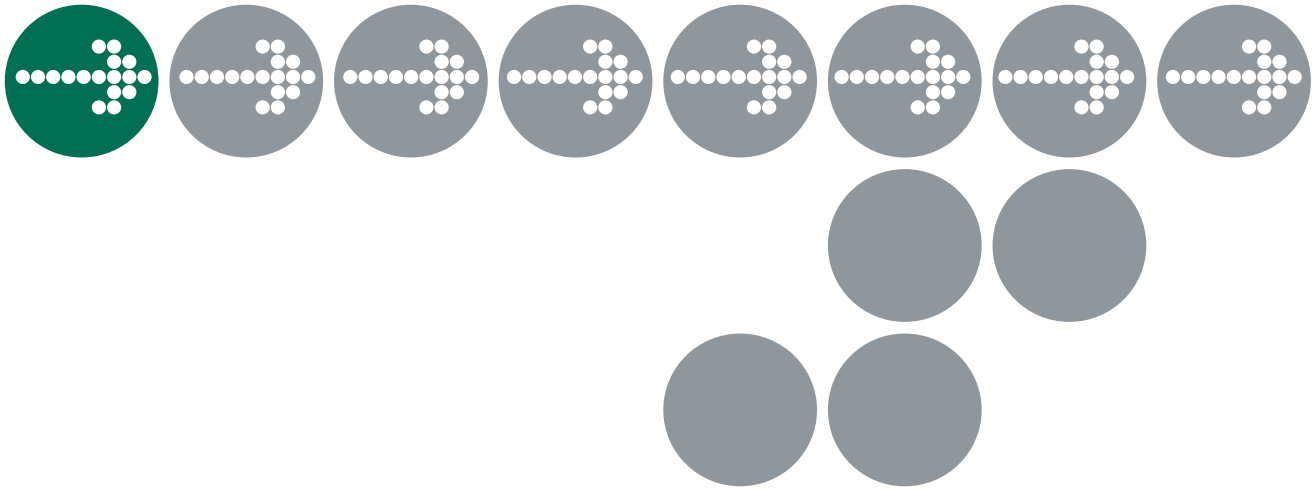
LiveDevices

RTA Binding Manual: Cosmic HC08



realogy
Real-time Architect

Contact Details

Great Britain

LiveDevices Ltd.
Atlas House
Link Business Park
Osbaldwick Link Road
Osbaldwick
York
YO10 3JB

Tel.: +44 (0) 19 04 56 25 80
Fax: +44 (0) 19 04 56 25 81

www.livedevices.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

Japan

ETAS K.K.
9-1 Ushikubo 3-chome
Tsuzuki-ku
Yokohama 224-0012

Tel.: +81 (45) 912-95 50
Fax: +81 (45) 912-95 52

www.etas.co.jp

USA

ETAS Inc.
3021 Miller Road
Ann Arbor, MI 48103

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

www.etasinc.com

France

ETAS S.A.S.
1, place des Etats 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: RM00019-001 .03

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

Real-Time Architect, RTA, RTArchitect, Realogy, the Time Compiler, SSX5 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 Manual

1.1 Purpose and Audience

This document provides port specific information for the HC08/COSMIC 16 task implementation of Realogy Real-Time Architect.

A port is defined as specific target microcontroller/target toolchain pairing. The document tells you about specific port implementation parameters, integration issues with your target toolchain and issues that you need to be aware of with using SSX5 on your target hardware. Port specific parameters of implementation are provided, giving the RAM and ROM requirements for each SSX5 object and execution times for each SSX5 API call.

It is assumed that you are a developer who wants to know low-level technical information about how to integrate SSX5 into your application.

1.2 Document Conventions

The properties of an object appear in normal typeface with ‘quotation marks’, for example, a task might have the ‘priority’ property.

Program code, such as header file names and C type names, appear in the `courier` typeface, as do C functions and SSX5 API call names. When the name of an object is made available to the programmer, the name also appears in `courier` typeface, so that a task named t1 appears as a task handle called `t1`.

1.3 Overview of Manual

This manual is divided into a number of sections. Section 2 tells you how to integrate SSX5 with your toolchain and Section 3 covers integrating SSX5 with your target hardware. Section 4 presents detailed information about the implementation parameters of SSX5, including details such as API call timings, linkable modules etc.

2 Toolchain Issues

This section details important things you need to know about SSX5 and your toolchain. A port of SSX5 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 then you should contact technical support to see whether this is possible.

The Motorola 68HC08 processor core can access variables more efficiently if they are located in 8-bit addressable memory (i.e. in the range 0x0000 to 0x00FF). This is sometimes referred to as "page 0" or the "direct page". SSX5 uses these instructions to access variables that are declared with the `OS_NEAR` modifier which, for the Cosmic cx6808 compiler, is defined to be `@tiny`. SSX5 variables that must be located in page 0 are stored in the `os_pnir` section, and so the entirety of this section must be located within the range 0x0000 to 0x00FF during linking.

The data to initialize `os_pnir` is stored in the `os_pnird` section. This section must be located in ROM, but it does not have to be placed at any particular address.

2.1 Compiler

The following compiler was used to build SSX5.

Vendor	Cosmic
Compiler	CX6808
Version	v4.4b

The following table details compulsory compiler options for application code.

Option	Description
`+nowiden`	Do not convert char parameters to integers

The C file generated by RTA from your OIL configuration file is called `osekdefs.c`. This file defines configuration parameters for SSX5 when running your application.

The following table details compulsory compiler options for `osekdefs.c`.

Option	Description
`+nowiden`	Do not convert char parameters to integers

Care is needed when reading the timer counter value within C routines. In simple expressions, the compiler-generated code correctly reads the high byte of the timer counter first. However, when the counter is subtracted from another value the generated code references the low byte of the counter first. This can result in incorrect timer values being read. This is particularly relevant when writing callback functions for advanced schedules in SSX5.

Therefore, if a subtraction involving the timer value is required, an assembler function must be written to read the timer value high-byte first. The most significant byte must be read into the X register, and the least significant byte must be read into the A register.

For example, to read the Timer A counter, the following inline assembler function could be used:

```
_asm("ldx $22\n  lda $23")
```

2.2 Assembler

The following assembler was used to build SSX5.

Vendor	Cosmic
Assembler	ca6808
Version	v4.2r

The assembly file generated by RTA from your OIL configuration file is called `osgen.s`. This file defines configuration parameters for SSX5 when running your application.

2.3 Linker/Locator

The following RTA sections must be located in addition to the sections used by application code.

Sections	ROM/RAM	Description
`os_pid`	ROM	SSX5 read-only data
`os_pird`	ROM	SSX5 initialization data
`os_pnird`	ROM	SSX5 initialization data
`os_pir`	RAM	SSX5 initialized data
`os_pnir`	Page 0 RAM	SSX5 initialized data (8-bit addressable)
`os_pur`	RAM	SSX5 uninitialized data
`os_vectbl`	ROM	Vector table if generated by RTArchitect

All Cosmic runtime libraries are compatible with SSX5.

2.4 Debugger

ORTI is the OSEK Run-Time Interface. Currently there are no ORTI compatible debuggers supported by RTA for this target.

3 Target Hardware Issues

3.1 Interrupts

For information about configuring interrupts for SSX5 please consult the *RTA User Guide*. Here, the implementation of the SSX5 interrupt model is explained.

3.1.1 Interrupt Levels

Interrupts in SSX5 are allocated an Interrupt Priority Level (IPL), which is a processor independent abstraction of the interrupt priorities available on the target hardware. You can find out more about IPLs in the *RTA User Guide*. You can find out more about the hardware interrupt controller in the *M68HC08 CPU Reference Manual*. The following table shows how SSX5 IPLs relate to interrupt priorities on the target hardware.

IPL Value	I Bit In Condition Code Register	Description
0	0	User level
1	1	OS level

3.1.2 Interrupt Vectors

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

Vector	Legality
0xFFFE	RESET cannot be used

The valid base addresses for the vector table are shown below:

Base Address	Notes
0xFF00	

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 Cosmic C compiler can generate an appropriate interrupt handling code for a C function decorated with the `@interrupt` function qualifier. See the compiler documentation for more information.

3.1.4 Category 2 Handlers

Category 2 ISRs are provided with a C function context by SSX5, as SSX5 handles the interrupt context itself. They are written using the OSEK standard `ISR()` macro as in the following example:

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

You must not insert a return from interrupt instruction in such a function – the return is handled automatically by SSX5.

3.1.5 Vector Table Issues

When you configure your application with RTArchitect you can select whether or not it generates a vector table within `osgen.s`. Note that this generated vector table omits the reset vector entry. If you choose to provide your own vector table then it must contain an entry for each interrupt handler, including the Category 2 interrupt handlers in SSX5. The following table gives the syntax for labels attached to SSX5 Category 2 interrupt handlers:

Vector Location	Label
0xVVVV	_os_wrapper_VVVV
eg : 0xFFF2	_os_wrapper_FFF2

In this table VVVV represents the 4 hex digit, upper-case, zero-padded value of the vector location.

The SSX5 configuration system does not control the reset vector for the HC08. Therefore, an appropriate vector must be created manually within the application source, and must also be placed correctly during linking. For an example of how to do this, see the files `reset.c` and `mx6808.lkf` in the example application installed as part of SSX5.

3.2 Register Settings

SSX5 does not require any registers to be initialized before calling `StartOS()`.

SSX5 does not reserve the use of any hardware registers.

