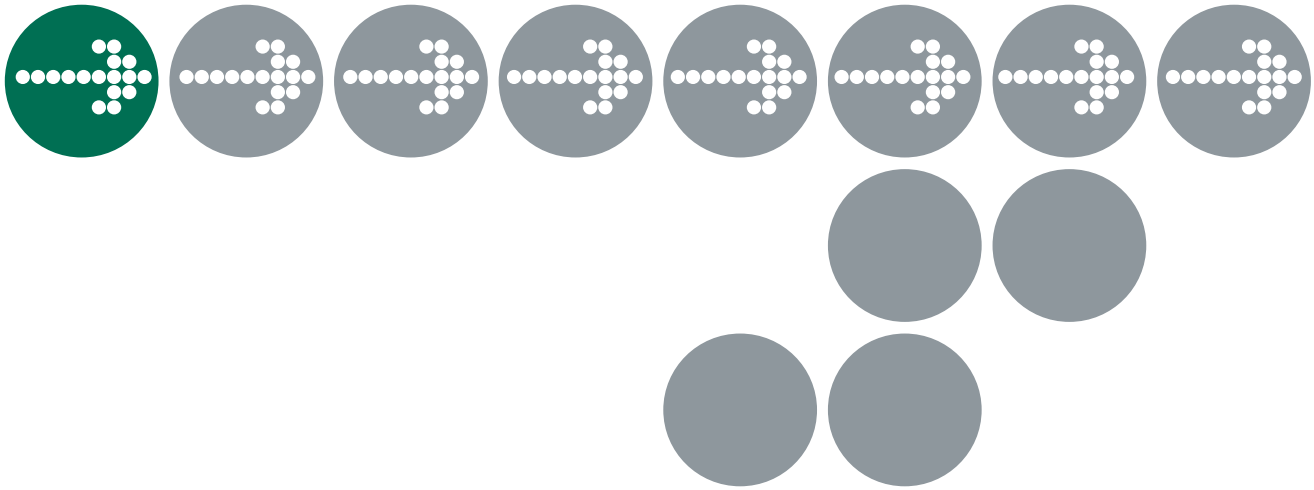
LiveDevices

Binding Manual: M16C/IAR



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: RM00041-001 .02

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 Guide

This guide provides port specific information for the M16C/IAR implementation of Realogy Real-Time Architect.

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 SSX5 on your target hardware. Port specific parameters of implementation are also provided, giving the RAM and ROM requirements for each SSX5 object and execution times for each SSX5 API call.

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 SSX5 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 SSX5.

In this guide you'll see that program code, header file names, C type names, C functions and SSX5 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 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, you should contact LiveDevices to confirm whether or not this is possible.

2.1 Memory Model

SSX5 is built using the (default) "near" memory model, but with all internal read-only data qualified as `far` to allow it to be located in internal ROM.

2.2 Compiler

SSX5 was built using the following compiler:

Vendor	IAR
Compiler	M16C C
Version	2.10A

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

Option	Description
`-e`	Enable extended keywords
`--data_model=near`	Select the "near" data model
`--constant_data=far`	Constant data is accessed using "far" mode addressing
`--calling_convention=normal`	Use "normal" style calling convention (this is the compiler default)
`--align_data=2`	Use "word" data alignment (this is the compiler default)

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

Option	Description
`--calling_convention=simple`	Use "simple" style calling convention
`--64bit_doubles`	Use 64-bit doubles

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

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

Option	Description
`-e`	Enable extended keywords
`-s6`	Set optimization level
`--data_model=near`	Select the "near" data model
`--constant_data=far`	Constant data is accessed using "far" mode addressing
`--calling_convention=normal`	Use "normal" style calling convention (this is the compiler default)
`--align_data=2`	Use "word" data alignment (this is the compiler default)

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

Option	Description
`-r`	Generate code for debugging
`-h`	Use SB as frame pointer (required for certain debuggers)
`--calling_convention=simple`	Use "simple" style calling convention
`--64bit_doubles`	Use 64-bit doubles

2.2.1 Data Alignment

SSX5 is built for word data alignment (this is the compiler default). A run-time C library that supports word alignment is therefore required.

