RTA-OSEK

Binding Manual: SH2/SHC

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: RM00057-002

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 SH2/SHC 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 Compiler

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

Vendor	Renesas (Hitachi)
Compiler	SH Series C/C++ compiler
Version	7.1.02.001

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.

Due a problem with this version of the compiler, `osekdefs.c` must first be compiled to an intermediate assembler file, using the compiler options `-C=A` and `-OP=0`, and also with the option `-cpu=SH2` or `-cpu=SH2E` as appropriate (see the example application for details of how this is done using an options file `copts.sub`). The intermediate file is then assembled to an object file.

The RTA-OSEK Component has been compiled with the following options:

Option	Meaning
`-cpu=SH2`	Compile for SH2
`-OP=1`	Turns on compiler optimization
`-DO=F`	Treat double as float
`-division=cpu=runtime`	Uses CPU division instruction, calls run-time routine

2.2 Assembler

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

Vendor	Renesas (Hitachi)
Assembler	SH Series Assembler
Version	5.1

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.

This file must be assembled with the option `-cpu=SH2` or `-cpu=SH2e` as appropriate. See the example application for details of how this is done using an options file `aopts.sub`.

The RTA-OSEK Component has been assembled with the following options:

Option	Meaning
`-cpu=SH2`	Assemble for SH2

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

In some cases, sections produced by the linker must be located according to special constraints. The following table indicates which sections must be located with which particular constraints:

Sections	Constraints
`STACK`	For ECC support, stack must be placed in the section named STACK

The startup file cstart.src supplied with the example program uses a suitable named section for the stack.

2.3.1 Linking SH2 or SH2e applications with the RTA-OSEK libraries

The RTA-OSEK runtime libraries are compiled to create object code to run on SH2 processor cores using the `-cpu=SH2` compile option. As the RTA-OSEK runtime libraries do not use the floating-point hardware, the resultant libraries can be used with either SH2 or SH2e processors. Applications have been successfully tested on both CPU derivatives.

2.3.2 Generating the standard library

Before the toolchain is used to build application, the Standard Library Generator (`lbgsh`) must be run to create a standard library. This must be built with the `-REE` (Generate re-entrant library) option, in addition to the options to match the users' application compilation process.

A re-entrant library requires some additional functions in order to link and work correctly. The compiler's manual states that these are `errno_addr()`, `wait_sem()` and `signal_sem()`. An example of `errno_addr()` is provided in example\target.c.

The `wait_sem()` and `signal_sem()` functions should only be required if you use the `malloc()`, `strtok()` or file IO library functions. If you need to use these functions that please contact our technical support.

2.4 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 SH2 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 *documenation supplied with the target hardware*.

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

IPL value	Status Register (SR), bits 4-7	Description
0	0	User level
1-15	0001 - 1111	Category 1 and 2 interrupts
16	Non-maskable interrupts, NMI, traps etc	Category 1 interrupts only

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	Not Allowed
0-3	Reset vectors; must be defined by the user outside RTA-OSEK
5, 7, 8, 15-31	Reserved

The valid base addresses for the vector table are:

Base Address	Notes
VBR	Aligned to 4-byte boundary. Vector Base Register must be set before `StartOS()` is called.

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 (Hitachi) C compiler can generate appropriate interrupt handling code for a C

function decorated with the `#pragma 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 Number	Label
4-255	`_os_wrapper_VVVV`
eg :	`_os_wrapper_02F0`

The SH2/SH2e has a vector table that may be placed at any location in memory with an address that is aligned to a 4-byte boundary. The vector table can be provided by RTA-OSEK or user defined. RTA-OSEK permits vectors to be placed in the range 4 to 255 (offsets 0x010 to 0x3FC).

The reset values for the processor registers defined in vectors 0-3 (offsets 0x0 to 0xC) must be provided and linked by the user.

3.1.6 Interrupt Priority Registers

The SH2/SH2e 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 (e.g. `TIER1A`, `TIER1B`, etc.), 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 12 values based at global symbol `os_ipr_values`.

Portability: The number of IPRs may vary according to the SH2/SH2e processor. The RTA-OSEK GUI always generates IPRs A to L inclusive, even though not all processors in the SH2 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 *Hitachi Hardware Manual* for the relevant SH2 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.

3.1.7 Category 1 ISRs

Category 1 interrupts can be declared at priorities 1 to 15 in the RTA-OSEK GUI with the exception of the following priorities:

Permitted priority	Vector offset
16	0x10 (General Illegal Instruction) 0x18 (Slot Illegal Instruction) 0x24 (CPU Address error) 0x28 (DMAC Address error) 0x2C (NMI) 0x34 (FPU exception – where applicable) 0x80 – 0xFC (All Software Traps)
15	0x30 (User Break) 0x38 (H-UDI)

3.2 Register Settings

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

Register	Required Value	Notes
IPRx	RTA-OSEK generates suitable values in `os_ipr_values`	Interrupt Priority Registers need to be set for all interrupt sources that use them

The RTA-OSEK Component does not reserve the use of any hardware registers.

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: `unsigned long`

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): 56

Timing

API max usage (bytes): 56

Extended

API max usage (bytes): 88

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.

3.4 Floating point

SH2e CPUs contain a hardware floating-point arithmetic unit that is not part of a standard SH2 CPU. When floating-point hardware is used for more than one task or ISR, the contents of the floating-point registers must be saved to prevent corruption of their values.

An example of how to save floating-point context can be found in `osfptgt.c` and `osfptgt.h` in the *<RTA-OSEK location>*`\shcsh2\inc` directory. Note that `osfptgt.c` must first be compiled to an intermediate assembler file, which is then assembled to produce an object file. For an application to use floating-point context saving, the appropriate tasks and Category 2 interrupt service routines must be marked as using floating-point operation in the RTA-OSEK GUI.

Note that the performance figures have been collected for a system not using hardware floating point.

3.5 Counters

Different SH2 variants provide different width (16 and 32 bits) hardware counters. RTA-OSEK assumes that all hardware counters used in applications are no larger than 16-bits.

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	**SH2e 7058**
Clock speed (MHz)	40
Code memory	on chip RAM
Read-only data memory	on chip ROM
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	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
Limits for the number of application modes	4294967295					

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	28	28	28	28	28	28
	ROM	172	172	172	172	172	172
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	40	40	40	40	40	40
	ROM	216	216	216	216	216	216
COM overhead	RAM	8	8	8	8	8	8
	ROM	16	16	16	16	16	16