3.3 Stack Usage

3.3.1 Number of Stacks

A single stack is used. The first argument to `StackFaultHook` is always 0 and `StackOffsetType` is a scalar, representing the number of bytes on the stack, with C type: `unsigned short`

3.3.2 Stack Usage within API Calls

The maximum stack usage within SSX5 API calls (not including calls to hooks and callbacks) is:

Standard

API max usage (bytes): 13

Timing

API max usage (bytes): 13

Extended

API max usage (bytes): 15

If your tasks use other library code, you may need to request information from the vendor about library call stack usage in order to correctly determine the stack usage within those tasks.

4 Parameters of Implementation

This section provides detailed information about the functionality, performance and memory demands of SSX5. SSX5 is highly scalable and different figures will be obtained when your application uses different sets of features. These feature sets give 6 classes of SSX5 depending on whether your application uses events, shared task priorities and/or multiple (queued) task activations. You should identify to which class your application belongs, then read the figures from the appropriate column.

The measurements contained in this section were made on the following hardware:

Processor	**68HC08**
Clock speed (MHz)	4
Code memory	Off-chip RAM
Read-only data memory	Off-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	16	16	16	16	16	16
Maximum number of not suspended tasks	16	16	16	16	16	16
Maximum number of priorities	16	16	16	16	16	16
Number of tasks per priority (for BCC2 and ECC2)	n/a	16	16	n/a	16	16
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	8	8	8
Limits for the number of alarm objects (per system / per task)	not limited by SSX5					
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 SSX5					
Limits for the number of nested resources (per system / per task)	255	255	255	255	255	255
Limits for the number of application modes (per system)	255	255	255	255	255	255

4.2 Hardware Resources

4.2.1 ROM and RAM Overheads

The following table gives the ROM and RAM overheads for SSX5. If you do not use messages then your application does not include the overhead for the parts of OSEK COM required to implement messaging, hence this figure is quoted separately. All figures are in bytes.

Standard

Configuration / Events / Shared Task Priorities / Multiple Task Activations		Application Uses					
		No			Yes		
		No		Yes	No		Yes
		No	Yes		No	Yes	
OS overhead	RAM	12	12	12	12	12	12
	ROM	101	101	101	101	101	101
COM overhead	RAM	2	2	2	2	2	2
	ROM	5	5	5	5	5	5

Timing

Configuration / Events / Shared Task Priorities / Multiple Task Activations		Application Uses					
		No			Yes		
		No		Yes	No		Yes
		No	Yes		No	Yes	
OS overhead	RAM	22	22	22	22	22	22
	ROM	136	136	136	136	136	136
COM overhead	RAM	2	2	2	2	2	2
	ROM	5	5	5	5	5	5

Extended

Configuration / Events / Shared Task Priorities / Multiple Task Activations		Application Uses					
		No			Yes		
		No		Yes	No		Yes
		No	Yes		No	Yes	
OS overhead	RAM	28	28	28	28	28	28
	ROM	156	156	156	156	156	156
COM overhead	RAM	2	2	2	2	2	2
	ROM	5	5	5	5	5	5

4.2.2 ROM and RAM for OSEK OS Objects

Each OSEK OS object requires ROM and/or RAM in addition to base OS overhead presented in Section 4.2.1. For each task type in OSEK (basic and extended), SSX5 provides additional sub task types that are determined by the offline configuration tools. These are:

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 SSX5. (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 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	17	17	17	17	17	17
BCC1 Heavyweight task	RAM	2	2	2	2	2	2
	ROM	19	19	19	19	19	19
BCC2 task	RAM	n/a	3	5	n/a	3	5
	ROM	n/a	20	24	n/a	20	24
ECC1, Integer task	RAM	n/a	n/a	n/a	8	8	8
	ROM	n/a	n/a	n/a	28	28	28
ECC1, floating point task	RAM	n/a	n/a	n/a	9	9	9
	ROM	n/a	n/a	n/a	28	28	28
ECC2, Integer task	RAM	n/a	n/a	n/a	n/a	n/a	10
	ROM	n/a	n/a	n/a	n/a	n/a	32
ECC2, floating point task	RAM	n/a	n/a	n/a	n/a	n/a	11
	ROM	n/a	n/a	n/a	n/a	n/a	32
Category 2 ISR	RAM	0	0	0	0	0	0
	ROM	64	64	64	64	64	64
Category 2 ISR, floating point	RAM	1	1	1	1	1	1
	ROM	73	73	73	73	73	73

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
Resource	RAM	0	0	0	0	0	0
	ROM	10	10	10	10	10	10
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	10	10	10	10	10	10
Alarm	RAM	5	5	5	5	5	5
	ROM	24	24	24	24	24	24
Counter	RAM	2	2	2	2	2	2
	ROM	56	56	56	56	56	56
Message	RAM	11	11	11	31	31	31
	ROM	10	10	10	26	26	26
Flag	RAM	1	1	1	1	1	1
	ROM	1	1	1	1	1	1
Message resource	RAM	0	0	0	0	0	0
	ROM	10	10	10	10	10	10
Event	RAM	0	0	0	0	0	0
	ROM	1	1	1	1	1	1
Priority level	RAM	0	0	4	0	4	4
	ROM	0	0	5	0	5	5
Arrivalpoint (readonly)	RAM	0	0	0	0	0	0
	ROM	6	6	6	6	6	6
Arrivalpoint (writable)	RAM	6	6	6	6	6	6
	ROM	6	6	6	6	6	6
Schedule	RAM	7	7	7	7	7	7
	ROM	16	16	16	16	16	16
Taskset (readonly)	RAM	0	0	0	0	0	0
	ROM	2	2	2	2	2	2
Taskset (writable)	RAM	2	2	2	2	2	2
	ROM	2	2	2	2	2	2

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	5	5	5	5	5	5
	ROM	22	22	22	22	22	22
BCC1 Heavyweight task	RAM	7	7	7	7	7	7
	ROM	24	24	24	24	24	24
BCC2 task	RAM	n/a	8	10	n/a	8	10
	ROM	n/a	25	29	n/a	25	29
ECC1, Integer task	RAM	n/a	n/a	n/a	13	13	13
	ROM	n/a	n/a	n/a	33	33	33
ECC1, floating point task	RAM	n/a	n/a	n/a	14	14	14
	ROM	n/a	n/a	n/a	33	33	33
ECC2, Integer task	RAM	n/a	n/a	n/a	n/a	n/a	15
	ROM	n/a	n/a	n/a	n/a	n/a	37
ECC2, floating point task	RAM	n/a	n/a	n/a	n/a	n/a	16
	ROM	n/a	n/a	n/a	n/a	n/a	37
Category 2 ISR	RAM	5	5	5	5	5	5
	ROM	82	82	82	82	82	82
Category 2 ISR, floating point	RAM	6	6	6	6	6	6
	ROM	88	88	88	88	88	88
Resource	RAM	0	0	0	0	0	0
	ROM	10	10	10	10	10	10
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	10	10	10	10	10	10
Alarm	RAM	5	5	5	5	5	5
	ROM	24	24	24	24	24	24
Counter	RAM	2	2	2	2	2	2
	ROM	56	56	56	56	56	56
Message	RAM	11	11	11	31	31	31
	ROM	10	10	10	26	26	26
Flag	RAM	1	1	1	1	1	1
	ROM	1	1	1	1	1	1
Message resource	RAM	0	0	0	0	0	0
	ROM	10	10	10	10	10	10
Event	RAM	0	0	0	0	0	0
	ROM	1	1	1	1	1	1
Priority level	RAM	0	0	4	0	4	4
	ROM	0	0	5	0	5	5

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
Arrivalpoint (readonly)	RAM	0	0	0	0	0	0
	ROM	6	6	6	6	6	6
Arrivalpoint (writable)	RAM	6	6	6	6	6	6
	ROM	6	6	6	6	6	6
Schedule	RAM	7	7	7	7	7	7
	ROM	16	16	16	16	16	16
Taskset (readonly)	RAM	0	0	0	0	0	0
	ROM	2	2	2	2	2	2
Taskset (writable)	RAM	2	2	2	2	2	2
	ROM	2	2	2	2	2	2

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	6	6	6	6	6	6
	ROM	26	26	26	26	26	26
BCC1 Heavyweight task	RAM	8	8	8	8	8	8
	ROM	26	26	26	26	26	26
BCC2 task	RAM	n/a	9	11	n/a	9	11
	ROM	n/a	27	31	n/a	27	31
ECC1, Integer task	RAM	n/a	n/a	n/a	14	14	14
	ROM	n/a	n/a	n/a	35	35	35
ECC1, floating point task	RAM	n/a	n/a	n/a	15	15	15
	ROM	n/a	n/a	n/a	35	35	35
ECC2, Integer task	RAM	n/a	n/a	n/a	n/a	n/a	16
	ROM	n/a	n/a	n/a	n/a	n/a	39
ECC2, floating point task	RAM	n/a	n/a	n/a	n/a	n/a	17
	ROM	n/a	n/a	n/a	n/a	n/a	39
Category 2 ISR	RAM	6	6	6	6	6	6
	ROM	86	86	86	86	86	86
Category 2 ISR, floating point	RAM	7	7	7	7	7	7
	ROM	92	92	92	92	92	92
Resource	RAM	3	3	3	3	3	3
	ROM	14	14	14	14	14	14
Internal resource	RAM	0	0	0	0	0	0
	ROM	0	0	0	0	0	0