The start address of stack-based data (`auto` variables) within SSX5 tasks and Category 2 ISRs, however, is not necessarily word aligned. This is because interrupts can occur when the stack pointer is either odd or even.

2.2.2 Floating-point

SSX5 is built for 32-bit `double` variables (this is the compiler default). A run-time C library that supports 32-bit `double` variables is therefore required.

2.3 Assembler

SSX5 was built using the following assembler:

Vendor	IAR
Assembler	M16C
Version	2.10A

The assembly file that RTA generates from your OIL configuration file is called `osgen.s34`. This file defines configuration parameters for SSX5 when running your application.

2.4 Linker/Locator

In addition to the sections used by application code, the following RTA sections must be located:

Sections	ROM/RAM	Description
os_pid	ROM	SSX5 read-only data
os_pird	ROM	SSX5 initialization data
INTVEC	ROM	Variable vector table if generated by RTArchitect
INTVEC1	ROM	Fixed vector table if generated by RTArchitect
os_pir	RAM	SSX5 initialized data
os_pur	RAM	SSX5 uninitialized data

The following compiler run-time library functions are required by SSX5:

C Library Functions	Description
setjmp	
longjmp	

2.4.1 Run-time Library and C-Startup

The run-time library used must be compatible with the “near” memory model, “near” variables, “far” constant data, “32-bit” doubles, “word” data alignment and the “normal” calling convention.

Any run-time library compatible with the above constraints may be used, for example, the clm16cnnffwc library is suitable.

SSX5 only uses the interrupt stack. If the IAR C-startup code from the IAR run-time library is used unmodified, then the “U” bit in the FLG register must be cleared before SSX5 is started. The example application is supplied with a modified version of the IAR C-startup code that omits the instruction to switch to the “User” stack.

As distributed, the IAR C-startup code defines the INTVEC1 vector table using pre-defined names for each vector entry. The INTVEC1 vector table is also created by RTArchitect to define Category 1 ISRs that have been bound to vectors in the range 0xFFFDC to 0xFFFFC.

With RTArchitect, it is possible to configure a Category 1 ISR name other than the pre-defined one. In this case the tables created by RTArchitect would conflict with the unmodified IAR table and the IAR definition must be removed from the C-startup in cstartup.asm.

2.4.2 Linker Control File

SSX5 writable data in `os_pir` and `os_pur` must be located in the first 64K of the address space (as it is declared to the compiler as "`near`" data).

SSX5 read-only data in `os_pid` and `os_pird` may be located anywhere, but must not span a 64K boundary (as it is declared to the compiler as "`far`" rather than "`huge`" data).

When extended tasks are configured, the label `OS_STACK_END` must be defined to the linker as the top of the Interrupt Stack. If the Interrupt Stack is located from address 0x400 to address 0xBFF, for example, the linker option would be `-DOS_STACK_END=0C00`.

2.5 Debugger

ORTI is the OSEK Run-Time Interface. RTA does not currently support ORTI compatible debuggers for this target.

2.5.1 KD30 Debugger

In order to use debuggers, such as the KD30 debugger that operate with the Mitsubishi ROM monitor v2.0, a Category 1 ISR must be configured to RTA. This ISR must be placed on vector `0x50` with priority `7` and entry address `0xFF900`. You can achieve this by naming the ISR "`kd30_uart1_receive`" and using the linker option "`-Dkd30_uart1_receive=0FF900`".

The KD30 debugger requires IEEE695 format files created with the modifier flags `-ylmbs`. Further information can be found in the IAR documentation for `xlink`.

3 Target Hardware Issues

3.1 Interrupts

This section explains the implementation of the SSX5 interrupt model. You can find out more about configuring interrupts for SSX5 in the *RTA User Guide*.

3.1.1 Interrupt Levels

Interrupts, in SSX5, 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 User Guide*. The hardware interrupt controller is explained in the *M16C/62 Group User's Manual*.