Extended

Configuration / Events / Shared Task Priorities / Multiple Task Activations		Application Uses					
		No			Yes		
		No		Yes	No		Yes
		No	Yes		No	Yes	
OS overhead	RAM	58	58	58	58	58	58
	ROM	262	262	262	262	262	262
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 / Events / Shared Task Priorities / Multiple Task Activations		Application Uses					
		No			Yes		
		No		Yes	No		Yes
		No	Yes		No	Yes	
BCC1 Lightweight task	RAM	0	0	0	0	0	0
	ROM	36	36	36	36	36	36
BCC1 Heavyweight task	RAM	4	4	4	4	4	4
	ROM	40	40	40	40	40	40
BCC2 task	RAM	n/a	8	10	n/a	8	10
	ROM	n/a	44	52	n/a	44	52
ECC1, Integer task	RAM	n/a	n/a	n/a	92	92	92
	ROM	n/a	n/a	n/a	60	60	60
ECC1, floating-point task	RAM	n/a	n/a	n/a	94	94	94
	ROM	n/a	n/a	n/a	60	60	60
ECC2, Integer task	RAM	n/a	n/a	n/a	n/a	n/a	94
	ROM	n/a	n/a	n/a	n/a	n/a	68
ECC2, floating-point task	RAM	n/a	n/a	n/a	n/a	n/a	96
	ROM	n/a	n/a	n/a	n/a	n/a	68
Category 2 ISR	RAM	0	0	0	0	0	0
	ROM	48	48	48	48	48	48
Category 2 ISR, floating-point	RAM	1	1	1	1	1	1
	ROM	78	78	78	78	78	78
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	8	8	8	8	8	8
	ROM	50	50	50	50	50	50
Counter	RAM	2	2	2	2	2	2
	ROM	28	28	28	28	28	28
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	12	12	12	12	12	12
	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	8	8	8	8	8	8
	ROM	48	48	48	48	48	48
BCC1 Heavyweight task	RAM	12	12	12	12	12	12
	ROM	52	52	52	52	52	52
BCC2 task	RAM	n/a	16	18	n/a	16	18
	ROM	n/a	56	64	n/a	56	64
ECC1, Integer task	RAM	n/a	n/a	n/a	100	100	100
	ROM	n/a	n/a	n/a	72	72	72
ECC1, floating-point task	RAM	n/a	n/a	n/a	102	102	102
	ROM	n/a	n/a	n/a	72	72	72
ECC2, Integer task	RAM	n/a	n/a	n/a	n/a	n/a	102
	ROM	n/a	n/a	n/a	n/a	n/a	80
ECC2, floating-point task	RAM	n/a	n/a	n/a	n/a	n/a	104
	ROM	n/a	n/a	n/a	n/a	n/a	80

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
Category 2 ISR	RAM	8	8	8	8	8	8
	ROM	100	100	100	100	100	100
Category 2 ISR, floating-point	RAM	10	10	10	10	10	10
	ROM	112	112	112	112	112	112
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	8	8	8	8	8	8
	ROM	50	50	50	50	50	50
Counter	RAM	2	2	2	2	2	2
	ROM	28	28	28	28	28	28
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	12	12	12	12	12	12
	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 Events Shared Task Priorities Multiple Task Activations		Application Uses					
		No			Yes		
		No		Yes	No		Yes
		No	Yes		No	Yes	
BCC1 Lightweight task	RAM	12	12	12	12	12	12
	ROM	60	60	60	60	60	60
BCC1 Heavyweight task	RAM	16	16	16	16	16	16
	ROM	60	60	60	60	60	60
BCC2 task	RAM	n/a	20	22	n/a	20	22
	ROM	n/a	64	72	n/a	64	72
ECC1, Integer task	RAM	n/a	n/a	n/a	104	104	104
	ROM	n/a	n/a	n/a	80	80	80
ECC1, floating-point task	RAM	n/a	n/a	n/a	106	106	106
	ROM	n/a	n/a	n/a	80	80	80
ECC2, Integer task	RAM	n/a	n/a	n/a	n/a	n/a	106
	ROM	n/a	n/a	n/a	n/a	n/a	88
ECC2, floating-point task	RAM	n/a	n/a	n/a	n/a	n/a	108
	ROM	n/a	n/a	n/a	n/a	n/a	88
Category 2 ISR	RAM	12	12	12	12	12	12
	ROM	112	112	112	112	112	112
Category 2 ISR, floating-point	RAM	14	14	14	14	14	14
	ROM	124	124	124	124	124	124
Resource	RAM	8	8	8	8	8	8
	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	8	8	8	8	8	8
	ROM	28	28	28	28	28	28
Alarm	RAM	8	8	8	8	8	8
	ROM	54	54	54	54	54	54
Counter	RAM	2	2	2	2	2	2
	ROM	32	32	32	32	32	32
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	8	8	8	8	8	8
	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	18	18	18	18	18	18
Arrivalpoint (writable)	RAM	18	18	18	18	18	18
	ROM	18	18	18	18	18	18
Schedule	RAM	14	14	14	14	14	14
	ROM	42	42	42	42	42	42
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	132	188	224	148	196	256
	NS		100	156	192	116	164	224
	KL	2	56	116	156	72	128	188

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	112	172	212	132	184	240
	SWH	1, 9	156	208	248	172	220	280
	NSL	8	112	172	212	132	184	240
	NSH	9	148	200	240	164	212	272
Schedule			100	100	128	100	100	128
GetTaskID			32	32	32	32	32	32
GetTaskState			84	84	84	100	100	100
EnableAllInterrupts			28	28	28	28	28	28
DisableAllInterrupts			24	24	24	24	24	24
ResumeAllInterrupts			44	44	44	44	44	44
SuspendAllInterrupts			48	48	48	48	48	48
ResumeOSInterrupts			44	44	44	44	44	44
SuspendOSInterrupts			68	68	68	68	68	68
GetResource	Task	7	24	24	28	24	24	28
	Combined	6	72	72	72	72	72	72
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
ReleaseResource	Task	7	88	88	88	88	88	88
	Combined	6	176	176	176	176	176	176
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	1	n/a	n/a	n/a	144	144	224
	NS		n/a	n/a	n/a	104	104	184
	NS1i	10	n/a	n/a	n/a	52	n/a	n/a
	KL	2	n/a	n/a	n/a	64	64	144
	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	276	276	528
	fp	11	n/a	n/a	n/a	308	308	600
	1i	10	n/a	n/a	n/a	24	n/a	n/a
GetAlarmBase			56	56	56	56	56	56
GetAlarm			108	108	108	108	108	108
SetRelAlarm			136	136	136	136	136	136
SetAbsAlarm			160	160	160	160	160	160
CancelAlarm			96	96	96	96	96	96
InitCounter			60	60	60	60	60	60

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
GetCounterValue			72	72	72	72	72	72
osek_tick_alarm	<default>		84	84	84	84	84	84
	KL	2	42	42	42	42	42	42
osek_incr_counter			36	36	36	36	36	36
GetActiveApplicationMode		30	n/a	n/a	n/a	n/a	n/a	n/a
StartOS			164	164	164	164	164	164
ShutdownOS	NoHook	12	40	40	40	40	40	40
	Hook	13	64	64	64	64	64	64
InitCOM			4	4	4	4	4	4
CloseCOM			4	4	4	4	4	4
StartCOM			44	44	44	44	44	44
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	84	84	84	212	212	212
	CCCB	15	212	212	212	212	212	212
GetMessageResource			44	44	44	44	44	44
ReleaseMessageResource			40	40	40	40	40	40
GetMessageStatus			38	38	38	38	38	38
SendMessage	SW CCCA	1, 14	116	116	116	256	256	256
	SW CCCB	1, 15	240	240	240	256	256	256
	NS CCCA	14	116	116	116	256	256	256
	NS CCCB	15	240	240	240	256	256	256
	KL CCCA	2, 14	80	80	80	220	220	220
	KL CCCB	2, 15	204	204	204	220	220	220
main_dispatch	NoHook	12	144	144	176	144	144	176
	Hook	13	200	200	232	200	200	232
sub_dispatch	B1LF	19	36	36	36	36	36	36
	B1HI	20	100	100	100	100	100	100
	B1HF	21	124	124	124	124	124	124
	B2LI	22	n/a	84	108	n/a	84	108
	B2LF	23	n/a	96	120	n/a	96	120
	B2HI	24	n/a	156	212	n/a	156	212
	B2HF	25	n/a	180	236	n/a	180	236
	E1HI	26	n/a	n/a	n/a	380	380	444
	E1HF	27	n/a	n/a	n/a	400	400	460
	E2HI	28	n/a	n/a	n/a	n/a	n/a	444