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
Linked resource	RAM	3	3	3	3	3	3
	ROM	14	14	14	14	14	14
Alarm	RAM	5	5	5	5	5	5
	ROM	26	26	26	26	26	26
Counter	RAM	2	2	2	2	2	2
	ROM	58	58	58	58	58	58
Message	RAM	11	11	11	31	31	31
	ROM	12	12	12	28	28	28
Flag	RAM	1	1	1	1	1	1
	ROM	1	1	1	1	1	1
Message resource	RAM	3	3	3	3	3	3
	ROM	14	14	14	14	14	14
Event	RAM	0	0	0	0	0	0
	ROM	1	1	1	1	1	1
Priority level	RAM	0	0	4	0	4	4
	ROM	0	0	5	0	5	5
Arrivalpoint (readonly)	RAM	0	0	0	0	0	0
	ROM	10	10	10	10	10	10
Arrivalpoint (writable)	RAM	10	10	10	10	10	10
	ROM	10	10	10	10	10	10
Schedule	RAM	9	9	9	9	9	9
	ROM	20	20	20	20	20	20
Taskset (readonly)	RAM	0	0	0	0	0	0
	ROM	2	2	2	2	2	2
Taskset (writable)	RAM	2	2	2	2	2	2
	ROM	2	2	2	2	2	2

4.2.3 Size of Linkable Modules

SSX5 is demand linked, that is each API call is placed into a separately linkable module. The following sections list the module sizes (in bytes) for each API call for each of the 3 SSX5 OS status types (standard, timing, and extended). In some cases there are multiple variants of particular API calls. This is because the offline configuration of SSX5 can determine when optimized versions of the API calls can be used and will select the smallest and fastest call. In these cases, modules sizes are given for each variant under the particular configuration of SSX5 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
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
No Params	`ErrorHook` uses `GetServiceID`, but does not use `GetServiceParameters`
No ServiceID	`ErrorHook` does not use `GetServiceID` or `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	107	221	338	124	237	407
	NS		92	206	322	109	222	391
	KL	2	75	185	303	94	201	374
TerminateTask	LExt	3	n/a	n/a	n/a	n/a	n/a	n/a
	H	5	28	28	28	28	28	28
ChainTask	SWL	1, 8	111	222	343	128	238	411
	SWH	1, 9	143	258	370	160	274	438
	NSL	8	111	222	343	128	238	411
	NSH	9	140	253	365	157	269	433

Configuration			Application Uses					
Events			No					
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
Schedule			69	69	121	69	69	121
GetTaskID			59	59	59	59	59	59
GetTaskState			81	81	81	120	120	120
EnableAllInterrupts			6	6	6	6	6	6
DisableAllInterrupts			6	6	6	6	6	6
ResumeAllInterrupts			11	11	11	11	11	11
SuspendAllInterrupts			15	15	15	15	15	15
ResumeOSInterrupts			11	11	11	11	11	11
SuspendOSInterrupts			15	15	15	15	15	15
GetResource	Task	7	57	57	69	57	57	69
	Combined	6	107	107	107	107	107	107
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
ReleaseResource	Task	7	48	48	48	48	48	48
	Combined	6	48	48	48	48	48	48
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	1	n/a	n/a	n/a	138	138	366
	NS		n/a	n/a	n/a	123	123	350
	NS1i	10	n/a	n/a	n/a	35	n/a	n/a
	KL	2	n/a	n/a	n/a	110	110	340
	KL1i	2, 10	n/a	n/a	n/a	25	n/a	n/a
ClearEvent			n/a	n/a	n/a	28	28	28
GetEvent			n/a	n/a	n/a	38	38	38
WaitEvent	<default>		n/a	n/a	n/a	342	342	643
	fp	11	n/a	n/a	n/a	392	392	754
	1i	10	n/a	n/a	n/a	41	n/a	n/a
GetAlarmBase			44	44	44	44	44	44
GetAlarm			116	116	116	116	116	116
SetRelAlarm			131	131	131	131	131	131
SetAbsAlarm			204	204	204	204	204	204
CancelAlarm			71	71	71	71	71	71
InitCounter			86	86	86	86	86	86
GetCounterValue			123	123	123	123	123	123
osek_tick_alarm	<default>		95	95	95	95	95	95
	KL	2	78	78	78	78	78	78
osek_incr_counter			99	99	99	99	99	99
GetActiveApplicationMode		31	n/a	n/a	n/a	n/a	n/a	n/a
StartOS			103	103	103	103	103	103
ShutdownOS	NoHook	12	18	18	18	18	18	18
	Hook	13	23	23	23	23	23	23
InitCOM			2	2	2	2	2	2

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
CloseCOM			2	2	2	2	2	2
StartCOM			15	15	15	15	15	15
StopCOM			17	17	17	17	17	17
ReadFlag		31	n/a	n/a	n/a	n/a	n/a	n/a
ResetFlag		31	n/a	n/a	n/a	n/a	n/a	n/a
ReceiveMessage	CCCA	14	76	76	76	389	389	389
	CCCB	15	389	389	389	389	389	389
GetMessageResource			42	42	42	42	42	42
ReleaseMessageResource			36	36	36	36	36	36
GetMessageStatus			83	83	83	83	83	83
SendMessage	SW CCCA	1, 14	113	113	113	444	444	444
	SW CCCB	1, 15	410	410	410	444	444	444
	NS CCCA	14	113	113	113	444	444	444
	NS CCCB	15	410	410	410	444	444	444
	KL CCCA	2, 14	101	101	101	433	433	433
	KL CCCB	2, 15	399	399	399	433	433	433
main_dispatch	NoHook	12	127	127	217	127	127	217
	Hook	13	166	166	256	166	166	256
sub_dispatch	B1LI	19	15	15	15	15	15	15
	B1LF	20	21	21	21	21	21	21
	B1HI	21	89	89	89	89	89	89
	B1HF	22	95	95	95	95	95	95
	B2LI	23	n/a	62	127	n/a	62	127
	B2LF	24	n/a	67	133	n/a	67	133
	B2HI	25	n/a	136	315	n/a	136	315
	B2HF	26	n/a	142	321	n/a	142	321
	E1HI	27	n/a	n/a	n/a	436	436	618
	E1HF	28	n/a	n/a	n/a	442	442	624
	E2HI	29	n/a	n/a	n/a	n/a	n/a	618
	E2HF	30	n/a	n/a	n/a	n/a	n/a	624
CAT2_wrapper			43	43	43	43	43	43
hook_support	No ServiceID	16	25	25	25	25	25	25
	No Parameters	17	30	30	30	30	30	30
		18	59	59	59	59	59	59
fp_support			22	22	22	22	22	22
utility_functions	common		95	95	95	95	95	95
	optional	32	n/a	n/a	n/a	n/a	n/a	n/a
	optional	32	47	47	47	47	47	47
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

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
Timing_termination		4	n/a	n/a	n/a	n/a	n/a	n/a
ActivateTaskset	SW	1	47	125	281	65	163	334
	NS		32	109	265	50	147	318
	KL	2	16	97	253	40	135	306
ChainTaskset	SWL	1, 8	59	137	293	59	157	328
	SWH	1, 9	112	219	375	112	239	410
	NSL	8	59	137	293	59	157	328
	NSH	9	107	213	369	107	233	404
GetTasksetRef			39	39	39	39	39	39
MergeTaskset			42	42	42	42	42	42
AssignTaskset			39	39	39	39	39	39
RemoveTaskset			44	44	44	44	44	44
TestSubTaskset			74	74	74	74	74	74
TestEquivalentTaskset			66	66	66	66	66	66
TickSchedule	SW	1	262	252	252	252	252	252
	NS		246	236	236	236	236	236
	KL	2	234	224	224	224	224	224
AdvanceSchedule	SW	1	192	182	182	182	182	182
	NS		177	167	167	167	167	167
	KL	2	161	151	151	151	151	151
StartSchedule			98	98	98	98	98	98
StopSchedule			46	46	46	46	46	46
GetScheduleStatus			142	142	142	142	142	142
GetScheduleValue			73	73	73	73	73	73
GetScheduleNext			44	44	44	44	44	44
SetScheduleNext			35	35	35	35	35	35
GetArrivalpointDelay			40	40	40	40	40	40
SetArrivalpointDelay			31	31	31	31	31	31
GetArrivalpointTasksetRef			30	30	30	30	30	30
GetArrivalpointNext			40	40	40	40	40	40
SetArrivalpointNext			31	31	31	31	31	31
TestArrivalpointWritable			37	37	37	37	37	37
GetExecutionTime			3	3	3	3	3	3
GetLargestExecutionTime			16	16	16	16	16	16
ResetLargestExecutionTime			2	2	2	2	2	2
GetStackOffset			51	51	51	51	51	51

