RTA-OSEK

Binding Manual: MicroBlaze/GCC

Contact Details

ETAS Group

www.etasgroup.com

Germany

ETAS GmbH
Borsigstraße 14
70469 Stuttgart

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

www.etas.de

USA

ETAS Inc.
3021 Miller Road
Ann Arbor, MI 48103

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

www.etasinc.com

Japan

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

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

www.etas.co.jp

France

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

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

www.etas.fr

Korea

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

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

www.etas.co.kr

Great Britain

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

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

www.etas-uk.net

Copyright Notice

Version: RM00073-001

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 MicroBlaze/GCC 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	GNU / Xilinx
Compiler	mb-gcc
Version	(GCC) 3.4.1 (Xilinx EDK 7.1.2 Build EDK_H.12.4)

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.

2.2 Assembler

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

Vendor	GNU/Xilinx
Assembler	mb-as
Version	GNU assembler 2.10.1 Xilinx EDK 7.1.1 Build EDK_H.11.3

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

2.3 Linker/Locator

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

Option	Description
-b elf32-microblaze	set output format

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_pur`	RAM	RTA-OSEK uninitialized data
`os_pir`	RAM	RTA-OSEK initialized data

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 Xilinx MicroBlaze 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.

2.5 Toolchain notes

mb-gdb does not reset the target hardware when it loads an executable image. If the interrupt controller has been initialized by a previous program execution and its HIE bit has been set then it may fail to initialize correctly. To work around this, issue a `rst` or `xreset` command at the XMD prompt after loading. The example application shipped with RTA-OSEK contains checks for this situation: If the application initialization code finds INTC is not correctly reset, it writes INTC_MER to the LEDs and enters a non-terminating loop.

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 *MicroBlaze Processor Reference Guide*.

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

IPL Value	Intc_Ier	Description
0	0xFFFFFFFF	User Level
1..31	os_uim & (0xFFFFFFFF >> IPL)	Category 1 and 2 interrupts
32	0	Category 1 handlers possible on exception or break

3.1.2 Interrupt Vectors

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

Vector	Legality
0x1000..0x101F	INTC handles Category 1 and Category 2 interrupts

The valid base addresses for the vector table are:

Base Address	Notes
0x0000	The core requires a small vector table at 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 GNU / Xilinx C compiler can generate appropriate interrupt handling code for a C

function decorated with the 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

The MicroBlaze reset vector is at location 0 and there is space for two machine instructions here.

RTA-OSEK requires that the MicroBlaze interrupt vector location 0x10 contains a branch to `_interrupt` which is an entry-point in the kernel library.

The `crt0.o` startup code provided with the compiler satisfies these requirements.

3.1.6 Handling multiple INTC interrupt sources

The INTC allows 32 external interrupt sources to be multiplexed onto the single MicroBlaze interrupt line. RTA-OSEK supports these interrupts with the interrupt handler function `_interrupt(void)` in the kernel. This function uses the interrupt handler stack convention and is the only function that needs to be compiled as such. When an interrupt occurs, `_interrupt()` reads the `INTC_IVR` register to determine which is the highest pending interrupt. Control then passes to a small kernel wrapper and finally to the application's Interrupt Service Routine.

For this reason, all user-supplied interrupt service routines for INTC interrupts must be regular C functions and must **not** be decorated with `__attribute__((interrupt_handler))`.

3.1.7 Break and Hardware Exception

RTA-OSEK assumes that the Break vector at 0x18 is only for use by an in-circuit debugger and no user code will appear at that vector.

RTA-OSEK assumes that the Hardware Exception vector at 0x20 will only be taken in the case of a serious error in application code and will result in a restart of the processor, or in hitting a debug breakpoint.

The user may implement code here but it is subject to the rules for a Category 1 ISR.

Because hardware exception will be taken only in extraordinary circumstances, extended build API calls do not check whether they are being called from the exception handler. This avoids an unnecessary reduction of kernel performance.

3.2 Register Settings

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

Register	Required Value
INTC_IER	os_oim if Category 1 interrupts are to be enabled throughout OS startup, or Set to 0 if Category 1 interrupts are to be enabled by the OS.
INTC_MER	3
MSR	Set IE bit

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

Registers Used	Notes
INTC_IER	RTA-OSEK manages the masking of interrupts. Application code must not write to this register.
INTC_MER	Application code must not alter INTC_MER
MSR.IE	Application code must not alter MSR.IE. Application ISRs must be implemented as regular C functions

3.3 Stack Usage

3.3.1 Number of Stacks

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

`osStackOffsetType` is a scalar, representing the number of bytes on the stack, with C type: `osUInt32Type` .

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): 112

Timing

API max usage (bytes): 112

Extended

API max usage (bytes): 160

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	**MicroBlaze**
Clock speed (MHz)	100
Code memory	On-chip RAM
Read-only data memory	On-chip RAM
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	255	255	255	255	255	255

Configuration	Application Uses					
Events	No			Yes		
Shared Task Priorities	No		Yes	No		Yes
Multiple Task Activations	No	Yes		No	Yes	
(per system / per task)						
Limits for the number of application modes	65535					

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	26	26	26	26	26	26
	ROM	184	184	184	184	184	184
COM overhead	RAM	4	4	4	4	4	4
	ROM	12	12	12	12	12	12

Timing

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
OS overhead	RAM	46	46	46	46	46	46
	ROM	256	256	256	256	256	256
COM overhead	RAM	4	4	4	4	4	4
	ROM	12	12	12	12	12	12

Extended

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
OS overhead	RAM	60	60	60	60	60	60
	ROM	300	300	300	300	300	300
COM overhead	RAM	4	4	4	4	4	4
	ROM	12	12	12	12	12	12

4.2.2 ROM and RAM for OSEK OS Objects

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

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

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

Standard

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
BCC1 Lightweight task	RAM	0	0	0	0	0	0
	ROM	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	76	76	76
	ROM	n/a	n/a	n/a	60	60	60
ECC1, floating-point task	RAM	n/a	n/a	n/a	78	78	78
	ROM	n/a	n/a	n/a	60	60	60
ECC2, Integer task	RAM	n/a	n/a	n/a	n/a	n/a	78
	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	80
	ROM	n/a	n/a	n/a	n/a	n/a	68
Category 2 ISR	RAM	0	0	0	0	0	0
	ROM	52	52	52	52	52	52
Category 2 ISR, floating-point	RAM	1	1	1	1	1	1
	ROM	108	108	108	108	108	108
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	84	84	84	84	84	84
Counter	RAM	4	4	4	4	4	4
	ROM	0	0	0	0	0	0
Message	RAM	11	11	11	51	51	51
	ROM	20	20	20	56	56	56
Flag	RAM	4	4	4	4	4	4
	ROM	2	2	2	2	2	2
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	48	48	48	48	48	48
BCC1 Heavyweight task	RAM	16	16	16	16	16	16
	ROM	52	52	52	52	52	52
BCC2 task	RAM	n/a	20	22	n/a	20	22
	ROM	n/a	56	64	n/a	56	64
ECC1, Integer task	RAM	n/a	n/a	n/a	88	88	88
	ROM	n/a	n/a	n/a	72	72	72
ECC1, floating-point task	RAM	n/a	n/a	n/a	90	90	90
	ROM	n/a	n/a	n/a	72	72	72
ECC2, Integer task	RAM	n/a	n/a	n/a	n/a	n/a	90
	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	92
	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	12	12	12	12	12	12
	ROM	160	160	160	160	160	160
Category 2 ISR, floating-point	RAM	14	14	14	14	14	14
	ROM	184	184	184	184	184	184
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	84	84	84	84	84	84
Counter	RAM	4	4	4	4	4	4
	ROM	0	0	0	0	0	0
Message	RAM	11	11	11	51	51	51
	ROM	20	20	20	56	56	56
Flag	RAM	4	4	4	4	4	4
	ROM	2	2	2	2	2	2
Message resource	RAM	0	0	0	0	0	0
	ROM	20	20	20	20	20	20
Event	RAM	0	0	0	0	0	0
	ROM	4	4	4	4	4	4
Priority level	RAM	0	0	6	0	6	6
	ROM	0	0	12	0	12	12
Arrivalpoint (readonly)	RAM	0	0	0	0	0	0
	ROM	12	12	12	12	12	12
Arrivalpoint (writable)	RAM	12	12	12	12	12	12
	ROM	12	12	12	12	12	12
Schedule	RAM	16	16	16	16	16	16
	ROM	36	36	36	36	36	36
Taskset (readonly)	RAM	0	0	0	0	0	0
	ROM	4	4	4	4	4	4
Taskset (writable)	RAM	4	4	4	4	4	4
	ROM	4	4	4	4	4	4