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	460
ErrorHook support		16	44	44	44	44	44	44
	ServiceID	17	52	52	52	52	52	52
	Parameters	18	72	72	72	72	72	72
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	92	144	208	104	168	244
	NS		48	104	168	60	128	204
	KL	2	20	72	136	32	96	172
ChainTaskset	SWL	1, 8	60	112	168	60	124	192
	SWH	1, 9	112	168	228	112	180	252
	NSL	8	60	112	168	60	124	192
	NSH	9	104	160	220	104	172	244
GetTasksetRef			8	8	8	8	8	8
MergeTaskset			48	48	48	48	48	48
AssignTaskset			8	8	8	8	8	8
RemoveTaskset			52	52	52	52	52	52
TestSubTaskset			60	60	60	60	60	60
TestEquivalentTaskset			56	56	56	56	56	56
TickSchedule	SW	1	156	160	160	160	160	160
	NS		112	120	120	120	120	120
	KL	2	80	88	88	88	88	88
AdvanceSchedule	SW	1	152	152	152	152	152	152
	NS		116	112	112	112	112	112
	KL	2	72	76	76	76	76	76
StartSchedule			84	84	84	84	84	84
StopSchedule			64	64	64	64	64	64
GetScheduleStatus			96	96	96	96	96	96
GetScheduleValue			76	76	76	76	76	76
GetScheduleNext			10	10	10	10	10	10
SetScheduleNext			8	8	8	8	8	8
GetArrivalpointDelay			8	8	8	8	8	8
SetArrivalpointDelay			8	8	8	8	8	8
GetArrivalpointTasksetRef			6	6	6	6	6	6
GetArrivalpointNext			8	8	8	8	8	8
SetArrivalpointNext			6	6	6	6	6	6

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			6	6	6	6	6	6
ResetLargestExecutionTime			4	4	4	4	4	4
GetStackOffset			28	28	28	28	28	28

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	132	188	224	148	196	256
	NS		100	156	192	116	164	224
	KL	2	56	116	156	72	128	188
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	112	172	212	132	184	240
	SWH	1, 9	156	208	248	172	220	280
	NSL	8	112	172	212	132	184	240
	NSH	9	148	200	240	164	212	272
Schedule			120	120	152	120	120	152
GetTaskID			32	32	32	32	32	32
GetTaskState			84	84	84	100	100	100
EnableAllInterrupts			28	28	28	28	28	28
DisableAllInterrupts			24	24	24	24	24	24
ResumeAllInterrupts			44	44	44	44	44	44
SuspendAllInterrupts			48	48	48	48	48	48
ResumeOSInterrupts			44	44	44	44	44	44
SuspendOSInterrupts			68	68	68	68	68	68
GetResource	Task	7	24	24	28	24	24	28
	Combined	6	72	72	72	72	72	72
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
ReleaseResource	Task	7	108	108	108	108	108	108
	Combined	6	212	212	212	212	212	212

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	1	n/a	n/a	n/a	144	144	224
	NS		n/a	n/a	n/a	104	104	184
	NS1i	10	n/a	n/a	n/a	52	n/a	n/a
	KL	2	n/a	n/a	n/a	64	64	144
	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	404	404	684
	fp	11	n/a	n/a	n/a	448	448	772
	1i	10	n/a	n/a	n/a	132	n/a	n/a
GetAlarmBase			56	56	56	56	56	56
GetAlarm			108	108	108	108	108	108
SetRelAlarm			136	136	136	136	136	136
SetAbsAlarm			160	160	160	160	160	160
CancelAlarm			96	96	96	96	96	96
InitCounter			60	60	60	60	60	60
GetCounterValue			72	72	72	72	72	72
osek_tick_alarm	<default>		84	84	84	84	84	84
	KL	2	42	42	42	42	42	42
osek_incr_counter			36	36	36	36	36	36
GetActiveApplicationMode		30	n/a	n/a	n/a	n/a	n/a	n/a
StartOS			220	220	220	220	220	220
ShutdownOS	NoHook	12	40	40	40	40	40	40
	Hook	13	64	64	64	64	64	64
InitCOM			4	4	4	4	4	4
CloseCOM			4	4	4	4	4	4
StartCOM			44	44	44	44	44	44
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	84	84	84	212	212	212
	CCCB	15	212	212	212	212	212	212
GetMessageResource			44	44	44	44	44	44
ReleaseMessageResource			40	40	40	40	40	40
GetMessageStatus			38	38	38	38	38	38
SendMessage	SW CCCA	1, 14	116	116	116	256	256	256

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
	SW CCCB	1, 15	240	240	240	256	256	256
	NS CCCA	14	116	116	116	256	256	256
	NS CCCB	15	240	240	240	256	256	256
	KL CCCA	2, 14	80	80	80	220	220	220
	KL CCCB	2, 15	204	204	204	220	220	220
main_dispatch	NoHook	12	196	196	232	196	196	232
	Hook	13	256	256	300	256	256	300
sub_dispatch	B1LF	19	32	32	32	32	32	32
	B1HI	20	112	112	112	112	112	112
	B1HF	21	132	132	132	132	132	132
	B2LI	22	n/a	60	84	n/a	60	84
	B2LF	23	n/a	76	100	n/a	76	100
	B2HI	24	n/a	144	204	n/a	144	204
	B2HF	25	n/a	168	224	n/a	168	224
	E1HI	26	n/a	n/a	n/a	432	432	496
	E1HF	27	n/a	n/a	n/a	456	456	516
	E2HI	28	n/a	n/a	n/a	n/a	n/a	496
	E2HF	29	n/a	n/a	n/a	n/a	n/a	516
ErrorHook support		16	44	44	44	44	44	44
	ServiceID	17	52	52	52	52	52	52
	Parameters	18	72	72	72	72	72	72
validity_checks		3	n/a	n/a	n/a	n/a	n/a	n/a
Timing_dispatch		4	84	84	84	84	84	84
Timing_termination		4	100	100	100	100	100	100
ActivateTaskset	SW	1	92	144	208	104	168	244
	NS		48	104	168	60	128	204
	KL	2	20	72	136	32	96	172
ChainTaskset	SWL	1, 8	60	112	168	60	124	192
	SWH	1, 9	112	168	228	112	180	252
	NSL	8	60	112	168	60	124	192
	NSH	9	104	160	220	104	172	244
GetTasksetRef			8	8	8	8	8	8
MergeTaskset			48	48	48	48	48	48
AssignTaskset			8	8	8	8	8	8
RemoveTaskset			52	52	52	52	52	52
TestSubTaskset			60	60	60	60	60	60
TestEquivalentTaskset			56	56	56	56	56	56

