
| Shared Task Priorities | | No | | Yes | No | | Yes |
| Multiple Task Activations | | No | Yes | | No | Yes | |
| MergeTaskset | | 476 | 476 | 476 | 476 | 476 | 476 |
| AssignTaskset | | 142 | 142 | 142 | 142 | 142 | 142 |
| RemoveTaskset | | 476 | 476 | 476 | 476 | 476 | 476 |
| TestSubTaskset | | 496 | 496 | 496 | 496 | 496 | 496 |
| TestEquivalentTaskset | | 494 | 494 | 494 | 494 | 494 | 494 |
| TickSchedule | SW | 612 | 2100 | 2170 | 1422 | 2134 | 2240 |
| | NS | 650 | 2162 | 2234 | 1486 | 2196 | 2302 |
| | KL | 290 | 1794 | 1868 | 1118 | 1830 | 1936 |
| | SW2 | 612 | 2098 | 2172 | 1424 | 2104 | 2214 |
| | NS2 | 650 | 2162 | 2234 | 1486 | 2166 | 2278 |
| | KL2 | 292 | 1796 | 1866 | 1120 | 1800 | 1910 |
| AdvanceSchedule | SW | 594 | 2086 | 2160 | 1410 | 2122 | 2226 |
| | NS | 640 | 2140 | 2214 | 1464 | 2174 | 2280 |
| | KL | 274 | 1778 | 1852 | 1104 | 1812 | 1920 |
| | SW2 | 594 | 2088 | 2158 | 1412 | 2092 | 2204 |
| | NS2 | 640 | 2142 | 2212 | 1466 | 2146 | 2258 |
| | KL2 | 274 | 1780 | 1850 | 1102 | 1784 | 1894 |
| StartSchedule | | 500 | 500 | 500 | 500 | 500 | 500 |
| StopSchedule | | 442 | 442 | 442 | 442 | 442 | 442 |
| GetScheduleStatus | | 470 | 470 | 472 | 472 | 472 | 470 |
| GetScheduleValue | | 462 | 462 | 460 | 460 | 460 | 462 |
| GetScheduleNext | | 94 | 94 | 94 | 94 | 94 | 94 |
| SetScheduleNext | | 144 | 144 | 146 | 146 | 146 | 144 |
| GetArrivalpointDelay | | 108 | 108 | 110 | 110 | 110 | 108 |
| SetArrivalpointDelay | | 126 | 126 | 126 | 126 | 126 | 126 |
| GetArrivalpointTasksetRef | | 94 | 94 | 96 | 96 | 96 | 94 |
| GetArrivalpointNext | | 98 | 98 | 100 | 100 | 100 | 98 |
| SetArrivalpointNext | | 150 | 150 | 150 | 150 | 150 | 150 |
| TestArrivalpointWritable | | 108 | 108 | 106 | 106 | 106 | 108 |
| GetExecutionTime | | 460 | 460 | 460 | 460 | 460 | 460 |
| GetLargestExecutionTime | | 330 | 330 | 330 | 330 | 330 | 330 |
| ResetLargestExecutionTime | | 322 | 322 | 322 | 322 | 322 | 322 |
| GetStackOffset | | 60 | 60 | 60 | 60 | 60 | 60 |

### 4.3.2 OS Start-up Time

OS start-up time is the time from the entry to the `StartOS()` function to the execution of the first instruction in a user task (including the idle task) without any hook routines being called. This time is always application dependent, since `StartOS()` may activate any number of tasks and start any number of user-specified alarms.

### 4.3.3 Interrupt Latencies

Interrupt latency is the time between an interrupt request being recognized by the target hardware and the execution of the first instruction of the user provided handler function. The following tables give the interrupt latencies (in CPU cycles).

## Standard

| Configuration | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|
| Events | | No | | | Yes | | |
| Shared Task Priorities | | No | | Yes | No | | Yes |
| Multiple Task Activations | | No | Yes | | No | Yes | |
| Operation | ISR Category | | | | | | |
| ISR Latency | Cat 1 | 90 | 90 | 90 | 90 | 90 | 90 |
| | Cat 2 | 214 | 252 | 252 | 252 | 252 | 254 |

## Timing

| Configuration | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|
| Events | | No | | | Yes | | |
| Shared Task Priorities | | No | | Yes | No | | Yes |
| Multiple Task Activations | | No | Yes | | No | Yes | |
| Operation | ISR Category | | | | | | |
| ISR Latency | Cat 1 | 90 | 90 | 90 | 90 | 90 | 90 |
| | Cat 2 | 624 | 652 | 652 | 652 | 652 | 652 |