The following table shows how SSX5 IPLs relate to interrupt priorities on the target hardware:

SSX5 IPL Value	Flag Register	Description
0	IPL field = 0, I bit = 1	User level
1..7	IPL field = 1..7, I bit = 1	Category 1 and 2 interrupts
8	I bit = 0	Category 1 interrupts only

3.1.2 Interrupt Vectors

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

Vector	Legality
0x4-0xFC	Category 1 or Category 2 asynchronous interrupts at SSX5 IPL 1-7
0x0-0xFC	Category 1 SWI at SSX5 IPL 8 (no API calls permitted)
0xFFFDC-0xFFFFC	Non-maskable Category 1 at SSX5 IPL 8 (no API calls permitted)

The valid base addresses for the vector table are:

Base Address	Notes
0xFFFDC	Fixed vector table (INTVEC1)
Any	Variable vector table (INTVEC)

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 IAR C compiler can generate appropriate interrupt handling code for a C function decorated with the `__interrupt` function qualifier. You can find out more in your compiler documentation.

3.1.4 Category 2 Handlers

Category 2 ISRs are provided with a C function context by SSX5, since SSX5 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 SSX5.

3.1.5 Vector Table Issues

When you configure your application with RTArchitect you can choose whether or not a pair of vector tables (one fixed and one variable) is generated within `osgen.s34`. Note that the generated fixed vector table omits the reset vector entry. If you choose to provide your own vector table, it must contain an entry for each interrupt handler, including the Category 2 interrupt handlers in SSX5.

The following table shows the syntax for labels attached to SSX5 Category 2 interrupt handlers (`VVVV` represents the 4 hex digit, upper-case, zero-padded value of the vector location).

Vector Location	Label
`0x0VVVV`	`os_wrapper_VVVV`
eg : `0x0005C`	`os_wrapper_005C`

3.1.6 Interrupt Control Registers

The Interrupt Control Register for Category 1 or Category 2 ISRs must be programmed before the OS is started. The IPL for ISR `<name>` is available as `OS_ISR_PRI_<name>`.

The example application uses this method to initialize the priority for the `TA0` timer interrupt (see the function `StartupHook()` in `target.c`).

3.1.7 Interrupt Handlers Use of Register Banks

SSX5 does not use register bank 1. This means that one or more Category 1 ISRs at a single IPL may use register bank 1 to improve their response time. Register bank 1 is selected when bit 4 (the "B" flag) is set in the `FLG` register.

The IAR extended keyword `__regbank_interrupt` can be used instead of `__interrupt` to declare a Category 1 interrupt function that uses the alternate register bank.

3.2 Register Settings

SSX5 requires the following registers to be initialized before calling `StartOS()`.

Register	Required Value	Notes
I bit of Flag Register	1	
B bit of Flag Register	0	SSX5 makes no use of register bank 1 though the user can set this bit for Category 1 ISRs that share the same IPL.
U bit of Flag Register	0	SSX5 applications use the Interrupt Stack
*xxx*IC Register	OS_ISR_PRI_<*ISR name*>	The `xxxIC` register corresponding to ISR <*ISR name*> must be programmed with the configured priority 1..7. The macro `OS_ISR_PRI_<ISR name>` expands to the appropriate value.

SSX5 uses the following hardware registers. They should not be altered by user code.

Register	Notes
Flag register's IPL field, I bit, B bit and U bit.	Category 1 ISRs may temporarily set the B bit.

3.3 Stack Usage

3.3.1 Number of Stacks

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

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

3.3.2 Stack Usage within API Calls

The maximum stack usage within SSX5 API calls, excluding calls to hooks and callbacks, is as follows:

Standard

API max usage (bytes): 48

Timing

API max usage (bytes): 48

Extended

API max usage (bytes): 65

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 SSX5.

