RTA-OSEK

Binding Manual: Hitachi/H8SX

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.
3F, Samseung Bldg. 61-1
Yangjae-dong, Seocho-gu
Seoul

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: RM00062-002

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 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 port specific information for the Hitachi/H8SX implementation of LiveDevices' RTA-OSEK.

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?

It is assumed that you are a developer. 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.

In this guide you'll see that program code, header file names, C type names, C functions 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

In this chapter, you'll see the important details that you need to know about RTA-OSEK and your toolchain. A port of the RTA-OSEK Component is specific to both the target hardware *and* the compiler toolchain. You must make sure that you build your application with this toolchain.

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

2.1 Memory Model

RTA-OSEK operates in the "Advanced" memory mode with 24-bit address space for code and data.

2.2 Compiler

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

Vendor	Renesas
Compiler	H8S, H8/300 SERIES C/C++ Compiler
Version	6.0.00.005

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

Option	Description
`-cpu=H8SXA:24:MD`	H8SX CPU core operating in advanced memory mode with 24 bit address space using the multiplier and the divider
`-op=1`	Optimize for size
`-C=A`	Compile via assembler. This is necessary to compile the FP context save and restores functions in osfptgt.h and osfptgt.c.
`-m`	Show all messages

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

Option	Description
`-REGP=3`	Change number of parameter registers from 2 to 3.

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
-cpu=H8SXA:24:MD	H8SX CPU core operating in advanced memory mode with 24 bit address space using the multiplier and the divider
-op=1	Optimize for size
-C=A	Compile via assembler.
-m	Show all messages

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

Option	Description
-REGP=3	Change number of parameter registers from 2 to 3.

The Renesas compiler used to build RTA-OSEK is included with the Renesas High-performance Embedded Workshop 3, which includes v6.0 of the Renesas H8 Tools.

The Renesas compiler assumes that the stack does not span a 64Kb boundary. Therefore care must be taken within the linker control file to ensure that the entire stack segment ("S") is located within a 64Kb 'page'.

The Renesas compiler includes an option to expand the number of registers used for parameter passing from 2 to 3. This option must *not* be used when compiling RTA-OSEK applications.

2.3 Assembler

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

Vendor	Renesas
Assembler	H8S, H8/300 SERIES CROSS ASSEMBLER
Version	6.0.01.005

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

Option	Description
-cpu=H8SXA:24:MD	H8SX CPU core operating in advanced memory mode with 24 bit address space using the multiplier and the divider
-op	Optimize (branch displacement, index displacement, absolute address width)

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

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

Option	Description
-cpu=H8SXA:24:MD	H8SX CPU core operating in advanced memory mode with 24 bit address space using the multiplier and the divider
-op	Optimize (branch displacement, index displacement, absolute address width)

2.4 Linker/Locator

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_intvec	ROM	Vector table if generated by RTA-OSEK GUI
os_pir	RAM	RTA-OSEK initialized data
os_pur	RAM	RTA-OSEK uninitialized data

The following compiler run-time library functions are required by the RTA-OSEK Component:

C Library Functions	Description
MVN3	C Structure copy
setjmp	Renesas library setjmp routine
longjmp	Renesas library longjmp routine

Some RTA-OSEK API calls require functions from the standard C library and so applications must be linked with it.

The Renesas C compiler does *not* ship with a precompiled C library. Instead the Renesas standard library generator must be used to create a custom C library containing the required functions. RTA-OSEK only requires functions from the RUNTIME section.

2.5 Debugger

Information about ORTI for RTA-OSEK can be found in the *RTA-OSEK ORTI Guide*

At the time of writing, we were not aware of any debuggers for the Renesas H8SX with support for ORTI.

If you are using an ORTI version 2.0 aware debugger on this platform you can use the "Unknown ORTI debugger" option in the RTA-OSEK GUI to generate an ORTI output file. The ORTI generated will not have been tested on the debugger and, therefore, is not guaranteed to work.

Please contact LiveDevices if you have any questions about ORTI support in RTA-OSEK.

3 Target Hardware Issues

3.1 Interrupts

This section explains the implementation of RTA-OSEK's interrupt model. 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 *H8SX/1520 Group Hardware Manual*.

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

IPL Value	EXR I0-I2 Bits	Description
0	0	User level
1-7	1-7	Category 1 and 2 interrupts
8	Not Applicable	NMI, General illegal instructions, Trace, Trap Instructions, CPU address error and DMA address error only (all Category 1).

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
0x0	The reset vector is outside the control of RTA-OSEK
0x4-0xFC	Category 1 interrupts at an IPL of 8
0x100-0x3FC	Category 1 and 2 interrupts at IPL 1-7

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 Renesas C compiler can generate appropriate interrupt handling code for a C function

decorated with the `__interrupt` 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.src`.

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 (VVVV represents the 4 hex digit, upper-case, zero-padded value of the vector location).

Vector Location	Label
0xVVVV	`_os_wrapper_VVVV`
eg : 0x01E0	`_os_wrapper_01E0`

The build process will generate a vector table covering all Category 1 and 2 ISRs with the exception of the reset vector at address offset 0x0. The generated vector table will cover the range 0x4 to 0x3fc inclusive.

The reset vector is not within the domain of RTA-OSEK and must be programmed by the user. The entry point for the "Power ON" reset should be marked within the application code as follows:

```
#pragma entry PowerON_Reset
```

Defining the "Power ON" reset function in this manner ensures that the C compiler generates code to set the stack pointer as the first instruction in the function. The same effect can also be achieved using the `__entry` qualifier.

The reset vector must be created within the application, for example:

```
#pragma section RESETVEC
typedef void (*interrupt_vector)(void);
```

```
const interrupt_vector reset_vector_table[] =
{PowerON_Reset};
```

The linker control file must ensure that the section (in this case `CRESETVEC`) is located at 0x0.

Important: When creating constant data sections in C, the Renesas C compiler automatically prefixes all section names with "C". Therefore the linker must locate the section `CRESETVEC` at address 0x0 rather than `RESETVEC`.

3.1.6 Interrupt Priority Registers

The H8SX has a set of interrupt priority registers (IPRs) used to specify the priority at which processor interrupts operate.

To permit user control over enabling and disabling of hardware interrupts, the RTA-OSEK Component does not initialize the peripheral interrupt enable registers, or the IPRs.

Important: The correct initialization of an interrupt source and the associated interrupt priority level in the IPRs is left as the responsibility of the user. Each IPR must be programmed with the interrupt priority level specified in the RTA-OSEK GUI.

As the interrupt priorities for Category 1 and 2 interrupts are specified in the RTA-OSEK GUI, RTA-OSEK generates the values for the IPR registers. The values are made available to user code, to be copied to the registers during system initialization, as a block of 18 values based at global symbol `os_ipr_values`.

Portability: The number of IPRs may vary according to the H8SX processor. The RTA-OSEK GUI always generates IPRs A to R inclusive, even though not all processors in the H8SX series may support them. Care must be taken when programming IPRs that the correct subset for the processor is used – refer to the Interrupt Controller Section of the *Renesas Hardware Manual* for the relevant H8SX processor for a list of valid IPRs and the vector numbers they represent.

Care must be taken when setting the priorities of vectors that share the same bits in an IPR. The RTA-OSEK GUI checks that vector priorities are consistent during the "Build checks" which precede building an application.

3.2 Register Settings

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

Register	Notes
Interrupt Control Register (INTCR)	Set INTM1 and INTM0 bits to 10 to enable interrupt mode 2
Vector Base Register (VBR)	Interrupt vector base address
Interrupt Priority Registers (IPRx)	Interrupt Priority Registers need to be set for all interrupt sources that use them. RTA-OSEK generates suitable values in os_ipr_values

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

Register	Notes
Extended Control Register (EXR)	The Interrupt Mask Bits I0-I2 are reserved by RTA-OSEK

3.3 Stack Usage

3.3.1 Number of Stacks

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

`StackOffsetType` is a scalar, representing the number of bytes on the stack, with C type: `UInt16Type`

3.3.2 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): 60

Timing

API max usage (bytes): 60

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 vendor to find out more about library call stack usage.

The stack pointer on the H8SX must be kept even at all times.