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	107	221	338	124	237	407
	NS		92	206	322	109	222	391
	KL	2	75	185	303	94	201	374
TerminateTask	LExt	3	n/a	n/a	n/a	n/a	n/a	n/a
	H	5	28	28	28	28	28	28
ChainTask	SWL	1, 8	111	222	343	128	238	411
	SWH	1, 9	143	258	370	160	274	438
	NSL	8	111	222	343	128	238	411
	NSH	9	140	253	365	157	269	433
Schedule			87	87	139	87	87	139
GetTaskID			59	59	59	59	59	59
GetTaskState			81	81	81	120	120	120
EnableAllInterrupts			6	6	6	6	6	6
DisableAllInterrupts			6	6	6	6	6	6
ResumeAllInterrupts			11	11	11	11	11	11
SuspendAllInterrupts			15	15	15	15	15	15
ResumeOSInterrupts			11	11	11	11	11	11
SuspendOSInterrupts			15	15	15	15	15	15
GetResource	Task	7	57	57	69	57	57	69
	Combined	6	107	107	107	107	107	107
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
ReleaseResource	Task	7	66	66	66	66	66	66
	Combined	6	66	66	66	66	66	66
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	1	n/a	n/a	n/a	138	138	366
	NS		n/a	n/a	n/a	123	123	350
	NS1i	10	n/a	n/a	n/a	35	n/a	n/a
	KL	2	n/a	n/a	n/a	110	110	340
	KL1i	2, 10	n/a	n/a	n/a	25	n/a	n/a
ClearEvent			n/a	n/a	n/a	28	28	28
GetEvent			n/a	n/a	n/a	38	38	38
WaitEvent	<default>		n/a	n/a	n/a	342	342	643
	fp	11	n/a	n/a	n/a	392	392	754
	1i	10	n/a	n/a	n/a	41	n/a	n/a
GetAlarmBase			44	44	44	44	44	44
GetAlarm			116	116	116	116	116	116
SetRelAlarm			131	131	131	131	131	131
SetAbsAlarm			204	204	204	204	204	204
CancelAlarm			71	71	71	71	71	71

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
InitCounter			86	86	86	86	86	86
GetCounterValue			123	123	123	123	123	123
osek_tick_alarm	<default>		95	95	95	95	95	95
	KL	2	78	78	78	78	78	78
osek_incr_counter			99	99	99	99	99	99
GetActiveApplicationMode		31	n/a	n/a	n/a	n/a	n/a	n/a
StartOS			150	150	150	150	150	150
ShutdownOS	NoHook	12	18	18	18	18	18	18
	Hook	13	23	23	23	23	23	23
InitCOM			2	2	2	2	2	2
CloseCOM			2	2	2	2	2	2
StartCOM			15	15	15	15	15	15
StopCOM			17	17	17	17	17	17
ReadFlag		31	n/a	n/a	n/a	n/a	n/a	n/a
ResetFlag		31	n/a	n/a	n/a	n/a	n/a	n/a
ReceiveMessage	CCCA	14	76	76	76	389	389	389
	CCCB	15	389	389	389	389	389	389
GetMessageResource			42	42	42	42	42	42
ReleaseMessageResource			36	36	36	36	36	36
GetMessageStatus			83	83	83	83	83	83
SendMessage	SW CCCA	1, 14	113	113	113	444	444	444
	SW CCCB	1, 15	410	410	410	444	444	444
	NS CCCA	14	113	113	113	444	444	444
	NS CCCB	15	410	410	410	444	444	444
	KL CCCA	2, 14	101	101	101	433	433	433
	KL CCCB	2, 15	399	399	399	433	433	433
main_dispatch	NoHook	12	264	264	351	264	264	351
	Hook	13	302	302	389	302	302	389
sub_dispatch	B1LI	19	3	3	3	3	3	3
	B1LF	20	9	9	9	9	9	9
	B1HI	21	90	90	90	90	90	90
	B1HF	22	96	96	96	96	96	96
	B2LI	23	n/a	49	114	n/a	49	114
	B2LF	24	n/a	54	120	n/a	54	120
	B2HI	25	n/a	133	312	n/a	133	312
	B2HF	26	n/a	139	318	n/a	139	318
	E1HI	27	n/a	n/a	n/a	466	466	648
	E1HF	28	n/a	n/a	n/a	472	472	654
	E2HI	29	n/a	n/a	n/a	n/a	n/a	648
	E2HF	30	n/a	n/a	n/a	n/a	n/a	654

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
CAT2_wrapper			183	183	183	183	183	183
hook_support	No ServiceID	16	25	25	25	25	25	25
	No Parameters	17	30	30	30	30	30	30
		18	59	59	59	59	59	59
fp_support			22	22	22	22	22	22
utility_functions	common		95	95	95	95	95	95
	optional	32	n/a	n/a	n/a	n/a	n/a	n/a
	optional	32	47	47	47	47	47	47
validity_checks		3	n/a	n/a	n/a	n/a	n/a	n/a
Timing_dispatch		4	61	61	61	61	61	61
Timing_termination		4	151	151	151	151	151	151
ActivateTaskset	SW	1	47	125	281	65	163	334
	NS		32	109	265	50	147	318
	KL	2	16	97	253	40	135	306
ChainTaskset	SWL	1, 8	59	137	293	59	157	328
	SWH	1, 9	112	219	375	112	239	410
	NSL	8	59	137	293	59	157	328
		9	107	213	369	107	233	404
GetTasksetRef			39	39	39	39	39	39
MergeTaskset			42	42	42	42	42	42
AssignTaskset			39	39	39	39	39	39
RemoveTaskset			44	44	44	44	44	44
TestSubTaskset			74	74	74	74	74	74
TestEquivalentTaskset			66	66	66	66	66	66
TickSchedule	SW	1	262	252	252	252	252	252
	NS		246	236	236	236	236	236
		2	234	224	224	224	224	224
AdvanceSchedule	SW	1	192	182	182	182	182	182
	NS		177	167	167	167	167	167
	KL	2	161	151	151	151	151	151
StartSchedule			98	98	98	98	98	98
StopSchedule			46	46	46	46	46	46
GetScheduleStatus			142	142	142	142	142	142
GetScheduleValue			73	73	73	73	73	73
GetScheduleNext			44	44	44	44	44	44
SetScheduleNext			35	35	35	35	35	35
GetArrivalpointDelay			40	40	40	40	40	40
SetArrivalpointDelay			31	31	31	31	31	31
GetArrivalpointTasksetRef			30	30	30	30	30	30
GetArrivalpointNext			40	40	40	40	40	40

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
SetArrivalpointNext			31	31	31	31	31	31
TestArrivalpointWritable			37	37	37	37	37	37
GetExecutionTime			96	96	96	96	96	96
GetLargestExecutionTime			50	50	50	50	50	50
ResetLargestExecutionTime			35	35	35	35	35	35
GetStackOffset			51	51	51	51	51	51

Extended

Configuration			Application Uses					
Events			No					
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
Service name	Variant	Notes						
ActivateTask	SW	1	196	303	418	212	319	494
	NS		329	443	558	345	459	626
	KL	2	171	278	394	187	295	471
TerminateTask	LExt	3	94	94	94	94	94	94
	H	5	132	132	132	132	132	132
ChainTask	SWL	1, 8	272	387	505	285	403	573
	SWH	1, 9	307	426	539	325	443	608
	NSL	8	422	538	658	439	554	726
	NSH	9	454	569	682	471	586	751
Schedule			160	160	213	160	160	213
GetTaskID			69	69	69	69	69	69
GetTaskState			205	205	205	226	226	226
EnableAllInterrupts			11	11	11	11	11	11
DisableAllInterrupts			11	11	11	11	11	11
ResumeAllInterrupts			43	43	43	43	43	43
SuspendAllInterrupts			20	20	20	20	20	20
ResumeOSInterrupts			43	43	43	43	43	43
SuspendOSInterrupts			20	20	20	20	20	20
GetResource	Task	7	523	523	437	523	523	437
	Combined	6	397	397	397	397	397	397
	CLEx	3	404	404	404	404	404	404
ReleaseResource	Task	7	374	374	374	374	374	374
	Combined	6	374	374	374	374	374	374
	CLEx	3	340	340	340	340	340	340

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
SetEvent	SW	1	n/a	n/a	n/a	267	267	494
	NS		n/a	n/a	n/a	409	409	636
	NS1i	10	n/a	n/a	n/a	169	n/a	n/a
	KL	2	n/a	n/a	n/a	238	238	466
	KL1i	2, 10	n/a	n/a	n/a	156	n/a	n/a
ClearEvent			n/a	n/a	n/a	100	100	100
GetEvent			n/a	n/a	n/a	203	203	203
WaitEvent	<default>		n/a	n/a	n/a	441	441	725
	fp	11	n/a	n/a	n/a	491	491	836
	1i	10	n/a	n/a	n/a	144	n/a	n/a
GetAlarmBase			110	110	110	110	110	110
GetAlarm			150	150	150	150	150	150
SetRelAlarm			249	249	249	249	249	249
SetAbsAlarm			323	323	323	323	323	323
CancelAlarm			106	106	106	106	106	106
InitCounter			158	158	158	158	158	158
GetCounterValue			183	183	183	183	183	183
osek_tick_alarm	<default>		95	95	95	95	95	95
	KL	2	78	78	78	78	78	78
osek_incr_counter			99	99	99	99	99	99
GetActiveApplicationMode		31	n/a	n/a	n/a	n/a	n/a	n/a
StartOS			160	160	160	160	160	160
ShutdownOS	NoHook	12	23	23	23	23	23	23
	Hook	13	28	28	28	28	28	28
InitCOM			2	2	2	2	2	2
CloseCOM			2	2	2	2	2	2
StartCOM			25	25	25	25	25	25
StopCOM			32	32	32	32	32	32
ReadFlag			18	18	18	18	18	18
ResetFlag			13	13	13	13	13	13
ReceiveMessage	CCCA	14	115	115	115	427	427	427
	CCCB	15	427	427	427	427	427	427
GetMessageResource			82	82	82	82	82	82
ReleaseMessageResource			82	82	82	82	82	82
GetMessageStatus			116	116	116	116	116	116
SendMessage	SW CCCA	1, 14	154	154	154	482	482	482
	SW CCCB	1, 15	448	448	448	482	482	482
	NS CCCA	14	154	154	154	482	482	482
	NS CCCB	15	448	448	448	482	482	482
	KL CCCA	2, 14	150	150	150	483	483	483
	KL CCCB	2, 15	449	449	449	483	483	483