Configuration / Events / Shared Task Priorities / Multiple Task Activations			Application Uses					
			No			Yes		
			No		Yes	No		Yes
			No	Yes		No	Yes	
TickSchedule	SW	1	156	160	160	160	160	160
	NS		112	120	120	120	120	120
	KL	2	80	88	88	88	88	88
AdvanceSchedule	SW	1	152	152	152	152	152	152
	NS		116	112	112	112	112	112
	KL	2	72	76	76	76	76	76
StartSchedule			84	84	84	84	84	84
StopSchedule			64	64	64	64	64	64
GetScheduleStatus			96	96	96	96	96	96
GetScheduleValue			76	76	76	76	76	76
GetScheduleNext			10	10	10	10	10	10
SetScheduleNext			8	8	8	8	8	8
GetArrivalpointDelay			8	8	8	8	8	8
SetArrivalpointDelay			8	8	8	8	8	8
GetArrivalpointTasksetRef			6	6	6	6	6	6
GetArrivalpointNext			8	8	8	8	8	8
SetArrivalpointNext			6	6	6	6	6	6
TestArrivalpointWritable			32	32	32	32	32	32
GetExecutionTime			116	116	116	116	116	116
GetLargestExecutionTime			12	12	12	12	12	12
ResetLargestExecutionTime			10	10	10	10	10	10
GetStackOffset			28	28	28	28	28	28

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	268	324	360	284	332	392
	NS		304	360	396	320	368	428
	KL	2	164	220	256	180	228	288
TerminateTask	LExt	3	112	112	112	112	112	112
	H	5	152	152	152	152	152	152
ChainTask	SWL	1, 8	280	340	380	300	352	408

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
	SWH	1, 9	352	408	444	368	416	476
	NSL	8	328	392	428	348	400	460
	NSH	9	400	456	492	416	464	524
Schedule			260	260	292	260	260	292
GetTaskID			48	48	48	48	48	48
GetTaskState			264	264	264	268	268	268
EnableAllInterrupts			40	40	40	40	40	40
DisableAllInterrupts			40	40	40	40	40	40
ResumeAllInterrupts			92	92	92	92	92	92
SuspendAllInterrupts			64	64	64	64	64	64
ResumeOSInterrupts			92	92	92	92	92	92
SuspendOSInterrupts			80	80	80	80	80	80
GetResource	Task	7	376	376	308	376	376	308
	Combined	6	352	352	352	352	352	352
	CLEx	3	328	328	328	328	328	328
ReleaseResource	Task	7	344	344	344	344	344	344
	Combined	6	452	452	452	452	452	452
	CLEx	3	300	300	300	300	300	300
SetEvent	SW	1	n/a	n/a	n/a	324	324	404
	NS		n/a	n/a	n/a	340	340	420
	NS1i	10	n/a	n/a	n/a	244	n/a	n/a
	KL	2	n/a	n/a	n/a	216	216	296
	KL1i	2, 10	n/a	n/a	n/a	180	n/a	n/a
ClearEvent			n/a	n/a	n/a	144	144	144
GetEvent			n/a	n/a	n/a	164	164	164
WaitEvent	<default>		n/a	n/a	n/a	472	472	704
	fp	11	n/a	n/a	n/a	512	512	784
	1i	10	n/a	n/a	n/a	236	n/a	n/a
GetAlarmBase			196	196	196	196	196	196
GetAlarm			196	196	196	196	196	196
SetRelAlarm			256	256	256	256	256	256
SetAbsAlarm			280	280	280	280	280	280
CancelAlarm			180	180	180	180	180	180
InitCounter			240	240	240	240	240	240
GetCounterValue			212	212	212	212	212	212
osek_tick_alarm	<default>		140	140	140	140	140	140
	KL	2	42	42	42	42	42	42

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
osek_incr_counter			36	36	36	36	36	36
GetActiveApplicationMode		30	n/a	n/a	n/a	n/a	n/a	n/a
StartOS			236	236	236	236	236	236
ShutdownOS	NoHook	12	48	48	48	48	48	48
	Hook	13	76	76	76	76	76	76
InitCOM			4	4	4	4	4	4
CloseCOM			4	4	4	4	4	4
StartCOM			60	60	60	60	60	60
StopCOM			48	48	48	48	48	48
ReadFlag			28	28	28	28	28	28
ResetFlag			32	32	32	32	32	32
ReceiveMessage	CCCA	14	200	200	200	324	324	324
	CCCB	15	324	324	324	324	324	324
GetMessageResource			96	96	96	96	96	96
ReleaseMessageResource			104	104	104	104	104	104
GetMessageStatus			104	104	104	104	104	104
SendMessage	SW CCCA	1, 14	244	244	244	380	380	380
	SW CCCB	1, 15	364	364	364	380	380	380
	NS CCCA	14	244	244	244	380	380	380
	NS CCCB	15	364	364	364	380	380	380
	KL CCCA	2, 14	172	172	172	308	308	308
	KL CCCB	2, 15	292	292	292	308	308	308
main_dispatch	NoHook	12	196	196	232	196	196	232
	Hook	13	256	256	300	256	256	300
sub_dispatch	B1LF	19	32	32	32	32	32	32
	B1HI	20	112	112	112	112	112	112
	B1HF	21	132	132	132	132	132	132
	B2LI	22	n/a	60	84	n/a	60	84
	B2LF	23	n/a	76	100	n/a	76	100
	B2HI	24	n/a	144	204	n/a	144	204
	B2HF	25	n/a	168	224	n/a	168	224
	E1HI	26	n/a	n/a	n/a	432	432	496
	E1HF	27	n/a	n/a	n/a	456	456	516
	E2HI	28	n/a	n/a	n/a	n/a	n/a	496
	E2HF	29	n/a	n/a	n/a	n/a	n/a	516
ErrorHook support		16	124	124	124	124	124	124
	ServiceID	17	136	136	136	136	136	136

Configuration / Events / Shared Task Priorities / Multiple Task Activations			Application Uses					
			No			Yes		
			No		Yes	No		Yes
			No	Yes		No	Yes	
	Parameters	18	168	168	168	168	168	168
validity_checks		3	22	22	22	22	22	22
Timing_dispatch		4	84	84	84	84	84	84
Timing_termination		4	100	100	100	100	100	100
ActivateTaskset	SW	1	368	428	476	392	460	532
	NS		404	464	516	428	492	580
	KL	2	276	336	388	300	368	444
ChainTaskset	SWL	1, 8	424	496	552	440	520	592
	SWH	1, 9	476	552	604	496	576	648
	NSL	8	488	564	624	508	588	664
	NSH	9	544	612	660	564	632	700
GetTasksetRef			132	132	132	132	132	132
MergeTaskset			288	288	288	288	288	288
AssignTaskset			180	180	180	180	180	180
RemoveTaskset			292	292	292	292	292	292
TestSubTaskset			308	308	308	308	308	308
TestEquivalentTaskset			304	304	304	304	304	304
TickSchedule	SW	1	340	292	292	292	292	292
	NS		372	336	336	336	336	336
	KL	2	256	204	204	204	204	204
AdvanceSchedule	SW	1	352	308	308	308	308	308
	NS		384	352	352	352	352	352
	KL	2	268	224	224	224	224	224
StartSchedule			260	260	260	260	260	260
StopSchedule			200	200	200	200	200	200
GetScheduleStatus			232	232	232	232	232	232
GetScheduleValue			196	196	196	196	196	196
GetScheduleNext			88	88	88	88	88	88
SetScheduleNext			172	172	172	172	172	172
GetArrivalpointDelay			132	132	132	132	132	132
SetArrivalpointDelay			152	152	152	152	152	152
GetArrivalpointTasksetRef			132	132	132	132	132	132
GetArrivalpointNext			132	132	132	132	132	132
SetArrivalpointNext			188	188	188	188	188	188
TestArrivalpointWritable			144	144	144	144	144	144
GetExecutionTime			172	172	172	172	172	172
GetLargestExecutionTime			112	112	112	112	112	112