SSX5 is highly scalable. As a result, 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 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	**M16C/62A**
Clock speed (MHz)	16
Code memory	On-chip FLASH
Read-only data memory	On-chip RAM
Read-write data memory	On-chip FLASH

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	16	16	16
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					

4.2 Hardware Resources

4.2.1 ROM and RAM Overheads

The following tables give the ROM and RAM overheads for SSX5 (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 / Events / Shared Task Priorities / Multiple Task Activations		Application Uses					
		No			Yes		
		No		Yes	No		Yes
		No	Yes		No	Yes	
OS overhead	RAM	16	16	16	16	16	16
	ROM	153	153	153	153	153	153
COM overhead	RAM	2	2	2	2	2	2
	ROM	7	7	7	7	7	7

Timing

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	193	193	193	193	193	193
COM overhead	RAM	2	2	2	2	2	2
	ROM	7	7	7	7	7	7

Extended

Configuration / Events / Shared Task Priorities / Multiple Task Activations		Application Uses					
		No			Yes		
		No		Yes	No		Yes
		No	Yes		No	Yes	
OS overhead	RAM	35	35	35	35	35	35
	ROM	225	225	225	225	225	225
COM overhead	RAM	2	2	2	2	2	2
	ROM	7	7	7	7	7	7

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. SSX5 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 SSX5. (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	30	30	30	30	30	30
BCC1 Heavyweight task	RAM	4	4	4	4	4	4
	ROM	34	34	34	34	34	34
BCC2 task	RAM	n/a	6	8	n/a	6	8
	ROM	n/a	36	44	n/a	36	44
ECC1, Integer task	RAM	n/a	n/a	n/a	28	28	28
	ROM	n/a	n/a	n/a	46	46	46
ECC1, floating-point task	RAM	n/a	n/a	n/a	29	29	29
	ROM	n/a	n/a	n/a	46	46	46
ECC2, Integer task	RAM	n/a	n/a	n/a	n/a	n/a	30
	ROM	n/a	n/a	n/a	n/a	n/a	54
ECC2, floating-point task	RAM	n/a	n/a	n/a	n/a	n/a	31
	ROM	n/a	n/a	n/a	n/a	n/a	54
Category 2 ISR	RAM	0	0	0	0	0	0
	ROM	69	69	69	69	69	69

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
Category 2 ISR, floating-point	RAM	1	1	1	1	1	1
	ROM	86	86	86	86	86	86
Resource	RAM	0	0	0	0	0	0
	ROM	18	18	18	18	18	18
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	18	18	18	18	18	18
Alarm	RAM	6	6	6	6	6	6
	ROM	44	44	44	44	44	44
Counter	RAM	2	2	2	2	2	2
	ROM	31	31	31	31	31	31
Message	RAM	11	11	11	31	31	31
	ROM	18	18	18	48	48	48
Flag	RAM	1	1	1	1	1	1
	ROM	2	2	2	2	2	2
Message resource	RAM	0	0	0	0	0	0
	ROM	18	18	18	18	18	18
Event	RAM	0	0	0	0	0	0
	ROM	2	2	2	2	2	2
Priority level	RAM	0	0	6	0	6	6
	ROM	0	0	8	0	8	8
Arrivalpoint (readonly)	RAM	0	0	0	0	0	0
	ROM	8	8	8	8	8	8
Arrivalpoint (writable)	RAM	8	8	8	8	8	8
	ROM	8	8	8	8	8	8
Schedule	RAM	10	10	10	10	10	10
	ROM	32	32	32	32	32	32
Taskset (readonly)	RAM	0	0	0	0	0	0
	ROM	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	6	6	6	6	6	6
	ROM	38	38	38	38	38	38
BCC1 Heavyweight task	RAM	10	10	10	10	10	10
	ROM	42	42	42	42	42	42
BCC2 task	RAM	n/a	12	14	n/a	12	14
	ROM	n/a	44	52	n/a	44	52
ECC1, Integer task	RAM	n/a	n/a	n/a	34	34	34
	ROM	n/a	n/a	n/a	54	54	54
ECC1, floating-point task	RAM	n/a	n/a	n/a	35	35	35
	ROM	n/a	n/a	n/a	54	54	54
ECC2, Integer task	RAM	n/a	n/a	n/a	n/a	n/a	36
	ROM	n/a	n/a	n/a	n/a	n/a	62
ECC2, floating-point task	RAM	n/a	n/a	n/a	n/a	n/a	37
	ROM	n/a	n/a	n/a	n/a	n/a	62
Category 2 ISR	RAM	6	6	6	6	6	6
	ROM	100	100	100	100	100	100
Category 2 ISR, floating-point	RAM	7	7	7	7	7	7
	ROM	108	108	108	108	108	108
Resource	RAM	0	0	0	0	0	0
	ROM	18	18	18	18	18	18
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	18	18	18	18	18	18
Alarm	RAM	6	6	6	6	6	6
	ROM	44	44	44	44	44	44
Counter	RAM	2	2	2	2	2	2
	ROM	31	31	31	31	31	31
Message	RAM	11	11	11	31	31	31
	ROM	18	18	18	48	48	48
Flag	RAM	1	1	1	1	1	1
	ROM	2	2	2	2	2	2
Message resource	RAM	0	0	0	0	0	0
	ROM	18	18	18	18	18	18
Event	RAM	0	0	0	0	0	0
	ROM	2	2	2	2	2	2
Priority level	RAM	0	0	6	0	6	6
	ROM	0	0	8	0	8	8