Extended

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
BCC1 Lightweight task	RAM	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	92	92	92
	ROM	n/a	n/a	n/a	80	80	80
ECC1, floating-point task	RAM	n/a	n/a	n/a	94	94	94
	ROM	n/a	n/a	n/a	80	80	80
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	88
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	88
Category 2 ISR	RAM	14	14	14	14	14	14
	ROM	172	172	172	172	172	172
Category 2 ISR, floating-point	RAM	16	16	16	16	16	16
	ROM	196	196	196	196	196	196
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	12	12	12	12	12	12
	ROM	88	88	88	88	88	88
Counter	RAM	4	4	4	4	4	4
	ROM	4	4	4	4	4	4
Message	RAM	11	11	11	51	51	51
	ROM	24	24	24	60	60	60
Flag	RAM	4	4	4	4	4	4
	ROM	2	2	2	2	2	2
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	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	292	376	476	304	388	520
	NS		240	332	432	260	344	476
	KL	2	104	200	300	124	212	344

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	48	48	48	48	48	48
ChainTask	SWL	1, 8	228	324	424	248	336	468
	SWH	1, 9	296	396	500	316	408	544
	NSL	8	228	324	424	248	336	468
	NSH	9	284	380	484	304	392	528
Schedule			204	204	276	204	204	276
GetTaskID			60	60	60	60	60	60
GetTaskState			160	160	160	196	196	196
EnableAllInterrupts			52	52	52	52	52	52
DisableAllInterrupts			60	60	60	60	60	60
ResumeAllInterrupts			84	84	84	84	84	84
SuspendAllInterrupts			92	92	92	92	92	92
ResumeOSInterrupts			84	84	84	84	84	84
SuspendOSInterrupts			112	112	112	112	112	112
GetResource	Task	7	44	44	52	44	44	52
	Combined	6	140	140	140	140	140	140
	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	428	428	428	428	428	428
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	1	n/a	n/a	n/a	268	268	428
	NS		n/a	n/a	n/a	200	200	360
	NS1i	10	n/a	n/a	n/a	116	n/a	n/a
	KL	2	n/a	n/a	n/a	112	112	272
	KL1i	2, 10	n/a	n/a	n/a	32	n/a	n/a
ClearEvent			n/a	n/a	n/a	124	124	124
GetEvent			n/a	n/a	n/a	20	20	20
WaitEvent	<default>		n/a	n/a	n/a	448	448	844
	fp	11	n/a	n/a	n/a	504	504	956
	1i	10	n/a	n/a	n/a	32	n/a	n/a
GetAlarmBase			120	120	120	120	120	120
GetAlarm			192	192	192	192	192	192
SetRelAlarm			240	240	240	240	240	240
SetAbsAlarm			288	288	288	288	288	288
CancelAlarm			180	180	180	180	180	180
InitCounter			124	124	124	124	124	124

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
GetCounterValue			152	152	152	152	152	152
osek_tick_alarm	<default>		148	148	148	148	148	148
	KL	2	56	56	56	56	56	56
osek_incr_counter			56	56	56	56	56	56
GetActiveApplicationMode		30	n/a	n/a	n/a	n/a	n/a	n/a
StartOS			272	272	272	272	272	272
ShutdownOS	NoHook	12	64	64	64	64	64	64
	Hook	13	88	88	88	88	88	88
InitCOM			8	8	8	8	8	8
CloseCOM			8	8	8	8	8	8
StartCOM			64	64	64	64	64	64
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	156	156	156	420	420	420
	CCCB	15	420	420	420	420	420	420
GetMessageResource			88	88	88	88	88	88
ReleaseMessageResource			76	76	76	76	76	76
GetMessageStatus			156	156	156	156	156	156
SendMessage	SW CCCA	1, 14	204	204	204	452	452	452
	SW CCCB	1, 15	428	428	428	452	452	452
	NS CCCA	14	204	204	204	452	452	452
	NS CCCB	15	428	428	428	452	452	452
	KL CCCA	2, 14	104	104	104	356	356	356
	KL CCCB	2, 15	332	332	332	356	356	356
main_dispatch	NoHook	12	292	292	372	292	292	372
	Hook	13	380	380	460	380	380	460
sub_dispatch	B1LF	19	64	64	64	64	64	64
	B1HI	20	172	172	172	172	172	172
	B1HF	21	196	196	196	196	196	196
	B2LI	22	n/a	160	232	n/a	160	232
	B2LF	23	n/a	184	256	n/a	184	256
	B2HI	24	n/a	340	508	n/a	340	508
	B2HF	25	n/a	364	532	n/a	364	532
	E1HI	26	n/a	n/a	n/a	700	700	872
	E1HF	27	n/a	n/a	n/a	724	724	896
	E2HI	28	n/a	n/a	n/a	n/a	n/a	872

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
	E2HF	29	n/a	n/a	n/a	n/a	n/a	896
ErrorHook support		16	76	76	76	76	76	76
	ServiceID	17	92	92	92	92	92	92
	Parameters	18	116	116	116	116	116	116
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	188	264	372	204	296	408
	NS		116	196	320	132	228	340
	KL	2	32	108	216	48	140	252
ChainTaskset	SWL	1, 8	104	176	280	104	192	300
	SWH	1, 9	208	288	396	208	304	416
	NSL	8	104	176	280	104	192	300
	NSH	9	196	276	384	196	292	404
GetTasksetRef			16	16	16	16	16	16
MergeTaskset			108	108	108	108	108	108
AssignTaskset			16	16	16	16	16	16
RemoveTaskset			112	112	112	112	112	112
TestSubTaskset			128	128	128	128	128	128
TestEquivalentTaskset			124	124	124	124	124	124
TickSchedule	SW	1	308	304	304	304	304	304
	NS		232	252	252	252	252	252
	KL	2	144	164	164	164	164	164
AdvanceSchedule	SW	1	276	288	288	288	288	288
	NS		224	236	236	236	236	236
	KL	2	132	144	144	144	144	144
StartSchedule			176	176	176	176	176	176
StopSchedule			148	148	148	148	148	148
GetScheduleStatus			196	196	196	196	196	196
GetScheduleValue			184	184	184	184	184	184
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			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
TestArrivalpointWritable			52	52	52	52	52	52
GetExecutionTime			8	8	8	8	8	8
GetLargestExecutionTime			12	12	12	12	12	12
ResetLargestExecutionTime			8	8	8	8	8	8
GetStackOffset			60	60	60	60	60	60
Interrupt support			340	340	340	340	340	340
longjmp/setjmp support			160	160	160	160	160	160
Stack Manipulation support			72	72	72	72	72	72