Configuration / Events / Shared Task Priorities / Multiple Task Activations			Application Uses					
			No			Yes		
			No		Yes	No		Yes
			No	Yes		No	Yes	
ResetLargestExecutionTime			104	104	104	104	104	104
GetStackOffset			28	28	28	28	28	28

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
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

Number	Note
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 SH2/SHC port of the RTA-OSEK Component was achieved using a timer running 2 times slower than the CPU clock speed. The figures in this section, therefore, have an uncertainty level of up to 2 CPU cycles. The actual times are between 0 and 2 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	37	53	66	42	51	72
	NS	29	46	58	35	42	65
	KL	19	37	49	24	34	55
TerminateTask	LExt	0	0	0	0	0	0
	H	73	72	75	73	73	76
ChainTask	SWL	121	138	161	143	155	184
	SWH	160	174	200	182	193	224

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
	NSL	121	138	161	143	155	184
	NSH	158	172	198	179	191	221
Schedule	SW	34	34	39	34	34	39
GetTaskID		15	15	15	15	15	15
GetTaskState		31	31	31	35	35	35
EnableAllInterrupts		13	13	13	13	13	13
DisableAllInterrupts		12	12	12	12	12	12
ResumeAllInterrupts		17	17	17	17	17	17
SuspendAllInterrupts		19	19	19	19	19	19
ResumeOSInterrupts		17	17	17	17	17	17
SuspendOSInterrupts		19	19	19	19	19	19
GetResource	Task	16	16	17	16	16	17
	Combined	23	23	23	23	23	23
	CLEx	n/a	n/a	n/a	n/a	n/a	n/a
ReleaseResource	Task	32	32	32	32	32	32
	Combined	46	46	46	46	46	46
	CLEx	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	n/a	n/a	n/a	45	45	45
	NS	n/a	n/a	n/a	39	39	39
	KL	n/a	n/a	n/a	28	28	28
ClearEvent		n/a	n/a	n/a	23	23	23
GetEvent		n/a	n/a	n/a	12	12	12
WaitEvent	<default>	n/a	n/a	n/a	201	201	223
	fp	n/a	n/a	n/a	202	202	224
GetAlarmBase		28	28	28	28	28	28
GetAlarm		38	38	38	38	38	38
SetRelAlarm		48	48	48	48	48	48
SetAbsAlarm		47	47	47	47	47	47
CancelAlarm		31	31	31	31	31	31
InitCounter		26	26	26	26	26	26
GetCounterValue		26	26	26	26	26	26
osek_tick_alarm	<default>	35	35	35	35	35	35
	KL	21	21	21	21	21	21
osek_incr_counter		12	12	12	12	12	12
GetActiveApplicationMode		7	7	7	7	7	7
StartOS		382	382	382	382	382	382

Configuration / Events / Shared Task Priorities / Multiple Task Activations		Application Uses					
		No			Yes		
		No		Yes	No		Yes
		No	Yes		No	Yes	
ShutdownOS	NoHook	n/a	n/a	n/a	n/a	n/a	n/a
	Hook	22	22	22	22	22	22
InitCOM		8	8	8	8	8	8
CloseCOM		7	7	7	7	7	7
StartCOM		22	22	22	80	80	81
StopCOM		12	12	12	12	12	12
ReadFlag		n/a	n/a	n/a	12	12	12
ResetFlag		n/a	n/a	n/a	11	11	11
ReceiveMessage		25	25	25	152	152	152
GetMessageResource		n/a	n/a	n/a	39	39	39
ReleaseMessageResource		n/a	n/a	n/a	52	52	52
GetMessageStatus		n/a	n/a	n/a	23	23	23
SendMessage	SW	66	82	94	193	202	224
	NS	58	75	87	187	194	217
	KL	38	55	68	164	175	196
ActivateTaskset	SW	33	216	243	36	218	280
	NS	19	206	233	22	207	269
	KL	12	197	223	15	198	262
	SW2	33	216	243	36	218	280
	NS2	19	206	233	22	207	269
	KL2	12	197	223	15	198	262
ChainTaskset	SWL	114	299	332	131	316	381
	SWH	159	342	374	177	358	427
	NSL	114	299	332	131	316	381
	NSH	157	341	372	174	357	425
GetTasksetRef		12	12	12	12	12	12
MergeTaskset		22	22	22	22	22	22
AssignTaskset		11	11	11	11	11	11
RemoveTaskset		22	22	22	22	22	22
TestSubTaskset		27	27	27	27	27	27
TestEquivalentTaskset		25	25	25	25	25	25
TickSchedule	SW	58	247	274	66	253	316
	NS	46	239	265	57	244	308
	KL	39	230	257	49	236	299
	SW2	58	247	274	66	249	313
	NS2	46	239	265	57	240	304

Configuration / Events / Shared Task Priorities / Multiple Task Activations		Application Uses					
		No			Yes		
		No		Yes	No		Yes
		No	Yes		No	Yes	
	KL2	39	230	257	49	232	296
AdvanceSchedule	SW	54	242	268	60	247	311
	NS	46	234	260	52	239	303
	KL	31	222	249	41	228	291
	SW2	54	242	268	60	243	307
	NS2	46	234	260	52	235	299
	KL2	31	222	249	41	224	288
StartSchedule		38	38	38	38	38	38
StopSchedule		30	30	30	30	30	30
GetScheduleStatus		37	37	37	37	37	37
GetScheduleValue		33	33	33	33	33	33
GetScheduleNext		12	12	12	12	12	12
SetScheduleNext		12	12	12	12	12	12
GetArrivalpointDelay		11	11	11	11	11	11
SetArrivalpointDelay		12	12	12	12	12	12
GetArrivalpointTasksetRef		10	10	10	10	10	10
GetArrivalpointNext		11	11	11	11	11	11
SetArrivalpointNext		11	11	11	11	11	11
TestArrivalpointWritable		15	15	15	15	15	15
GetExecutionTime		8	8	8	8	8	8
GetLargestExecutionTime		11	11	11	11	11	11
ResetLargestExecutionTime		10	10	10	10	10	10
GetStackOffset		16	16	16	16	16	16

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	37	53	66	42	51	72
	NS	29	46	58	35	42	65
	KL	19	37	49	24	34	55
TerminateTask	LExt	0	0	0	0	0	0