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	**H8SX/1527**
Clock speed (MHz)	10
Code memory	On-chip FLASH
Read-only data memory	On-chip FLASH
Read-write data memory	On-chip 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	16	16	16
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
Limits for the number of application modes	65535					

Configuration	Application Uses					
Events	No			Yes		
Shared Task Priorities	No		Yes	No		Yes
Multiple Task Activations	No	Yes		No	Yes	
(per system)						

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	24	24	24	24	24	24
	ROM	153	153	153	153	153	153
COM overhead	RAM	4	4	4	4	4	4
	ROM	10	10	10	10	10	10

Timing

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
OS overhead	RAM	36	36	36	36	36	36
	ROM	193	193	193	193	193	193
COM overhead	RAM	4	4	4	4	4	4
	ROM	10	10	10	10	10	10

Extended

Configuration / Events / Shared Task Priorities / Multiple Task Activations		Application Uses					
		No			Yes		
		No		Yes	No		Yes
		No	Yes		No	Yes	
OS overhead	RAM	47	47	47	47	47	47
	ROM	232	232	232	232	232	232
COM overhead	RAM	4	4	4	4	4	4
	ROM	10	10	10	10	10	10

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	34	34	34	34	34	34
BCC1 Heavyweight task	RAM	4	4	4	4	4	4
	ROM	38	38	38	38	38	38
BCC2 task	RAM	n/a	6	8	n/a	6	8
	ROM	n/a	40	48	n/a	40	48
ECC1, Integer task	RAM	n/a	n/a	n/a	40	40	40
	ROM	n/a	n/a	n/a	54	54	54
ECC1, floating-point task	RAM	n/a	n/a	n/a	48	48	48
	ROM	n/a	n/a	n/a	54	54	54
ECC2, Integer task	RAM	n/a	n/a	n/a	n/a	n/a	42
	ROM	n/a	n/a	n/a	n/a	n/a	62
ECC2, floating-point task	RAM	n/a	n/a	n/a	n/a	n/a	50
	ROM	n/a	n/a	n/a	n/a	n/a	62
Category 2 ISR	RAM	0	0	0	0	0	0
	ROM	58	58	58	58	58	58
Category 2 ISR, floating-point	RAM	8	8	8	8	8	8
	ROM	70	70	70	70	70	70
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	6	6	6	6	6	6
	ROM	38	38	38	38	38	38
Counter	RAM	2	2	2	2	2	2
	ROM	40	40	40	40	40	40
Message	RAM	11	11	11	31	31	31
	ROM	18	18	18	48	48	48
Flag	RAM	2	2	2	2	2	2
	ROM	2	2	2	2	2	2
Message resource	RAM	0	0	0	0	0	0
	ROM	20	20	20	20	20	20

Configuration / Events / Shared Task Priorities / Multiple Task Activations		Application Uses					
		No			Yes		
		No		Yes	No		Yes
		No	Yes		No	Yes	
Event	RAM	0	0	0	0	0	0
	ROM	2	2	2	2	2	2
Priority level	RAM	0	0	6	0	6	6
	ROM	0	0	10	0	10	10
Arrivalpoint (readonly)	RAM	0	0	0	0	0	0
	ROM	10	10	10	10	10	10
Arrivalpoint (writable)	RAM	10	10	10	10	10	10
	ROM	10	10	10	10	10	10
Schedule	RAM	10	10	10	10	10	10
	ROM	32	32	32	32	32	32
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 / Events / Shared Task Priorities / Multiple Task Activations		Application Uses					
		No			Yes		
		No		Yes	No		Yes
		No	Yes		No	Yes	
BCC1 Lightweight task	RAM	6	6	6	6	6	6
	ROM	42	42	42	42	42	42
BCC1 Heavyweight task	RAM	10	10	10	10	10	10
	ROM	46	46	46	46	46	46
BCC2 task	RAM	n/a	12	14	n/a	12	14
	ROM	n/a	48	56	n/a	48	56
ECC1, Integer task	RAM	n/a	n/a	n/a	46	46	46
	ROM	n/a	n/a	n/a	62	62	62
ECC1, floating-point task	RAM	n/a	n/a	n/a	54	54	54
	ROM	n/a	n/a	n/a	62	62	62
ECC2, Integer task	RAM	n/a	n/a	n/a	n/a	n/a	48
	ROM	n/a	n/a	n/a	n/a	n/a	70
ECC2, floating-point task	RAM	n/a	n/a	n/a	n/a	n/a	56
	ROM	n/a	n/a	n/a	n/a	n/a	70

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
Category 2 ISR	RAM	6	6	6	6	6	6
	ROM	90	90	90	90	90	90
Category 2 ISR, floating-point	RAM	14	14	14	14	14	14
	ROM	98	98	98	98	98	98
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	6	6	6	6	6	6
	ROM	38	38	38	38	38	38
Counter	RAM	2	2	2	2	2	2
	ROM	40	40	40	40	40	40
Message	RAM	11	11	11	31	31	31
	ROM	18	18	18	48	48	48
Flag	RAM	2	2	2	2	2	2
	ROM	2	2	2	2	2	2
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	2	2	2	2	2	2
Priority level	RAM	0	0	6	0	6	6
	ROM	0	0	10	0	10	10
Arrivalpoint (readonly)	RAM	0	0	0	0	0	0
	ROM	10	10	10	10	10	10
Arrivalpoint (writable)	RAM	10	10	10	10	10	10
	ROM	10	10	10	10	10	10
Schedule	RAM	10	10	10	10	10	10
	ROM	32	32	32	32	32	32
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 Events Shared Task Priorities Multiple Task Activations		Application Uses					
		No			Yes		
		No		Yes	No		Yes
		No	Yes		No	Yes	
BCC1 Lightweight task	RAM	8	8	8	8	8	8
	ROM	50	50	50	50	50	50
BCC1 Heavyweight task	RAM	12	12	12	12	12	12
	ROM	50	50	50	50	50	50
BCC2 task	RAM	n/a	14	16	n/a	14	16
	ROM	n/a	52	60	n/a	52	60
ECC1, Integer task	RAM	n/a	n/a	n/a	48	48	48
	ROM	n/a	n/a	n/a	66	66	66
ECC1, floating-point task	RAM	n/a	n/a	n/a	56	56	56
	ROM	n/a	n/a	n/a	66	66	66
ECC2, Integer task	RAM	n/a	n/a	n/a	n/a	n/a	50
	ROM	n/a	n/a	n/a	n/a	n/a	74
ECC2, floating-point task	RAM	n/a	n/a	n/a	n/a	n/a	58
	ROM	n/a	n/a	n/a	n/a	n/a	74
Category 2 ISR	RAM	8	8	8	8	8	8
	ROM	98	98	98	98	98	98
Category 2 ISR, floating-point	RAM	16	16	16	16	16	16
	ROM	106	106	106	106	106	106
Resource	RAM	6	6	6	6	6	6
	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	6	6	6	6	6	6
	ROM	28	28	28	28	28	28
Alarm	RAM	6	6	6	6	6	6
	ROM	42	42	42	42	42	42
Counter	RAM	2	2	2	2	2	2
	ROM	44	44	44	44	44	44
Message	RAM	11	11	11	31	31	31
	ROM	22	22	22	52	52	52
Flag	RAM	2	2	2	2	2	2
	ROM	2	2	2	2	2	2
Message resource	RAM	6	6	6	6	6	6
	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	2	2	2	2	2	2
Priority level	RAM	0	0	6	0	6	6
	ROM	0	0	10	0	10	10
Arrivalpoint (readonly)	RAM	0	0	0	0	0	0
	ROM	16	16	16	16	16	16
Arrivalpoint (writable)	RAM	16	16	16	16	16	16
	ROM	16	16	16	16	16	16
Schedule	RAM	12	12	12	12	12	12
	ROM	38	38	38	38	38	38
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	144	210	282	152	218	326
	NS		118	184	256	126	192	300
	KL	2	74	138	204	82	146	252