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
main_dispatch	NoHook	12	264	264	351	264	264	351
	Hook	13	302	302	389	302	302	389
sub_dispatch	B1LI	19	3	3	3	3	3	3
	B1LF	20	9	9	9	9	9	9
	B1HI	21	91	91	91	91	91	91
	B1HF	22	97	97	97	97	97	97
	B2LI	23	n/a	49	114	n/a	49	114
	B2LF	24	n/a	54	120	n/a	54	120
	B2HI	25	n/a	134	313	n/a	134	313
	B2HF	26	n/a	140	319	n/a	140	319
	E1HI	27	n/a	n/a	n/a	467	467	649
	E1HF	28	n/a	n/a	n/a	473	473	655
	E2HI	29	n/a	n/a	n/a	n/a	n/a	649
	E2HF	30	n/a	n/a	n/a	n/a	n/a	655
CAT2_wrapper			183	183	183	183	183	183
hook_support	No ServiceID	16	63	63	63	63	63	63
	No Parameters	17	68	68	68	68	68	68
		18	98	98	98	98	98	98
fp_support			22	22	22	22	22	22
utility_functions	common		95	95	95	95	95	95
	optional	32	n/a	n/a	n/a	n/a	n/a	n/a
	optional	32	47	47	47	47	47	47
validity_checks		3	60	60	60	60	60	60
Timing_dispatch		4	61	61	61	61	61	61
Timing_termination		4	151	151	151	151	151	151
ActivateTaskset	SW	1	310	406	568	342	467	648
	NS		419	519	673	450	570	751
	KL	2	287	383	545	319	444	625
ChainTaskset	SWL	1, 8	446	553	707	471	594	776
	SWH	1, 9	503	632	786	528	679	855
	NSL	8	562	663	817	581	705	887
	NSH	9	614	736	890	633	784	960
GetTasksetRef			153	153	153	153	153	153
MergeTaskset			239	239	239	239	239	239
AssignTaskset			232	232	232	232	232	232
RemoveTaskset			241	241	241	241	241	241
TestSubTaskset			232	232	232	232	232	232
TestEquivalentTaskset			223	223	223	223	223	223
TickSchedule	SW	1	461	335	335	335	335	335
	NS		611	507	507	507	507	507
	KL	2	442	316	316	316	316	316

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
AdvanceSchedule	SW	1	430	301	301	301	301	301
	NS		582	478	478	478	478	478
	KL	2	411	282	282	282	282	282
StartSchedule			222	222	222	222	222	222
StopSchedule			110	110	110	110	110	110
GetScheduleStatus			211	211	211	211	211	211
GetScheduleValue			136	136	136	136	136	136
GetScheduleNext			107	107	107	107	107	107
SetScheduleNext			197	197	197	197	197	197
GetArrivalpointDelay			137	137	137	137	137	137
SetArrivalpointDelay			179	179	179	179	179	179
GetArrivalpointTasksetRef			127	127	127	127	127	127
GetArrivalpointNext			137	137	137	137	137	137
SetArrivalpointNext			244	244	244	244	244	244
TestArrivalpointWritable			135	135	135	135	135	135
GetExecutionTime			106	106	106	106	106	106
GetLargestExecutionTime			116	116	116	116	116	116
ResetLargestExecutionTime			97	97	97	97	97	97
GetStackOffset			51	51	51	51	51	51

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.

Number	Note
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, integer tasks
20	Linked only for basic, single-activation, lightweight, floating point tasks
21	Linked only for basic, single-activation, heavyweight, integer tasks
22	Linked only for basic, single-activation, heavyweight, floating point tasks
23	Linked only for basic, multiple-activation, lightweight, integer tasks
24	Linked only for basic, multiple-activation, lightweight, floating point tasks
25	Linked only for basic, multiple-activation, heavyweight, integer tasks
26	Linked only for basic, multiple-activation, heavyweight, floating point tasks
27	Linked only for extended, unique priority, integer tasks
28	Linked only for extended, unique priority, floating point tasks
29	Linked only for extended, shared priority, integer tasks
30	Linked only for extended, shared priority, floating point tasks
31	Implemented as a macro, so no code is linked
32	Not required on some targets

4.2.4 Reserved Hardware Resources

No timer units, interrupts, traps or other hardware resources are reserved by SSX5.

4.3 Performance

4.3.1 Execution Times for SSX5 API Calls

The following tables give the execution time in CPU cycles for each API call. The OSEK COM class was set to CCCA for systems without events, CCCB for systems with events. `ShutdownOS()` enters a tight loop, and its execution time up to `ShutdownHook()` (when called) is measured.

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	136	251	428	166	261	521
	NS	120	235	412	150	245	505
	KL	100	212	394	133	223	490
TerminateTask	Lext	0	0	0	0	0	0
	H	326	326	327	326	326	327
ChainTask	SWL	489	601	922	647	742	1138
	SWH	728	843	1159	886	983	1375
	NSL	489	601	922	647	742	1138
	NSH	724	836	1152	882	976	1368
Schedule	SW	112	112	159	112	112	159
GetTaskID		82	82	82	82	82	82
GetTaskState		132	132	132	177	177	177
EnableAllInterrupts		17	17	17	17	17	17
DisableAllInterrupts		17	17	17	17	17	17
ResumeAllInterrupts		24	24	24	24	24	24
SuspendAllInterrupts		30	30	30	30	30	30
ResumeOSInterrupts		24	24	24	24	24	24
SuspendOSInterrupts		30	30	30	30	30	30
GetResource	Task	108	108	122	108	108	122
	Combined	91	91	91	91	91	91
	CLEx	1	1	1	1	1	1
ReleaseResource	Task	85	85	85	85	85	85
	Combined	106	106	106	106	106	106
	CLEx	1	1	1	1	1	1
SetEvent	SW	n/a	n/a	n/a	187	187	192
	NS	n/a	n/a	n/a	187	187	192
	KL	n/a	n/a	n/a	169	169	177
ClearEvent		n/a	n/a	n/a	56	56	56
GetEvent		n/a	n/a	n/a	84	84	84
WaitEvent	<default>	n/a	n/a	n/a	932	932	1072
	fp	n/a	n/a	n/a	956	956	1096
GetAlarmBase		292	292	293	292	292	292
GetAlarm		166	166	166	166	166	166
SetRelAlarm		187	187	187	187	187	187
SetAbsAlarm		174	174	174	174	174	174
CancelAlarm		94	94	94	94	94	94
InitCounter		106	106	106	106	106	106

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
GetCounterValue		125	125	125	125	125	125
osek_tick_alarm	<default>	168	168	168	168	168	168
	KL	141	141	141	141	141	141
osek_incr_counter		32	32	32	32	32	32
GetActiveApplicationMode		9	9	9	9	9	9
StartOS		2613	2613	2613	2613	2618	2613
ShutdownOS	NoHook	n/a	n/a	n/a	n/a	n/a	n/a
	Hook	39	39	39	39	39	39
InitCOM		11	11	11	11	11	11
CloseCOM		11	11	11	11	11	11
StartCOM		44	44	44	494	494	494
StopCOM		33	33	33	33	33	33
ReadFlag		n/a	n/a	n/a	14	14	14
ResetFlag		n/a	n/a	n/a	11	11	11
ReceiveMessage		85	85	85	1088	1088	1088
GetMessageResource		n/a	n/a	n/a	226	226	226
ReleaseMessageResource		n/a	n/a	n/a	178	178	178
GetMessageStatus		n/a	n/a	n/a	133	133	133
SendMessage	SW	254	369	546	1220	1315	1575
	NS	238	353	530	1204	1299	1559
	KL	202	314	496	1171	1261	1528
ActivateTaskset	SW	78	703	1037	107	724	1084
	NS	62	687	1021	91	708	1068
	KL	37	669	1003	75	690	1050
	SW2	78	703	1037	107	724	1084
	NS2	62	687	1021	91	708	1068
	KL2	37	669	1003	75	690	1050
ChainTaskset	SWL	479	1105	1577	609	1230	1724
	SWH	725	1352	1824	855	1477	1971
	NSL	485	1111	1583	615	1236	1730
	NSH	720	1345	1817	850	1470	1964
GetTasksetRef		86	86	86	86	86	86
MergeTaskset		90	90	90	90	90	90
AssignTaskset		85	85	85	85	85	85
RemoveTaskset		92	92	92	92	92	92
TestSubTaskset		145	145	145	145	145	145
TestEquivalentTaskset		129	129	129	129	129	129
TickSchedule	SW	397	1040	1374	446	1093	1460
	NS	376	1019	1353	425	1072	1439
	KL	358	1001	1335	407	1054	1421