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
	H	146	145	149	146	146	149
ChainTask	SWL	205	222	249	228	240	273
	SWH	244	258	289	268	279	313
	NSL	205	222	249	228	240	273
	NSH	239	253	283	262	274	308
Schedule	SW	33	33	39	33	33	39
GetTaskID		15	15	15	15	15	15
GetTaskState		31	31	31	35	35	35
EnableAllInterrupts		13	13	13	13	13	13
DisableAllInterrupts		12	12	12	12	12	12
ResumeAllInterrupts		17	17	17	17	17	17
SuspendAllInterrupts		19	19	19	19	19	19
ResumeOSInterrupts		17	17	17	17	17	17
SuspendOSInterrupts		19	19	19	19	19	19
GetResource	Task	16	16	17	16	16	17
	Combined	23	23	23	23	23	23
	CLEx	n/a	n/a	n/a	n/a	n/a	n/a
ReleaseResource	Task	32	32	32	32	32	32
	Combined	46	46	46	46	46	46
	CLEx	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	n/a	n/a	n/a	45	45	45
	NS	n/a	n/a	n/a	39	39	39
	KL	n/a	n/a	n/a	28	28	28
ClearEvent		n/a	n/a	n/a	23	23	23
GetEvent		n/a	n/a	n/a	12	12	12
WaitEvent	<default>	n/a	n/a	n/a	290	290	316
	fp	n/a	n/a	n/a	294	294	320
GetAlarmBase		28	28	28	28	28	28
GetAlarm		38	38	38	38	38	38
SetRelAlarm		48	48	48	48	48	48
SetAbsAlarm		47	47	47	47	47	47
CancelAlarm		31	31	31	31	31	31
InitCounter		26	26	26	26	26	26
GetCounterValue		26	26	26	26	26	26
osek_tick_alarm	<default>	35	35	35	35	35	35
	KL	21	21	21	21	21	21

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
osek_incr_counter		12	12	12	12	12	12
GetActiveApplicationMode		7	7	7	7	7	7
StartOS		779	779	779	779	779	779
ShutdownOS	NoHook	n/a	n/a	n/a	n/a	n/a	n/a
	Hook	22	22	22	22	22	22
InitCOM		8	8	8	8	8	8
CloseCOM		7	7	7	7	7	7
StartCOM		22	22	22	81	81	80
StopCOM		12	12	12	12	12	12
ReadFlag		n/a	n/a	n/a	12	12	12
ResetFlag		n/a	n/a	n/a	11	11	11
ReceiveMessage		25	25	25	152	152	152
GetMessageResource		n/a	n/a	n/a	39	39	39
ReleaseMessageResource		n/a	n/a	n/a	52	52	52
GetMessageStatus		n/a	n/a	n/a	23	23	23
SendMessage	SW	66	82	94	193	202	224
	NS	58	75	87	187	194	217
	KL	38	55	68	164	175	196
ActivateTaskset	SW	33	216	243	36	218	280
	NS	19	206	233	22	207	269
	KL	12	197	223	15	198	262
	SW2	33	216	243	36	218	280
	NS2	19	206	233	22	207	269
	KL2	12	197	223	15	198	262
ChainTaskset	SWL	198	383	420	217	402	470
	SWH	243	426	463	263	444	516
	NSL	198	383	420	217	402	470
	NSH	238	422	457	257	443	512
GetTasksetRef		12	12	12	12	12	12
MergeTaskset		22	22	22	22	22	22
AssignTaskset		11	11	11	11	11	11
RemoveTaskset		22	22	22	22	22	22
TestSubTaskset		27	27	27	27	27	27
TestEquivalentTaskset		25	25	25	25	25	25
TickSchedule	SW	58	247	274	66	253	316
	NS	46	239	265	57	244	308

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
	KL	39	230	257	49	236	299
	SW2	58	247	274	66	249	313
	NS2	46	239	265	57	240	304
	KL2	39	230	257	49	232	296
AdvanceSchedule	SW	54	242	268	60	247	311
	NS	46	234	260	52	239	303
	KL	31	222	249	41	228	291
	SW2	54	242	268	60	243	307
	NS2	46	234	260	52	235	299
	KL2	31	222	249	41	224	288
StartSchedule		38	38	38	38	38	38
StopSchedule		30	30	30	30	30	30
GetScheduleStatus		37	37	37	37	37	37
GetScheduleValue		33	33	33	33	33	33
GetScheduleNext		12	12	12	12	12	12
SetScheduleNext		12	12	12	12	12	12
GetArrivalpointDelay		11	11	11	11	11	11
SetArrivalpointDelay		12	12	12	12	12	12
GetArrivalpointTasksetRef		10	10	10	10	10	10
GetArrivalpointNext		11	11	11	11	11	11
SetArrivalpointNext		11	11	11	11	11	11
TestArrivalpointWritable		15	15	15	15	15	15
GetExecutionTime		44	44	44	44	44	44
GetLargestExecutionTime		13	13	13	13	13	13
ResetLargestExecutionTime		12	12	12	12	12	12
GetStackOffset		16	16	16	16	16	16

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	117	133	146	122	129	152
	NS	125	141	155	129	139	160
	KL	99	114	129	102	113	132
TerminateTask	LExt	157	157	159	157	157	160
	H	189	189	191	189	189	192
ChainTask	SWL	315	332	359	338	351	384
	SWH	355	371	397	376	388	422
	NSL	329	344	372	352	364	395
	NSH	366	383	410	388	399	433
Schedule	SW	59	59	65	59	59	65
GetTaskID		19	19	19	19	19	19
GetTaskState		126	126	126	127	127	127
EnableAllInterrupts		17	17	17	17	17	17
DisableAllInterrupts		17	17	17	17	17	17
ResumeAllInterrupts		28	28	28	28	28	28
SuspendAllInterrupts		24	24	24	24	24	24
ResumeOSInterrupts		28	28	28	28	28	28
SuspendOSInterrupts		24	24	24	24	24	24
GetResource	Task	182	182	108	198	198	124
	Combined	104	104	104	121	121	121
	CLEx	115	115	115	132	132	132
ReleaseResource	Task	112	112	112	128	128	128
	Combined	113	113	113	130	130	130
	CLEx	102	102	102	118	118	118
SetEvent	SW	n/a	n/a	n/a	129	129	129
	NS	n/a	n/a	n/a	138	138	138
	KL	n/a	n/a	n/a	117	117	117
ClearEvent		n/a	n/a	n/a	46	46	46
GetEvent		n/a	n/a	n/a	102	102	102
WaitEvent	<default>	n/a	n/a	n/a	324	324	346
	fp	n/a	n/a	n/a	327	327	350
GetAlarmBase		94	94	94	94	94	94
GetAlarm		100	100	100	100	100	100

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
SetRelAlarm		118	118	118	118	118	118
SetAbsAlarm		114	114	114	114	114	114
CancelAlarm		90	90	90	90	90	90
InitCounter		149	149	149	149	149	149
GetCounterValue		88	88	88	88	88	88
osek_tick_alarm	<default>	50	50	50	50	50	50
	KL	21	21	21	21	21	21
osek_incr_counter		12	12	12	12	12	12
GetActiveApplicationMode		7	7	7	7	7	7
StartOS		902	902	902	902	902	902
ShutdownOS	NoHook	n/a	n/a	n/a	n/a	n/a	n/a
	Hook	26	26	26	26	26	26
InitCOM		8	8	8	8	8	8
CloseCOM		7	7	7	7	7	7
StartCOM		30	30	30	92	92	89
StopCOM		17	17	17	17	17	17
ReadFlag		n/a	n/a	n/a	18	18	18
ResetFlag		n/a	n/a	n/a	17	17	17
ReceiveMessage		77	77	77	201	201	201
GetMessageResource		n/a	n/a	n/a	180	180	180
ReleaseMessageResource		n/a	n/a	n/a	180	180	180
GetMessageStatus		n/a	n/a	n/a	61	61	61
SendMessage	SW	195	211	224	321	328	351
	NS	204	219	233	329	339	359
	KL	161	176	191	277	288	307
ActivateTaskset	SW	336	579	597	347	612	656
	NS	345	620	640	354	588	662
	KL	318	592	578	325	559	651
	SW2	336	579	597	347	612	656
	NS2	345	620	640	354	588	662
	KL2	318	592	578	325	559	651
ChainTaskset	SWL	571	863	882	595	882	922
	SWH	608	883	916	633	917	971
	NSL	587	883	918	612	883	941
	NSH	625	915	948	649	916	974
GetTasksetRef		94	94	94	94	94	94