## Extended

| Configuration | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|
| Events | | No | | | Yes | | |
| Shared Task Priorities | | No | | Yes | No | | Yes |
| Multiple Task Activations | | No | Yes | | No | Yes | |
| Operation | ISR Category | | | | | | |
| ISR Latency | Cat 1 | 90 | 90 | 90 | 90 | 90 | 90 |
| | Cat 2 | 626 | 654 | 654 | 652 | 652 | 652 |

### 4.3.4 Task Switching Times

Task switching time is the time between the last instruction of the previous task and the first instruction of the next task. The switching time differs, depending on the switching contexts (e.g. an `ActivateTask()` versus a `ChainTask()`).

RTA-OSEK sub-task types also affect the switching time. The tables in this section show the switching times (in CPU cycles) for all system classes for basic, lightweight tasks and for basic and extended heavyweight tasks.

Figures 1 to 8 show the RTA-OSEK switching contexts measured.

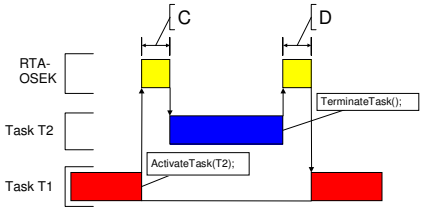


**Figure 1: Task Activates a Higher Priority Task which Terminates Normally**

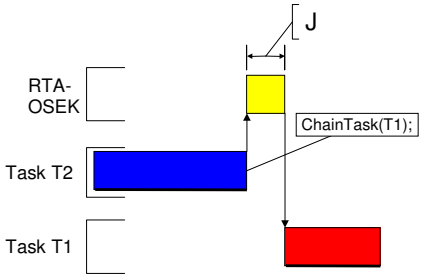


**Figure 2: Task Chaining**

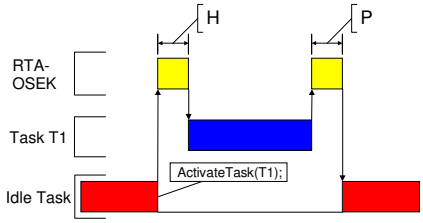


**Figure 3: Task Activation from Idle Task**

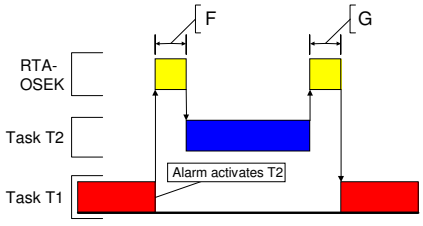


**Figure 4: Task Activation from an Alarm**

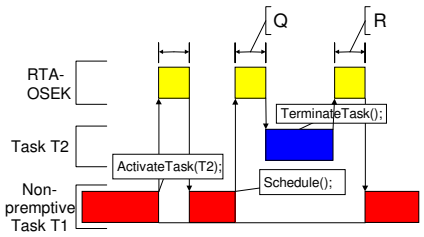


**Figure 5: Non-Premptive Task Calls Schedule()**

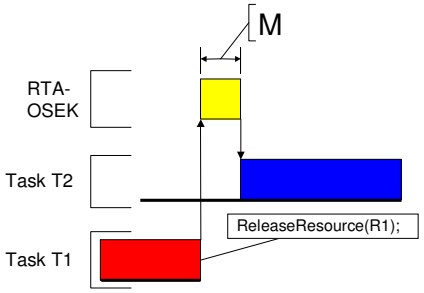


**Figure 6: Blocked Task Activated by ReleaseResource()**

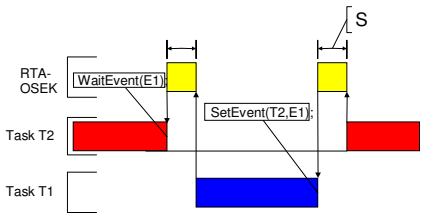


**Figure 7: Waiting Task Activated by SetEvent()**

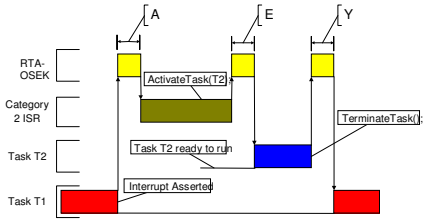


**Figure 8: Category 2 ISR Activates a Higher Priority Task**

## Standard

| Configuration | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|
| Events | | No | | | Yes | | |
| Shared Task Priorities | | No | | Yes | No | | Yes |
| Multiple Task Activations | Task Attributes | No | Yes | | No | Yes | |
| Normal termination | Light, Basic | 312 | 478 | 524 | 312 | 478 | 528 |
| Figure 1: D | Heavy, Basic/Extended | 398 | 548 | 598 | 562 | 560 | 608 |
| ChainTask | Light, Basic | 598 | 686 | 792 | 610 | 684 | 812 |
| Figure 2: J | Heavy, Basic/Extended | 1228 | 1472 | 1628 | 1402 | 1484 | 1660 |
| Pre-emption | Light, Basic | 518 | 604 | 734 | 528 | 604 | 762 |
| Figure 1: C | Heavy, Basic/Extended | 608 | 670 | 796 | 706 | 736 | 890 |
| From idle task | Light, Basic | 518 | 604 | 734 | 528 | 604 | 762 |
| Figure 3: H | Heavy, Basic/Extended | 608 | 670 | 796 | 706 | 736 | 888 |