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	8	8	8	8	8	8
Arrivalpoint (writable)	RAM	8	8	8	8	8	8
	ROM	8	8	8	8	8	8
Schedule	RAM	10	10	10	10	10	10
	ROM	32	32	32	32	32	32
Taskset (readonly)	RAM	0	0	0	0	0	0
	ROM	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	7	7	7	7	7	7
	ROM	44	44	44	44	44	44
BCC1 Heavyweight task	RAM	12	12	12	12	12	12
	ROM	44	44	44	44	44	44
BCC2 task	RAM	n/a	14	16	n/a	14	16
	ROM	n/a	46	54	n/a	46	54
ECC1, Integer task	RAM	n/a	n/a	n/a	36	36	36
	ROM	n/a	n/a	n/a	56	56	56
ECC1, floating-point task	RAM	n/a	n/a	n/a	37	37	37
	ROM	n/a	n/a	n/a	56	56	56
ECC2, Integer task	RAM	n/a	n/a	n/a	n/a	n/a	38
	ROM	n/a	n/a	n/a	n/a	n/a	64
ECC2, floating-point task	RAM	n/a	n/a	n/a	n/a	n/a	39
	ROM	n/a	n/a	n/a	n/a	n/a	64
Category 2 ISR	RAM	7	7	7	7	7	7
	ROM	106	106	106	106	106	106
Category 2 ISR, floating-point	RAM	8	8	8	8	8	8
	ROM	114	114	114	114	114	114
Resource	RAM	4	4	4	4	4	4
	ROM	24	24	24	24	24	24
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	4	4	4	4	4	4
	ROM	24	24	24	24	24	24
Alarm	RAM	6	6	6	6	6	6
	ROM	48	48	48	48	48	48
Counter	RAM	2	2	2	2	2	2
	ROM	35	35	35	35	35	35
Message	RAM	11	11	11	31	31	31
	ROM	22	22	22	52	52	52
Flag	RAM	1	1	1	1	1	1
	ROM	2	2	2	2	2	2
Message resource	RAM	4	4	4	4	4	4
	ROM	24	24	24	24	24	24
Event	RAM	0	0	0	0	0	0
	ROM	2	2	2	2	2	2
Priority level	RAM	0	0	6	0	6	6
	ROM	0	0	8	0	8	8
Arrivalpoint (readonly)	RAM	0	0	0	0	0	0
	ROM	14	14	14	14	14	14
Arrivalpoint (writable)	RAM	14	14	14	14	14	14
	ROM	14	14	14	14	14	14
Schedule	RAM	12	12	12	12	12	12
	ROM	38	38	38	38	38	38
Taskset (readonly)	RAM	0	0	0	0	0	0
	ROM	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. 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 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. The smallest and fastest call will be selected. 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.
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	123	257	414	129	263	441
	NS		105	239	395	111	245	422
	KL	2	72	221	374	78	227	401
TerminateTask	LExt	3	n/a	n/a	n/a	n/a	n/a	n/a
	H	5	55	55	55	55	55	55
ChainTask	SWL	1, 8	121	262	423	127	268	447
	SWH	1, 9	157	278	432	163	284	458
	NSL	8	121	262	423	127	268	447
	NSH	9	153	274	428	159	280	454