Timing

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
Service name	Variant	Notes						
ActivateTask	SW	1	292	376	476	304	388	520
	NS		240	332	432	260	344	476
	KL	2	104	200	300	124	212	344
TerminateTask	LExt	3	n/a	n/a	n/a	n/a	n/a	n/a
	H	5	48	48	48	48	48	48
ChainTask	SWL	1, 8	228	324	424	248	336	468
	SWH	1, 9	296	396	500	316	408	544
	NSL	8	228	324	424	248	336	468
	NSH	9	284	380	484	304	392	528
Schedule			232	232	304	232	232	304
GetTaskID			60	60	60	60	60	60
GetTaskState			160	160	160	196	196	196
EnableAllInterrupts			52	52	52	52	52	52
DisableAllInterrupts			60	60	60	60	60	60
ResumeAllInterrupts			84	84	84	84	84	84
SuspendAllInterrupts			92	92	92	92	92	92
ResumeOSInterrupts			84	84	84	84	84	84
SuspendOSInterrupts			112	112	112	112	112	112
GetResource	Task	7	44	44	52	44	44	52
	Combined	6	140	140	140	140	140	140

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
ReleaseResource	Task	7	204	204	204	204	204	204
	Combined	6	484	484	484	484	484	484
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	1	n/a	n/a	n/a	268	268	428
	NS		n/a	n/a	n/a	200	200	360
	NS1i	10	n/a	n/a	n/a	116	n/a	n/a
	KL	2	n/a	n/a	n/a	112	112	272
	KL1i	2, 10	n/a	n/a	n/a	32	n/a	n/a
ClearEvent			n/a	n/a	n/a	124	124	124
GetEvent			n/a	n/a	n/a	20	20	20
WaitEvent	<default>		n/a	n/a	n/a	620	620	1016
	fp	11	n/a	n/a	n/a	676	676	1128
	1i	10	n/a	n/a	n/a	224	n/a	n/a
GetAlarmBase			120	120	120	120	120	120
GetAlarm			192	192	192	192	192	192
SetRelAlarm			240	240	240	240	240	240
SetAbsAlarm			288	288	288	288	288	288
CancelAlarm			180	180	180	180	180	180
InitCounter			124	124	124	124	124	124
GetCounterValue			152	152	152	152	152	152
osek_tick_alarm	<default>		148	148	148	148	148	148
	KL	2	56	56	56	56	56	56
osek_incr_counter			56	56	56	56	56	56
GetActiveApplicationMode		30	n/a	n/a	n/a	n/a	n/a	n/a
StartOS			340	340	340	340	340	340
ShutdownOS	NoHook	12	64	64	64	64	64	64
	Hook	13	88	88	88	88	88	88
InitCOM			8	8	8	8	8	8
CloseCOM			8	8	8	8	8	8
StartCOM			64	64	64	64	64	64
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	156	156	156	420	420	420
	CCCB	15	420	420	420	420	420	420
GetMessageResource			88	88	88	88	88	88

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
ReleaseMessageResource			76	76	76	76	76	76
GetMessageStatus			156	156	156	156	156	156
SendMessage	SW CCCA	1, 14	204	204	204	452	452	452
	SW CCCB	1, 15	428	428	428	452	452	452
	NS CCCA	14	204	204	204	452	452	452
	NS CCCB	15	428	428	428	452	452	452
	KL CCCA	2, 14	104	104	104	356	356	356
	KL CCCB	2, 15	332	332	332	356	356	356
main_dispatch	NoHook	12	320	320	400	320	320	400
	Hook	13	416	416	496	416	416	496
sub_dispatch	B1LF	19	56	56	56	56	56	56
	B1HI	20	180	180	180	180	180	180
	B1HF	21	204	204	204	204	204	204
	B2LI	22	n/a	108	180	n/a	108	180
	B2LF	23	n/a	132	204	n/a	132	204
	B2HI	24	n/a	256	424	n/a	256	424
	B2HF	25	n/a	280	448	n/a	280	448
	E1HI	26	n/a	n/a	n/a	708	708	880
	E1HF	27	n/a	n/a	n/a	732	732	904
	E2HI	28	n/a	n/a	n/a	n/a	n/a	880
	E2HF	29	n/a	n/a	n/a	n/a	n/a	904
ErrorHook support		16	76	76	76	76	76	76
	ServiceID	17	92	92	92	92	92	92
	Parameters	18	116	116	116	116	116	116
validity_checks		3	n/a	n/a	n/a	n/a	n/a	n/a
Timing_dispatch		4	160	160	160	160	160	160
Timing_termination		4	172	172	172	172	172	172
ActivateTaskset	SW	1	188	264	372	204	296	408
	NS		116	196	320	132	228	340
	KL	2	32	108	216	48	140	252
ChainTaskset	SWL	1, 8	104	176	280	104	192	300
	SWH	1, 9	208	288	396	208	304	416
	NSL	8	104	176	280	104	192	300
	NSH	9	196	276	384	196	292	404
GetTasksetRef			16	16	16	16	16	16
MergeTaskset			108	108	108	108	108	108
AssignTaskset			16	16	16	16	16	16

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
RemoveTaskset			112	112	112	112	112	112
TestSubTaskset			128	128	128	128	128	128
TestEquivalentTaskset			124	124	124	124	124	124
TickSchedule	SW	1	308	304	304	304	304	304
	NS		232	252	252	252	252	252
	KL	2	144	164	164	164	164	164
AdvanceSchedule	SW	1	276	288	288	288	288	288
	NS		224	236	236	236	236	236
	KL	2	132	144	144	144	144	144
StartSchedule			176	176	176	176	176	176
StopSchedule			148	148	148	148	148	148
GetScheduleStatus			196	196	196	196	196	196
GetScheduleValue			184	184	184	184	184	184
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			52	52	52	52	52	52
GetExecutionTime			204	204	204	204	204	204
GetLargestExecutionTime			24	24	24	24	24	24
ResetLargestExecutionTime			20	20	20	20	20	20
GetStackOffset			60	60	60	60	60	60
Interrupt support			340	340	340	340	340	340
longjmp/setjmp support			160	160	160	160	160	160
Stack Manipulation support			72	72	72	72	72	72

Extended

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
Service name	Variant	Notes						
ActivateTask	SW	1	424	520	612	440	532	656
	NS		496	592	684	512	604	728
	KL	2	304	400	492	320	412	536
TerminateTask	LExt	3	172	172	172	172	172	172
	H	5	228	228	228	228	228	228
ChainTask	SWL	1, 8	464	564	656	488	576	700
	SWH	1, 9	544	648	740	568	660	784
	NSL	8	580	680	772	604	692	816
	NSH	9	648	752	844	672	764	888
Schedule			360	360	432	360	360	432
GetTaskID			80	80	80	80	80	80
GetTaskState			416	416	416	408	408	408
EnableAllInterrupts			76	76	76	76	76	76
DisableAllInterrupts			84	84	84	84	84	84
ResumeAllInterrupts			168	168	168	168	168	168
SuspendAllInterrupts			116	116	116	116	116	116
ResumeOSInterrupts			168	168	168	168	168	168
SuspendOSInterrupts			136	136	136	136	136	136
GetResource	Task	7	652	652	592	652	652	592
	Combined	6	660	660	660	660	660	660
	CLEx	3	548	548	548	548	548	548
ReleaseResource	Task	7	592	592	592	592	592	592
	Combined	6	856	856	856	856	856	856
	CLEx	3	536	536	536	536	536	536
SetEvent	SW	1	n/a	n/a	n/a	536	536	696
	NS		n/a	n/a	n/a	608	608	768
	NS1i	10	n/a	n/a	n/a	416	n/a	n/a
	KL	2	n/a	n/a	n/a	416	416	584
	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	324	324	324
WaitEvent	<default>		n/a	n/a	n/a	836	836	1208
	fp	11	n/a	n/a	n/a	892	892	1320