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
TickSchedule	SW2	397	1040	1374	446	1061	1421
	NS2	376	1019	1353	425	1040	1400
	KL2	358	1001	1335	407	1022	1382
AdvanceSchedule	SW	259	902	1236	308	955	1322
	NS	243	886	1220	292	939	1306
	KL	223	866	1200	272	919	1286
	SW2	259	902	1236	308	923	1283
	NS2	243	886	1220	292	907	1267
	KL2	223	866	1200	272	887	1247
StartSchedule		198	198	198	198	198	198
StopSchedule		93	93	93	93	93	93
GetScheduleStatus		199	199	199	199	199	199
GetScheduleValue		116	116	116	116	116	116
GetScheduleNext		91	91	91	91	91	91
SetScheduleNext		81	81	81	81	81	81
GetArrivalpointDelay		83	83	83	83	83	83
		73	73	73	73	73	73
GetArrivalpointTasksetRef		69	69	69	69	69	69
GetArrivalpointNext		83	83	83	83	83	83
SetArrivalpointNext		73	73	73	73	73	73
TestArrivalpointWritable		70	70	70	70	70	70
GetExecutionTime		12	12	12	12	12	12
GetLargestExecutionTime		52	52	52	52	52	52
ResetLargestExecutionTime		19	19	19	19	19	19
GetStackOffset		89	89	89	89	89	89

Timing

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
		No	Yes		No	Yes	
Service	Variant						
ActivateTask		136	251	428	166	261	521
	NS	120	235	412	150	245	505
	KL	100	212	394	133	223	490
TerminateTask	Lext	0	0	0	0	0	0
	H	611	611	612	611	611	612

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
ChainTask	SWL	871	983	1300	985	1124	1472
	SWH	1110	1181	1537	1268	1365	1753
	NSL	871	939	1256	985	1124	1516
	NSH	1062	1218	1486	1220	1314	1702
Schedule	SW	112	112	159	112	112	159
GetTaskID		82	82	82	82	82	82
GetTaskState		132	132	132	177	177	177
EnableAllInterrupts		17	17	17	17	17	17
DisableAllInterrupts		17	17	17	17	17	17
ResumeAllInterrupts		24	24	24	24	24	24
SuspendAllInterrupts		30	30	30	30	30	30
ResumeOSInterrupts		24	24	24	24	24	24
SuspendOSInterrupts		30	30	30	30	30	30
GetResource	Task	108	108	122	108	108	122
	Combined	91	91	91	91	91	91
	CLEx	1	1	1	1	1	1
ReleaseResource	Task	85	85	85	85	85	85
	Combined	106	106	106	106	106	106
	CLEx	1	1	1	1	1	1
SetEvent	SW	n/a	n/a	n/a	187	187	192
	NS	n/a	n/a	n/a	187	187	192
	KL	n/a	n/a	n/a	169	169	177
ClearEvent		n/a	n/a	n/a	56	56	56
GetEvent		n/a	n/a	n/a	84	84	84
WaitEvent	<default>	n/a	n/a	n/a	1259	1259	1399
	fp	n/a	n/a	n/a	1283	1239	1423
GetAlarmBase		292	292	292	292	292	292
GetAlarm		166	166	166	166	166	166
SetRelAlarm		187	187	187	187	187	187
SetAbsAlarm		174	174	174	174	174	174
CancelAlarm		94	94	94	94	94	94
InitCounter		106	106	106	106	106	106
GetCounterValue		125	125	125	125	125	125
osek_tick_alarm	<default>	168	168	168	168	168	168
	KL	141	141	141	141	141	141
osek_incr_counter		32	32	32	32	32	32
GetActiveApplicationMode		9	9	9	9	9	9
StartOS		5961	5961	5956	5956	5956	5956
ShutdownOS	NoHook	n/a	n/a	n/a	n/a	n/a	n/a
	Hook	39	39	39	39	39	39

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
InitCOM		11	11	11	11	11	11
CloseCOM		11	11	11	11	11	11
StartCOM		44	44	44	494	494	494
StopCOM		33	33	33	33	33	33
ReadFlag		n/a	n/a	n/a	14	14	14
ResetFlag		n/a	n/a	n/a	11	11	11
ReceiveMessage		85	85	85	1093	1093	1093
GetMessageResource		n/a	n/a	n/a	226	226	226
ReleaseMessageResource		n/a	n/a	n/a	178	178	178
GetMessageStatus		n/a	n/a	n/a	133	133	133
SendMessage	SW	254	369	546	1225	1320	1580
	NS	238	353	530	1209	1304	1564
	KL	202	314	496	1176	1266	1533
ActivateTaskset	SW	78	703	1037	107	724	1080
	NS	62	687	1021	91	708	1064
	KL	37	669	1003	75	690	1046
	SW2	78	703	1037	107	724	1080
	NS2	62	687	1021	91	708	1064
	KL2	37	669	1003	75	690	1046
ChainTaskset	SWL	817	1487	1911	947	1612	2054
	SWH	1063	1690	2158	1193	1815	2301
	NSL	867	1493	1917	953	1618	2060
	NSH	1058	1683	2151	1188	1808	2294
GetTasksetRef		86	86	86	86	86	86
MergeTaskset		90	90	90	90	90	90
AssignTaskset		85	85	85	85	85	85
RemoveTaskset		92	92	92	92	92	92
TestSubTaskset		145	145	145	145	145	145
TestEquivalentTaskset		129	129	129	129	129	129
TickSchedule	SW	397	1040	1374	446	1093	1456
	NS	376	1019	1353	425	1072	1435
	KL	358	1001	1335	407	1054	1417
	SW2	397	1040	1379	446	1061	1417
	NS2	376	1019	1358	425	1040	1396
	KL2	358	1001	1340	407	1022	1378
AdvanceSchedule	SW	259	902	1236	308	955	1318
	NS	243	886	1220	292	939	1302
	KL	223	866	1200	272	919	1282
	SW2	259	902	1236	308	923	1279
	NS2	243	886	1220	292	907	1263
	KL2	223	866	1200	272	887	1243

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
StartSchedule		198	198	198	198	198	198
StopSchedule		93	93	93	93	93	93
GetScheduleStatus		199	199	199	199	199	199
GetScheduleValue		116	116	116	116	116	116
GetScheduleNext		91	91	91	91	91	91
SetScheduleNext		81	81	81	81	81	81
GetArrivalpointDelay		83	83	83	83	83	83
SetArrivalpointDelay		73	73	73	73	73	73
GetArrivalpointTasksetRef		69	69	69	69	69	69
GetArrivalpointNext		83	83	83	83	83	83
SetArrivalpointNext		73	73	73	73	73	73
TestArrivalpointWritable		70	70	70	70	70	70
GetExecutionTime		163	163	163	163	163	163
GetLargestExecutionTime		106	106	106	106	106	106
ResetLargestExecutionTime		73	73	73	73	73	73
GetStackOffset		89	89	89	89	89	89

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	602	705	881	628	715	981
	NS	727	842	1020	753	852	1108
	KL	574	677	853	600	687	953
TerminateTask	Lext	491	491	535	491	491	535
	H	710	754	754	754	754	754
ChainTask	SWL	1420	1536	1802	1577	1676	2016
	SWH	1616	1731	2041	1774	1871	2257
	NSL	1561	1632	1988	1719	1772	2204
	NSH	1750	1865	2221	1952	2005	2393
Schedule	SW	161	161	208	161	161	208
GetTaskID		94	94	94	94	94	94
GetTaskState		611	611	611	636	636	636
EnableAllInterrupts		23	23	23	23	23	23
DisableAllInterrupts		23	23	23	23	23	23
ResumeAllInterrupts		43	43	43	43	43	43

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
SuspendAllInterrupts		36	36	36	36	36	36
ResumeOSInterrupts		43	43	43	43	43	43
SuspendOSInterrupts		36	36	36	36	36	36
GetResource	Task	1545	1545	714	1650	1650	819
	Combined	568	568	568	673	673	673
	CLEx	695	695	695	800	800	800
ReleaseResource	Task	624	624	624	729	729	729
	Combined	551	551	551	656	656	656
	CLEx	592	592	592	697	697	697
SetEvent	SW	n/a	n/a	n/a	646	646	649
	NS	n/a	n/a	n/a	722	722	725
	KL	n/a	n/a	n/a	634	634	637
ClearEvent		n/a	n/a	n/a	99	99	99
GetEvent		n/a	n/a	n/a	612	612	612
WaitEvent	<default>	n/a	n/a	n/a	1346	1346	1465
	fp	n/a	n/a	n/a	1370	1370	1445
GetAlarmBase		612	612	612	612	612	612
GetAlarm		486	486	486	486	486	486
SetRelAlarm		630	630	630	630	630	630
SetAbsAlarm		582	582	582	582	582	582
CancelAlarm		414	414	414	414	414	414
InitCounter		415	415	415	415	418	415
GetCounterValue		398	398	398	398	401	398
osek_tick_alarm	<default>	168	168	168	168	168	168
	KL	141	141	141	141	141	141
osek_incr_counter		32	32	32	32	32	32
GetActiveApplicationMode		9	9	9	9	9	9
StartOS		6221	6221	6216	6216	6216	6216
ShutdownOS	NoHook	n/a	n/a	n/a	n/a	n/a	n/a
	Hook	45	45	45	45	45	45
InitCOM		11	11	11	11	11	11
CloseCOM		11	11	11	11	11	11
StartCOM		62	62	62	512	512	512
StopCOM		49	49	49	49	49	49
ReadFlag		n/a	n/a	n/a	42	42	42
ResetFlag		n/a	n/a	n/a	28	28	28
ReceiveMessage		306	306	306	1298	1298	1298
GetMessageResource		n/a	n/a	n/a	1086	1086	1086
ReleaseMessageResource		n/a	n/a	n/a	1023	1023	1023
GetMessageStatus		n/a	n/a	n/a	348	348	348

