RTA-OSEK

Binding Manual: MPC56x/Diab

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: RM00025-004

Disclaimer

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

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

Trademarks

RTA-OSEK 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 MPC56x/Diab 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	Diab
Compiler	DCC
Version	5.0.3

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

Option	Description
`-tPPC555EH:simple`	Selects the correct target for code generation etc.

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
`-tPPC555EH:simple`	Selects the correct target for code generation etc.
`-g0`	Turn debug mode off.

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

Option	Description
`-g`	Debugging must be disabled.

2.2 Assembler

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

Vendor	Diab
Assembler	DAS
Version	5.0.3

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

Option	Description
`-tPPC555EH:simple`	Selects the correct target for code generation etc.

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

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

Option	Description
`-tPPC555EH:simple`	Selects the correct target for code generation etc.

2.3 Linker/Locator

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

Option	Description
`-tPPC555EH:simple`	Ensures the correct run-time libraries are selected.

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_pnird`	ROM	RTA-OSEK near initialization data
`os_vectbl`	ROM	Vector table if generated by RTA-OSEK
`os_pir`	RAM	RTA-OSEK initialized data
`os_pur`	RAM	RTA-OSEK uninitialized data
`os_vec_ETR`	ROM	RTA-OSEK RCPU vector table for use when the BBCMCR[ETRE] bit is 1
`.abs.0000XX00`	ROM	RTA-OSEK RCPU vector table for use when MSR[IP] bit is 0
`.abs.FFF0XX00`	ROM	RTA-OSEK RCPU vector table for use when MSR[IP] bit is 1

Furthermore the symbol `__OS_INTERNAL_BASE` must be defined in the linker command file as the start address of the processor's internal memory space. This must correspond to the value set in the ISB field of the processor's Internal Memory Map Register. This is illustrated in the supplied example application.

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

C Library Functions
`setjmp`
`longjmp`

For improved performance, the RTA-OSEK GUI outputs a small amount of data in compiler sections `.sdata` and `.sdata2`. Please refer to the compiler documentation on these sections.

2.4 Debugger

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

ORTI compatible debuggers	Lauterbach TRACE32

The RTA-OSEK GUI outputs a file with the extension `.ort`. This file should be loaded into the debugger with the command `Task.ORTI <file>`. Please refer to the debugger documentation for further details on its support for ORTI.

3 Target Hardware Issues

3.1 Interrupts

This section explains the implementation of RTA-OSEK's interrupt model. 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 *MPC565/566 Reference Manual*.

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

IPL Value	Simask2 / Simask3	Description
0	SIMASK3 bits 16-31 = 0	User level
1	SIMASK3 bits 15-31 = 0	IMB_IRQ31
2	SIMASK3 bits 14-31 = 0	IMB_IRQ30
3	SIMASK3 bits 13-31 = 0	IMB_IRQ29
4	SIMASK3 bits 12-31 = 0	IMB_IRQ28
5	SIMASK3 bits 11-31 = 0	Level 7
6	SIMASK3 bits 10-31 = 0	EXT_IRQ 7
…	…	…
16	SIMASK3 bits 0-31 = 0	IMB_IRQ20
17	SIMASK2 bit 31 = 0, SIMASK3 bits 0-31 = 0	Level 5
…	…	…
45	SIMASK2 bits 3-31 = 0, SIMASK3 bits 0-31 = 0	IMB_IRQ1
46	SIMASK2 bits 2-31 = 0, SIMASK3 bits 0-31 = 0	IMB_IRQ0
47	SIMASK2 bits 1-31 = 0, SIMASK3 bits 0-31 = 0	Level 0 Interrupt
48	MSR[EE] bit clear	Category 1 Interrupts
49	Not applicable	Synchronous Exceptions

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 Table	Constraints
RCPU	Category 1 ISRs only
EEIR	Category 2 ISRs and Category 1 ISRs

The valid base addresses for the vector table are:

Base Address	Notes
set by EIBADR	EEIR vector table
0x00000100	RCPU vector table (MSR[IP] set to 0, BBCMCR[ETRE] set to 0)
0xFFF00100	RCPU vector table (MSR[IP] set to 1, BBCMCR[ETRE] set to 0)

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 Diab 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.s`. Note that this generated vector table is for EEIR vectors only. 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 the RTA-OSEK Component.

In addition, the RTA-OSEK GUI outputs three files `osvechi.s`, `osveclow.s` and `osvecETR.s`. These files contain the RCPU vector table for each operating mode of the interrupt hardware. You should link against the appropriate file for your target's configuration. The following table shows

when each file should be used. Note that these files do not include the reset vector.

Vector File	Notes
`osveclow.s`	Contains vectors located from 0x0000200 upwards. MSR[IP] = 0 and BBCMCR[ETRE] = 0
`osvechi.s`	Contains vectors located from 0xFFF0200 upwards. MSR[IP] = 1 and BBCMCR[ETRE] = 0
`osvecETR.s`	Contains a relocatable vector segment. This should be located by the linker command file. BBCMCR[ETRE] = 1

The table below shows the relationship and allowed interrupt categories for vectors in the RCPU vector table:

Interrupt Vector (`Osvechi.S`)	Interrupt Vector (`Osveclow.S`)	Interrupt Vector Offset (`Osvecetr.S`)	Description	Category Permitted	Priority
0xFFF00200	0x00000200	0x0010	Machine Check	1	49
0xFFF00300	0x00000300	0x0018	Data Storage	1	49
0xFFF00400	0x00000400	0x0020	Illegal	N/A	N/A
0xFFF00500	0x00000500	0x0028	Illegal	N/A	N/A
0xFFF00600	0x00000600	0x0030	Alignment	1	49
0xFFF00700	0x00000700	0x0038	Program	1	49
0xFFF00800	0x00000800	0x0040	Floating Point Unavailable	1	49
0xFFF00900	0x00000900	0x0048	Decrementer	1	48
0xFFF00A00	0x00000A00	0x0050	Illegal	N/A	N/A
0xFFF00B00	0x00000B00	0x0058	Illegal	N/A	N/A
0xFFF00C00	0x00000C00	0x0060	System Call	1	49
0xFFF00D00	0x00000D00	0x0068	Trace	1	49
0xFFF00E00	0x00000E00	0x0070	Floating point assist	1	49
0xFFF00F00	0x00000F00	0x0078	Illegal	N/A	N/A

Interrupt Vector (`Osvechi.S`)	Interrupt Vector (`Osveclow.S`)	Interrupt Vector Offset (`Osvecetr.S`)	Description	Category Permitted	Priority
0xFFF01000	0x00001000	0x0080	Implementation dependent software emulation	1	49
0xFFF01100	0x00001100	0x0088	Illegal	N/A	N/A
0xFFF01200	0x00001200	0x0090	Illegal	N/A	N/A
0xFFF01300	0x00001300	0x0098	Implementation dependent instruction storage protection error	1	49
0xFFF01400	0x00001400	0x00A0	Implementation dependent data storage protection error	1	49
0xFFF01500	0x00001500	0x00A8	Illegal	N/A	N/A
0xFFF01600	0x00001600	0x00B0	Illegal	N/A	N/A
0xFFF01700	0x00001700	0x00B8	Illegal	N/A	N/A
0xFFF01800	0x00001800	0x00C0	Illegal	N/A	N/A
0xFFF01900	0x00001900	0x00C8	Illegal	N/A	N/A
0xFFF01A00	0x00001A00	0x00D0	Illegal	N/A	N/A
0xFFF01B00	0x00001B00	0x00D8	Illegal	N/A	N/A
0xFFF01C00	0x00001C00	0x00E0	Implementation dependent data breakpoint	1	49

Interrupt Vector (`Osvechi.S`)	Interrupt Vector (`Osveclow.S`)	Interrupt Vector Offset (`Osvecetr.S`)	Description	Category Permitted	Priority
0xFFF01D00	0x00001D00	0x00E8	Implementation dependent instruction breakpoint	1	49
0xFFF01E00	0x00001E00	0x00F0	Implementation dependent maskable external breakpoint	1	49
0xFFF01F00	0x00001F00	0x00F8	Non-maskable external breakpoint	1	49

The table below shows the relationship and allowed interrupt categories for vectors in the EEIR vector table:

Interrupt Vector Offset	Vector Description	Categories Permitted	Cat 1 Priority	Cat 2 Priority
0x0000	EXT_IRQ 0	1	49	n/a
0x0008	Level 0	1,2	48	47
0x0010	IMB_IRQ 0	1,2	48	46
0x0018	IMB_IRQ 1	1,2	48	45
0x0020	IMB_IRQ 2	1,2	48	44
0x0028	IMB_IRQ 3	1,2	48	43
0x0030	EXT_IRQ 1	1,2	48	42
0x0038	Level 1	1,2	48	41
0x0040	IMB_IRQ 4	1,2	48	40
0x0048	IMB_IRQ 5	1,2	48	39
0x0050	IMB_IRQ 6	1,2	48	38