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
	1i	10	n/a	n/a	n/a	436	n/a	n/a
GetAlarmBase			288	288	288	288	288	288
GetAlarm			260	260	260	260	260	260
SetRelAlarm			364	364	364	364	364	364
SetAbsAlarm			408	408	408	408	408	408
CancelAlarm			228	228	228	228	228	228
InitCounter			328	328	328	328	328	328
GetCounterValue			316	316	316	316	316	316
osek_tick_alarm	<default>		168	168	168	168	168	168
	KL	2	56	56	56	56	56	56
osek_incr_counter			56	56	56	56	56	56
GetActiveApplicationMode		30	n/a	n/a	n/a	n/a	n/a	n/a
StartOS			364	364	364	364	364	364
ShutdownOS	NoHook	12	76	76	76	76	76	76
	Hook	13	100	100	100	100	100	100
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	240	240	240	488	488	488
	CCCB	15	488	488	488	488	488	488
GetMessageResource			172	172	172	172	172	172
ReleaseMessageResource			172	172	172	172	172	172
GetMessageStatus			256	256	256	256	256	256
SendMessage	SW CCCA	1, 14	316	316	316	564	564	564
	SW CCCB	1, 15	540	540	540	564	564	564
	NS CCCA	14	316	316	316	564	564	564
	NS CCCB	15	540	540	540	564	564	564
	KL CCCA	2, 14	232	232	232	480	480	480
	KL CCCB	2, 15	456	456	456	480	480	480
main_dispatch	NoHook	12	320	320	400	320	320	400
	Hook	13	416	416	496	416	416	496
sub_dispatch	B1LF	19	56	56	56	56	56	56
	B1HI	20	180	180	180	180	180	180
	B1HF	21	204	204	204	204	204	204

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
	B2LI	22	n/a	108	180	n/a	108	180
	B2LF	23	n/a	132	204	n/a	132	204
	B2HI	24	n/a	256	424	n/a	256	424
	B2HF	25	n/a	280	448	n/a	280	448
	E1HI	26	n/a	n/a	n/a	708	708	880
	E1HF	27	n/a	n/a	n/a	732	732	904
	E2HI	28	n/a	n/a	n/a	n/a	n/a	880
	E2HF	29	n/a	n/a	n/a	n/a	n/a	904
ErrorHook support		16	224	224	224	224	224	224
	ServiceID	17	240	240	240	240	240	240
	Parameters	18	264	264	264	264	264	264
validity_checks		3	104	104	104	104	104	104
Timing_dispatch		4	160	160	160	160	160	160
Timing_termination		4	172	172	172	172	172	172
ActivateTaskset	SW	1	500	600	704	536	644	764
	NS		560	660	764	596	704	824
	KL	2	364	464	576	400	516	636
ChainTaskset	SWL	1, 8	576	672	776	596	700	820
	SWH	1, 9	696	800	904	716	828	948
	NSL	8	680	776	880	700	804	924
	NSH	9	788	892	996	808	920	1040
GetTasksetRef			256	256	256	256	256	256
MergeTaskset			424	424	424	424	424	424
AssignTaskset			280	280	280	280	280	280
RemoveTaskset			428	428	428	428	428	428
TestSubTaskset			452	452	452	452	452	452
TestEquivalentTaskset			448	448	448	448	448	448
TickSchedule	SW	1	512	400	400	400	400	400
	NS		584	500	500	500	500	500
	KL	2	436	324	324	324	324	324
AdvanceSchedule	SW	1	528	412	412	412	412	412
	NS		600	512	512	512	512	512
	KL	2	448	328	328	328	328	328
StartSchedule			364	364	364	364	364	364
StopSchedule			272	272	272	272	272	272
GetScheduleStatus			332	332	332	332	332	332
GetScheduleValue			264	264	264	264	264	264

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
GetScheduleNext			132	132	132	132	132	132
SetScheduleNext			244	244	244	244	244	244
GetArrivalpointDelay			184	184	184	184	184	184
SetArrivalpointDelay			204	204	204	204	204	204
GetArrivalpointTasksetRef			180	180	180	180	180	180
GetArrivalpointNext			184	184	184	184	184	184
SetArrivalpointNext			280	280	280	280	280	280
TestArrivalpointWritable			220	220	220	220	220	220
GetExecutionTime			204	204	204	204	204	204
GetLargestExecutionTime			180	180	180	180	180	180
ResetLargestExecutionTime			164	164	164	164	164	164
GetStackOffset			60	60	60	60	60	60
Interrupt support			340	340	340	340	340	340
longjmp/setjmp support			160	160	160	160	160	160
Stack Manipulation support			72	72	72	72	72	72

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`

Number	Note
	and `USEPARAMETERACCESS = FALSE`
17	Linked only if `USEGETSERVICEID = TRUE` and `USEPARAMETERACCESS = FALSE`
18	Linked only if `USEGETSERVICEID = TRUE` and `USEPARAMETERACCESS = TRUE`
19	Linked only for basic, single-activation, lightweight, floating-point tasks
20	Linked only for basic, single-activation, heavyweight, integer tasks
21	Linked only for basic, single-activation, heavyweight, floating-point tasks
22	Linked only for basic, multiple-activation, lightweight, integer tasks
23	Linked only for basic, multiple-activation, lightweight, floating-point tasks
24	Linked only for basic, multiple-activation, heavyweight, integer tasks
25	Linked only for basic, multiple-activation, heavyweight, floating-point tasks
26	Linked only for extended, unique priority, integer tasks
27	Linked only for extended, unique priority, floating-point tasks
28	Linked only for extended, shared priority, integer tasks
29	Linked only for extended, shared priority, floating-point tasks
30	Implemented as a macro, so no code is linked
31	Not required on some targets

4.2.4 Reserved Hardware Resources

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

4.3 Performance

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	499	619	991	523	613	1045
	NS	415	553	925	457	547	979
	KL	217	379	751	259	373	805
TerminateTask	LExt	0	0	0	0	0	0
	H	801	789	831	801	789	831
ChainTask	SWL	1327	1489	2023	1633	1747	2341
	SWH	1709	1883	2423	2015	2135	2741
	NSL	1327	1489	2023	1633	1747	2341
	NSH	1685	1853	2393	1991	2111	2711
Schedule	SW	427	427	523	427	427	523
GetTaskID		151	151	151	151	151	151
GetTaskState		355	355	355	421	421	421
EnableAllInterrupts		125	125	125	125	125	125
DisableAllInterrupts		141	141	141	141	141	141
ResumeAllInterrupts		185	185	185	185	185	185
SuspendAllInterrupts		207	207	207	207	207	207
ResumeOSInterrupts		185	185	185	185	185	185
SuspendOSInterrupts		207	207	207	207	207	207
GetResource	Task	157	157	169	157	157	169
	Combined	237	237	237	237	237	237
	CLEx	n/a	n/a	n/a	n/a	n/a	n/a
ReleaseResource	Task	379	379	379	379	379	379
	Combined	603	603	603	603	603	603
	CLEx	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	n/a	n/a	n/a	469	469	469
	NS	n/a	n/a	n/a	409	409	409
	KL	n/a	n/a	n/a	259	259	259
ClearEvent		n/a	n/a	n/a	285	285	285
GetEvent		n/a	n/a	n/a	103	103	103
WaitEvent	<default>	n/a	n/a	n/a	3365	3353	3761
	fp	n/a	n/a	n/a	3407	3395	3803
GetAlarmBase		307	307	307	307	307	307
GetAlarm		421	421	421	421	421	421