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	24	24	24	24	24	24
ChainTask	SWL	1, 8	106	190	256	114	198	300
	SWH	1, 9	154	226	294	162	234	334
	NSL	8	106	190	256	114	198	300
	NSH	9	142	214	282	150	222	322
Schedule			102	102	148	102	102	148
GetTaskID			34	34	34	34	34	34
GetTaskState			86	86	86	110	110	110
EnableAllInterrupts			22	22	22	22	22	22
DisableAllInterrupts			16	16	16	16	16	16
ResumeAllInterrupts			38	38	38	38	38	38
SuspendAllInterrupts			34	34	34	34	34	34
ResumeOSInterrupts			38	38	38	38	38	38
SuspendOSInterrupts			52	52	52	52	52	52
GetResource	Task	7	38	38	44	38	38	44
	Combined	6	80	80	80	80	80	80
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
ReleaseResource	Task	7	78	78	78	78	78	78
	Combined	6	162	162	162	162	162	162
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	1	n/a	n/a	n/a	136	136	294
	NS		n/a	n/a	n/a	110	110	256
	NS1i	10	n/a	n/a	n/a	56	n/a	n/a
	KL	2	n/a	n/a	n/a	78	78	206
	KL1i	2, 10	n/a	n/a	n/a	24	n/a	n/a
ClearEvent			n/a	n/a	n/a	48	48	48
GetEvent			n/a	n/a	n/a	12	12	12
WaitEvent	<default>		n/a	n/a	n/a	268	268	544
	fp	11	n/a	n/a	n/a	310	310	622
	1i	10	n/a	n/a	n/a	26	n/a	n/a
GetAlarmBase			54	54	54	54	54	54
GetAlarm			108	108	108	108	108	108
SetRelAlarm			138	138	138	138	138	138
SetAbsAlarm			154	154	154	154	154	154
CancelAlarm			90	90	90	90	90	90
InitCounter			68	68	68	68	68	68

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>		80	80	80	80	80	80
	KL	2	42	42	42	42	42	42
osek_incr_counter			46	46	46	46	46	46
GetActiveApplicationMode		30	n/a	n/a	n/a	n/a	n/a	n/a
StartOS			138	138	138	138	138	138
ShutdownOS	NoHook	12	24	24	24	24	24	24
	Hook	13	28	28	28	28	28	28
InitCOM			4	4	4	4	4	4
CloseCOM			4	4	4	4	4	4
StartCOM			20	20	20	20	20	20
StopCOM			14	14	14	14	14	14
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	70	70	70	182	182	182
	CCCB	15	182	182	182	182	182	182
GetMessageResource			38	38	38	38	38	38
ReleaseMessageResource			32	32	32	32	32	32
GetMessageStatus			46	46	46	46	46	46
SendMessage	SW CCCA	1, 14	88	88	88	238	238	238
	SW CCCB	1, 15	220	220	220	238	238	238
	NS CCCA	14	88	88	88	238	238	238
	NS CCCB	15	220	220	220	238	238	238
	KL CCCA	2, 14	58	58	58	204	204	204
	KL CCCB	2, 15	184	184	184	204	204	204
main_dispatch	NoHook	12	136	136	204	136	136	204
	Hook	13	176	176	254	176	176	254
sub_dispatch	B1LF	19	22	22	22	22	22	22
	B1HI	20	90	90	90	90	90	90
	B1HF	21	98	98	98	98	98	98
	B2LI	22	n/a	82	120	n/a	82	120
	B2LF	23	n/a	90	128	n/a	90	128
	B2HI	24	n/a	164	282	n/a	164	282
	B2HF	25	n/a	172	290	n/a	172	290
	E1HI	26	n/a	n/a	n/a	360	360	482
	E1HF	27	n/a	n/a	n/a	368	368	490
	E2HI	28	n/a	n/a	n/a	n/a	n/a	482

Configuration / Events / Shared Task Priorities / Multiple Task Activations			Application Uses					
			No			Yes		
			No		Yes	No		Yes
			No	Yes		No	Yes	
	E2HF	29	n/a	n/a	n/a	n/a	n/a	490
ErrorHook support		16	32	32	32	32	32	32
	ServiceID	17	42	42	42	42	42	42
	Parameters	18	66	66	66	66	66	66
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	76	132	238	94	162	280
	NS		44	106	204	66	136	228
	KL	2	14	72	156	36	102	190
ChainTaskset	SWL	1, 8	64	112	198	64	126	210
	SWH	1, 9	100	164	252	100	178	280
	NSL	8	64	112	198	64	126	210
	NSH	9	88	152	240	88	166	268
GetTasksetRef			12	12	12	12	12	12
MergeTaskset			46	46	46	46	46	46
AssignTaskset			18	18	18	18	18	18
RemoveTaskset			52	52	52	52	52	52
TestSubTaskset			68	68	68	68	68	68
TestEquivalentTaskset			60	60	60	60	60	60
TickSchedule	SW	1	166	160	160	160	160	160
	NS		140	134	134	134	134	134
	KL	2	106	102	102	102	102	102
AdvanceSchedule	SW	1	152	146	146	146	146	146
	NS		126	120	120	120	120	120
	KL	2	94	88	88	88	88	88
StartSchedule			88	88	88	88	88	88
StopSchedule			66	66	66	66	66	66
GetScheduleStatus			104	104	104	104	104	104
GetScheduleValue			70	70	70	70	70	70
GetScheduleNext			12	12	12	12	12	12
SetScheduleNext			12	12	12	12	12	12
GetArrivalpointDelay			10	10	10	10	10	10
SetArrivalpointDelay			8	8	8	8	8	8
GetArrivalpointTasksetRef			8	8	8	8	8	8
GetArrivalpointNext			14	14	14	14	14	14
SetArrivalpointNext			10	10	10	10	10	10

Configuration / Events / Shared Task Priorities / Multiple Task Activations			Application Uses					
			No			Yes		
			No		Yes	No		Yes
			No	Yes		No	Yes	
TestArrivalpointWritable			32	32	32	32	32	32
GetExecutionTime			4	4	4	4	4	4
GetLargestExecutionTime			8	8	8	8	8	8
ResetLargestExecutionTime			4	4	4	4	4	4
GetStackOffset			26	26	26	26	26	26
Utility function			66	66	66	66	66	66
Interrupt support			52	52	52	52	52	52

Timing

Configuration / Events / Shared Task Priorities / Multiple Task Activations			Application Uses					
			No			Yes		
			No		Yes	No		Yes
			No	Yes		No	Yes	
Service name	Variant	Notes						
ActivateTask	SW	1	144	210	282	152	218	326
	NS		118	184	256	126	192	300
	KL	2	74	138	204	82	146	252
TerminateTask	LExt	3	n/a	n/a	n/a	n/a	n/a	n/a
	H	5	24	24	24	24	24	24
ChainTask	SWL	1, 8	106	190	256	114	198	300
	SWH	1, 9	154	226	294	162	234	334
	NSL	8	106	190	256	114	198	300
	NSH	9	142	214	282	150	222	322
Schedule			114	114	160	114	114	160
GetTaskID			34	34	34	34	34	34
GetTaskState			86	86	86	110	110	110
EnableAllInterrupts			22	22	22	22	22	22
DisableAllInterrupts			16	16	16	16	16	16
ResumeAllInterrupts			38	38	38	38	38	38
SuspendAllInterrupts			34	34	34	34	34	34
ResumeOSInterrupts			38	38	38	38	38	38
SuspendOSInterrupts			52	52	52	52	52	52
GetResource	Task	7	38	38	44	38	38	44
	Combined	6	80	80	80	80	80	80
	CLEx	3	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	
ReleaseResource	Task	7	90	90	90	90	90	90
	Combined	6	186	186	186	186	186	186
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	1	n/a	n/a	n/a	136	136	294
	NS		n/a	n/a	n/a	110	110	256
	NS1i	10	n/a	n/a	n/a	56	n/a	n/a
	KL	2	n/a	n/a	n/a	78	78	206
	KL1i	2, 10	n/a	n/a	n/a	24	n/a	n/a
ClearEvent			n/a	n/a	n/a	48	48	48
GetEvent			n/a	n/a	n/a	12	12	12
WaitEvent	<default>		n/a	n/a	n/a	356	356	602
	fp	11	n/a	n/a	n/a	396	396	680
	1i	10	n/a	n/a	n/a	92	n/a	n/a
GetAlarmBase			54	54	54	54	54	54
GetAlarm			108	108	108	108	108	108
SetRelAlarm			138	138	138	138	138	138
SetAbsAlarm			154	154	154	154	154	154
CancelAlarm			90	90	90	90	90	90
InitCounter			68	68	68	68	68	68
GetCounterValue			68	68	68	68	68	68
osek_tick_alarm	<default>		80	80	80	80	80	80
	KL	2	42	42	42	42	42	42
osek_incr_counter			46	46	46	46	46	46
GetActiveApplicationMode		30	n/a	n/a	n/a	n/a	n/a	n/a
StartOS			182	182	182	182	182	182
ShutdownOS	NoHook	12	24	24	24	24	24	24
	Hook	13	28	28	28	28	28	28
InitCOM			4	4	4	4	4	4
CloseCOM			4	4	4	4	4	4
StartCOM			20	20	20	20	20	20
StopCOM			14	14	14	14	14	14
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	70	70	70	182	182	182
	CCCB	15	182	182	182	182	182	182
GetMessageResource			38	38	38	38	38	38
ReleaseMessageResource			32	32	32	32	32	32