Interrupt Vector Offset	Vector Description	Categories Permitted	Cat 1 Priority	Cat 2 Priority
0x0058	IMB_IRQ 7	1,2	48	37
0x0060	EXT_IRQ 2	1,2	48	36
0x0068	Level 2	1,2	48	35
0x0070	IMB_IRQ 8	1,2	48	34
0x0078	IMB_IRQ 9	1,2	48	33
0x0080	IMB_IRQ 10	1,2	48	32
0x0088	IMB_IRQ 11	1,2	48	31
0x0090	EXT_IRQ 3	1,2	48	30
0x0098	Level 3	1,2	48	29
0x00A0	IMB_IRQ 12	1,2	48	28
0x00A8	IMB_IRQ 13	1,2	48	27
0x00B0	IMB_IRQ 14	1,2	48	26
0x00B8	IMB_IRQ 15	1,2	48	25
0x00C0	EXT_IRQ 4	1,2	48	24
0x00C8	Level 4	1,2	48	23
0x00D0	IMB_IRQ 16	1,2	48	22
0x00D8	IMB_IRQ 17	1,2	48	21
0x00E0	IMB_IRQ 18	1,2	48	20
0x00E8	IMB_IRQ 19	1,2	48	19
0x00F0	EXT_IRQ 5	1,2	48	18
0x00F8	Level 5	1,2	48	17
0x0100	IMB_IRQ 20	1,2	48	16
0x0108	IMB_IRQ 21	1,2	48	15
0x0110	IMB_IRQ 22	1,2	48	14
0x0118	IMB_IRQ 23	1,2	48	13
0x0120	EXT_IRQ 6	1,2	48	12
0x0128	Level 6	1,2	48	11
0x0130	IMB_IRQ 24	1,2	48	10
0x0138	IMB_IRQ 25	1,2	48	9
0x0140	IMB_IRQ 26	1,2	48	8
0x0148	IMB_IRQ 27	1,2	48	7
0x0150	EXT_IRQ 7	1,2	48	6
0x0158	Level 7	1,2	48	5
0x0160	IMB_IRQ 28	1,2	48	4
0x0168	IMB_IRQ 29	1,2	48	3
0x0170	IMB_IRQ 30	1,2	48	2
0x0178	IMB_IRQ 31	1,2	48	1

For a full explanation of the MPC565 interrupt architecture, please refer to the MPC565 Reference Manual sections 4, 6 and in particular 4.5.3.

3.1.6 Category 2 Interrupt Pre-emption

Category 2 interrupts are run with the recoverable exception bit of the machine status register set to 1 (MSR[RI]) to allow debugging exceptions and higher priority interrupts to safely pre-empt them.

3.1.7 Floating-point

Tasks and Category 2 ISRs inherit the MSR[FP] bit setting from the environment they preempt. Thus, if MSR[FP] is initialized to 1 before calling StartOS, all tasks and Category 2 ISRs can use the PowerPC floating-point hardware.

In contrast, any Category 1 interrupt handler that uses the floating-point hardware must re-enable MSR[FP] before using the floating-point hardware. This is because the PowerPC exception handling mechanism disables floating-point.

3.2 Register Settings

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

Register	Required Value	Notes
SIUMCR[EIC] bit	1	
SIUMCR[LPMASK_EN] bit	0	
BBCMCR[EIR] bit	1	See section 3.1.5
MSR[IP]	1	See section 3.1.5
MSR[FP]	0/1	Must be 1 if hardware floating-point is used
EIBADR	Address of EEIR vector table	

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

Register	Notes
SIMASK2	Interrupt mask register.
SIMASK3	Interrupt mask register.
MSR[EE] bit	Global interrupt enable bit.

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): 64

Timing

API max usage (bytes): 80

Extended

API max usage (bytes): 96

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.

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	**Mpc565**
Clock speed (MHz)	40
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	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	255					

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	140	140	140	140	140	140
COM overhead	RAM	2	2	2	2	2	2
	ROM	9	9	9	9	9	9

Timing

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
OS overhead	RAM	44	44	44	44	44	44
	ROM	212	212	212	212	212	212
COM overhead	RAM	2	2	2	2	2	2
	ROM	9	9	9	9	9	9

Extended

Configuration / Events / Shared Task Priorities / Multiple Task Activations		Application Uses					
		No			Yes		
		No		Yes	No		Yes
		No	Yes		No	Yes	
OS overhead	RAM	66	66	66	66	66	66
	ROM	252	252	252	252	252	252
COM overhead	RAM	2	2	2	2	2	2
	ROM	9	9	9	9	9	9

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	40	40	40	40	40	40
BCC1 Heavyweight task	RAM	4	4	4	4	4	4
	ROM	44	44	44	44	44	44
BCC2 task	RAM	n/a	8	10	n/a	8	10
	ROM	n/a	48	56	n/a	48	56
ECC1, Integer task	RAM	n/a	n/a	n/a	268	268	268
	ROM	n/a	n/a	n/a	64	64	64
ECC1, floating-point task	RAM	n/a	n/a	n/a	388	388	388
	ROM	n/a	n/a	n/a	64	64	64
ECC2, Integer task	RAM	n/a	n/a	n/a	n/a	n/a	270
	ROM	n/a	n/a	n/a	n/a	n/a	72
ECC2, floating-point task	RAM	n/a	n/a	n/a	n/a	n/a	390
	ROM	n/a	n/a	n/a	n/a	n/a	72
Category 2 ISR	RAM	0	0	0	0	0	0
	ROM	40	40	40	40	40	40
Category 2 ISR, floating-point	RAM	120	120	120	120	120	120
	ROM	76	76	76	76	76	76
Resource	RAM	0	0	0	0	0	0
	ROM	20	20	20	20	20	20
Internal resource	RAM	0	0	0	0	0	0
	ROM	0	0	0	0	0	0
Linked resource	RAM	0	0	0	0	0	0
	ROM	20	20	20	20	20	20
Alarm	RAM	12	12	12	12	12	12
	ROM	44	44	44	44	44	44
Counter	RAM	4	4	4	4	4	4
	ROM	64	64	64	64	64	64
Message	RAM	11	11	11	51	51	51
	ROM	20	20	20	56	56	56
Flag	RAM	2	2	2	2	2	2
	ROM	1	1	1	1	1	1
Message resource	RAM	0	0	0	0	0	0
	ROM	20	20	20	20	20	20

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

Timing

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
BCC1 Lightweight task	RAM	12	12	12	12	12	12
	ROM	52	52	52	52	52	52
BCC1 Heavyweight task	RAM	16	16	16	16	16	16
	ROM	56	56	56	56	56	56
BCC2 task	RAM	n/a	20	22	n/a	20	22
	ROM	n/a	60	68	n/a	60	68
ECC1, Integer task	RAM	n/a	n/a	n/a	280	280	280
	ROM	n/a	n/a	n/a	76	76	76
ECC1, floating-point task	RAM	n/a	n/a	n/a	400	400	400
	ROM	n/a	n/a	n/a	76	76	76
ECC2, Integer task	RAM	n/a	n/a	n/a	n/a	n/a	282
	ROM	n/a	n/a	n/a	n/a	n/a	84
ECC2, floating-point task	RAM	n/a	n/a	n/a	n/a	n/a	402
	ROM	n/a	n/a	n/a	n/a	n/a	84

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
Category 2 ISR	RAM	12	12	12	12	12	12
	ROM	112	112	112	112	112	112
Category 2 ISR, floating-point	RAM	132	132	132	132	132	132
	ROM	144	144	144	144	144	144
Resource	RAM	0	0	0	0	0	0
	ROM	20	20	20	20	20	20
Internal resource	RAM	0	0	0	0	0	0
	ROM	0	0	0	0	0	0
Linked resource	RAM	0	0	0	0	0	0
	ROM	20	20	20	20	20	20
Alarm	RAM	12	12	12	12	12	12
	ROM	44	44	44	44	44	44
Counter	RAM	4	4	4	4	4	4
	ROM	64	64	64	64	64	64
Message	RAM	11	11	11	51	51	51
	ROM	20	20	20	56	56	56
Flag	RAM	2	2	2	2	2	2
	ROM	1	1	1	1	1	1
Message resource	RAM	0	0	0	0	0	0
	ROM	20	20	20	20	20	20
Event	RAM	0	0	0	0	0	0
	ROM	4	4	4	4	4	4
Priority level	RAM	0	0	6	0	6	6
	ROM	0	0	12	0	12	12
Arrivalpoint (readonly)	RAM	0	0	0	0	0	0
	ROM	12	12	12	12	12	12
Arrivalpoint (writable)	RAM	12	12	12	12	12	12
	ROM	12	12	12	12	12	12
Schedule	RAM	16	16	16	16	16	16
	ROM	36	36	36	36	36	36
Taskset (readonly)	RAM	0	0	0	0	0	0
	ROM	4	4	4	4	4	4
Taskset (writable)	RAM	4	4	4	4	4	4
	ROM	4	4	4	4	4	4