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
SetRelAlarm		445	445	445	445	445	445
SetAbsAlarm		451	451	451	451	451	451
CancelAlarm		355	355	355	355	355	355
InitCounter		271	271	271	271	271	271
GetCounterValue		301	301	301	301	301	301
osek_tick_alarm	<default>	331	331	331	331	331	331
	KL	163	163	163	163	163	163
osek_incr_counter		67	67	67	67	67	67
GetActiveApplicationMode		43	43	43	43	43	43
StartOS		5315	5315	5315	5315	5315	5315
ShutdownOS	NoHook	n/a	n/a	n/a	n/a	n/a	n/a
	Hook	195	195	195	195	195	195
InitCOM		49	49	49	49	49	49
CloseCOM		49	49	49	49	49	49
StartCOM		205	205	205	625	625	625
StopCOM		91	91	91	91	91	91
ReadFlag		n/a	n/a	n/a	67	67	67
ResetFlag		n/a	n/a	n/a	49	49	49
ReceiveMessage		331	331	331	1663	1663	1663
GetMessageResource		n/a	n/a	n/a	415	415	415
ReleaseMessageResource		n/a	n/a	n/a	601	601	601
GetMessageStatus		n/a	n/a	n/a	553	553	553
SendMessage	SW	871	991	1363	2035	2125	2557
	NS	787	925	1297	1969	2059	2491
	KL	415	577	949	1597	1711	2143
ActivateTaskset	SW	391	2029	2431	421	2041	2845
	NS	259	1903	2341	289	1915	2719
	KL	115	1753	2155	145	1765	2569
	SW2	391	2029	2431	421	2041	2845
	NS2	259	1903	2341	289	1915	2719
	KL2	115	1753	2155	145	1765	2569
ChainTaskset	SWL	1255	2881	3439	1519	3127	4087
	SWH	1685	3317	3881	1949	3563	4529
	NSL	1255	2881	3439	1519	3127	4087
	NSH	1661	3293	3857	1925	3539	4505
GetTasksetRef		103	103	103	103	103	103

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
MergeTaskset		259	259	259	259	259	259
AssignTaskset		97	97	97	97	97	97
RemoveTaskset		265	265	265	265	265	265
TestSubTaskset		295	295	295	295	295	295
TestEquivalentTaskset		289	289	289	289	289	289
TickSchedule	SW	685	2389	2791	781	2449	3241
	NS	541	2305	2707	697	2365	3157
	KL	391	2137	2539	529	2197	2989
	SW2	685	2389	2791	781	2401	3205
	NS2	541	2305	2707	697	2317	3121
	KL2	391	2137	2539	529	2149	2953
AdvanceSchedule	SW	571	2311	2713	703	2371	3163
	NS	487	2227	2629	619	2287	3079
	KL	313	2053	2455	445	2113	2905
	SW2	571	2311	2713	703	2323	3127
	NS2	487	2227	2629	619	2239	3043
	KL2	313	2053	2455	445	2065	2869
StartSchedule		397	397	397	397	397	397
StopSchedule		361	361	361	361	361	361
GetScheduleStatus		415	415	415	415	415	415
GetScheduleValue		421	421	421	421	421	421
GetScheduleNext		109	109	109	109	109	109
SetScheduleNext		103	103	103	103	103	103
GetArrivalpointDelay		97	97	97	97	97	97
SetArrivalpointDelay		91	91	91	91	91	91
GetArrivalpointTasksetRef		85	85	85	85	85	85
GetArrivalpointNext		97	97	97	97	97	97
SetArrivalpointNext		91	91	91	91	91	91
TestArrivalpointWritable		145	145	145	145	145	145
GetExecutionTime		49	49	49	49	49	49
GetLargestExecutionTime		79	79	79	79	79	79
ResetLargestExecutionTime		67	67	67	67	67	67
GetStackOffset		187	187	187	187	187	187

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	499	619	991	523	613	1045
	NS	415	553	925	457	547	979
	KL	217	379	751	259	373	805
TerminateTask	LExt	0	0	0	0	0	0
	H	1585	1573	1615	1585	1573	1615
ChainTask	SWL	2279	2441	2975	2591	2705	3299
	SWH	2631	2805	3345	2943	3063	3669
	NSL	2279	2441	2975	2591	2705	3299
	NSH	2601	2769	3309	2913	3033	3633
Schedule	SW	427	427	523	427	427	523
GetTaskID		151	151	151	151	151	151
GetTaskState		355	355	355	421	421	421
EnableAllInterrupts		125	125	125	125	125	125
DisableAllInterrupts		141	141	141	141	141	141
ResumeAllInterrupts		185	185	185	185	185	185
SuspendAllInterrupts		207	207	207	207	207	207
ResumeOSInterrupts		185	185	185	185	185	185
SuspendOSInterrupts		207	207	207	207	207	207
GetResource	Task	157	157	169	157	157	169
	Combined	237	237	237	237	237	237
	CLEx	n/a	n/a	n/a	n/a	n/a	n/a
ReleaseResource	Task	379	379	379	379	379	379
	Combined	603	603	603	603	603	603
	CLEx	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	n/a	n/a	n/a	469	469	469
	NS	n/a	n/a	n/a	409	409	409
	KL	n/a	n/a	n/a	259	259	259
ClearEvent		n/a	n/a	n/a	285	285	285
GetEvent		n/a	n/a	n/a	103	103	103
WaitEvent	<default>	n/a	n/a	n/a	4097	4085	4493
	fp	n/a	n/a	n/a	4139	4127	4535
GetAlarmBase		307	307	307	307	307	307
GetAlarm		421	421	421	421	421	421

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
SetRelAlarm		445	445	445	445	445	445
SetAbsAlarm		451	451	451	451	451	451
CancelAlarm		355	355	355	355	355	355
InitCounter		271	271	271	271	271	271
GetCounterValue		301	301	301	301	301	301
osek_tick_alarm	<default>	331	331	331	331	331	331
	KL	163	163	163	163	163	163
osek_incr_counter		67	67	67	67	67	67
GetActiveApplicationMode		43	43	43	43	43	43
StartOS		13015	13015	13015	13015	13015	13015
ShutdownOS	NoHook	n/a	n/a	n/a	n/a	n/a	n/a
	Hook	195	195	195	195	195	195
InitCOM		49	49	49	49	49	49
CloseCOM		49	49	49	49	49	49
StartCOM		205	205	205	625	625	625
StopCOM		91	91	91	91	91	91
ReadFlag		n/a	n/a	n/a	67	67	67
ResetFlag		n/a	n/a	n/a	49	49	49
ReceiveMessage		331	331	331	1663	1663	1663
GetMessageResource		n/a	n/a	n/a	415	415	415
ReleaseMessageResource		n/a	n/a	n/a	601	601	601
GetMessageStatus		n/a	n/a	n/a	553	553	553
SendMessage	SW	871	991	1363	2035	2125	2557
	NS	787	925	1297	1969	2059	2491
	KL	415	577	949	1597	1711	2143
ActivateTaskset	SW	391	2029	2431	421	2041	2845
	NS	259	1903	2341	289	1915	2719
	KL	115	1753	2155	145	1765	2569
	SW2	391	2029	2431	421	2041	2845
	NS2	259	1903	2341	289	1915	2719
	KL2	115	1753	2155	145	1765	2569
ChainTaskset	SWL	2201	3833	4391	2471	4085	5045
	SWH	2607	4239	4803	2877	4491	5457
	NSL	2201	3833	4391	2471	4085	5045
	NSH	2577	4209	4773	2847	4461	5427
GetTasksetRef		103	103	103	103	103	103