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
Schedule			96	96	155	96	96	155
GetTaskID			58	58	58	58	58	58
GetTaskState			80	80	80	93	93	93
EnableAllInterrupts			17	17	17	17	17	17
DisableAllInterrupts			15	15	15	15	15	15
ResumeAllInterrupts			26	26	26	26	26	26
SuspendAllInterrupts			24	24	24	24	24	24
ResumeOSInterrupts			45	45	45	45	45	45
SuspendOSInterrupts			54	54	54	54	54	54
GetResource	Task	7	66	66	76	66	66	76
	Combined	6	118	118	118	118	118	118
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
ReleaseResource	Task	7	75	75	75	75	75	75
	Combined	6	168	168	168	168	168	168
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	1	n/a	n/a	n/a	183	183	410
	NS		n/a	n/a	n/a	161	161	391
	NS1i	10	n/a	n/a	n/a	68	n/a	n/a
	KL	2	n/a	n/a	n/a	150	150	374
	KL1i	2, 10	n/a	n/a	n/a	56	n/a	n/a
ClearEvent			n/a	n/a	n/a	60	60	60
GetEvent			n/a	n/a	n/a	42	42	42
WaitEvent	<default>		n/a	n/a	n/a	364	364	691
	fp	11	n/a	n/a	n/a	432	432	821
	1i	10	n/a	n/a	n/a	43	n/a	n/a
GetAlarmBase			56	56	56	56	56	56
GetAlarm			155	155	155	155	155	155
SetRelAlarm			170	170	170	170	170	170
SetAbsAlarm			207	207	207	207	207	207
CancelAlarm			110	110	110	110	110	110
InitCounter			90	90	90	90	90	90
GetCounterValue			107	107	107	107	107	107
osek_tick_alarm	<default>		107	107	107	107	107	107
	KL	2	78	78	78	78	78	78
osek_incr_counter			87	87	87	87	87	87
GetActiveApplicationMode		30	n/a	n/a	n/a	n/a	n/a	n/a
StartOS			89	89	89	89	89	89
ShutdownOS	NoHook	12	17	17	17	17	17	17
	Hook	13	27	27	27	27	27	27
InitCOM			6	6	6	6	6	6

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
CloseCOM			6	6	6	6	6	6
StartCOM			25	25	25	25	25	25
StopCOM			14	14	14	14	14	14
ReadFlag		30	n/a	n/a	n/a	n/a	n/a	n/a
ResetFlag		30	n/a	n/a	n/a	n/a	n/a	n/a
ReceiveMessage	CCCA	14	92	92	92	479	479	479
	CCCB	15	479	479	479	479	479	479
GetMessageResource			76	76	76	76	76	76
ReleaseMessageResource			79	79	79	79	79	79
GetMessageStatus			123	123	123	123	123	123
SendMessage	SW CCCA	1, 14	146	146	146	575	575	575
	SW CCCB	1, 15	529	529	529	575	575	575
	NS CCCA	14	146	146	146	575	575	575
	NS CCCB	15	529	529	529	575	575	575
	KL CCCA	2, 14	132	132	132	561	561	561
	KL CCCB	2, 15	515	515	515	561	561	561
main_dispatch	NoHook	12	154	154	265	154	154	265
	Hook	13	214	214	325	214	214	325
sub_dispatch	B1LF	19	41	41	41	41	41	41
	B1HI	20	123	123	123	123	123	123
	B1HF	21	131	131	131	131	131	131
	B2LI	22	n/a	141	227	n/a	141	227
	B2LF	23	n/a	149	235	n/a	149	235
	B2HI	24	n/a	237	501	n/a	237	501
	B2HF	25	n/a	245	509	n/a	245	509
	E1HI	26	n/a	n/a	n/a	484	484	730
	E1HF	27	n/a	n/a	n/a	492	492	738
	E2HI	28	n/a	n/a	n/a	n/a	n/a	730
	E2HF	29	n/a	n/a	n/a	n/a	n/a	738
ErrorHook support		16	23	23	23	23	23	23
	ServiceID	17	33	33	33	33	33	33
	Parameters	18	70	70	70	70	70	70
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	58	162	358	66	179	399
	NS		32	143	339	40	160	380
	KL	2	20	129	325	28	146	366
ChainTaskset	SWL	1, 8	59	152	348	59	161	381
	SWH	1, 9	113	209	406	113	218	439
	NSL	8	59	152	348	59	161	381
	NSH	9	109	205	402	109	214	435