Extended

Configuration Events Shared Task Priorities Multiple Task Activations		Application Uses					
		No			Yes		
		No		Yes	No		Yes
		No	Yes		No	Yes	
BCC1 Lightweight task	RAM	14	14	14	14	14	14
	ROM	60	60	60	60	60	60
BCC1 Heavyweight task	RAM	20	20	20	20	20	20
	ROM	60	60	60	60	60	60
BCC2 task	RAM	n/a	24	26	n/a	24	26
	ROM	n/a	64	72	n/a	64	72
ECC1, Integer task	RAM	n/a	n/a	n/a	284	284	284
	ROM	n/a	n/a	n/a	80	80	80
ECC1, floating-point task	RAM	n/a	n/a	n/a	404	404	404
	ROM	n/a	n/a	n/a	80	80	80
ECC2, Integer task	RAM	n/a	n/a	n/a	n/a	n/a	286
	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	406
	ROM	n/a	n/a	n/a	n/a	n/a	88
Category 2 ISR	RAM	14	14	14	14	14	14
	ROM	120	120	120	120	120	120
Category 2 ISR, floating-point	RAM	134	134	134	134	134	134
	ROM	152	152	152	152	152	152
Resource	RAM	16	16	16	16	16	16
	ROM	28	28	28	28	28	28
Internal resource	RAM	0	0	0	0	0	0
	ROM	0	0	0	0	0	0
Linked resource	RAM	16	16	16	16	16	16
	ROM	28	28	28	28	28	28
Alarm	RAM	12	12	12	12	12	12
	ROM	48	48	48	48	48	48
Counter	RAM	4	4	4	4	4	4
	ROM	68	68	68	68	68	68
Message	RAM	11	11	11	51	51	51
	ROM	24	24	24	60	60	60
Flag	RAM	2	2	2	2	2	2
	ROM	1	1	1	1	1	1
Message resource	RAM	16	16	16	16	16	16
	ROM	28	28	28	28	28	28

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

4.2.3 Size of Linkable Modules

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

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

The call variants are as follows:

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

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

Standard

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
Service name	Variant	Notes						
ActivateTask	SW	1	248	344	400	256	352	432
	NS		200	296	344	208	304	376
	KL	2	88	180	232	96	188	260

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	20	20	20	20	20	20
ChainTask	SWL	1, 8	192	296	344	200	304	384
	SWH	1, 9	236	336	392	244	344	432
	NSL	8	192	296	344	200	304	384
	NSH	9	228	328	384	236	336	424
Schedule			172	172	220	172	172	220
GetTaskID			44	44	44	44	44	44
GetTaskState			132	132	132	156	156	156
EnableAllInterrupts			32	32	32	32	32	32
DisableAllInterrupts			36	36	36	36	36	36
ResumeAllInterrupts			60	60	60	60	60	60
SuspendAllInterrupts			64	64	64	64	64	64
ResumeOSInterrupts			80	80	80	80	80	80
SuspendOSInterrupts			112	112	112	112	112	112
GetResource	Task	7	32	32	40	32	32	40
	Combined	6	148	148	148	148	148	148
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
ReleaseResource	Task	7	176	176	176	176	176	176
	Combined	6	340	340	340	340	340	340
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	1	n/a	n/a	n/a	248	248	336
	NS		n/a	n/a	n/a	160	160	280
	NS1i	10	n/a	n/a	n/a	100	n/a	n/a
	KL	2	n/a	n/a	n/a	76	76	192
	KL1i	2, 10	n/a	n/a	n/a	32	n/a	n/a
ClearEvent			n/a	n/a	n/a	96	96	96
GetEvent			n/a	n/a	n/a	20	20	20
WaitEvent	<default>		n/a	n/a	n/a	352	352	612
	fp	11	n/a	n/a	n/a	400	400	724
	1i	10	n/a	n/a	n/a	24	n/a	n/a
GetAlarmBase			124	124	124	124	124	124
GetAlarm			200	200	200	200	200	200
SetRelAlarm			244	244	244	244	244	244
SetAbsAlarm			256	256	256	256	256	256
CancelAlarm			188	188	188	188	188	188
InitCounter			96	96	96	96	96	96

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
GetCounterValue			140	140	140	140	140	140
osek_tick_alarm	<default>		148	148	148	148	148	148
	KL	2	68	68	68	68	68	68
osek_incr_counter			60	60	60	60	60	60
GetActiveApplicationMode		30	n/a	n/a	n/a	n/a	n/a	n/a
StartOS			284	284	284	284	284	284
ShutdownOS	NoHook	12	92	92	92	92	92	92
	Hook	13	116	116	116	116	116	116
InitCOM			8	8	8	8	8	8
CloseCOM			8	8	8	8	8	8
StartCOM			60	60	60	60	60	60
StopCOM			32	32	32	32	32	32
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	148	148	148	304	304	304
	CCCB	15	304	304	304	304	304	304
GetMessageResource			88	88	88	88	88	88
ReleaseMessageResource			80	80	80	80	80	80
GetMessageStatus			76	76	76	76	76	76
SendMessage	SW CCCA	1, 14	188	188	188	376	376	376
	SW CCCB	1, 15	352	352	352	376	376	376
	NS CCCA	14	188	188	188	376	376	376
	NS CCCB	15	352	352	352	376	376	376
	KL CCCA	2, 14	116	116	116	324	324	324
	KL CCCB	2, 15	300	300	300	324	324	324
main_dispatch	NoHook	12	216	216	276	216	216	276
	Hook	13	264	264	324	264	264	324
sub_dispatch	B1LF	19	48	48	48	48	48	48
	B1HI	20	144	144	144	144	144	144
	B1HF	21	152	152	152	152	152	152
	B2LI	22	n/a	120	160	n/a	120	160
	B2LF	23	n/a	124	164	n/a	124	164
	B2HI	24	n/a	252	340	n/a	252	340
	B2HF	25	n/a	260	348	n/a	260	348
	E1HI	26	n/a	n/a	n/a	504	504	592
	E1HF	27	n/a	n/a	n/a	512	512	600
	E2HI	28	n/a	n/a	n/a	n/a	n/a	592

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	600
ErrorHook support		16	80	80	80	80	80	80
	ServiceID	17	88	88	88	88	88	88
	Parameters	18	108	108	108	108	108	108
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	184	264	312	192	292	348
	NS		104	184	264	112	204	308
	KL	2	24	100	168	32	120	216
ChainTaskset	SWL	1, 8	72	144	212	72	156	236
	SWH	1, 9	136	216	280	136	228	308
	NSL	8	72	144	212	72	156	236
	NSH	9	128	208	272	128	220	300
GetTasksetRef			16	16	16	16	16	16
MergeTaskset			104	104	104	104	104	104
AssignTaskset			16	16	16	16	16	16
RemoveTaskset			104	104	104	104	104	104
TestSubTaskset			124	124	124	124	124	124
TestEquivalentTaskset			128	128	128	128	128	128
TickSchedule	SW	1	280	288	288	288	288	288
	NS		192	236	236	236	236	236
	KL	2	112	152	152	152	152	152
AdvanceSchedule	SW	1	268	244	244	244	244	244
	NS		216	192	192	192	192	192
	KL	2	128	136	136	136	136	136
StartSchedule			192	192	192	192	192	192
StopSchedule			164	164	164	164	164	164
GetScheduleStatus			216	216	216	216	216	216
GetScheduleValue			152	152	152	152	152	152
GetScheduleNext			20	20	20	20	20	20
SetScheduleNext			16	16	16	16	16	16
GetArrivalpointDelay			16	16	16	16	16	16
SetArrivalpointDelay			12	12	12	12	12	12
GetArrivalpointTasksetRef			12	12	12	12	12	12
GetArrivalpointNext			16	16	16	16	16	16
SetArrivalpointNext			12	12	12	12	12	12

Configuration / Events / Shared Task Priorities / Multiple Task Activations			Application Uses					
			No			Yes		
			No		Yes	No		Yes
			No	Yes		No	Yes	
TestArrivalpointWritable			56	56	56	56	56	56
GetExecutionTime			8	8	8	8	8	8
GetLargestExecutionTime			12	12	12	12	12	12
ResetLargestExecutionTime			8	8	8	8	8	8
GetStackOffset			20	20	20	20	20	20
Stack manipulation			116	116	116	116	116	116
Interrupt support			224	224	224	224	224	224