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
MergeTaskset		259	259	259	259	259	259
AssignTaskset		97	97	97	97	97	97
RemoveTaskset		265	265	265	265	265	265
TestSubTaskset		295	295	295	295	295	295
TestEquivalentTaskset		289	289	289	289	289	289
TickSchedule	SW	685	2389	2791	781	2449	3241
	NS	541	2305	2707	697	2365	3157
	KL	391	2137	2539	529	2197	2989
	SW2	685	2389	2791	781	2401	3205
	NS2	541	2305	2707	697	2317	3121
	KL2	391	2137	2539	529	2149	2953
AdvanceSchedule	SW	571	2311	2713	703	2371	3163
	NS	487	2227	2629	619	2287	3079
	KL	313	2053	2455	445	2113	2905
	SW2	571	2311	2713	703	2323	3127
	NS2	487	2227	2629	619	2239	3043
	KL2	313	2053	2455	445	2065	2869
StartSchedule		397	397	397	397	397	397
StopSchedule		361	361	361	361	361	361
GetScheduleStatus		415	415	415	415	415	415
GetScheduleValue		421	421	421	421	421	421
GetScheduleNext		109	109	109	109	109	109
SetScheduleNext		103	103	103	103	103	103
GetArrivalpointDelay		97	97	97	97	97	97
SetArrivalpointDelay		91	91	91	91	91	91
GetArrivalpointTasksetRef		85	85	85	85	85	85
GetArrivalpointNext		97	97	97	97	97	97
SetArrivalpointNext		91	91	91	91	91	91
TestArrivalpointWritable		145	145	145	145	145	145
GetExecutionTime		495	495	495	495	495	495
GetLargestExecutionTime		115	115	115	115	115	115
ResetLargestExecutionTime		103	103	103	103	103	103
GetStackOffset		187	187	187	187	187	187

Extended

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
Service	Variant						
ActivateTask	SW	2335	2491	2857	2365	2485	2911
	NS	2431	2587	2953	2461	2581	3007
	KL	1985	2141	2507	2015	2135	2561
TerminateTask	LExt	1847	1847	1889	1847	1847	1889
	H	2117	2117	2159	2117	2117	2159
ChainTask	SWL	4557	4731	5259	4875	4989	5583
	SWH	4903	5077	5605	5221	5335	5929
	NSL	4719	4893	5421	5037	5151	5745
	NSH	5041	5215	5743	5359	5473	6067
Schedule	SW	757	757	853	757	757	853
GetTaskID		193	193	193	193	193	193
GetTaskState		2443	2443	2443	2383	2383	2383
EnableAllInterrupts		173	173	173	173	173	173
DisableAllInterrupts		189	189	189	189	189	189
ResumeAllInterrupts		305	305	305	305	305	305
SuspendAllInterrupts		255	255	255	255	255	255
ResumeOSInterrupts		305	305	305	305	305	305
SuspendOSInterrupts		255	255	255	255	255	255
GetResource	Task	2701	2701	2005	3115	3115	2419
	Combined	1849	1849	1849	2263	2263	2263
	CLEx	2011	2011	2011	2425	2425	2425
ReleaseResource	Task	1951	1951	1951	2365	2365	2365
	Combined	2049	2049	2049	2463	2463	2463
	CLEx	1879	1879	1879	2293	2293	2293
SetEvent	SW	n/a	n/a	n/a	2323	2323	2323
	NS	n/a	n/a	n/a	2431	2431	2431
	KL	n/a	n/a	n/a	2039	2039	2039
ClearEvent		n/a	n/a	n/a	607	607	607
GetEvent		n/a	n/a	n/a	1891	1891	1891
WaitEvent	<default>	n/a	n/a	n/a	4683	4683	5037
	fp	n/a	n/a	n/a	4725	4725	5079
GetAlarmBase		1615	1615	1615	1615	1615	1615
GetAlarm		1627	1627	1627	1627	1627	1627

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
SetRelAlarm		1765	1765	1765	1765	1765	1765
SetAbsAlarm		1735	1735	1735	1735	1735	1735
CancelAlarm		1537	1537	1537	1537	1537	1537
InitCounter		1939	1939	1939	1939	1939	1939
GetCounterValue		1465	1465	1465	1465	1465	1465
osek_tick_alarm	<default>	625	625	625	625	625	625
	KL	163	163	163	163	163	163
osek_incr_counter		67	67	67	67	67	67
GetActiveApplicationMode		43	43	43	43	43	43
StartOS		13543	13543	13543	13543	13543	13543
ShutdownOS	NoHook	n/a	n/a	n/a	n/a	n/a	n/a
	Hook	219	219	219	219	219	219
InitCOM		49	49	49	49	49	49
CloseCOM		49	49	49	49	49	49
StartCOM		283	283	283	703	703	703
StopCOM		157	157	157	157	157	157
ReadFlag		n/a	n/a	n/a	169	169	169
ResetFlag		n/a	n/a	n/a	151	151	151
ReceiveMessage		1171	1171	1171	2455	2455	2455
GetMessageResource		n/a	n/a	n/a	3217	3217	3217
ReleaseMessageResource		n/a	n/a	n/a	3151	3151	3151
GetMessageStatus		n/a	n/a	n/a	1213	1213	1213
SendMessage	SW	3535	3691	4057	4681	4801	5227
	NS	3631	3787	4153	4777	4897	5323
	KL	2895	3051	3417	4041	4161	4587
ActivateTaskset	SW	2449	4177	4573	2521	4171	4993
	NS	2533	4261	4657	2605	4255	5077
	KL	2069	3797	4199	2141	3797	4619
	SW2	2449	4177	4573	2521	4171	4993
	NS2	2533	4261	4657	2605	4255	5077
	KL2	2069	3797	4199	2141	3797	4619
ChainTaskset	SWL	4833	6549	7107	5145	6783	7767
	SWH	5209	6931	7489	5521	7165	8149
	NSL	4983	6699	7257	5295	6933	7917
	NSH	5335	7057	7615	5647	7291	8275
GetTasksetRef		1891	1891	1891	1891	1891	1891