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
SendMessage	SW	941	1044	1220	1892	1979	2245
	NS	1066	1181	1359	2017	2116	2372
	KL	903	1006	1182	1854	1941	2207
ActivateTaskset	SW	822	1541	1880	870	1522	1894
	NS	923	1646	1979	970	1619	1991
	KL	794	1513	1852	843	1494	1866
	SW2	822	1541	1880	870	1522	1894
	NS2	923	1646	1979	970	1619	1991
	KL2	794	1513	1852	843	1494	1866
ChainTaskset	SWL	1727	2453	2920	1888	2541	3043
	SWH	1977	2691	3158	2138	2790	3281
	NSL	1836	2602	3025	1993	2691	3149
	NSH	2075	2783	3250	2276	2883	3418
GetTasksetRef		546	546	546	546	546	546
MergeTaskset		214	214	214	214	214	214
AssignTaskset		209	209	209	209	209	209
RemoveTaskset		216	216	216	216	216	216
TestSubTaskset		248	248	248	248	248	248
TestEquivalentTaskset		231	231	231	231	231	231
TickSchedule	SW	607	1935	2274	1265	2014	2392
	NS	748	2078	2417	1408	2157	2535
	KL	576	1904	2243	1234	1983	2361
	SW2	607	1935	2274	1265	1916	2288
	NS2	748	2078	2417	1408	2059	2431
	KL2	576	1904	2243	1234	1885	2257
AdvanceSchedule	SW	500	1828	2167	1158	1907	2285
	NS	641	1971	2310	1301	2050	2428
	KL	469	1797	2136	1127	1876	2254
	SW2	500	1828	2167	1158	1809	2181
	NS2	641	1971	2310	1301	1952	2324
	KL2	469	1797	2136	1127	1778	2150
StartSchedule		302	302	302	302	302	302
StopSchedule		131	131	131	131	131	131
GetScheduleStatus		237	237	237	237	237	237
GetScheduleValue		153	153	153	153	153	153
GetScheduleNext		129	129	129	129	129	129
SetScheduleNext		197	197	197	197	197	197
GetArrivalpointDelay		141	141	141	141	141	141
SetArrivalpointDelay		194	194	194	194	194	194
GetArrivalpointTasksetRef		125	125	125	125	125	125

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
GetArrivalpointNext		141	141	141	141	141	141
SetArrivalpointNext		244	244	244	244	244	244
TestArrivalpointWritable		132	132	132	132	132	132
GetExecutionTime		175	175	175	175	175	175
GetLargestExecutionTime		531	531	531	531	531	531
ResetLargestExecutionTime		498	498	498	498	498	498
GetStackOffset		89	89	89	89	89	89

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 dependant, since `StartOS` may activate any number of tasks and start any number of user specified alarms.

4.3.3 Interrupt Latencies

The 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 table gives 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	19	19	19	19	19	19
	Cat 2	75	75	75	75	75	75

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	19	19	19	19	19	19
	Cat 2	246	246	246	246	246	246

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	19	19	19	19	19	19
	Cat 2	246	246	246	246	246	246

4.3.4 Task Switching Times

The 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 is different for different switching contexts (e.g. an `ActivateTask` versus a `ChainTask`).

SSX5 sub-task types also affect the switching time. The tables below show the switching times in CPU cycles for all system classes for basic, lightweight tasks and for basic and extended heavyweight tasks.

The 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 is different for different switching. Figures 1 through 8 show the SSX5 switching contexts measured.

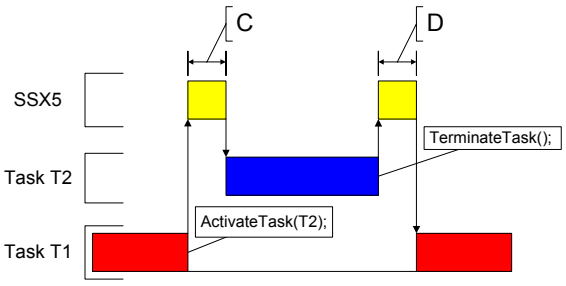


Figure 1: Task Activates a Higher Priority Task which Terminates Normally

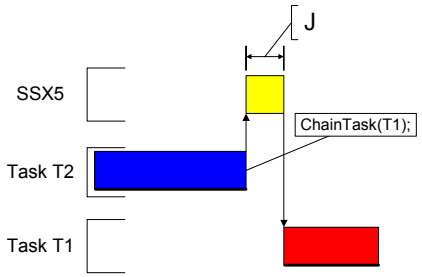


Figure 2: Task Chaining

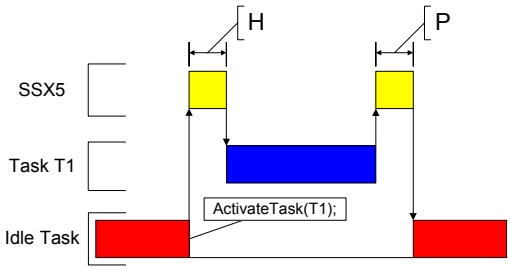


Figure 3: Task Activation from Idle Task

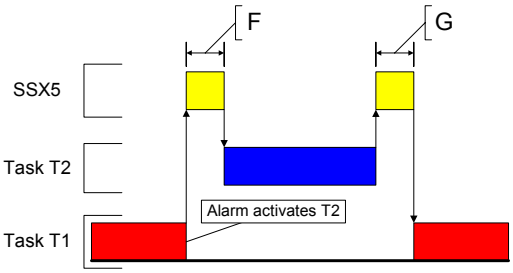


Figure 4: Task Activation from an Alarm

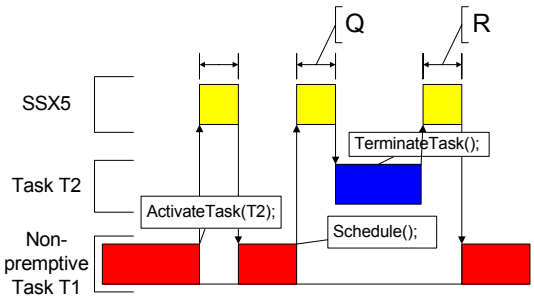


Figure 5: Non-Premptive Task Calls Schedule()

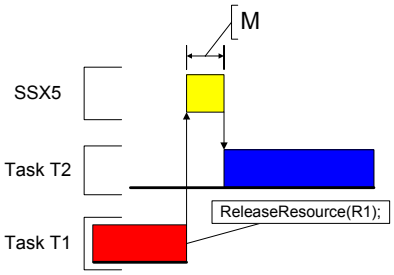


Figure 6: Blocked Task Activated by ReleaseResource()

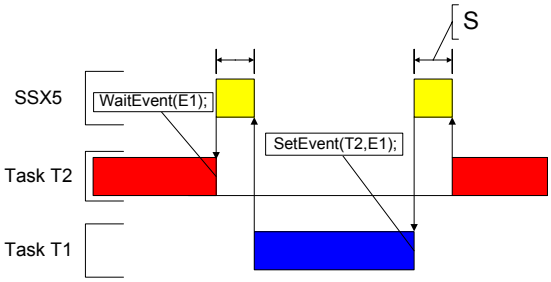


Figure 7: Waiting Task Activated by SetEvent()

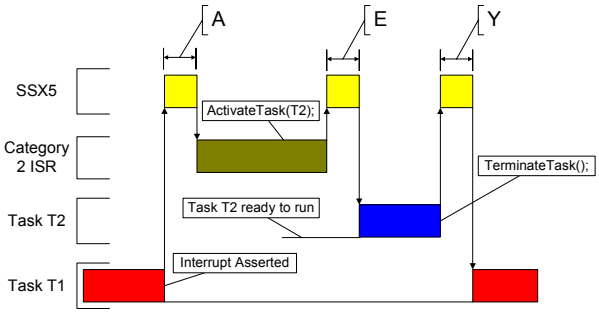


Figure 8: Category 2 ISR Activates a Higher Priority Task

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	106	141	240	106	141	240
Figure 1: D	Heavy, Basic/Extended	326	357	454	398	398	495
ChainTask	Light, Basic	294	432	751	322	432	829
Figure 2: J	Heavy, Basic/Extended	887	1059	1470	987	1100	1589
Pre-emption	Light, Basic	283	424	737	313	424	818
Figure 1: C	Heavy, Basic/Extended	478	593	908	638	733	1127
From idle task	Light, Basic	283	424	737	313	424	818
Figure 3: H	Heavy, Basic/Extended	478	593	908	638	733	1127
Triggered by alarm	Light, Basic	471	612	925	501	612	1006
Figure 4: F	Heavy, Basic/Extended	666	781	1096	826	921	1315
Schedule	Light, Basic	222	248	431	222	248	431
Figure 5: Q	Heavy, Basic/Extended	417	417	602	547	547	728
Release resource	Light, Basic	263	289	425	263	289	425
Figure 6: M	Heavy, Basic/Extended	458	458	596	588	588	722
SetEvent							
Figure 7: S	Heavy, Extended	n/a	n/a	n/a	1095	1095	1549
From category 2 ISR	Light, Basic	173	199	335	173	199	335
Figure 8: E	Heavy, Basic/Extended	368	368	506	498	498	632