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
MergeTaskset		68	68	68	68	68	68
AssignTaskset		45	45	45	45	45	45
RemoveTaskset		69	69	69	69	69	69
TestSubTaskset		81	81	81	81	81	81
TestEquivalentTaskset		79	79	79	79	79	79
TickSchedule	SW	96	668	654	401	642	733
	NS	107	678	664	411	652	743
	KL	75	646	633	380	621	711
	SW2	96	668	654	401	635	726
	NS2	107	678	664	411	645	736
	KL2	75	646	633	380	614	704
AdvanceSchedule	SW	92	661	648	395	636	726
	NS	101	670	656	403	644	735
	KL	69	641	627	374	615	706
	SW2	92	661	648	395	629	719
	NS2	101	670	656	403	637	728
	KL2	69	641	627	374	608	699
StartSchedule		70	70	70	70	70	70
StopSchedule		56	56	56	56	56	56
GetScheduleStatus		62	62	62	62	62	62
GetScheduleValue		58	58	58	58	58	58
GetScheduleNext		29	29	29	29	29	29
SetScheduleNext		48	48	48	48	48	48
GetArrivalpointDelay		37	37	37	37	37	37
SetArrivalpointDelay		42	42	42	42	42	42
GetArrivalpointTasksetRef		29	29	29	29	29	29
GetArrivalpointNext		30	30	30	30	30	30
SetArrivalpointNext		52	52	52	52	52	52
TestArrivalpointWritable		32	32	32	32	32	32
GetExecutionTime		60	60	60	60	60	60
GetLargestExecutionTime		87	87	87	87	87	87
ResetLargestExecutionTime		82	82	82	82	82	82
GetStackOffset		16	16	16	16	16	16

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	48	48	48	48	48	48
	Cat 2	70	70	70	70	70	70

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	48	48	48	48	48	48
	Cat 2	180	180	180	180	180	180

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	48	48	48	48	48	48
	Cat 2	180	180	180	180	180	180

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.

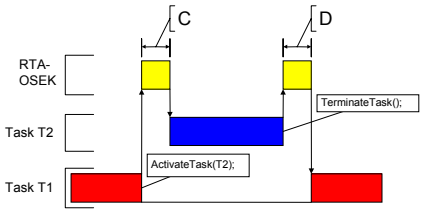

Figure 1: Task Activates a Higher Priority Task which Terminates Normally

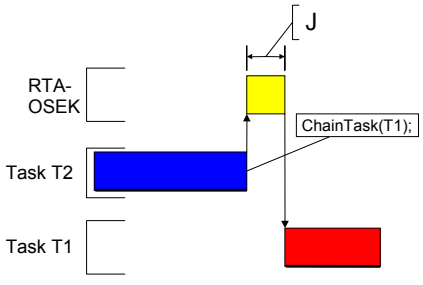

Figure 2: Task Chaining

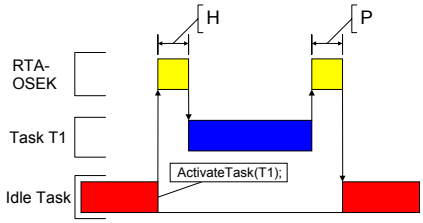

Figure 3: Task Activation from Idle Task

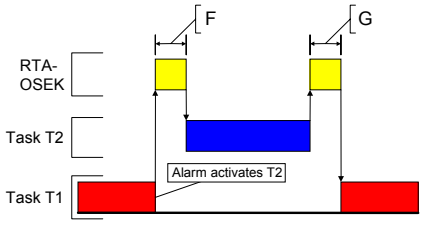

Figure 4: Task Activation from an Alarm

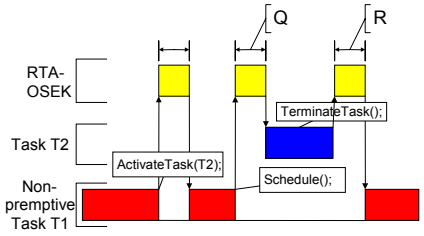

Figure 5: Non-Premptive Task Calls Schedule()

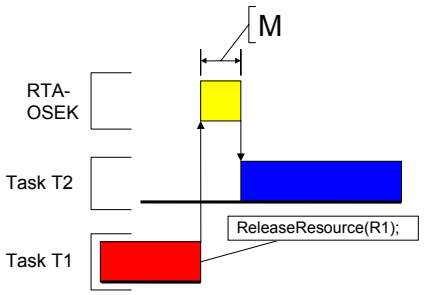

Figure 6: Blocked Task Activated by ReleaseResource()

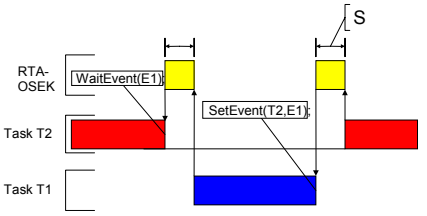

Figure 7: Waiting Task Activated by SetEvent()

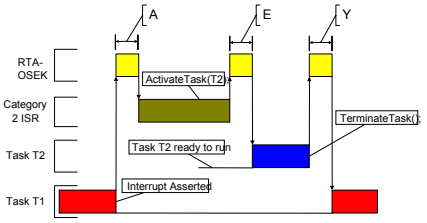

Figure 8: Category 2 ISR Activates a Higher Priority Task

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	33	49	61	32	49	61
Figure 1: D	Heavy, Basic/Extended	72	86	97	87	87	98
ChainTask	Light, Basic	76	98	122	80	98	127
Figure 2: J	Heavy, Basic/Extended	200	233	271	219	237	278
Pre-emption	Light, Basic	77	99	125	82	99	132
Figure 1: C	Heavy, Basic/Extended	124	140	166	147	156	191
From idle task	Light, Basic	78	100	126	82	99	132
Figure 3: H	Heavy, Basic/Extended	125	141	167	147	156	191
Triggered by alarm	Light, Basic	119	140	167	123	139	173
Figure 4: F	Heavy, Basic/Extended	166	182	208	188	197	232
Schedule	Light, Basic	70	76	94	70	76	94
Figure 5: Q	Heavy, Basic/Extended	117	117	136	135	135	153
Release resource	Light, Basic	78	84	97	79	85	98
Figure 6: M	Heavy, Basic/Extended	125	125	139	144	144	157
SetEvent							
Figure 7: S	Heavy, Extended	n/a	n/a	n/a	239	239	284

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	67	72	86	67	72	86
Figure 8: E	Heavy, Basic/Extended	114	114	127	131	131	145