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	248	344	400	256	352	432
	NS		200	296	344	208	304	376
	KL	2	88	180	232	96	188	260
TerminateTask	LExt	3	n/a	n/a	n/a	n/a	n/a	n/a
	H	5	20	20	20	20	20	20
ChainTask	SWL	1, 8	192	296	344	200	304	384
	SWH	1, 9	236	336	392	244	344	432
	NSL	8	192	296	344	200	304	384
	NSH	9	228	328	384	236	336	424
Schedule			200	200	248	200	200	248
GetTaskID			44	44	44	44	44	44
GetTaskState			132	132	132	156	156	156
EnableAllInterrupts			32	32	32	32	32	32
DisableAllInterrupts			36	36	36	36	36	36
ResumeAllInterrupts			60	60	60	60	60	60
SuspendAllInterrupts			64	64	64	64	64	64
ResumeOSInterrupts			80	80	80	80	80	80
SuspendOSInterrupts			112	112	112	112	112	112
GetResource	Task	7	32	32	40	32	32	40
	Combined	6	148	148	148	148	148	148
	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	204	204	204	204	204	204
	Combined	6	396	396	396	396	396	396
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	1	n/a	n/a	n/a	248	248	336
	NS		n/a	n/a	n/a	160	160	280
	NS1i	10	n/a	n/a	n/a	100	n/a	n/a
	KL	2	n/a	n/a	n/a	76	76	192
	KL1i	2, 10	n/a	n/a	n/a	32	n/a	n/a
ClearEvent			n/a	n/a	n/a	96	96	96
GetEvent			n/a	n/a	n/a	20	20	20
WaitEvent	<default>		n/a	n/a	n/a	468	468	748
	fp	11	n/a	n/a	n/a	512	512	848
	1i	10	n/a	n/a	n/a	180	n/a	n/a
GetAlarmBase			124	124	124	124	124	124
GetAlarm			200	200	200	200	200	200
SetRelAlarm			244	244	244	244	244	244
SetAbsAlarm			256	256	256	256	256	256
CancelAlarm			188	188	188	188	188	188
InitCounter			96	96	96	96	96	96
GetCounterValue			140	140	140	140	140	140
osek_tick_alarm	<default>		148	148	148	148	148	148
	KL	2	68	68	68	68	68	68
osek_incr_counter			60	60	60	60	60	60
GetActiveApplicationMode		30	n/a	n/a	n/a	n/a	n/a	n/a
StartOS			336	336	336	336	336	336
ShutdownOS	NoHook	12	92	92	92	92	92	92
	Hook	13	116	116	116	116	116	116
InitCOM			8	8	8	8	8	8
CloseCOM			8	8	8	8	8	8
StartCOM			60	60	60	60	60	60
StopCOM			32	32	32	32	32	32
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	148	148	148	304	304	304
	CCCB	15	304	304	304	304	304	304
GetMessageResource			88	88	88	88	88	88
ReleaseMessageResource			80	80	80	80	80	80

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
GetMessageStatus			76	76	76	76	76	76
SendMessage	SW CCCA	1, 14	188	188	188	376	376	376
	SW CCCB	1, 15	352	352	352	376	376	376
	NS CCCA	14	188	188	188	376	376	376
	NS CCCB	15	352	352	352	376	376	376
	KL CCCA	2, 14	116	116	116	324	324	324
	KL CCCB	2, 15	300	300	300	324	324	324
main_dispatch	NoHook	12	248	248	308	248	248	308
	Hook	13	300	300	360	300	300	360
sub_dispatch	B1LF	19	36	36	36	36	36	36
	B1HI	20	132	132	132	132	132	132
	B1HF	21	140	140	140	140	140	140
	B2LI	22	n/a	84	128	n/a	84	128
	B2LF	23	n/a	88	132	n/a	88	132
	B2HI	24	n/a	184	276	n/a	184	276
	B2HF	25	n/a	192	284	n/a	192	284
	E1HI	26	n/a	n/a	n/a	544	544	632
	E1HF	27	n/a	n/a	n/a	552	552	640
	E2HI	28	n/a	n/a	n/a	n/a	n/a	632
	E2HF	29	n/a	n/a	n/a	n/a	n/a	640
ErrorHook support		16	80	80	80	80	80	80
	ServiceID	17	88	88	88	88	88	88
	Parameters	18	108	108	108	108	108	108
validity_checks		3	n/a	n/a	n/a	n/a	n/a	n/a
Timing_dispatch		4	136	136	136	136	136	136
Timing_termination		4	140	140	140	140	140	140
ActivateTaskset	SW	1	184	264	312	192	292	348
	NS		104	184	264	112	204	308
	KL	2	24	100	168	32	120	216
ChainTaskset	SWL	1, 8	72	144	212	72	156	236
	SWH	1, 9	136	216	280	136	228	308
	NSL	8	72	144	212	72	156	236
	NSH	9	128	208	272	128	220	300
GetTasksetRef			16	16	16	16	16	16
MergeTaskset			104	104	104	104	104	104
AssignTaskset			16	16	16	16	16	16
RemoveTaskset			104	104	104	104	104	104

Configuration / Events / Shared Task Priorities / Multiple Task Activations			Application Uses					
			No			Yes		
			No		Yes	No		Yes
			No	Yes		No	Yes	
TestSubTaskset			124	124	124	124	124	124
TestEquivalentTaskset			128	128	128	128	128	128
TickSchedule	SW	1	280	288	288	288	288	288
	NS		192	236	236	236	236	236
	KL	2	112	152	152	152	152	152
AdvanceSchedule	SW	1	268	244	244	244	244	244
	NS		216	192	192	192	192	192
	KL	2	128	136	136	136	136	136
StartSchedule			192	192	192	192	192	192
StopSchedule			164	164	164	164	164	164
GetScheduleStatus			216	216	216	216	216	216
GetScheduleValue			152	152	152	152	152	152
GetScheduleNext			20	20	20	20	20	20
SetScheduleNext			16	16	16	16	16	16
GetArrivalpointDelay			16	16	16	16	16	16
SetArrivalpointDelay			12	12	12	12	12	12
GetArrivalpointTasksetRef			12	12	12	12	12	12
GetArrivalpointNext			16	16	16	16	16	16
SetArrivalpointNext			12	12	12	12	12	12
TestArrivalpointWritable			56	56	56	56	56	56
GetExecutionTime			200	200	200	200	200	200
GetLargestExecutionTime			24	24	24	24	24	24
ResetLargestExecutionTime			20	20	20	20	20	20
GetStackOffset			20	20	20	20	20	20
Stack manipulation			116	116	116	116	116	116
Interrupt support			224	224	224	224	224	224

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	416	516	560	424	524	596
	NS		512	612	656	520	620	692

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
	KL	2	312	412	456	320	420	492
TerminateTask	LExt	3	216	216	216	216	216	216
	H	5	268	268	268	268	268	268
ChainTask	SWL	1, 8	492	600	644	500	608	688
	SWH	1, 9	540	636	680	548	644	716
	NSL	8	608	716	760	616	724	804
	NSH	9	652	748	792	660	756	828
Schedule			404	404	452	404	404	452
GetTaskID			68	68	68	68	68	68
GetTaskState			392	392	392	400	400	400
EnableAllInterrupts			56	56	56	56	56	56
DisableAllInterrupts			60	60	60	60	60	60
ResumeAllInterrupts			152	152	152	152	152	152
SuspendAllInterrupts			88	88	88	88	88	88
ResumeOSInterrupts			172	172	172	172	172	172
SuspendOSInterrupts			136	136	136	136	136	136
GetResource	Task	7	664	664	604	664	664	604
	Combined	6	656	656	656	656	656	656
	CLEx	3	588	588	588	588	588	588
ReleaseResource	Task	7	592	592	592	592	592	592
	Combined	6	792	792	792	792	792	792
	CLEx	3	552	552	552	552	552	552
SetEvent	SW	1	n/a	n/a	n/a	480	480	604
	NS		n/a	n/a	n/a	568	568	692
	NS1i	10	n/a	n/a	n/a	384	n/a	n/a
	KL	2	n/a	n/a	n/a	392	392	516
	KL1i	2, 10	n/a	n/a	n/a	340	n/a	n/a
ClearEvent			n/a	n/a	n/a	308	308	308
GetEvent			n/a	n/a	n/a	264	264	264
WaitEvent	<default>		n/a	n/a	n/a	672	672	932
	fp	11	n/a	n/a	n/a	716	716	1032
	1i	10	n/a	n/a	n/a	384	n/a	n/a
GetAlarmBase			304	304	304	304	304	304
GetAlarm			308	308	308	308	308	308
SetRelAlarm			396	396	396	396	396	396
SetAbsAlarm			420	420	420	420	420	420
CancelAlarm			288	288	288	288	288	288