| Triggered by alarm | Light, Basic | 884 | 972 | 1102 | 896 | 972 | 1130 |
| Figure 4: F | Heavy, Basic/Extended | 974 | 1038 | 1164 | 1074 | 1104 | 1256 |
| Schedule | Light, Basic | 486 | 518 | 602 | 486 | 518 | 602 |
| Figure 5: Q | Heavy, Basic/Extended | 576 | 584 | 664 | 664 | 664 | 746 |
| Release resource | Light, Basic | 542 | 574 | 640 | 542 | 574 | 640 |
| Figure 6: M | Heavy, Basic/Extended | 632 | 640 | 702 | 720 | 720 | 784 |
| SetEvent | | | | | | | |

| Configuration | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|
| Events | | No | | | Yes | | |
| Shared Task Priorities | | No | | Yes | No | | Yes |
| Multiple Task Activations | Task Attributes | No | Yes | | No | Yes | |
| Figure 7: S | Heavy, Extended | n/a | n/a | n/a | 1110 | 1108 | 1272 |
| From category 2 ISR | Light, Basic | 386 | 460 | 526 | 430 | 460 | 528 |
| Figure 8: E | Heavy, Basic/Extended | 478 | 526 | 588 | 606 | 606 | 670 |

## Timing

| Configuration | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|
| Events | | No | | | Yes | | |
| Shared Task Priorities | | No | | Yes | No | | Yes |
| Multiple Task Activations | Task Attributes | No | Yes | | No | Yes | |
| Normal termination | Light, Basic | 832 | 888 | 934 | 832 | 888 | 938 |
| Figure 1: D | Heavy, Basic/Extended | 904 | 936 | 986 | 956 | 956 | 1004 |
| ChainTask | Light, Basic | 1170 | 1236 | 1344 | 1182 | 1238 | 1364 |
| Figure 2: J | Heavy, Basic/Extended | 2300 | 2406 | 2564 | 2360 | 2422 | 2598 |
| Pre-emption | Light, Basic | 710 | 776 | 906 | 720 | 776 | 934 |
| Figure 1: C | Heavy, Basic/Extended | 786 | 840 | 968 | 882 | 912 | 1066 |
| From idle task | Light, Basic | 710 | 776 | 906 | 720 | 776 | 934 |
| Figure 3: H | Heavy, Basic/Extended | 784 | 840 | 968 | 882 | 912 | 1064 |
| Triggered by alarm | Light, Basic | 1078 | 1146 | 1276 | 1088 | 1144 | 1302 |
| Figure 4: F | Heavy, Basic/Extended | 1154 | 1210 | 1338 | 1250 | 1280 | 1432 |
| Schedule | Light, Basic | 678 | 690 | 774 | 678 | 690 | 774 |
| Figure 5: Q | Heavy, Basic/Extended | 754 | 754 | 836 | 840 | 840 | 922 |
| Release resource | Light, Basic | 740 | 752 | 818 | 740 | 752 | 818 |
| Figure 6: M | Heavy, Basic/Extended | 814 | 816 | 880 | 902 | 900 | 964 |
| SetEvent | | | | | | | |
| Figure 7: S | Heavy, Extended | n/a | n/a | n/a | 1134 | 1136 | 1300 |
| From category 2 ISR | Light, Basic | 1172 | 1212 | 1278 | 1200 | 1212 | 1278 |
| Figure 8: E | Heavy, Basic/Extended | 1248 | 1276 | 1340 | 1362 | 1362 | 1426 |