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
MergeTaskset		883	883	883	883	883	883
AssignTaskset		469	469	469	469	469	469
RemoveTaskset		889	889	889	889	889	889
TestSubTaskset		943	943	943	943	943	943
TestEquivalentTaskset		937	937	937	937	937	937
TickSchedule	SW	1177	4685	5087	3029	4775	5585
	NS	1267	4775	5177	3119	4865	5675
	KL	815	4323	4725	2667	4413	5223
	SW2	1177	4685	5087	3029	4685	5507
	NS2	1267	4775	5177	3119	4775	5597
	KL2	815	4323	4725	2667	4323	5145
AdvanceSchedule	SW	1135	4649	5051	2993	4739	5549
	NS	1225	4739	5141	3083	4829	5639
	KL	767	4281	4683	2625	4371	5181
	SW2	1135	4649	5051	2993	4649	5471
	NS2	1225	4739	5141	3083	4739	5561
	KL2	767	4281	4683	2625	4281	5103
StartSchedule		835	835	835	835	835	835
StopSchedule		655	655	655	655	655	655
GetScheduleStatus		739	739	739	739	739	739
GetScheduleValue		691	691	691	691	691	691
GetScheduleNext		247	247	247	247	247	247
SetScheduleNext		415	415	415	415	415	415
GetArrivalpointDelay		313	313	313	313	313	313
SetArrivalpointDelay		373	373	373	373	373	373
GetArrivalpointTasksetRef		271	271	271	271	271	271
GetArrivalpointNext		283	283	283	283	283	283
SetArrivalpointNext		463	463	463	463	463	463
TestArrivalpointWritable		331	331	331	331	331	331
GetExecutionTime		711	711	711	711	711	711
GetLargestExecutionTime		1609	1609	1609	1609	1609	1609
ResetLargestExecutionTime		1567	1567	1567	1567	1567	1567
GetStackOffset		187	187	187	187	187	187

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	585	585	585	585	585	585
	Cat 2	717	717	717	717	717	717

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	585	585	585	585	585	585
	Cat 2	1317	1317	1317	1317	1317	1317

Extended

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
Operation	ISR Category						
ISR Latency	Cat 1	585	585	585	585	585	585
	Cat 2	1317	1317	1317	1317	1317	1317

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.

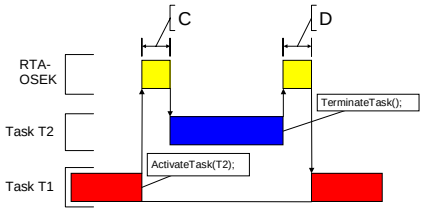

Figure 1: Task Activates a Higher Priority Task which Terminates Normally

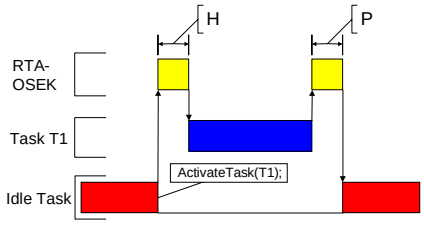

Figure 3: Task Activation from Idle Task

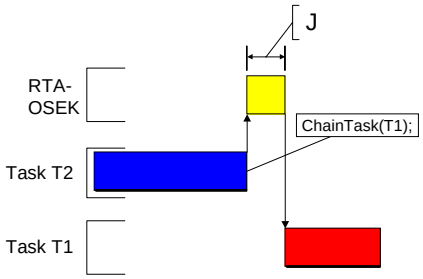

Figure 2: Task Chaining

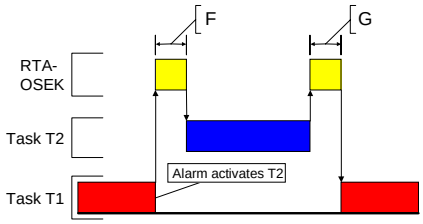

Figure 4: Task Activation from an Alarm

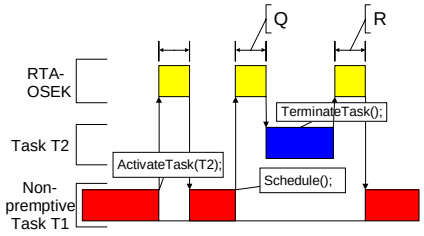

Figure 5: Non-Premptive Task Calls Schedule()

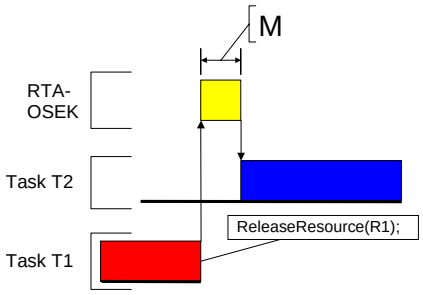

Figure 6: Blocked Task Activated by ReleaseResource()

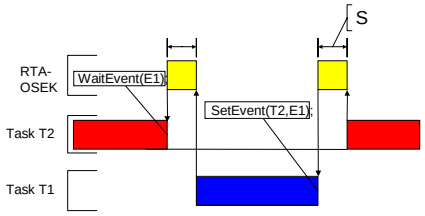

Figure 7: Waiting Task Activated by SetEvent()

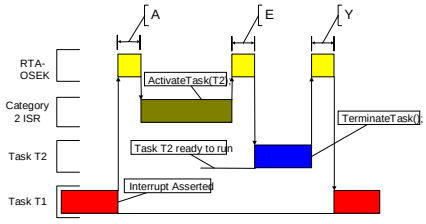

Figure 8: Category 2 ISR Activates a Higher Priority Task

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	435	613	967	435	613	967
Figure 1: D	Heavy, Basic/Extended	801	949	1303	1009	997	1351
ChainTask	Light, Basic	955	1183	1717	997	1183	1777
Figure 2: J	Heavy, Basic/Extended	2247	2635	3529	2497	2683	3637
Pre-emption	Light, Basic	949	1147	1741	973	1147	1801
Figure 1: C	Heavy, Basic/Extended	1315	1447	2041	1603	1705	2359
From idle task	Light, Basic	949	1147	1741	973	1147	1801
Figure 3: H	Heavy, Basic/Extended	1315	1447	2041	1603	1705	2359
Triggered by alarm	Light, Basic	1339	1537	2131	1363	1537	2191
Figure 4: F	Heavy, Basic/Extended	1705	1837	2431	1993	2095	2749
Schedule	Light, Basic	829	895	1213	829	895	1213
Figure 5: Q	Heavy, Basic/Extended	1195	1195	1513	1459	1459	1777
Release resource	Light, Basic	883	949	1171	883	949	1171
Figure 6: M	Heavy, Basic/Extended	1249	1249	1471	1513	1513	1735
SetEvent							

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations	Task Attributes	No	Yes		No	Yes	
Figure 7: S	Heavy, Extended	n/a	n/a	n/a	3797	3797	4517
From category 2 ISR	Light, Basic	777	843	1065	777	843	1065
Figure 8: E	Heavy, Basic/Extended	1143	1143	1365	1407	1407	1629

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	1249	1351	1705	1249	1351	1705
Figure 1: D	Heavy, Basic/Extended	1585	1657	2011	1717	1705	2059
ChainTask	Light, Basic	1931	2135	2669	1973	2135	2729
Figure 2: J	Heavy, Basic/Extended	3983	4271	5165	4151	4313	5267
Pre-emption	Light, Basic	1403	1577	2171	1427	1577	2231
Figure 1: C	Heavy, Basic/Extended	1745	1877	2471	2039	2141	2795
From idle task	Light, Basic	1403	1577	2171	1427	1577	2231
Figure 3: H	Heavy, Basic/Extended	1745	1877	2471	2039	2141	2795
Triggered by alarm	Light, Basic	1793	1967	2561	1817	1967	2621
Figure 4: F	Heavy, Basic/Extended	2135	2267	2861	2429	2531	3185
Schedule	Light, Basic	1283	1325	1643	1283	1325	1643
Figure 5: Q	Heavy, Basic/Extended	1625	1625	1943	1895	1895	2213
Release resource	Light, Basic	1337	1379	1601	1337	1379	1601
Figure 6: M	Heavy, Basic/Extended	1679	1679	1901	1949	1949	2171
SetEvent							
Figure 7: S	Heavy, Extended	n/a	n/a	n/a	4091	4091	4811
From category 2 ISR	Light, Basic	2075	2117	2339	2075	2117	2339
Figure 8: E	Heavy, Basic/Extended	2417	2417	2639	2687	2687	2909