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
GetMessageStatus			46	46	46	46	46	46
SendMessage	SW CCCA	1, 14	88	88	88	238	238	238
	SW CCCB	1, 15	220	220	220	238	238	238
	NS CCCA	14	88	88	88	238	238	238
	NS CCCB	15	220	220	220	238	238	238
	KL CCCA	2, 14	58	58	58	204	204	204
	KL CCCB	2, 15	184	184	184	204	204	204
main_dispatch	NoHook	12	210	210	290	210	210	290
	Hook	13	256	256	330	256	256	330
sub_dispatch	B1LF	19	12	12	12	12	12	12
	B1HI	20	92	92	92	92	92	92
	B1HF	21	100	100	100	100	100	100
	B2LI	22	n/a	62	102	n/a	62	102
	B2LF	23	n/a	68	110	n/a	68	110
	B2HI	24	n/a	138	250	n/a	138	250
	B2HF	25	n/a	146	258	n/a	146	258
	E1HI	26	n/a	n/a	n/a	384	384	500
	E1HF	27	n/a	n/a	n/a	392	392	508
	E2HI	28	n/a	n/a	n/a	n/a	n/a	500
	E2HF	29	n/a	n/a	n/a	n/a	n/a	508
ErrorHook support		16	32	32	32	32	32	32
	ServiceID	17	42	42	42	42	42	42
	Parameters	18	66	66	66	66	66	66
validity_checks		3	n/a	n/a	n/a	n/a	n/a	n/a
Timing_dispatch		4	68	68	68	68	68	68
Timing_termination		4	96	96	96	96	96	96
ActivateTaskset	SW	1	76	132	238	94	162	280
	NS		44	106	204	66	136	228
	KL	2	14	72	156	36	102	190
ChainTaskset	SWL	1, 8	64	112	198	64	126	210
	SWH	1, 9	100	164	252	100	178	280
	NSL	8	64	112	198	64	126	210
	NSH	9	88	152	240	88	166	268
GetTasksetRef			12	12	12	12	12	12
MergeTaskset			46	46	46	46	46	46
AssignTaskset			18	18	18	18	18	18
RemoveTaskset			52	52	52	52	52	52

Configuration / Events / Shared Task Priorities / Multiple Task Activations			Application Uses					
			No			Yes		
			No		Yes	No		Yes
			No	Yes		No	Yes	
TestSubTaskset			68	68	68	68	68	68
TestEquivalentTaskset			60	60	60	60	60	60
TickSchedule	SW	1	166	160	160	160	160	160
	NS		140	134	134	134	134	134
	KL	2	106	102	102	102	102	102
AdvanceSchedule	SW	1	152	146	146	146	146	146
	NS		126	120	120	120	120	120
	KL	2	94	88	88	88	88	88
StartSchedule			88	88	88	88	88	88
StopSchedule			66	66	66	66	66	66
GetScheduleStatus			104	104	104	104	104	104
GetScheduleValue			70	70	70	70	70	70
GetScheduleNext			12	12	12	12	12	12
SetScheduleNext			12	12	12	12	12	12
GetArrivalpointDelay			10	10	10	10	10	10
SetArrivalpointDelay			8	8	8	8	8	8
GetArrivalpointTasksetRef			8	8	8	8	8	8
GetArrivalpointNext			14	14	14	14	14	14
SetArrivalpointNext			10	10	10	10	10	10
TestArrivalpointWritable			32	32	32	32	32	32
GetExecutionTime			100	100	100	100	100	100
GetLargestExecutionTime			22	22	22	22	22	22
ResetLargestExecutionTime			16	16	16	16	16	16
GetStackOffset			26	26	26	26	26	26
Utility function			66	66	66	66	66	66
Interrupt support			204	204	204	204	204	204

Extended

Configuration / Events / Shared Task Priorities / Multiple Task Activations			Application Uses					
			No			Yes		
			No		Yes	No		Yes
			No	Yes		No	Yes	
Service name	Variant	Notes						
ActivateTask	SW	1	232	308	384	242	316	430
	NS		276	354	430	288	362	476

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
	KL	2	158	228	304	168	238	350
TerminateTask	LExt	3	118	118	118	118	118	118
	H	5	158	158	158	158	158	158
ChainTask	SWL	1, 8	280	372	440	290	380	482
	SWH	1, 9	336	416	492	346	424	532
	NSL	8	352	440	506	362	448	548
	NSH	9	388	466	542	398	474	582
Schedule			230	230	278	230	230	278
GetTaskID			46	46	46	46	46	46
GetTaskState			214	214	214	224	224	224
EnableAllInterrupts			34	34	34	34	34	34
DisableAllInterrupts			32	32	32	32	32	32
ResumeAllInterrupts			88	88	88	88	88	88
SuspendAllInterrupts			50	50	50	50	50	50
ResumeOSInterrupts			88	88	88	88	88	88
SuspendOSInterrupts			68	68	68	68	68	68
GetResource	Task	7	364	364	330	364	364	330
	Combined	6	352	352	352	352	352	352
	CLEx	3	316	316	316	316	316	316
ReleaseResource	Task	7	310	310	310	310	310	310
	Combined	6	420	420	420	420	420	420
	CLEx	3	280	280	280	280	280	280
SetEvent	SW	1	n/a	n/a	n/a	298	298	446
	NS		n/a	n/a	n/a	344	344	502
	NS1i	10	n/a	n/a	n/a	212	n/a	n/a
	KL	2	n/a	n/a	n/a	218	218	378
	KL1i	2, 10	n/a	n/a	n/a	164	n/a	n/a
ClearEvent			n/a	n/a	n/a	136	136	136
GetEvent			n/a	n/a	n/a	160	160	160
WaitEvent	<default>		n/a	n/a	n/a	498	498	754
	fp	11	n/a	n/a	n/a	540	540	834
	1i	10	n/a	n/a	n/a	222	n/a	n/a
GetAlarmBase			162	162	162	162	162	162
GetAlarm			160	160	160	160	160	160
SetRelAlarm			230	230	230	230	230	230
SetAbsAlarm			240	240	240	240	240	240
CancelAlarm			146	146	146	146	146	146

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
InitCounter			200	200	200	200	200	200
GetCounterValue			172	172	172	172	172	172
osek_tick_alarm	<default>		106	106	106	106	106	106
	KL	2	42	42	42	42	42	42
osek_incr_counter			46	46	46	46	46	46
GetActiveApplicationMode		30	n/a	n/a	n/a	n/a	n/a	n/a
StartOS			198	198	198	198	198	198
ShutdownOS	NoHook	12	32	32	32	32	32	32
	Hook	13	36	36	36	36	36	36
InitCOM			4	4	4	4	4	4
CloseCOM			4	4	4	4	4	4
StartCOM			36	36	36	36	36	36
StopCOM			38	38	38	38	38	38
ReadFlag			26	26	26	26	26	26
ResetFlag			28	28	28	28	28	28
ReceiveMessage	CCCA	14	146	146	146	278	278	278
	CCCB	15	278	278	278	278	278	278
GetMessageResource			84	84	84	84	84	84
ReleaseMessageResource			84	84	84	84	84	84
GetMessageStatus			84	84	84	84	84	84
SendMessage	SW CCCA	1, 14	180	180	180	314	314	314
	SW CCCB	1, 15	296	296	296	314	314	314
	NS CCCA	14	180	180	180	314	314	314
	NS CCCB	15	296	296	296	314	314	314
	KL CCCA	2, 14	126	126	126	266	266	266
	KL CCCB	2, 15	248	248	248	266	266	266
main_dispatch	NoHook	12	210	210	290	210	210	290
	Hook	13	256	256	330	256	256	330
sub_dispatch	B1LF	19	12	12	12	12	12	12
	B1HI	20	96	96	96	96	96	96
	B1HF	21	104	104	104	104	104	104
	B2LI	22	n/a	66	106	n/a	66	106
	B2LF	23	n/a	72	114	n/a	72	114
	B2HI	24	n/a	146	260	n/a	146	260
	B2HF	25	n/a	154	268	n/a	154	268
	E1HI	26	n/a	n/a	n/a	392	392	508
	E1HF	27	n/a	n/a	n/a	400	400	516