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
InitCounter			376	376	376	376	376	376
GetCounterValue			332	332	332	332	332	332
osek_tick_alarm	<default>		236	236	236	236	236	236
	KL	2	68	68	68	68	68	68
osek_incr_counter			60	60	60	60	60	60
GetActiveApplicationMode		30	n/a	n/a	n/a	n/a	n/a	n/a
StartOS			368	368	368	368	368	368
ShutdownOS	NoHook	12	100	100	100	100	100	100
	Hook	13	124	124	124	124	124	124
InitCOM			8	8	8	8	8	8
CloseCOM			8	8	8	8	8	8
StartCOM			92	92	92	92	92	92
StopCOM			68	68	68	68	68	68
ReadFlag			52	52	52	52	52	52
ResetFlag			52	52	52	52	52	52
ReceiveMessage	CCCA	14	284	284	284	452	452	452
	CCCB	15	452	452	452	452	452	452
GetMessageResource			160	160	160	160	160	160
ReleaseMessageResource			160	160	160	160	160	160
GetMessageStatus			184	184	184	184	184	184
SendMessage	SW CCCA	1, 14	344	344	344	544	544	544
	SW CCCB	1, 15	520	520	520	544	544	544
	NS CCCA	14	344	344	344	544	544	544
	NS CCCB	15	520	520	520	544	544	544
	KL CCCA	2, 14	292	292	292	492	492	492
	KL CCCB	2, 15	468	468	468	492	492	492
main_dispatch	NoHook	12	248	248	308	248	248	308
	Hook	13	300	300	360	300	300	360
sub_dispatch	B1LF	19	36	36	36	36	36	36
	B1HI	20	132	132	132	132	132	132
	B1HF	21	140	140	140	140	140	140
	B2LI	22	n/a	84	128	n/a	84	128
	B2LF	23	n/a	88	132	n/a	88	132
	B2HI	24	n/a	184	276	n/a	184	276
	B2HF	25	n/a	192	284	n/a	192	284
	E1HI	26	n/a	n/a	n/a	544	544	632
	E1HF	27	n/a	n/a	n/a	552	552	640

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	632
	E2HF	29	n/a	n/a	n/a	n/a	n/a	640
ErrorHook support		16	240	240	240	240	240	240
	ServiceID	17	248	248	248	248	248	248
	Parameters	18	276	276	276	276	276	276
validity_checks		3	56	56	56	56	56	56
Timing_dispatch		4	136	136	136	136	136	136
Timing_termination		4	140	140	140	140	140	140
ActivateTaskset	SW	1	568	636	696	580	660	764
	NS		656	724	784	668	760	860
	KL	2	460	552	588	476	580	648
ChainTaskset	SWL	1, 8	644	712	772	652	732	824
	SWH	1, 9	720	804	868	728	824	924
	NSL	8	756	832	896	764	852	964
	NSH	9	872	952	1012	880	972	1076
GetTasksetRef			228	228	228	228	228	228
MergeTaskset			508	508	508	508	508	508
AssignTaskset			344	344	344	344	344	344
RemoveTaskset			508	508	508	508	508	508
TestSubTaskset			532	532	532	532	532	532
TestEquivalentTaskset			528	528	528	528	528	528
TickSchedule	SW	1	552	468	468	468	468	468
	NS		640	580	580	580	580	580
	KL	2	460	376	376	376	376	376
AdvanceSchedule	SW	1	568	480	480	480	480	480
	NS		656	596	596	596	596	596
	KL	2	488	400	400	400	400	400
StartSchedule			428	428	428	428	428	428
StopSchedule			344	344	344	344	344	344
GetScheduleStatus			400	400	400	400	400	400
GetScheduleValue			340	340	340	340	340	340
GetScheduleNext			160	160	160	160	160	160
SetScheduleNext			324	324	324	324	324	324
GetArrivalpointDelay			240	240	240	240	240	240
SetArrivalpointDelay			268	268	268	268	268	268
GetArrivalpointTasksetRef			236	236	236	236	236	236
GetArrivalpointNext			240	240	240	240	240	240

Configuration Events Shared Task Priorities Multiple Task Activations			Application Uses					
			No			Yes		
			No		Yes	No		Yes
			No	Yes		No	Yes	
SetArrivalpointNext			372	372	372	372	372	372
TestArrivalpointWritable			272	272	272	272	272	272
GetExecutionTime			272	272	272	272	272	272
GetLargestExecutionTime			184	184	184	184	184	184
ResetLargestExecutionTime			168	168	168	168	168	168
GetStackOffset			20	20	20	20	20	20
Stack manipulation			116	116	116	116	116	116
Interrupt support			224	224	224	224	224	224

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

The collection of performance data for the MPC56x/Diab port of the RTA-OSEK Component was achieved using a timer running two times slower than the CPU clock speed. The figures in this section, therefore, have an uncertainty level of up to two CPU cycles. The actual times are between 0 and two cycles shorter than those reported in the remainder of this section.

4.3.1 Execution Times for RTA-OSEK API Calls

The following tables give the execution time (in CPU cycles) for each API call. (Note that: (1) the OSEK COM class was set to CCCA for systems without events and to CCCB for systems with events; (2) `ShutdownOS()` enters an infinite loop; the execution time for `ShutdownOS()` reported below is the time up to the point at which `ShutdownOS()` calls `ShutdownHook()`).

Standard

Configuration / Events / Shared Task Priorities / Multiple Task Activations		Application Uses					
		No			Yes		
		No		Yes	No		Yes
		No	Yes		No	Yes	
Service	Variant						
ActivateTask	SW	114	152	180	120	138	180
	NS	94	130	156	96	118	150
	KL	32	66	98	32	54	90
TerminateTask	LExt	0	0	0	0	0	0
	H	192	192	194	192	194	194
ChainTask	SWL	262	306	362	290	318	382
	SWH	370	412	468	398	426	486
	NSL	262	308	360	288	318	386
	NSH	370	408	466	394	422	482
Schedule	SW	102	108	110	102	104	114
GetTaskID		24	26	24	22	26	24
GetTaskState		84	86	84	92	92	94
EnableAllInterrupts		14	16	14	14	16	16
DisableAllInterrupts		16	18	16	16	18	18
ResumeAllInterrupts		24	24	22	24	26	24
SuspendAllInterrupts		24	24	22	24	26	24
ResumeOSInterrupts		24	24	22	24	24	26
SuspendOSInterrupts		26	26	24	24	24	26
GetResource	Task	28	30	30	30	30	30
	Combined	82	84	80	82	82	84
	CLEx	n/a	n/a	n/a	n/a	n/a	n/a
ReleaseResource	Task	104	110	106	108	108	108
	Combined	184	188	182	182	182	184
	CLEx	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	n/a	n/a	n/a	116	118	120
	NS	n/a	n/a	n/a	94	98	98
	KL	n/a	n/a	n/a	34	38	38
ClearEvent		n/a	n/a	n/a	72	68	74
GetEvent		n/a	n/a	n/a	18	20	18
WaitEvent	<default>	n/a	n/a	n/a	464	458	496
	fp	n/a	n/a	n/a	478	470	506
GetAlarmBase		100	104	98	100	104	102
GetAlarm		110	112	112	110	112	110

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
SetRelAlarm		118	120	114	118	118	118
SetAbsAlarm		116	116	116	118	114	118
CancelAlarm		96	98	96	94	96	98
InitCounter		84	82	76	80	82	82
GetCounterValue		88	88	86	86	86	88
osek_tick_alarm	<default>	90	96	94	92	94	90
	KL	30	30	30	30	30	32
osek_incr_counter		10	12	8	10	12	12
GetActiveApplicationMode		4	6	2	2	6	4
StartOS		704	674	774	704	738	708
ShutdownOS	NoHook	n/a	n/a	n/a	n/a	n/a	n/a
	Hook	76	78	74	76	78	74
InitCOM		8	8	8	8	10	10
CloseCOM		8	8	6	8	10	10
StartCOM		34	36	34	62	64	62
StopCOM		18	18	16	16	18	18
ReadFlag		n/a	n/a	n/a	12	12	12
ResetFlag		n/a	n/a	n/a	14	14	14
ReceiveMessage		88	92	86	220	218	220
GetMessageResource		n/a	n/a	n/a	86	86	86
ReleaseMessageResource		n/a	n/a	n/a	170	172	174
GetMessageStatus		n/a	n/a	n/a	34	34	36
SendMessage	SW	228	268	298	366	380	424
	NS	208	244	270	340	362	394
	KL	84	120	152	220	242	280
ActivateTaskset	SW	106	480	502	108	442	512
	NS	76	420	532	80	442	482
	KL	14	356	404	18	350	416
	SW2	106	480	502	106	442	512
	NS2	76	420	534	78	442	482
	KL2	14	356	404	18	348	416
ChainTaskset	SWL	242	588	726	270	602	698
	SWH	366	710	850	392	756	818
	NSL	244	588	664	270	666	726
	NSH	364	738	784	394	752	814
GetTasksetRef		16	18	18	16	18	18