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	1835	1949	2303	1835	1949	2303
Figure 1: D	Heavy, Basic/Extended	2117	2201	2555	2249	2249	2603
ChainTask	Light, Basic	4209	4425	4953	4257	4425	5013
Figure 2: J	Heavy, Basic/Extended	6787	7087	7969	6967	7135	8077
Pre-emption	Light, Basic	3191	3389	3977	3221	3389	4037
Figure 1: C	Heavy, Basic/Extended	3533	3689	4277	3833	3953	4601
From idle task	Light, Basic	3191	3389	3977	3221	3389	4037
Figure 3: H	Heavy, Basic/Extended	3533	3689	4277	3833	3953	4601
Triggered by alarm	Light, Basic	3875	4073	4661	3905	4073	4721
Figure 4: F	Heavy, Basic/Extended	4217	4373	4961	4517	4637	5285
Schedule	Light, Basic	1565	1607	1925	1565	1607	1925
Figure 5: Q	Heavy, Basic/Extended	1907	1907	2225	2177	2177	2495
Release resource	Light, Basic	2795	2837	3059	3209	3251	3473
Figure 6: M	Heavy, Basic/Extended	3137	3137	3359	3821	3821	4043
SetEvent							
Figure 7: S	Heavy, Extended	n/a	n/a	n/a	6089	6089	6809
From category 2 ISR	Light, Basic	2291	2333	2555	2291	2333	2555
Figure 8: E	Heavy, Basic/Extended	2633	2633	2855	2903	2903	3125

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	228	228	276	228	228	276
BCC1 lightweight, floating-point	276	276	324	276	276	324
BCC1 heavyweight, integer	340	340	388	340	340	388
BCC1 heavyweight, floating-point	340	340	388	340	340	388
BCC2 lightweight, integer	n/a	276	324	n/a	276	324
BCC2 lightweight, floating-point	n/a	276	324	n/a	276	324
BCC2 heavyweight, integer	n/a	340	388	n/a	340	388
BCC2 heavyweight, floating-point	n/a	340	388	n/a	340	388
ECC1 heavyweight, integer	n/a	n/a	n/a	396	396	444
ECC1 heavyweight, floating-point	n/a	n/a	n/a	396	396	444
ECC2 heavyweight, integer	n/a	n/a	n/a	n/a	n/a	444
ECC2 heavyweight, floating-point	n/a	n/a	n/a	n/a	n/a	444
Pre- and/or Post-Task hooks used						
Task type						
BCC1 lightweight, integer	276	276	276	276	276	276
BCC1 lightweight, floating-point	324	324	324	324	324	324
BCC1 heavyweight, integer	388	388	388	388	388	388
BCC1 heavyweight, floating-point	388	388	388	388	388	388
BCC2 lightweight, integer	n/a	324	324	n/a	324	324
BCC2 lightweight, floating-point	n/a	324	324	n/a	324	324
BCC2 heavyweight, integer	n/a	388	388	n/a	388	388
BCC2 heavyweight, floating-point	n/a	388	388	n/a	388	388
ECC1 heavyweight, integer	n/a	n/a	n/a	444	444	444
ECC1 heavyweight, floating-point	n/a	n/a	n/a	444	444	444
ECC2 heavyweight, integer	n/a	n/a	n/a	n/a	n/a	444
ECC2 heavyweight, floating-point	n/a	n/a	n/a	n/a	n/a	444

Timing

Configuration	Application Uses					
Events	No			Yes		
Shared Task Priorities	No		Yes	No		Yes
Multiple Task Activations	No	Yes		No	Yes	
Pre- and Post-Task hooks not used						
Task type						
BCC1 lightweight, integer	284	284	340	284	284	340
BCC1 lightweight, floating-point	332	332	388	332	332	388
BCC1 heavyweight, integer	396	396	452	396	396	452
BCC1 heavyweight, floating-point	396	396	452	396	396	452
BCC2 lightweight, integer	n/a	332	388	n/a	332	388
BCC2 lightweight, floating-point	n/a	332	388	n/a	332	388
BCC2 heavyweight, integer	n/a	396	452	n/a	396	452
BCC2 heavyweight, floating-point	n/a	396	452	n/a	396	452
ECC1 heavyweight, integer	n/a	n/a	n/a	452	452	508
ECC1 heavyweight, floating-point	n/a	n/a	n/a	452	452	508
ECC2 heavyweight, integer	n/a	n/a	n/a	n/a	n/a	508
ECC2 heavyweight, floating-point	n/a	n/a	n/a	n/a	n/a	508
Pre- and/or Post-Task hooks used						
Task type						
BCC1 lightweight, integer	332	332	340	332	332	340
BCC1 lightweight, floating-point	380	380	388	380	380	388
BCC1 heavyweight, integer	444	444	452	444	444	452
BCC1 heavyweight, floating-point	444	444	452	444	444	452
BCC2 lightweight, integer	n/a	380	388	n/a	380	388
BCC2 lightweight, floating-point	n/a	380	388	n/a	380	388
BCC2 heavyweight, integer	n/a	444	452	n/a	444	452
BCC2 heavyweight, floating-point	n/a	444	452	n/a	444	452
ECC1 heavyweight, integer	n/a	n/a	n/a	500	500	508
ECC1 heavyweight, floating-point	n/a	n/a	n/a	500	500	508
ECC2 heavyweight, integer	n/a	n/a	n/a	n/a	n/a	508
ECC2 heavyweight, floating-point	n/a	n/a	n/a	n/a	n/a	508

Extended

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	284	284	340	284	284	340
BCC1 lightweight, floating-point	332	332	388	332	332	388
BCC1 heavyweight, integer	396	396	452	396	396	452
BCC1 heavyweight, floating-point	396	396	452	396	396	452
BCC2 lightweight, integer	n/a	332	388	n/a	332	388
BCC2 lightweight, floating-point	n/a	332	388	n/a	332	388
BCC2 heavyweight, integer	n/a	396	452	n/a	396	452
BCC2 heavyweight, floating-point	n/a	396	452	n/a	396	452
ECC1 heavyweight, integer	n/a	n/a	n/a	452	452	508
ECC1 heavyweight, floating-point	n/a	n/a	n/a	452	452	508
ECC2 heavyweight, integer	n/a	n/a	n/a	n/a	n/a	508
ECC2 heavyweight, floating-point	n/a	n/a	n/a	n/a	n/a	508
Pre- and/or Post-Task hooks used						
Task type						
BCC1 lightweight, integer	332	332	340	332	332	340
BCC1 lightweight, floating-point	380	380	388	380	380	388
BCC1 heavyweight, integer	444	444	452	444	444	452
BCC1 heavyweight, floating-point	444	444	452	444	444	452
BCC2 lightweight, integer	n/a	380	388	n/a	380	388
BCC2 lightweight, floating-point	n/a	380	388	n/a	380	388
BCC2 heavyweight, integer	n/a	444	452	n/a	444	452
BCC2 heavyweight, floating-point	n/a	444	452	n/a	444	452
ECC1 heavyweight, integer	n/a	n/a	n/a	500	500	508
ECC1 heavyweight, floating-point	n/a	n/a	n/a	500	500	508
ECC2 heavyweight, integer	n/a	n/a	n/a	n/a	n/a	508
ECC2 heavyweight, floating-point	n/a	n/a	n/a	n/a	n/a	508

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.