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	391	424	523	391	424	523
Figure 1: D	Heavy, Basic/Extended	611	640	737	681	681	778
ChainTask	Light, Basic	700	814	1129	728	814	1207
Figure 2: J	Heavy, Basic/Extended	1578	1724	2087	1676	1765	2206
Pre-emption	Light, Basic	570	687	996	600	687	1077
Figure 1: C	Heavy, Basic/Extended	741	856	1167	901	996	1386
From idle task	Light, Basic	570	687	996	600	687	1077
Figure 3: H	Heavy, Basic/Extended	741	856	1167	901	996	1386
Triggered by alarm	Light, Basic	758	875	1184	788	875	1265
Figure 4: F	Heavy, Basic/Extended	929	1044	1355	1089	1184	1574
Schedule	Light, Basic	509	511	690	509	511	690
Figure 5: Q	Heavy, Basic/Extended	680	680	861	810	810	987

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations	Task Attributes	No	Yes		No	Yes	
Release resource	Light, Basic	550	552	684	550	552	684
Figure 6: M	Heavy, Basic/Extended	721	721	855	851	851	981
SetEvent							
Figure 7: S	Heavy, Extended	n/a	n/a	n/a	1343	1343	1793
From category 2 ISR	Light, Basic	761	763	895	761	763	895
Figure 8: E	Heavy, Basic/Extended	932	932	1066	1062	1062	1192

Extended

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
Normal termination	Light, Basic	491	524	622	491	524	622
Figure 1: D	Heavy, Basic/Extended	710	739	835	780	780	876
ChainTask	Light, Basic	1204	1366	1674	1275	1366	1750
Figure 2: J	Heavy, Basic/Extended	2226	2372	2776	2324	2369	2851
Pre-emption	Light, Basic	1030	1135	1443	1056	1135	1531
Figure 1: C	Heavy, Basic/Extended	1202	1305	1615	1358	1445	1841
From idle task	Light, Basic	1030	1135	1443	1056	1135	1531
Figure 3: H	Heavy, Basic/Extended	1202	1305	1615	1358	1445	1841
Triggered by alarm	Light, Basic	1218	1323	1631	1244	1323	1719
Figure 4: F	Heavy, Basic/Extended	1390	1493	1803	1546	1633	2029
Schedule	Light, Basic	551	553	732	551	553	732
Figure 5: Q	Heavy, Basic/Extended	723	723	904	853	853	1030
Release resource	Light, Basic	1049	1051	1183	1154	1156	1288
Figure 6: M	Heavy, Basic/Extended	1221	1221	1355	1456	1456	1586
SetEvent							
Figure 7: S	Heavy, Extended	n/a	n/a	n/a	1815	1815	2263
From category 2 ISR	Light, Basic	773	775	907	773	775	907
Figure 8: E	Heavy, Basic/Extended	945	945	1079	1075	1075	1205

4.4 Configuration of Runtime Context

The runtime contexts of all tasks reside on the same stack and are recovered when the task terminates. This causes runtime contexts of mutually exclusive tasks to be effectively overlaid. RTArchitect can calculate the worst-case stack requirement for the entire application based on the declared stack usage, priorities and resource occupation of individual tasks.

The size of the runtime 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			
Multiple Task Activations	No	Yes		No	Yes	
Pre- and Post-Task hooks not used						
Task type						
BCC1 lightweight, integer	19	19	19	19	19	19
BCC1 lightweight, floating point	21	21	21	21	21	21
BCC1 heavyweight, integer	25	25	25	25	25	25
BCC1 heavyweight, floating point	25	25	25	25	25	25
BCC2 lightweight, integer	n/a	21	23	n/a	21	23
BCC2 lightweight, floating point	n/a	21	23	n/a	21	23
BCC2 heavyweight, integer	n/a	25	27	n/a	25	27
BCC2 heavyweight, floating point	n/a	25	27	n/a	25	27
ECC1 heavyweight, integer	n/a	n/a	n/a	28	28	28
ECC1 heavyweight, floating point	n/a	n/a	n/a	28	28	28
ECC2 heavyweight, integer	n/a	n/a	n/a	n/a	n/a	28
ECC2 heavyweight, floating point	n/a	n/a	n/a	n/a	n/a	28
Pre- and/or Post-Task hooks used						
Task type						
BCC1 lightweight, integer	19	19	19	19	19	19
BCC1 lightweight, floating point	21	21	21	21	21	21
BCC1 heavyweight, integer	25	25	25	25	25	25
BCC1 heavyweight, floating point	25	25	25	25	25	25
	n/a	21	23	n/a	21	23
BCC2 lightweight, floating point	n/a	21	23	n/a	21	23
BCC2 heavyweight, integer	n/a	25	27	n/a	25	27
BCC2 heavyweight, floating point	n/a	25	27	n/a	25	27
ECC1 heavyweight, integer	n/a	n/a	n/a	28	28	28

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		
Multiple Task Activations		No	Yes		No	Yes	
ECC1 heavyweight, floating point		n/a	n/a	n/a	28	28	28
ECC2 heavyweight, integer		n/a	n/a	n/a	n/a	n/a	28
ECC2 heavyweight, floating point		n/a	n/a	n/a	n/a	n/a	28

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		25	25	25	25	25	25
BCC1 lightweight, floating point		27	27	27	27	27	27
BCC1 heavyweight, integer		31	31	31	31	31	31
BCC1 heavyweight, floating point		31	31	31	31	31	31
BCC2 lightweight, integer		n/a	27	29	n/a	27	29
BCC2 lightweight, floating point		n/a	27	29	n/a	27	29
BCC2 heavyweight, integer		n/a	31	33	n/a	31	33
BCC2 heavyweight, floating point		n/a	31	33	n/a	31	33
ECC1 heavyweight, integer		n/a	n/a	n/a	34	34	34
ECC1 heavyweight, floating point		n/a	n/a	n/a	34	34	34
ECC2 heavyweight, integer		n/a	n/a	n/a	n/a	n/a	34
ECC2 heavyweight, floating point		n/a	n/a	n/a	n/a	n/a	34
Pre- and/or Post-Task hooks used							
Task type							
BCC1 lightweight, integer		25	25	25	25	25	25
BCC1 lightweight, floating point		27	27	27	27	27	27
BCC1 heavyweight, integer		31	31	31	31	31	31
BCC1 heavyweight, floating point		31	31	31	31	31	31
BCC2 lightweight, integer		n/a	27	29	n/a	27	29
BCC2 lightweight, floating point		n/a	27	29	n/a	27	29
BCC2 heavyweight, integer		n/a	31	33	n/a	31	33
BCC2 heavyweight, floating point		n/a	31	33	n/a	31	33
ECC1 heavyweight, integer		n/a	n/a	n/a	34	34	34
ECC1 heavyweight, floating point		n/a	n/a	n/a	34	34	34
ECC2 heavyweight, integer		n/a	n/a	n/a	n/a	n/a	34
ECC2 heavyweight, floating point		n/a	n/a	n/a	n/a	n/a	34

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	25	25	25	25	25	25
BCC1 lightweight, floating point	27	27	27	27	27	27
BCC1 heavyweight, integer	31	31	31	31	31	31
BCC1 heavyweight, floating point	31	31	31	31	31	31
BCC2 lightweight, integer	n/a	27	29	n/a	27	29
BCC2 lightweight, floating point	n/a	27	29	n/a	27	29
BCC2 heavyweight, integer	n/a	31	33	n/a	31	33
BCC2 heavyweight, floating point	n/a	31	33	n/a	31	33
ECC1 heavyweight, integer	n/a	n/a	n/a	34	34	34
ECC1 heavyweight, floating point	n/a	n/a	n/a	34	34	34
ECC2 heavyweight, integer	n/a	n/a	n/a	n/a	n/a	34
ECC2 heavyweight, floating point	n/a	n/a	n/a	n/a	n/a	34
Pre- and/or Post-Task hooks used						
Task type						
BCC1 lightweight, integer	25	25	25	25	25	25
BCC1 lightweight, floating point	27	27	27	27	27	27
BCC1 heavyweight, integer	31	31	31	31	31	31
BCC1 heavyweight, floating point	31	31	31	31	31	31
BCC2 lightweight, integer	n/a	27	29	n/a	27	29
BCC2 lightweight, floating point	n/a	27	29	n/a	27	29
BCC2 heavyweight, integer	n/a	31	33	n/a	31	33
BCC2 heavyweight, floating point	n/a	31	33	n/a	31	33
ECC1 heavyweight, integer	n/a	n/a	n/a	34	34	34
ECC1 heavyweight, floating point	n/a	n/a	n/a	34	34	34
ECC2 heavyweight, integer	n/a	n/a	n/a	n/a	n/a	34
ECC2 heavyweight, floating point	n/a	n/a	n/a	n/a	n/a	34

Support Details

Getting Help

There are a number of ways to contact LiveDevices for technical support. When you contact our support team, please provide your customer number.

Email

The preferred method for dealing with support inquiries is via email. Any issues should be sent to **support@livedevices.com**

Telephone

You can contact us by telephone during our normal office hours (0900-1730 GMT/BST). Our telephone number is +44 (0) 19 04 56 26 24

Fax

Our Fax number is +44 (0) 19 04 56 25 81

World Wide Web

You can keep up with the latest developments by looking at our web site **www.livedevices.com**

Write to Us

You can write to us at:

LiveDevices Ltd.
Atlas House
Link Business Park
Osbaldwick Link Road
Osbaldwick
York
YO10 3JB