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
	E2HI	28	n/a	n/a	n/a	n/a	n/a	508
	E2HF	29	n/a	n/a	n/a	n/a	n/a	516
ErrorHook support		16	100	100	100	100	100	100
	ServiceID	17	110	110	110	110	110	110
	Parameters	18	134	134	134	134	134	134
validity_checks		3	30	30	30	30	30	30
Timing_dispatch		4	68	68	68	68	68	68
Timing_termination		4	96	96	96	96	96	96
ActivateTaskset	SW	1	318	394	508	342	434	550
	NS		360	438	546	384	476	602
	KL	2	244	322	434	268	356	482
ChainTaskset	SWL	1, 8	394	466	540	402	490	590
	SWH	1, 9	452	540	628	460	562	662
	NSL	8	494	566	640	502	588	688
	NSH	9	538	626	730	546	646	768
GetTasksetRef			124	124	124	124	124	124
MergeTaskset			246	246	246	246	246	246
AssignTaskset			176	176	176	176	176	176
RemoveTaskset			252	252	252	252	252	252
TestSubTaskset			264	264	264	264	264	264
TestEquivalentTaskset			248	248	248	248	248	248
TickSchedule	SW	1	338	268	268	268	268	268
	NS		384	338	338	338	338	338
	KL	2	288	212	212	212	212	212
AdvanceSchedule	SW	1	346	268	268	268	268	268
	NS		392	336	336	336	336	336
	KL	2	286	212	212	212	212	212
StartSchedule			230	230	230	230	230	230
StopSchedule			180	180	180	180	180	180
GetScheduleStatus			218	218	218	218	218	218
GetScheduleValue			180	180	180	180	180	180
GetScheduleNext			102	102	102	102	102	102
SetScheduleNext			168	168	168	168	168	168
GetArrivalpointDelay			118	118	118	118	118	118
SetArrivalpointDelay			140	140	140	140	140	140
GetArrivalpointTasksetRef			118	118	118	118	118	118
GetArrivalpointNext			122	122	122	122	122	122

Configuration / Events / Shared Task Priorities / Multiple Task Activations			Application Uses					
			No			Yes		
			No		Yes	No		Yes
			No	Yes		No	Yes	
SetArrivalpointNext			194	194	194	194	194	194
TestArrivalpointWritable			134	134	134	134	134	134
GetExecutionTime			134	134	134	134	134	134
GetLargestExecutionTime			106	106	106	106	106	106
ResetLargestExecutionTime			104	104	104	104	104	104
GetStackOffset			26	26	26	26	26	26
Utility function			66	66	66	66	66	66
Interrupt support			204	204	204	204	204	204

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 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

Number	Note
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

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	71	102	158	72	95	164
	NS	56	91	147	58	81	150
	KL	38	73	128	42	63	132
TerminateTask	LExt	0	0	0	0	0	0
	H	111	105	125	110	110	132
ChainTask	SWL	197	223	335	235	269	379

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
	SWH	248	270	391	284	313	436
	NSL	197	223	336	235	269	379
	NSH	244	267	388	280	309	433
Schedule	SW	57	53	71	55	54	75
GetTaskID		26	24	29	25	24	28
GetTaskState		61	60	70	67	66	72
EnableAllInterrupts		15	15	15	15	15	15
DisableAllInterrupts		14	14	14	14	14	14
ResumeAllInterrupts		22	22	22	22	22	22
SuspendAllInterrupts		19	19	19	19	19	19
ResumeOSInterrupts		22	22	22	22	22	22
SuspendOSInterrupts		19	19	19	19	19	19
GetResource	Task	38	32	41	34	34	40
	Combined	46	43	46	43	43	43
	CLEx	n/a	n/a	n/a	n/a	n/a	n/a
ReleaseResource	Task	54	49	51	51	51	51
	Combined	72	67	70	69	69	69
	CLEx	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	n/a	n/a	n/a	87	89	99
	NS	n/a	n/a	n/a	84	83	93
	KL	n/a	n/a	n/a	65	66	74
ClearEvent		n/a	n/a	n/a	41	42	42
GetEvent		n/a	n/a	n/a	23	23	23
WaitEvent	<default>	n/a	n/a	n/a	356	374	387
	fp	n/a	n/a	n/a	363	386	399
GetAlarmBase		48	50	56	49	47	55
GetAlarm		62	64	71	64	62	72
SetRelAlarm		74	73	82	73	74	86
SetAbsAlarm		66	66	74	66	66	76
CancelAlarm		45	46	49	46	45	49
InitCounter		46	48	55	48	46	54
GetCounterValue		50	52	61	52	50	60
osek_tick_alarm	<default>	62	60	67	60	62	70
	KL	35	36	39	36	35	39
osek_incr_counter		11	11	12	11	11	13
GetActiveApplicationMode		8	9	7	9	8	6

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
StartOS		264	260	261	262	262	262
ShutdownOS	NoHook	n/a	n/a	n/a	n/a	n/a	n/a
	Hook	23	22	22	22	23	23
InitCOM		11	12	16	12	11	15
CloseCOM		11	12	16	11	12	16
StartCOM		36	37	37	109	110	108
StopCOM		16	18	22	17	19	23
ReadFlag		n/a	n/a	n/a	10	11	13
ResetFlag		n/a	n/a	n/a	8	9	11
ReceiveMessage		51	52	66	138	136	150
GetMessageResource		n/a	n/a	n/a	71	71	83
ReleaseMessageResource		n/a	n/a	n/a	85	85	95
GetMessageStatus		n/a	n/a	n/a	40	40	44
SendMessage	SW	136	172	218	216	238	319
	NS	122	159	205	202	223	304
	KL	92	127	174	172	199	262
ActivateTaskset	SW	54	363	445	62	371	482
	NS	32	352	427	46	358	422
	KL	19	336	404	31	342	405
	SW2	54	363	445	62	371	482
	NS2	32	352	427	46	358	422
	KL2	19	336	404	31	342	405
ChainTaskset	SWL	191	486	615	224	533	649
	SWH	244	537	669	276	586	742
	NSL	190	487	616	223	534	650
	NSH	241	534	665	272	582	736
GetTasksetRef		17	18	18	16	18	17
MergeTaskset		41	37	37	41	40	40
AssignTaskset		22	20	20	22	22	22
RemoveTaskset		42	39	39	42	42	42
TestSubTaskset		56	55	55	56	57	57
TestEquivalentTaskset		49	47	47	49	49	49
TickSchedule	SW	127	444	516	147	466	543
	NS	114	430	502	134	451	528
	KL	97	414	484	118	435	510
	SW2	127	446	522	147	456	527

Configuration / Events / Shared Task Priorities / Multiple Task Activations		Application Uses					
		No			Yes		
		No		Yes	No		Yes
		No	Yes		No	Yes	
	NS2	114	432	508	134	441	512
	KL2	97	416	490	118	425	494
AdvanceSchedule	SW	92	419	492	113	432	511
	NS	78	406	480	98	417	496
	KL	61	391	463	81	402	479
	SW2	92	419	496	113	420	493
	NS2	78	406	484	98	405	478
	KL2	61	391	467	81	390	461
StartSchedule		62	63	63	62	63	63
StopSchedule		53	52	52	53	52	52
GetScheduleStatus		66	66	66	66	66	66
GetScheduleValue		55	55	55	55	55	55
GetScheduleNext		18	19	19	18	18	18
SetScheduleNext		18	18	18	18	17	17
GetArrivalpointDelay		16	16	16	16	16	16
SetArrivalpointDelay		15	15	15	15	15	15
GetArrivalpointTasksetRef		15	15	15	15	15	15
GetArrivalpointNext		18	19	19	18	18	18
SetArrivalpointNext		16	18	18	16	17	17
TestArrivalpointWritable		27	27	27	27	27	27
GetExecutionTime		10	10	10	10	10	10
GetLargestExecutionTime		15	14	15	14	14	14
ResetLargestExecutionTime		12	11	12	11	11	11
GetStackOffset		22	22	22	22	22	22