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
MergeTaskset		78	82	78	80	80	82
AssignTaskset		12	14	12	12	14	12
RemoveTaskset		76	76	74	76	74	76
TestSubTaskset		80	82	80	80	82	82
TestEquivalentTaskset		80	86	80	82	84	82
TickSchedule	SW	138	492	538	150	496	556
	NS	104	470	514	128	474	534
	KL	42	404	450	64	410	470
	SW2	138	492	538	150	484	548
	NS2	104	472	516	128	462	526
	KL2	44	404	452	64	398	464
AdvanceSchedule	SW	130	476	522	134	482	544
	NS	104	454	500	114	460	522
	KL	40	396	442	58	400	462
	SW2	130	476	526	134	468	536
	NS2	104	454	502	114	446	514
	KL2	40	396	444	58	388	456
StartSchedule		106	108	106	108	108	108
StopSchedule		98	100	100	102	100	100
GetScheduleStatus		106	104	100	100	102	102
GetScheduleValue		98	102	96	98	98	98
GetScheduleNext		14	16	14	14	16	14
SetScheduleNext		12	12	10	12	14	14
GetArrivalpointDelay		16	16	14	16	16	16
SetArrivalpointDelay		12	12	12	12	14	12
GetArrivalpointTasksetRef		12	12	10	12	12	14
GetArrivalpointNext		14	14	14	14	14	14
SetArrivalpointNext		12	14	10	10	14	14
TestArrivalpointWritable		24	24	22	24	24	24
GetExecutionTime		8	8	6	8	10	10
GetLargestExecutionTime		18	18	18	14	18	18
ResetLargestExecutionTime		10	12	8	12	10	10
GetStackOffset		14	14	12	12	14	14

Timing

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
Service	Variant						
ActivateTask	SW	114	154	184	118	138	178
	NS	94	132	162	96	116	150
	KL	32	68	100	36	54	88
TerminateTask	LExt	0	0	0	0	0	0
	H	384	384	384	386	380	382
ChainTask	SWL	494	538	592	534	550	624
	SWH	600	640	696	640	656	728
	NSL	492	536	592	532	550	628
	NSH	600	636	698	642	652	726
Schedule	SW	106	108	112	110	104	112
GetTaskID		26	26	24	28	24	24
GetTaskState		86	86	86	92	90	92
EnableAllInterrupts		16	18	18	16	16	16
DisableAllInterrupts		18	18	18	18	16	16
ResumeAllInterrupts		24	26	26	26	22	24
SuspendAllInterrupts		24	26	28	26	24	24
ResumeOSInterrupts		24	24	26	26	24	22
SuspendOSInterrupts		26	28	26	26	26	26
GetResource	Task	30	30	32	30	28	32
	Combined	84	84	84	86	82	80
	CLEx	n/a	n/a	n/a	n/a	n/a	n/a
ReleaseResource	Task	110	112	112	110	108	108
	Combined	182	184	182	188	180	184
	CLEx	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	n/a	n/a	n/a	120	118	120
	NS	n/a	n/a	n/a	100	96	100
	KL	n/a	n/a	n/a	36	36	36
ClearEvent		n/a	n/a	n/a	74	72	70
GetEvent		n/a	n/a	n/a	20	18	18
WaitEvent	<default>	n/a	n/a	n/a	634	628	664
	fp	n/a	n/a	n/a	644	640	668
GetAlarmBase		100	102	104	106	100	104
GetAlarm		108	114	110	112	110	110

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
SetRelAlarm		118	118	120	120	116	118
SetAbsAlarm		118	118	116	120	116	116
CancelAlarm		98	98	100	100	96	100
InitCounter		82	82	86	80	80	80
GetCounterValue		90	90	88	90	86	88
osek_tick_alarm	<default>	90	96	94	96	90	92
	KL	32	30	32	32	30	30
osek_incr_counter		10	12	12	14	10	10
GetActiveApplicationMode		6	4	6	4	2	4
StartOS		1574	1476	1754	1758	1574	1758
ShutdownOS	NoHook	n/a	n/a	n/a	n/a	n/a	n/a
	Hook	76	74	76	80	74	78
InitCOM		8	10	10	10	10	10
CloseCOM		10	10	10	10	8	10
StartCOM		36	38	36	64	62	62
StopCOM		18	18	20	18	16	16
ReadFlag		n/a	n/a	n/a	14	12	12
ResetFlag		n/a	n/a	n/a	14	14	12
ReceiveMessage		92	92	90	218	218	218
GetMessageResource		n/a	n/a	n/a	86	86	82
ReleaseMessageResource		n/a	n/a	n/a	174	168	172
GetMessageStatus		n/a	n/a	n/a	36	34	32
SendMessage	SW	228	270	298	366	380	426
	NS	210	244	274	346	358	398
	KL	84	122	154	220	242	276
ActivateTaskset	SW	106	450	562	108	440	512
	NS	78	418	504	78	440	482
	KL	16	356	470	18	380	418
	SW2	106	450	560	108	440	512
	NS2	76	416	502	78	440	482
	KL2	16	358	470	18	380	418
ChainTaskset	SWL	474	820	930	514	840	972
	SWH	596	938	1046	636	1020	1060
	NSL	476	818	960	514	840	1000
	NSH	596	938	1074	632	1018	1054
GetTasksetRef		18	20	20	20	16	16

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
MergeTaskset		78	78	80	80	80	80
AssignTaskset		14	14	14	16	12	12
RemoveTaskset		76	78	78	76	76	76
TestSubTaskset		80	84	82	84	80	82
TestEquivalentTaskset		82	84	84	88	82	82
TickSchedule	SW	138	492	606	154	524	558
	NS	108	470	584	134	502	534
	KL	44	404	520	66	438	466
	SW2	138	492	604	156	512	556
	NS2	108	470	582	134	490	532
	KL2	44	404	518	66	426	466
AdvanceSchedule	SW	130	478	592	136	512	542
	NS	104	454	566	116	488	520
	KL	42	398	510	60	432	458
	SW2	128	478	590	136	498	540
	NS2	104	454	564	116	478	518
	KL2	42	398	508	60	420	456
StartSchedule		106	108	108	108	106	108
StopSchedule		100	102	100	102	100	100
GetScheduleStatus		102	102	102	108	102	108
GetScheduleValue		100	98	98	100	100	98
GetScheduleNext		16	16	16	18	14	14
SetScheduleNext		14	16	14	14	14	14
GetArrivalpointDelay		16	18	16	18	16	16
SetArrivalpointDelay		14	14	14	14	12	12
GetArrivalpointTasksetRef		12	14	14	16	12	12
GetArrivalpointNext		12	16	16	14	14	14
SetArrivalpointNext		14	14	14	14	10	10
TestArrivalpointWritable		26	24	24	26	24	24
GetExecutionTime		126	128	128	128	130	126
GetLargestExecutionTime		24	28	28	26	22	24
ResetLargestExecutionTime		18	18	20	18	18	16
GetStackOffset		14	16	16	14	12	12

Extended

Configuration / Events / Shared Task Priorities / Multiple Task Activations		Application Uses					
		No			Yes		
		No		Yes	No		Yes
		No	Yes		No	Yes	
Service	Variant						
ActivateTask	SW	342	370	404	318	366	402
	NS	376	408	448	356	404	442
	KL	272	300	338	246	294	330
TerminateTask	LExt	430	440	428	428	432	434
	H	516	516	512	514	518	518
ChainTask	SWL	822	854	908	828	886	952
	SWH	918	946	1008	926	980	1040
	NSL	866	896	952	872	924	996
	NSH	960	982	1040	960	1016	1084
Schedule	SW	178	174	180	178	180	186
GetTaskID		30	34	30	32	32	32
GetTaskState		332	322	330	312	334	336
EnableAllInterrupts		24	24	20	24	24	24
DisableAllInterrupts		24	24	22	26	24	24
ResumeAllInterrupts		44	46	42	44	46	46
SuspendAllInterrupts		34	30	30	32	32	34
ResumeOSInterrupts		44	46	44	44	46	46
SuspendOSInterrupts		32	32	30	32	32	34
GetResource	Task	540	534	334	568	614	380
	Combined	346	338	342	376	388	386
	CLEx	342	336	340	368	394	392
ReleaseResource	Task	316	316	318	346	366	366
	Combined	378	376	382	410	432	428
	CLEx	294	294	298	326	348	344
SetEvent	SW	n/a	n/a	n/a	332	360	362
	NS	n/a	n/a	n/a	356	382	382
	KL	n/a	n/a	n/a	270	298	300
ClearEvent		n/a	n/a	n/a	146	146	144
GetEvent		n/a	n/a	n/a	224	248	250
WaitEvent	<default>	n/a	n/a	n/a	748	762	776
	fp	n/a	n/a	n/a	760	772	782
GetAlarmBase		274	270	272	256	274	272
GetAlarm		272	264	266	252	272	270