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
GetTasksetRef			53	53	53	53	53	53
MergeTaskset			47	47	47	47	47	47
AssignTaskset			24	24	24	24	24	24
RemoveTaskset			49	49	49	49	49	49
TestSubTaskset			60	60	60	60	60	60
TestEquivalentTaskset			58	58	58	58	58	58
TickSchedule	SW	1	328	380	380	380	380	380
	NS		309	361	361	361	361	361
	KL	2	295	347	347	347	347	347
AdvanceSchedule	SW	1	262	286	286	286	286	286
	NS		243	267	267	267	267	267
	KL	2	229	253	253	253	253	253
StartSchedule			149	149	149	149	149	149
StopSchedule			93	93	93	93	93	93
GetScheduleStatus			154	154	154	154	154	154
GetScheduleValue			99	99	99	99	99	99
GetScheduleNext			61	61	61	61	61	61
SetScheduleNext			70	70	70	70	70	70
GetArrivalpointDelay			24	24	24	24	24	24
SetArrivalpointDelay			20	20	20	20	20	20
GetArrivalpointTasksetRef			29	29	29	29	29	29
GetArrivalpointNext			49	49	49	49	49	49
SetArrivalpointNext			48	48	48	48	48	48
TestArrivalpointWritable			44	44	44	44	44	44
GetExecutionTime			7	7	7	7	7	7
GetLargestExecutionTime			8	8	8	8	8	8
ResetLargestExecutionTime			6	6	6	6	6	6
GetStackOffset			23	23	23	23	23	23
Floating-point support			28	28	28	28	28	28
Interrupt support			53	53	53	53	53	53
Utility functions			67	67	67	67	67	67
Utility functions			23	23	23	23	23	23

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	123	257	414	129	263	441
	NS		105	239	395	111	245	422
	KL	2	72	221	374	78	227	401
TerminateTask	LExt	3	n/a	n/a	n/a	n/a	n/a	n/a
	H	5	55	55	55	55	55	55
ChainTask	SWL	1, 8	121	262	423	127	268	447
	SWH	1, 9	157	278	432	163	284	458
	NSL	8	121	262	423	127	268	447
	NSH	9	153	274	428	159	280	454
Schedule			105	105	164	105	105	164
GetTaskID			58	58	58	58	58	58
GetTaskState			80	80	80	93	93	93
EnableAllInterrupts			17	17	17	17	17	17
DisableAllInterrupts			15	15	15	15	15	15
ResumeAllInterrupts			26	26	26	26	26	26
SuspendAllInterrupts			24	24	24	24	24	24
ResumeOSInterrupts			45	45	45	45	45	45
SuspendOSInterrupts			54	54	54	54	54	54
GetResource	Task	7	66	66	76	66	66	76
	Combined	6	118	118	118	118	118	118
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
ReleaseResource	Task	7	84	84	84	84	84	84
	Combined	6	186	186	186	186	186	186
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	1	n/a	n/a	n/a	183	183	410
	NS		n/a	n/a	n/a	161	161	391
	NS1i	10	n/a	n/a	n/a	68	n/a	n/a
	KL	2	n/a	n/a	n/a	150	150	374
	KL1i	2, 10	n/a	n/a	n/a	56	n/a	n/a
ClearEvent			n/a	n/a	n/a	60	60	60
GetEvent			n/a	n/a	n/a	42	42	42
WaitEvent	<default>		n/a	n/a	n/a	364	364	691
	fp	11	n/a	n/a	n/a	432	432	821
	1i	10	n/a	n/a	n/a	43	n/a	n/a
GetAlarmBase			56	56	56	56	56	56
GetAlarm			155	155	155	155	155	155
SetRelAlarm			170	170	170	170	170	170
SetAbsAlarm			207	207	207	207	207	207
CancelAlarm			110	110	110	110	110	110