### Extended

| Configuration | | Application Uses | | | | | |
|---|---|---|---|---|---|---|---|
| Events | | No | | | Yes | | |
| Shared Task Priorities | | No | | Yes | No | | Yes |
| Multiple Task Activations | Task Attributes | No | Yes | | No | Yes | |
| Normal termination | Light, Basic | 1086 | 1142 | 1188 | 1086 | 1142 | 1192 |
| Figure 1: D | Heavy, Basic/Extended | 1118 | 1152 | 1202 | 1172 | 1170 | 1218 |
| ChainTask | Light, Basic | 1704 | 1768 | 1874 | 1716 | 1768 | 1900 |
| Figure 2: J | Heavy, Basic/Extended | 3040 | 3140 | 3296 | 3106 | 3160 | 3338 |
| Pre-emption | Light, Basic | 1024 | 1096 | 1222 | 1040 | 1096 | 1252 |
| Figure 1: C | Heavy, Basic/Extended | 1100 | 1160 | 1284 | 1202 | 1234 | 1388 |
| From idle task | Light, Basic | 1024 | 1096 | 1222 | 1040 | 1096 | 1252 |
| Figure 3: H | Heavy, Basic/Extended | 1100 | 1160 | 1284 | 1202 | 1232 | 1388 |
| Triggered by alarm | Light, Basic | 1458 | 1528 | 1654 | 1474 | 1530 | 1686 |
| Figure 4: F | Heavy, Basic/Extended | 1534 | 1592 | 1716 | 1632 | 1664 | 1820 |
| Schedule | Light, Basic | 784 | 796 | 880 | 784 | 794 | 878 |
| Figure 5: Q | Heavy, Basic/Extended | 858 | 860 | 942 | 946 | 946 | 1028 |
| Release resource | Light, Basic | 1070 | 1082 | 1148 | 1140 | 1152 | 1218 |
| Figure 6: M | Heavy, Basic/Extended | 1146 | 1146 | 1210 | 1302 | 1302 | 1366 |
| SetEvent | | | | | | | |
| Figure 7: S | Heavy, Extended | n/a | n/a | n/a | 1524 | 1524 | 1648 |
| From category 2 ISR | Light, Basic | 1244 | 1284 | 1350 | 1272 | 1284 | 1350 |
| Figure 8: E | Heavy, Basic/Extended | 1320 | 1348 | 1412 | 1434 | 1434 | 1498 |

## 4.4 Configuration of Run-time Context

The run-time contexts of all tasks reside on the same stack and are recovered when the task terminates. As a result, run-time contexts of mutually exclusive tasks are effectively overlaid. The RTA-OSEK GUI is able to calculate the worst-case stack requirement for the entire application, based on the declared stack usage, the priorities and the resource occupation of individual tasks.

The size of the run-time context of a task depends on the task type and the system configuration. The following tables give the sizes (in bytes) for different OS status and configurations:

## Standard

| Configuration | Application Uses | | | | | |
|---|---|---|---|---|---|---|
| Events | No | | | Yes | | |
| Shared Task Priorities | No | | Yes | No | | Yes |
| Multiple Task Activations | No | Yes | | No | Yes | |
| **Pre- and Post-Task hooks not used** | | | | | | |
| Task type | | | | | | |
| BCC1 lightweight, integer | 88 | 88 | 96 | 88 | 88 | 96 |
| BCC1 lightweight, floating-point | 96 | 96 | 104 | 96 | 96 | 104 |
| BCC1 heavyweight, integer | 144 | 144 | 152 | 144 | 144 | 152 |
| BCC1 heavyweight, floating-point | 144 | 144 | 152 | 144 | 144 | 152 |
| BCC2 lightweight, integer | n/a | 96 | 112 | n/a | 96 | 112 |
| BCC2 lightweight, floating-point | n/a | 96 | 112 | n/a | 96 | 112 |
| BCC2 heavyweight, integer | n/a | 152 | 152 | n/a | 152 | 152 |
| BCC2 heavyweight, floating-point | n/a | 152 | 152 | n/a | 152 | 152 |
| ECC1 heavyweight, integer | n/a | n/a | n/a | 180 | 180 | 188 |
| ECC1 heavyweight, floating-point | n/a | n/a | n/a | 180 | 180 | 188 |
| ECC2 heavyweight, integer | n/a | n/a | n/a | n/a | n/a | 188 |
| ECC2 heavyweight, floating-point | n/a | n/a | n/a | n/a | n/a | 188 |
| | | | | | | |
| **Pre- and/or Post-Task hooks used** | | | | | | |
| Task type | | | | | | |
| BCC1 lightweight, integer | 104 | 104 | 96 | 104 | 104 | 96 |
| BCC1 lightweight, floating-point | 112 | 112 | 104 | 112 | 112 | 104 |
| BCC1 heavyweight, integer | 160 | 160 | 152 | 160 | 160 | 152 |
| BCC1 heavyweight, floating-point | 160 | 160 | 152 | 160 | 160 | 152 |
| BCC2 lightweight, integer | n/a | 112 | 112 | n/a | 112 | 112 |
| BCC2 lightweight, floating-point | n/a | 112 | 112 | n/a | 112 | 112 |
| BCC2 heavyweight, integer | n/a | 168 | 152 | n/a | 168 | 152 |
| BCC2 heavyweight, floating-point | n/a | 168 | 152 | n/a | 168 | 152 |
| ECC1 heavyweight, integer | n/a | n/a | n/a | 196 | 196 | 188 |
| ECC1 heavyweight, floating-point | n/a | n/a | n/a | 196 | 196 | 188 |
| ECC2 heavyweight, integer | n/a | n/a | n/a | n/a | n/a | 188 |
| ECC2 heavyweight, floating-point | n/a | n/a | n/a | n/a | n/a | 188 |

## Timing

| Configuration | Application Uses | | | | | |
|---|---|---|---|---|---|---|
| Events | No | | | Yes | | |
| Shared Task Priorities | No | | Yes | No | | Yes |
| Multiple Task Activations | No | Yes | | No | Yes | |
| **Pre- and Post-Task hooks not used** | | | | | | |
| Task type | | | | | | |
| BCC1 lightweight, integer | 112 | 112 | 120 | 112 | 112 | 120 |
| BCC1 lightweight, floating-point | 120 | 120 | 128 | 120 | 120 | 128 |
| BCC1 heavyweight, integer | 168 | 168 | 176 | 168 | 168 | 176 |
| BCC1 heavyweight, floating-point | 168 | 168 | 176 | 168 | 168 | 176 |
| BCC2 lightweight, integer | n/a | 120 | 136 | n/a | 120 | 136 |
| BCC2 lightweight, floating-point | n/a | 120 | 136 | n/a | 120 | 136 |
| BCC2 heavyweight, integer | n/a | 168 | 176 | n/a | 168 | 176 |
| BCC2 heavyweight, floating-point | n/a | 168 | 176 | n/a | 168 | 176 |
| ECC1 heavyweight, integer | n/a | n/a | n/a | 208 | 208 | 216 |
| ECC1 heavyweight, floating-point | n/a | n/a | n/a | 208 | 208 | 216 |
| ECC2 heavyweight, integer | n/a | n/a | n/a | n/a | n/a | 216 |
| ECC2 heavyweight, floating-point | n/a | n/a | n/a | n/a | n/a | 216 |
| | | | | | | |
| **Pre- and/or Post-Task hooks used** | | | | | | |
| Task type | | | | | | |
| BCC1 lightweight, integer | 120 | 120 | 120 | 120 | 120 | 120 |
| BCC1 lightweight, floating-point | 128 | 128 | 128 | 128 | 128 | 128 |
| BCC1 heavyweight, integer | 176 | 176 | 176 | 176 | 176 | 176 |
| BCC1 heavyweight, floating-point | 176 | 176 | 176 | 176 | 176 | 176 |
| BCC2 lightweight, integer | n/a | 128 | 136 | n/a | 128 | 136 |
| BCC2 lightweight, floating-point | n/a | 128 | 136 | n/a | 128 | 136 |
| BCC2 heavyweight, integer | n/a | 176 | 176 | n/a | 176 | 176 |
| BCC2 heavyweight, floating-point | n/a | 176 | 176 | n/a | 176 | 176 |
| ECC1 heavyweight, integer | n/a | n/a | n/a | 216 | 216 | 216 |
| ECC1 heavyweight, floating-point | n/a | n/a | n/a | 216 | 216 | 216 |
| ECC2 heavyweight, integer | n/a | n/a | n/a | n/a | n/a | 216 |
| ECC2 heavyweight, floating-point | n/a | n/a | n/a | n/a | n/a | 216 |