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
SetRelAlarm		300	298	296	282	296	300
SetAbsAlarm		294	286	290	276	292	296
CancelAlarm		254	250	252	236	254	252
InitCounter		410	388	406	394	402	392
GetCounterValue		248	242	246	234	248	248
osek_tick_alarm	<default>	130	134	130	132	134	134
	KL	30	32	30	30	30	32
osek_incr_counter		8	12	8	10	8	10
GetActiveApplicationMode		4	4	2	4	6	4
StartOS		1632	1530	1536	1636	1534	1730
ShutdownOS	NoHook	n/a	n/a	n/a	n/a	n/a	n/a
	Hook	78	76	74	80	82	80
InitCOM		10	10	8	10	8	10
CloseCOM		10	8	8	8	12	10
StartCOM		46	46	44	74	74	76
StopCOM		28	26	24	28	28	30
ReadFlag		n/a	n/a	n/a	28	30	30
ResetFlag		n/a	n/a	n/a	28	26	28
ReceiveMessage		210	204	206	326	332	330
GetMessageResource		n/a	n/a	n/a	498	522	520
ReleaseMessageResource		n/a	n/a	n/a	480	514	508
GetMessageStatus		n/a	n/a	n/a	124	126	130
SendMessage	SW	570	596	634	670	728	760
	NS	608	632	672	708	766	798
	KL	452	478	520	544	598	644
ActivateTaskset	SW	598	914	992	568	868	1002
	NS	598	980	1060	632	938	980
	KL	486	842	892	490	828	932
	SW2	598	914	992	568	868	1000
	NS2	598	980	1058	632	938	980
	KL2	486	840	892	490	828	932
ChainTaskset	SWL	1056	1438	1534	1118	1422	1588
	SWH	1124	1538	1580	1188	1526	1616
	NSL	1090	1450	1546	1094	1530	1564
	NSH	1168	1610	1644	1200	1592	1670
GetTasksetRef		236	226	230	206	232	234

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
MergeTaskset		200	202	198	202	204	202
AssignTaskset		98	98	94	98	98	96
RemoveTaskset		196	196	196	194	198	198
TestSubTaskset		206	204	206	202	210	206
TestEquivalentTaskset		202	202	202	206	208	206
TickSchedule	SW	214	1012	1066	664	1028	1122
	NS	242	1040	1094	690	1056	1154
	KL	140	942	998	590	954	1054
	SW2	214	1014	1066	664	1002	1104
	NS2	242	1040	1096	690	1032	1136
	KL2	140	942	998	592	928	1036
AdvanceSchedule	SW	208	1010	1064	658	1024	1120
	NS	242	1042	1096	690	1054	1150
	KL	140	940	992	588	952	1050
	SW2	208	1010	1064	658	1000	1102
	NS2	240	1042	1096	692	1028	1132
	KL2	140	940	992	588	926	1032
StartSchedule		178	176	174	178	182	178
StopSchedule		152	154	150	152	154	152
GetScheduleStatus		160	160	160	160	164	166
GetScheduleValue		162	160	162	162	160	162
GetScheduleNext		46	46	44	46	46	48
SetScheduleNext		88	86	84	86	88	90
GetArrivalpointDelay		70	68	66	66	66	68
SetArrivalpointDelay		78	76	72	76	78	74
GetArrivalpointTasksetRef		52	54	50	54	52	54
GetArrivalpointNext		54	54	52	52	54	56
SetArrivalpointNext		100	100	98	102	98	100
TestArrivalpointWritable		62	64	60	64	62	64
GetExecutionTime		166	164	162	162	168	164
GetLargestExecutionTime		216	208	216	192	216	218
ResetLargestExecutionTime		208	200	204	184	208	208
GetStackOffset		14	14	12	14	14	14

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	47	47	47	47	47	47
	Cat 2	109	113	109	111	109	113

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	47	47	47	47	47	47
	Cat 2	237	235	233	237	237	237

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	47	47	47	47	47	47
	Cat 2	231	235	235	233	237	237

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.

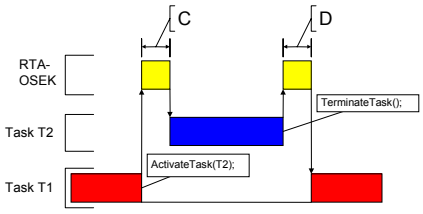

Figure 1: Task Activates a Higher Priority Task which Terminates Normally

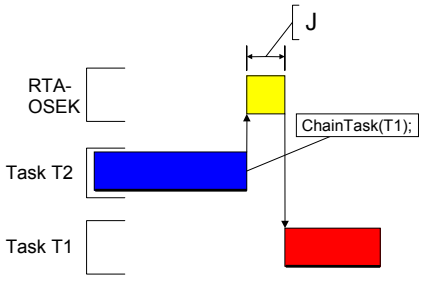

Figure 2: Task Chaining

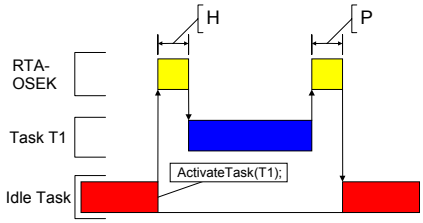

Figure 3: Task Activation from Idle Task

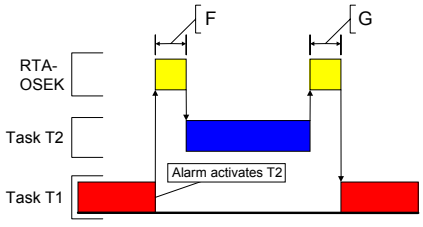

Figure 4: Task Activation from an Alarm

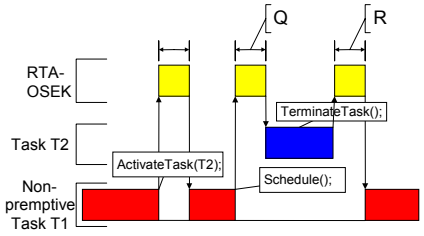

Figure 5: Non-Premptive Task Calls Schedule()

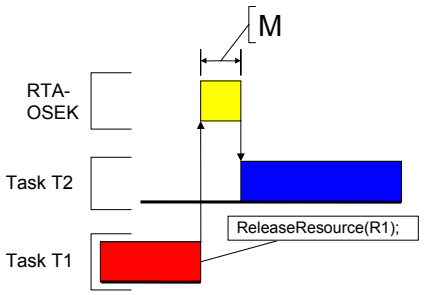

Figure 6: Blocked Task Activated by ReleaseResource()

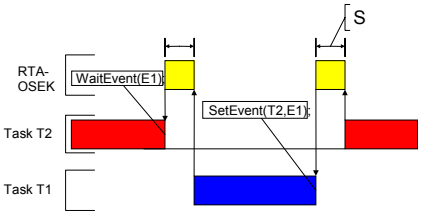

Figure 7: Waiting Task Activated by SetEvent()

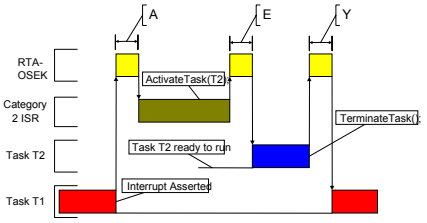

Figure 8: Category 2 ISR Activates a Higher Priority Task

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	92	142	160	94	140	168
Figure 1: D	Heavy, Basic/Extended	190	238	256	242	238	262
ChainTask	Light, Basic	172	228	282	170	226	290
Figure 2: J	Heavy, Basic/Extended	500	606	678	552	602	692
Pre-emption	Light, Basic	168	224	286	172	220	292
Figure 1: C	Heavy, Basic/Extended	266	308	370	296	324	394
From idle task	Light, Basic	168	226	286	172	222	292
Figure 3: H	Heavy, Basic/Extended	268	310	370	296	324	394
Triggered by alarm	Light, Basic	268	326	390	274	322	392
Figure 4: F	Heavy, Basic/Extended	368	412	474	398	424	492
Schedule	Light, Basic	150	172	208	148	168	208
Figure 5: Q	Heavy, Basic/Extended	250	256	294	274	282	320
Release resource	Light, Basic	184	204	232	182	198	234
Figure 6: M	Heavy, Basic/Extended	282	288	316	306	312	348
SetEvent							
Figure 7: S	Heavy, Extended	n/a	n/a	n/a	488	492	586