Timing

Configuration / Events / Shared Task Priorities / Multiple Task Activations		Application Uses					
		No			Yes		
		No		Yes	No		Yes
		No	Yes		No	Yes	
Service	Variant						
ActivateTask	SW	68	98	149	73	95	164
	NS	54	87	138	60	82	150
	KL	38	69	119	40	62	132

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
TerminateTask	LExt	0	0	0	0	0	0
	H	219	213	254	216	222	255
ChainTask	SWL	327	351	474	370	406	520
	SWH	373	396	529	418	458	581
	NSL	327	351	466	370	406	520
	NSH	363	387	520	408	447	572
Schedule	SW	55	50	73	54	55	76
GetTaskID		25	25	28	24	25	29
GetTaskState		61	59	68	67	70	72
EnableAllInterrupts		15	15	15	15	15	15
DisableAllInterrupts		14	14	14	14	14	14
ResumeAllInterrupts		22	22	22	22	22	22
SuspendAllInterrupts		19	19	19	19	19	19
ResumeOSInterrupts		22	22	22	22	22	22
SuspendOSInterrupts		19	19	19	19	19	19
GetResource	Task	35	35	38	34	34	40
	Combined	43	46	43	43	43	43
	CLEx	n/a	n/a	n/a	n/a	n/a	n/a
ReleaseResource	Task	52	51	49	51	51	51
	Combined	69	70	67	69	69	69
	CLEx	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	n/a	n/a	n/a	89	87	99
	NS	n/a	n/a	n/a	83	84	93
	KL	n/a	n/a	n/a	66	65	74
ClearEvent		n/a	n/a	n/a	42	41	42
GetEvent		n/a	n/a	n/a	23	23	23
WaitEvent	<default>	n/a	n/a	n/a	482	507	493
	fp	n/a	n/a	n/a	493	515	505
GetAlarmBase		48	48	58	47	49	55
GetAlarm		61	61	74	62	64	72
SetRelAlarm		70	70	85	74	73	86
SetAbsAlarm		64	64	76	66	66	76
CancelAlarm		45	45	50	45	46	49
InitCounter		47	47	56	46	48	54
GetCounterValue		51	51	62	50	52	60
osek_tick_alarm	<default>	59	59	68	62	60	70

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
	KL	35	35	40	35	36	39
osek_incr_counter		10	10	13	11	11	13
GetActiveApplicationMode		9	9	7	8	9	6
StartOS		450	449	445	450	450	448
ShutdownOS	NoHook	n/a	n/a	n/a	n/a	n/a	n/a
	Hook	22	22	22	23	22	23
InitCOM		12	12	16	11	12	15
CloseCOM		12	12	16	12	11	16
StartCOM		37	37	37	110	109	108
StopCOM		19	19	22	19	17	22
ReadFlag		n/a	n/a	n/a	11	10	13
ResetFlag		n/a	n/a	n/a	9	8	11
ReceiveMessage		51	52	66	136	138	150
GetMessageResource		n/a	n/a	n/a	71	71	83
ReleaseMessageResource		n/a	n/a	n/a	85	85	95
GetMessageStatus		n/a	n/a	n/a	40	40	44
SendMessage	SW	131	166	211	216	239	319
	NS	115	153	198	202	226	304
	KL	85	121	167	172	196	262
ActivateTaskset	SW	54	363	435	62	371	482
	NS	34	352	418	48	356	422
	KL	21	336	395	33	340	405
	SW2	54	363	435	62	371	482
	NS2	34	352	418	48	356	422
	KL2	21	336	395	33	340	405
ChainTaskset	SWL	314	617	754	353	673	796
	SWH	370	666	808	404	727	887
	NSL	321	618	755	354	672	797
	NSH	359	657	798	399	725	875
GetTasksetRef		17	17	17	17	17	17
MergeTaskset		40	37	37	40	41	40
AssignTaskset		22	20	20	22	22	22
RemoveTaskset		42	39	39	42	42	42
TestSubTaskset		57	55	55	57	56	57
TestEquivalentTaskset		49	47	47	49	49	49
TickSchedule	SW	128	445	517	147	466	544

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
	NS	112	432	503	132	453	528
	KL	95	416	485	116	437	510
	SW2	128	445	513	147	456	528
	NS2	112	432	499	132	443	512
	KL2	95	416	481	116	427	494
AdvanceSchedule	SW	91	421	494	111	432	511
	NS	77	408	481	97	418	496
	KL	62	393	464	82	401	479
	SW2	91	419	488	111	420	493
	NS2	77	406	475	97	406	478
	KL2	62	391	458	82	389	461
StartSchedule		63	63	63	63	62	63
StopSchedule		52	52	52	52	53	52
GetScheduleStatus		66	66	66	66	66	66
GetScheduleValue		55	55	55	55	55	55
GetScheduleNext		18	19	19	18	18	18
SetScheduleNext		17	18	18	17	18	17
GetArrivalpointDelay		16	16	16	16	16	16
SetArrivalpointDelay		15	15	15	15	15	15
GetArrivalpointTasksetRef		15	15	15	15	15	15
GetArrivalpointNext		18	19	19	18	18	18
SetArrivalpointNext		17	18	18	17	16	17
TestArrivalpointWritable		27	27	27	27	27	27
GetExecutionTime		63	64	63	63	63	63
GetLargestExecutionTime		26	27	27	26	26	26
ResetLargestExecutionTime		19	20	20	19	19	19
GetStackOffset		22	22	22	22	22	22

Extended

Configuration / Events / Shared Task Priorities / Multiple Task Activations		Application Uses					
		No			Yes		
		No		Yes	No		Yes
Service	Variant	No	Yes		No	Yes	
ActivateTask	SW	199	251	279	220	241	311
	NS	220	268	297	242	258	327
	KL	175	218	252	193	209	282
TerminateTask	LExt	227	240	243	238	240	245
	H	270	280	286	285	286	287
ChainTask	SWL	495	560	630	578	586	709
	SWH	538	598	671	628	642	756
	NSL	525	590	662	608	627	739
	NSH	566	626	702	657	670	783
Schedule	SW	80	85	102	88	89	101
GetTaskID		28	31	30	31	32	30
GetTaskState		197	208	200	213	211	210
EnableAllInterrupts		19	19	18	19	19	18
DisableAllInterrupts		19	19	19	19	19	19
ResumeAllInterrupts		31	31	32	31	31	32
SuspendAllInterrupts		25	25	25	25	25	25
ResumeOSInterrupts		31	31	32	31	31	32
SuspendOSInterrupts		25	25	25	25	25	25
GetResource	Task	317	347	206	385	382	252
	Combined	201	215	201	245	245	245
	CLEx	211	223	216	255	255	255
ReleaseResource	Task	186	201	189	232	232	232
	Combined	196	210	198	240	240	240
	CLEx	181	190	183	222	222	222
SetEvent	SW	n/a	n/a	n/a	257	255	258
	NS	n/a	n/a	n/a	271	272	273
	KL	n/a	n/a	n/a	234	233	242
ClearEvent		n/a	n/a	n/a	67	66	67
GetEvent		n/a	n/a	n/a	209	209	209
WaitEvent	<default>	n/a	n/a	n/a	556	558	611
	fp	n/a	n/a	n/a	564	569	619
GetAlarmBase		149	150	161	148	150	149
GetAlarm		154	153	165	154	156	153