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	106	117	130	105	117	130
Figure 1: D	Heavy, Basic/Extended	145	152	166	154	154	165
ChainTask	Light, Basic	162	182	208	166	181	213
Figure 2: J	Heavy, Basic/Extended	361	387	428	372	388	432
Pre-emption	Light, Basic	123	143	172	128	143	179
Figure 1: C	Heavy, Basic/Extended	168	184	214	193	202	240
From idle task	Light, Basic	124	144	173	128	143	179
Figure 3: H	Heavy, Basic/Extended	169	185	215	193	202	240
Triggered by alarm	Light, Basic	164	184	214	168	183	220
Figure 4: F	Heavy, Basic/Extended	209	225	256	234	243	281
Schedule	Light, Basic	117	121	142	117	121	142
Figure 5: Q	Heavy, Basic/Extended	162	162	184	182	182	203
Release resource	Light, Basic	124	128	144	125	129	145
Figure 6: M	Heavy, Basic/Extended	169	169	186	190	190	206
SetEvent							
Figure 7: S	Heavy, Extended	n/a	n/a	n/a	294	294	346
From category 2 ISR	Light, Basic	185	189	205	185	189	205
Figure 8: E	Heavy, Basic/Extended	230	230	247	250	250	266

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	155	168	179	155	168	179
Figure 1: D	Heavy, Basic/Extended	189	197	208	198	198	208
ChainTask	Light, Basic	272	293	319	275	292	325
Figure 2: J	Heavy, Basic/Extended	516	543	581	525	541	585
Pre-emption	Light, Basic	199	219	248	204	219	255
Figure 1: C	Heavy, Basic/Extended	244	260	290	269	277	316
From idle task	Light, Basic	200	220	249	204	219	255
Figure 3: H	Heavy, Basic/Extended	245	261	291	269	277	316
Triggered by alarm	Light, Basic	255	275	305	260	274	311
Figure 4: F	Heavy, Basic/Extended	300	316	347	325	333	372
Schedule	Light, Basic	138	142	163	138	142	163
Figure 5: Q	Heavy, Basic/Extended	183	183	205	203	203	224
Release resource	Light, Basic	193	197	214	211	215	231
Figure 6: M	Heavy, Basic/Extended	238	238	256	276	276	292
SetEvent							
Figure 7: S	Heavy, Extended	n/a	n/a	n/a	376	376	430
From category 2 ISR	Light, Basic	202	206	222	202	206	222
Figure 8: E	Heavy, Basic/Extended	247	247	264	267	267	283

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	84	84	88	84	84	88
BCC1 lightweight, floating-point	88	88	92	88	88	92
BCC1 heavyweight, integer	168	168	172	168	168	172
BCC1 heavyweight, floating-point	168	168	172	168	168	172
BCC2 lightweight, integer	n/a	88	92	n/a	88	92
BCC2 lightweight, floating-point	n/a	88	92	n/a	88	92
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	212
ECC1 heavyweight, floating-point	n/a	n/a	n/a	208	208	212
ECC2 heavyweight, integer	n/a	n/a	n/a	n/a	n/a	212
ECC2 heavyweight, floating-point	n/a	n/a	n/a	n/a	n/a	212
Pre- and/or Post-Task hooks used						
Task type						
BCC1 lightweight, integer	92	92	92	92	92	92
BCC1 lightweight, floating-point	96	96	96	96	96	96
BCC1 heavyweight, integer	176	176	176	176	176	176
BCC1 heavyweight, floating-point	176	176	176	176	176	176
BCC2 lightweight, integer	n/a	96	96	n/a	96	96
BCC2 lightweight, floating-point	n/a	96	96	n/a	96	96
BCC2 heavyweight, integer	n/a	176	180	n/a	176	180
BCC2 heavyweight, floating-point	n/a	176	180	n/a	176	180
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

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		116	116	120	116	116	120
BCC1 lightweight, floating-point		120	120	124	120	120	124
BCC1 heavyweight, integer		200	200	204	200	200	204
BCC1 heavyweight, floating-point		200	200	204	200	200	204
BCC2 lightweight, integer		n/a	120	124	n/a	120	124
BCC2 lightweight, floating-point		n/a	120	124	n/a	120	124
BCC2 heavyweight, integer		n/a	200	208	n/a	200	208
BCC2 heavyweight, floating-point		n/a	200	208	n/a	200	208
ECC1 heavyweight, integer		n/a	n/a	n/a	256	256	260
ECC1 heavyweight, floating-point		n/a	n/a	n/a	256	256	260
ECC2 heavyweight, integer		n/a	n/a	n/a	n/a	n/a	260
ECC2 heavyweight, floating-point		n/a	n/a	n/a	n/a	n/a	260
Pre- and/or Post-Task hooks used							
Task type							
BCC1 lightweight, integer		120	120	120	120	120	120
BCC1 lightweight, floating-point		124	124	124	124	124	124
BCC1 heavyweight, integer		204	204	204	204	204	204
BCC1 heavyweight, floating-point		204	204	204	204	204	204
BCC2 lightweight, integer		n/a	124	124	n/a	124	124
BCC2 lightweight, floating-point		n/a	124	124	n/a	124	124
BCC2 heavyweight, integer		n/a	204	208	n/a	204	208
BCC2 heavyweight, floating-point		n/a	204	208	n/a	204	208
ECC1 heavyweight, integer		n/a	n/a	n/a	260	260	260
ECC1 heavyweight, floating-point		n/a	n/a	n/a	260	260	260
ECC2 heavyweight, integer		n/a	n/a	n/a	n/a	n/a	260
ECC2 heavyweight, floating-point		n/a	n/a	n/a	n/a	n/a	260

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		116	116	120	116	116	120
BCC1 lightweight, floating-point		120	120	124	120	120	124
BCC1 heavyweight, integer		200	200	204	200	200	204
BCC1 heavyweight, floating-point		200	200	204	200	200	204
BCC2 lightweight, integer		n/a	120	124	n/a	120	124
BCC2 lightweight, floating-point		n/a	120	124	n/a	120	124
BCC2 heavyweight, integer		n/a	200	208	n/a	200	208
BCC2 heavyweight, floating-point		n/a	200	208	n/a	200	208
ECC1 heavyweight, integer		n/a	n/a	n/a	260	260	264
ECC1 heavyweight, floating-point		n/a	n/a	n/a	260	260	264
ECC2 heavyweight, integer		n/a	n/a	n/a	n/a	n/a	264
ECC2 heavyweight, floating-point		n/a	n/a	n/a	n/a	n/a	264
Pre- and/or Post-Task hooks used							
Task type							
BCC1 lightweight, integer		120	120	120	120	120	120
BCC1 lightweight, floating-point		124	124	124	124	124	124
BCC1 heavyweight, integer		204	204	204	204	204	204
BCC1 heavyweight, floating-point		204	204	204	204	204	204
BCC2 lightweight, integer		n/a	124	124	n/a	124	124
BCC2 lightweight, floating-point		n/a	124	124	n/a	124	124
BCC2 heavyweight, integer		n/a	204	208	n/a	204	208
BCC2 heavyweight, floating-point		n/a	204	208	n/a	204	208
ECC1 heavyweight, integer		n/a	n/a	n/a	264	264	264
ECC1 heavyweight, floating-point		n/a	n/a	n/a	264	264	264
ECC2 heavyweight, integer		n/a	n/a	n/a	n/a	n/a	264
ECC2 heavyweight, floating-point		n/a	n/a	n/a	n/a	n/a	264

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.