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	136	156	186	134	154	188
Figure 8: E	Heavy, Basic/Extended	236	240	270	260	268	300

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	284	312	334	288	306	334
Figure 1: D	Heavy, Basic/Extended	382	396	422	406	402	428
ChainTask	Light, Basic	412	460	510	414	452	526
Figure 2: J	Heavy, Basic/Extended	932	990	1064	958	990	1086
Pre-emption	Light, Basic	280	326	392	286	322	404
Figure 1: C	Heavy, Basic/Extended	370	408	476	410	426	510
From idle task	Light, Basic	280	326	392	286	322	402
Figure 3: H	Heavy, Basic/Extended	370	406	476	410	426	508
Triggered by alarm	Light, Basic	380	430	494	388	424	502
Figure 4: F	Heavy, Basic/Extended	470	512	578	512	526	610
Schedule	Light, Basic	258	268	308	266	264	314
Figure 5: Q	Heavy, Basic/Extended	348	350	394	390	382	434
Release resource	Light, Basic	290	300	336	298	298	340
Figure 6: M	Heavy, Basic/Extended	380	384	422	422	416	462
SetEvent							
Figure 7: S	Heavy, Extended	n/a	n/a	n/a	562	562	658
From category 2 ISR	Light, Basic	444	454	490	448	452	490
Figure 8: E	Heavy, Basic/Extended	534	538	574	570	570	610

Extended

Configuration Events Shared Task Priorities Multiple Task Activations	Task Attributes	Application Uses					
		No			Yes		
		No		Yes	No		Yes
		No	Yes		No	Yes	
Normal termination	Light, Basic	428	460	470	424	452	474
Figure 1: D	Heavy, Basic/Extended	516	532	552	532	540	560
ChainTask	Light, Basic	736	778	832	714	794	850
Figure 2: J	Heavy, Basic/Extended	1376	1430	1516	1372	1454	1528
Pre-emption	Light, Basic	490	532	596	468	544	614
Figure 1: C	Heavy, Basic/Extended	584	614	680	594	642	720
From idle task	Light, Basic	490	534	596	470	544	612
Figure 3: H	Heavy, Basic/Extended	584	616	678	596	644	718
Triggered by alarm	Light, Basic	632	676	736	610	688	756
Figure 4: F	Heavy, Basic/Extended	726	758	822	736	786	862
Schedule	Light, Basic	316	324	362	320	328	368
Figure 5: Q	Heavy, Basic/Extended	410	406	446	446	444	488
Release resource	Light, Basic	482	492	524	512	544	578
Figure 6: M	Heavy, Basic/Extended	576	576	608	638	664	700
SetEvent							
Figure 7: S	Heavy, Extended	n/a	n/a	n/a	768	794	886
From category 2 ISR	Light, Basic	480	490	516	478	490	524
Figure 8: E	Heavy, Basic/Extended	572	574	598	604	608	646

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	160	160	160	160	160	160
BCC1 lightweight, floating-point	176	176	176	176	176	176
BCC1 heavyweight, integer	432	432	432	432	432	432
BCC1 heavyweight, floating-point	432	432	432	432	432	432
BCC2 lightweight, integer	n/a	176	176	n/a	176	176
BCC2 lightweight, floating-point	n/a	176	176	n/a	176	176
BCC2 heavyweight, integer	n/a	432	432	n/a	432	432
BCC2 heavyweight, floating-point	n/a	432	432	n/a	432	432
ECC1 heavyweight, integer	n/a	n/a	n/a	448	448	448
ECC1 heavyweight, floating-point	n/a	n/a	n/a	448	448	448
ECC2 heavyweight, integer	n/a	n/a	n/a	n/a	n/a	448
ECC2 heavyweight, floating-point	n/a	n/a	n/a	n/a	n/a	448
Pre- and/or Post-Task hooks used						
Task type						
BCC1 lightweight, integer	160	160	176	160	160	176
BCC1 lightweight, floating-point	176	176	192	176	176	192
BCC1 heavyweight, integer	432	432	448	432	432	448
BCC1 heavyweight, floating-point	432	432	448	432	432	448
BCC2 lightweight, integer	n/a	176	192	n/a	176	192
BCC2 lightweight, floating-point	n/a	176	192	n/a	176	192
BCC2 heavyweight, integer	n/a	432	448	n/a	432	448
BCC2 heavyweight, floating-point	n/a	432	448	n/a	432	448
ECC1 heavyweight, integer	n/a	n/a	n/a	448	448	464
ECC1 heavyweight, floating-point	n/a	n/a	n/a	448	448	464
ECC2 heavyweight, integer	n/a	n/a	n/a	n/a	n/a	464
ECC2 heavyweight, floating-point	n/a	n/a	n/a	n/a	n/a	464

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		192	192	192	192	192	192
BCC1 lightweight, floating-point		208	208	208	208	208	208
BCC1 heavyweight, integer		464	464	464	464	464	464
BCC1 heavyweight, floating-point		464	464	464	464	464	464
BCC2 lightweight, integer		n/a	208	208	n/a	208	208
BCC2 lightweight, floating-point		n/a	208	208	n/a	208	208
BCC2 heavyweight, integer		n/a	464	464	n/a	464	464
BCC2 heavyweight, floating-point		n/a	464	464	n/a	464	464
ECC1 heavyweight, integer		n/a	n/a	n/a	480	480	480
ECC1 heavyweight, floating-point		n/a	n/a	n/a	480	480	480
ECC2 heavyweight, integer		n/a	n/a	n/a	n/a	n/a	480
ECC2 heavyweight, floating-point		n/a	n/a	n/a	n/a	n/a	480
Pre- and/or Post-Task hooks used							
Task type							
BCC1 lightweight, integer		192	192	192	192	192	192
BCC1 lightweight, floating-point		208	208	208	208	208	208
BCC1 heavyweight, integer		464	464	464	464	464	464
BCC1 heavyweight, floating-point		464	464	464	464	464	464
BCC2 lightweight, integer		n/a	208	208	n/a	208	208
BCC2 lightweight, floating-point		n/a	208	208	n/a	208	208
BCC2 heavyweight, integer		n/a	464	464	n/a	464	464
BCC2 heavyweight, floating-point		n/a	464	464	n/a	464	464
ECC1 heavyweight, integer		n/a	n/a	n/a	480	480	480
ECC1 heavyweight, floating-point		n/a	n/a	n/a	480	480	480
ECC2 heavyweight, integer		n/a	n/a	n/a	n/a	n/a	480
ECC2 heavyweight, floating-point		n/a	n/a	n/a	n/a	n/a	480

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		192	192	192	192	192	192
BCC1 lightweight, floating-point		208	208	208	208	208	208
BCC1 heavyweight, integer		464	464	464	464	464	464
BCC1 heavyweight, floating-point		464	464	464	464	464	464
BCC2 lightweight, integer		n/a	208	208	n/a	208	208
BCC2 lightweight, floating-point		n/a	208	208	n/a	208	208
BCC2 heavyweight, integer		n/a	464	464	n/a	464	464
BCC2 heavyweight, floating-point		n/a	464	464	n/a	464	464
ECC1 heavyweight, integer		n/a	n/a	n/a	480	480	480
ECC1 heavyweight, floating-point		n/a	n/a	n/a	480	480	480
ECC2 heavyweight, integer		n/a	n/a	n/a	n/a	n/a	480
ECC2 heavyweight, floating-point		n/a	n/a	n/a	n/a	n/a	480
Pre- and/or Post-Task hooks used							
Task type							
BCC1 lightweight, integer		192	192	192	192	192	192
BCC1 lightweight, floating-point		208	208	208	208	208	208
BCC1 heavyweight, integer		464	464	464	464	464	464
BCC1 heavyweight, floating-point		464	464	464	464	464	464
BCC2 lightweight, integer		n/a	208	208	n/a	208	208
BCC2 lightweight, floating-point		n/a	208	208	n/a	208	208
BCC2 heavyweight, integer		n/a	464	464	n/a	464	464
BCC2 heavyweight, floating-point		n/a	464	464	n/a	464	464
ECC1 heavyweight, integer		n/a	n/a	n/a	480	480	480
ECC1 heavyweight, floating-point		n/a	n/a	n/a	480	480	480
ECC2 heavyweight, integer		n/a	n/a	n/a	n/a	n/a	480
ECC2 heavyweight, floating-point		n/a	n/a	n/a	n/a	n/a	480

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.