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
SetRelAlarm		188	189	199	188	187	189
SetAbsAlarm		173	172	182	173	173	172
CancelAlarm		144	143	152	144	145	143
InitCounter		228	228	251	228	230	228
GetCounterValue		141	141	148	141	143	139
osek_tick_alarm	<default>	71	71	70	71	69	71
	KL	34	34	36	34	35	34
osek_incr_counter		10	10	11	10	10	10
GetActiveApplicationMode		8	8	9	8	9	8
StartOS		467	472	470	472	472	472
ShutdownOS	NoHook	n/a	n/a	n/a	n/a	n/a	n/a
	Hook	26	26	25	26	25	26
InitCOM		11	11	12	11	12	11
CloseCOM		11	11	12	12	11	12
StartCOM		43	43	44	118	116	118
StopCOM		24	24	26	27	25	27
ReadFlag		n/a	n/a	n/a	24	23	24
ResetFlag		n/a	n/a	n/a	21	23	21
ReceiveMessage		135	136	134	227	225	227
GetMessageResource		n/a	n/a	n/a	359	359	359
ReleaseMessageResource		n/a	n/a	n/a	343	343	343
GetMessageStatus		n/a	n/a	n/a	110	110	110
SendMessage	SW	338	393	421	440	464	531
	NS	360	411	437	463	479	548
	KL	320	365	384	417	437	495
ActivateTaskset	SW	405	734	816	423	733	835
	NS	423	752	829	437	750	845
	KL	382	716	782	400	710	792
	SW2	405	734	816	423	733	835
	NS2	423	752	829	437	750	845
	KL2	382	716	782	400	710	792
ChainTaskset	SWL	750	1153	1237	827	1189	1303
	SWH	759	1105	1194	843	1141	1257
	NSL	771	1175	1256	846	1207	1324
	NSH	804	1149	1241	884	1187	1304
GetTasksetRef		157	168	158	169	169	169

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
MergeTaskset		110	110	112	113	109	110
AssignTaskset		78	80	80	80	80	80
RemoveTaskset		107	109	110	110	109	109
TestSubTaskset		116	117	119	118	118	117
TestEquivalentTaskset		106	107	108	108	107	107
TickSchedule	SW	166	853	912	535	865	935
	NS	186	873	931	556	884	956
	KL	139	828	885	510	838	910
	SW2	166	849	904	534	841	916
	NS2	186	868	922	554	859	936
	KL2	139	824	877	509	814	891
AdvanceSchedule	SW	165	833	891	515	843	915
	NS	183	852	911	535	863	935
	KL	137	809	869	492	822	891
	SW2	165	833	887	516	823	900
	NS2	183	853	908	537	844	921
	KL2	137	809	865	493	802	876
StartSchedule		107	106	107	106	107	106
StopSchedule		84	84	83	84	83	84
GetScheduleStatus		92	92	92	92	92	92
GetScheduleValue		89	89	89	89	89	89
GetScheduleNext		50	49	49	49	49	49
SetScheduleNext		78	77	76	77	76	77
GetArrivalpointDelay		51	51	51	51	51	51
SetArrivalpointDelay		63	62	62	62	62	62
GetArrivalpointTasksetRef		46	46	46	46	46	46
GetArrivalpointNext		49	48	48	48	48	48
SetArrivalpointNext		89	87	88	87	88	87
TestArrivalpointWritable		53	53	53	53	53	53
GetExecutionTime		72	73	72	73	73	73
GetLargestExecutionTime		149	160	150	161	161	161
ResetLargestExecutionTime		144	155	145	156	156	156
GetStackOffset		22	22	22	22	22	22

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	20	20	20	20	20	20
	Cat 2	41	42	42	41	42	42

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	20	20	20	20	20	20
	Cat 2	120	121	120	120	118	120

Extended

Configuration / Events / Shared Task Priorities / Multiple Task Activations		Application Uses					
		No			Yes		
		No		Yes	No		Yes
		No	Yes		No	Yes	
Operation	ISR Category						
ISR Latency	Cat 1	21	21	21	21	21	21
	Cat 2	122	123	120	123	121	123

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.

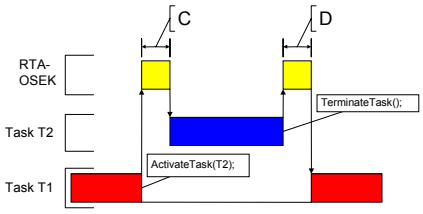

Figure 1: Task Activates a Higher Priority Task which Terminates Normally

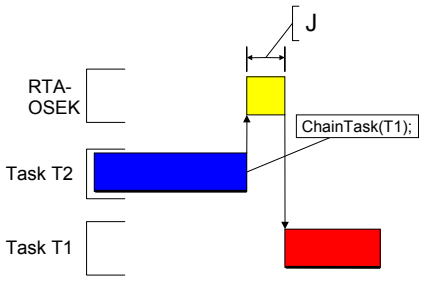

Figure 2: Task Chaining

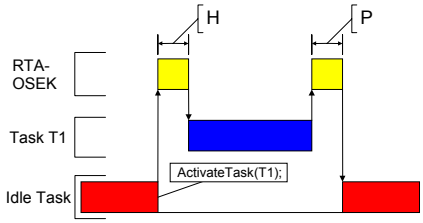

Figure 3: Task Activation from Idle Task

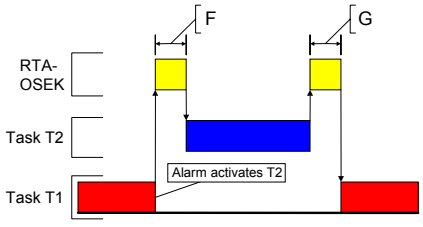

Figure 4: Task Activation from an Alarm

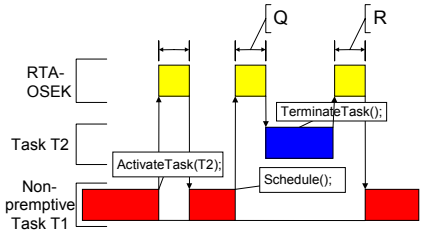

Figure 5: Non-Premptive Task Calls Schedule()

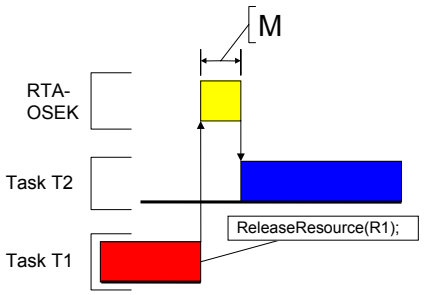

Figure 6: Blocked Task Activated by ReleaseResource()

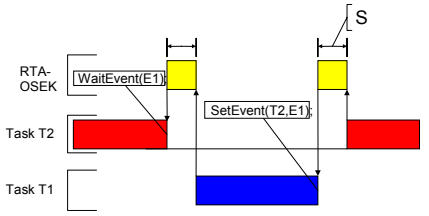

Figure 7: Waiting Task Activated by SetEvent()

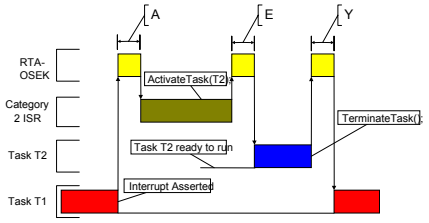

Figure 8: Category 2 ISR Activates a Higher Priority Task

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	72	102	124	71	113	139
Figure 1: D	Heavy, Basic/Extended	135	158	180	161	170	184
ChainTask	Light, Basic	157	190	298	156	200	315
Figure 2: J	Heavy, Basic/Extended	331	381	527	354	400	568
Pre-emption	Light, Basic	151	187	265	151	204	303
Figure 1: C	Heavy, Basic/Extended	219	244	327	257	284	371
From idle task	Light, Basic	138	170	266	138	187	304
Figure 3: H	Heavy, Basic/Extended	192	215	328	230	255	372
Triggered by alarm	Light, Basic	224	258	330	223	275	367
Figure 4: F	Heavy, Basic/Extended	291	316	393	328	356	436
Schedule	Light, Basic	129	132	191	125	144	208
Figure 5: Q	Heavy, Basic/Extended	198	187	247	232	239	292
Release resource	Light, Basic	151	151	180	144	162	191
Figure 6: M	Heavy, Basic/Extended	220	206	225	251	257	265
SetEvent							
Figure 7: S	Heavy, Extended	n/a	n/a	n/a	420	440	515

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations	Task Attributes	No	Yes		No	Yes	
From category 2 ISR	Light, Basic	120	124	151	117	135	165
Figure 8: E	Heavy, Basic/Extended	189	179	196	224	230	239