## Extended

| Configuration / Events / Shared Task Priorities / Multiple Task Activations | Application Uses | | | | | |
|---|---|---|---|---|---|---|
| | No | | | Yes | | |
| | No | | Yes | No | | Yes |
| | No | Yes | | No | Yes | |
| **Pre- and Post-Task hooks not used** | | | | | | |
| Task type | | | | | | |
| BCC1 lightweight, integer | 112 | 112 | 120 | 112 | 112 | 120 |
| BCC1 lightweight, floating-point | 120 | 120 | 128 | 120 | 120 | 128 |
| BCC1 heavyweight, integer | 168 | 168 | 176 | 168 | 168 | 176 |
| BCC1 heavyweight, floating-point | 168 | 168 | 176 | 168 | 168 | 176 |
| BCC2 lightweight, integer | n/a | 120 | 136 | n/a | 120 | 136 |
| BCC2 lightweight, floating-point | n/a | 120 | 136 | n/a | 120 | 136 |
| BCC2 heavyweight, integer | n/a | 168 | 176 | n/a | 168 | 176 |
| BCC2 heavyweight, floating-point | n/a | 168 | 176 | n/a | 168 | 176 |
| ECC1 heavyweight, integer | n/a | n/a | n/a | 216 | 216 | 224 |
| ECC1 heavyweight, floating-point | n/a | n/a | n/a | 216 | 216 | 224 |
| ECC2 heavyweight, integer | n/a | n/a | n/a | n/a | n/a | 224 |
| ECC2 heavyweight, floating-point | n/a | n/a | n/a | n/a | n/a | 224 |
| | | | | | | |
| **Pre- and/or Post-Task hooks used** | | | | | | |
| Task type | | | | | | |
| BCC1 lightweight, integer | 120 | 120 | 120 | 120 | 120 | 120 |
| BCC1 lightweight, floating-point | 128 | 128 | 128 | 128 | 128 | 128 |
| BCC1 heavyweight, integer | 176 | 176 | 176 | 176 | 176 | 176 |
| BCC1 heavyweight, floating-point | 176 | 176 | 176 | 176 | 176 | 176 |
| BCC2 lightweight, integer | n/a | 128 | 136 | n/a | 128 | 136 |
| BCC2 lightweight, floating-point | n/a | 128 | 136 | n/a | 128 | 136 |
| BCC2 heavyweight, integer | n/a | 176 | 176 | n/a | 176 | 176 |
| BCC2 heavyweight, floating-point | n/a | 176 | 176 | n/a | 176 | 176 |
| ECC1 heavyweight, integer | n/a | n/a | n/a | 224 | 224 | 224 |
| ECC1 heavyweight, floating-point | n/a | n/a | n/a | 224 | 224 | 224 |
| ECC2 heavyweight, integer | n/a | n/a | n/a | n/a | n/a | 224 |
| ECC2 heavyweight, floating-point | n/a | n/a | n/a | n/a | n/a | 224 |

# 5 Compatibility with Pre-v3.26 Kernels

## 5.1 Updating the Application Version

To convert an existing OIL configuration file that uses the v3.20 or v3.25 kernel libraries to use the v3.26 kernel libraries, load the file into the RTA-OSEK GUI, select the 'OS Configuration' option in the 'Application' menu and change the 'Kernel Version' to v3.26. When the OIL configuration file is saved it will then use the v3.26 kernel libraries. This process can be reversed to move back to earlier kernel versions.

## 5.2 ARM ABI

In the move from v3.20 to v3.25 the ABI used by the compiler when generating code is changed. Pre v3.25 versions use the TI ABI which is not compatible with the ARM ABI used in later versions. The TI code generation tools prevent applications being created that mix the two ABIs. The main differences between the ABIs are as follows:

- The stack is always aligned on a 64-bit boundary; this is maintained throughout an application.
- The linker creates veneers automatically
- Assembler functions are no longer pre-pended with the '&' and '_' characters to denote the instruction set used
- The assembler directives have been modified
- The linker now generates the addresses for 16-bit instruction set functions automatically with the +1 to assist the instruction set change with the BX instruction
- The register use for passing arguments into functions has been modified to support 64-bit variables
- The output format is now ELF rather than COFF

The ARM ABI and TI code generation tools documentation should be referenced for a detailed description for the differences between the ABIs.

# Support

For product support, please contact your local ETAS representative.

Office locations and contact details can be found at the front of this manual and on the ETAS Group website www.etasgroup.com.