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	198	228	266	195	232	273
Figure 1: D	Heavy, Basic/Extended	251	270	299	277	288	304
ChainTask	Light, Basic	294	330	432	296	325	459
Figure 2: J	Heavy, Basic/Extended	574	625	768	597	643	830
Pre-emption	Light, Basic	229	271	356	233	278	381
Figure 1: C	Heavy, Basic/Extended	287	317	389	330	363	444
From idle task	Light, Basic	212	251	358	216	258	383
Figure 3: H	Heavy, Basic/Extended	257	284	391	302	333	446
Triggered by alarm	Light, Basic	300	343	420	304	350	445
Figure 4: F	Heavy, Basic/Extended	359	390	454	402	434	509
Schedule	Light, Basic	208	210	279	209	221	289
Figure 5: Q	Heavy, Basic/Extended	267	262	322	307	319	368
Release resource	Light, Basic	227	234	256	227	236	264
Figure 6: M	Heavy, Basic/Extended	286	286	288	325	334	336
SetEvent							
Figure 7: S	Heavy, Extended	n/a	n/a	n/a	488	507	579
From category 2 ISR	Light, Basic	321	326	342	322	334	348
Figure 8: E	Heavy, Basic/Extended	380	378	374	420	432	420

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	246	296	330	257	287	323
Figure 1: D	Heavy, Basic/Extended	303	340	367	340	339	376
ChainTask	Light, Basic	465	540	598	504	544	643
Figure 2: J	Heavy, Basic/Extended	790	894	986	875	906	1043
Pre-emption	Light, Basic	359	432	514	393	424	535
Figure 1: C	Heavy, Basic/Extended	415	478	549	492	507	631
From idle task	Light, Basic	335	405	483	369	397	504
Figure 3: H	Heavy, Basic/Extended	378	438	507	457	470	593
Triggered by alarm	Light, Basic	440	514	596	475	507	617
Figure 4: F	Heavy, Basic/Extended	497	561	630	575	589	714
Schedule	Light, Basic	213	230	293	231	226	286
Figure 5: Q	Heavy, Basic/Extended	257	269	327	320	309	377
Release resource	Light, Basic	359	392	426	417	417	467
Figure 6: M	Heavy, Basic/Extended	416	444	471	517	510	565
SetEvent							
Figure 7: S	Heavy, Extended	n/a	n/a	n/a	687	684	827
From category 2 ISR	Light, Basic	325	344	397	345	342	396
Figure 8: E	Heavy, Basic/Extended	382	396	442	445	435	494

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		48	48	48	48	48	48
BCC1 lightweight, floating-point		52	52	52	52	52	52
BCC1 heavyweight, integer		92	92	92	92	92	92
BCC1 heavyweight, floating-point		92	92	92	92	92	92
BCC2 lightweight, integer		n/a	56	68	n/a	56	68
BCC2 lightweight, floating-point		n/a	56	68	n/a	56	68
BCC2 heavyweight, integer		n/a	92	100	n/a	92	100
BCC2 heavyweight, floating-point		n/a	92	100	n/a	92	100
ECC1 heavyweight, integer		n/a	n/a	n/a	124	124	124
ECC1 heavyweight, floating-point		n/a	n/a	n/a	124	124	124
ECC2 heavyweight, integer		n/a	n/a	n/a	n/a	n/a	128
ECC2 heavyweight, floating-point		n/a	n/a	n/a	n/a	n/a	128
Pre- and/or Post-Task hooks used							
Task type							
BCC1 lightweight, integer		48	48	48	48	48	48
BCC1 lightweight, floating-point		52	52	52	52	52	52
BCC1 heavyweight, integer		92	92	92	92	92	92
BCC1 heavyweight, floating-point		92	92	92	92	92	92
BCC2 lightweight, integer		n/a	56	68	n/a	56	68
BCC2 lightweight, floating-point		n/a	56	68	n/a	56	68
BCC2 heavyweight, integer		n/a	92	100	n/a	92	100
BCC2 heavyweight, floating-point		n/a	92	100	n/a	92	100
ECC1 heavyweight, integer		n/a	n/a	n/a	124	124	124
ECC1 heavyweight, floating-point		n/a	n/a	n/a	124	124	124
ECC2 heavyweight, integer		n/a	n/a	n/a	n/a	n/a	128
ECC2 heavyweight, floating-point		n/a	n/a	n/a	n/a	n/a	128

Timing

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		68	68	68	68	68	68
BCC1 lightweight, floating-point		72	72	72	72	72	72
BCC1 heavyweight, integer		108	108	108	108	108	108
BCC1 heavyweight, floating-point		108	108	108	108	108	108
BCC2 lightweight, integer		n/a	72	88	n/a	72	88
BCC2 lightweight, floating-point		n/a	72	88	n/a	72	88
BCC2 heavyweight, integer		n/a	112	120	n/a	112	120
BCC2 heavyweight, floating-point		n/a	112	120	n/a	112	120
ECC1 heavyweight, integer		n/a	n/a	n/a	144	144	144
ECC1 heavyweight, floating-point		n/a	n/a	n/a	144	144	144
ECC2 heavyweight, integer		n/a	n/a	n/a	n/a	n/a	148
ECC2 heavyweight, floating-point		n/a	n/a	n/a	n/a	n/a	148
Pre- and/or Post-Task hooks used							
Task type							
BCC1 lightweight, integer		68	68	68	68	68	68
BCC1 lightweight, floating-point		72	72	72	72	72	72
BCC1 heavyweight, integer		108	108	108	108	108	108
BCC1 heavyweight, floating-point		108	108	108	108	108	108
BCC2 lightweight, integer		n/a	72	88	n/a	72	88
BCC2 lightweight, floating-point		n/a	72	88	n/a	72	88
BCC2 heavyweight, integer		n/a	112	120	n/a	112	120
BCC2 heavyweight, floating-point		n/a	112	120	n/a	112	120
ECC1 heavyweight, integer		n/a	n/a	n/a	144	144	144
ECC1 heavyweight, floating-point		n/a	n/a	n/a	144	144	144
ECC2 heavyweight, integer		n/a	n/a	n/a	n/a	n/a	148
ECC2 heavyweight, floating-point		n/a	n/a	n/a	n/a	n/a	148

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		68	68	68	68	68	68
BCC1 lightweight, floating-point		72	72	72	72	72	72
BCC1 heavyweight, integer		108	108	108	108	108	108
BCC1 heavyweight, floating-point		108	108	108	108	108	108
BCC2 lightweight, integer		n/a	72	88	n/a	72	88
BCC2 lightweight, floating-point		n/a	72	88	n/a	72	88
BCC2 heavyweight, integer		n/a	112	120	n/a	112	120
BCC2 heavyweight, floating-point		n/a	112	120	n/a	112	120
ECC1 heavyweight, integer		n/a	n/a	n/a	160	160	160
ECC1 heavyweight, floating-point		n/a	n/a	n/a	160	160	160
ECC2 heavyweight, integer		n/a	n/a	n/a	n/a	n/a	164
ECC2 heavyweight, floating-point		n/a	n/a	n/a	n/a	n/a	164
Pre- and/or Post-Task hooks used							
Task type							
BCC1 lightweight, integer		68	68	68	68	68	68
BCC1 lightweight, floating-point		72	72	72	72	72	72
BCC1 heavyweight, integer		108	108	108	108	108	108
BCC1 heavyweight, floating-point		108	108	108	108	108	108
BCC2 lightweight, integer		n/a	72	88	n/a	72	88
BCC2 lightweight, floating-point		n/a	72	88	n/a	72	88
BCC2 heavyweight, integer		n/a	112	120	n/a	112	120
BCC2 heavyweight, floating-point		n/a	112	120	n/a	112	120
ECC1 heavyweight, integer		n/a	n/a	n/a	160	160	160
ECC1 heavyweight, floating-point		n/a	n/a	n/a	160	160	160
ECC2 heavyweight, integer		n/a	n/a	n/a	n/a	n/a	164
ECC2 heavyweight, floating-point		n/a	n/a	n/a	n/a	n/a	164

Support

For product support, please contact your local ETAS representative.

Office locations and contact details can be found on the ETAS Group website www.etasgroup.com.