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
InitCounter			90	90	90	90	90	90
GetCounterValue			107	107	107	107	107	107
osek_tick_alarm	<default>		107	107	107	107	107	107
	KL	2	78	78	78	78	78	78
osek_incr_counter			87	87	87	87	87	87
GetActiveApplicationMode		30	n/a	n/a	n/a	n/a	n/a	n/a
StartOS			154	154	154	154	154	154
ShutdownOS	NoHook	12	17	17	17	17	17	17
	Hook	13	27	27	27	27	27	27
InitCOM			6	6	6	6	6	6
CloseCOM			6	6	6	6	6	6
StartCOM			25	25	25	25	25	25
StopCOM			14	14	14	14	14	14
ReadFlag		30	n/a	n/a	n/a	n/a	n/a	n/a
ResetFlag		30	n/a	n/a	n/a	n/a	n/a	n/a
ReceiveMessage	CCCA	14	92	92	92	479	479	479
	CCCB	15	479	479	479	479	479	479
GetMessageResource			76	76	76	76	76	76
ReleaseMessageResource			79	79	79	79	79	79
GetMessageStatus			123	123	123	123	123	123
SendMessage	SW CCCA	1, 14	146	146	146	575	575	575
	SW CCCB	1, 15	529	529	529	575	575	575
	NS CCCA	14	146	146	146	575	575	575
	NS CCCB	15	529	529	529	575	575	575
	KL CCCA	2, 14	132	132	132	561	561	561
	KL CCCB	2, 15	515	515	515	561	561	561
main_dispatch	NoHook	12	294	294	405	294	294	405
	Hook	13	358	358	469	358	358	469
sub_dispatch	B1LF	19	17	17	17	17	17	17
	B1HI	20	116	116	116	116	116	116
	B1HF	21	124	124	124	124	124	124
	B2LI	22	n/a	117	203	n/a	117	203
	B2LF	23	n/a	125	211	n/a	125	211
	B2HI	24	n/a	221	485	n/a	221	485
	B2HF	25	n/a	229	493	n/a	229	493
	E1HI	26	n/a	n/a	n/a	518	518	764
	E1HF	27	n/a	n/a	n/a	526	526	772
	E2HI	28	n/a	n/a	n/a	n/a	n/a	764
	E2HF	29	n/a	n/a	n/a	n/a	n/a	772
ErrorHook support		16	23	23	23	23	23	23
	ServiceID	17	33	33	33	33	33	33
	Parameters	18	70	70	70	70	70	70

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
validity_checks		3	n/a	n/a	n/a	n/a	n/a	n/a
Timing_dispatch		4	100	100	100	100	100	100
Timing_termination		4	180	180	180	180	180	180
ActivateTaskset	SW	1	58	162	358	66	179	399
	NS		32	143	339	40	160	380
	KL	2	20	129	325	28	146	366
ChainTaskset	SWL	1, 8	59	152	348	59	161	381
	SWH	1, 9	113	209	406	113	218	439
	NSL	8	59	152	348	59	161	381
	NSH	9	109	205	402	109	214	435
GetTasksetRef			53	53	53	53	53	53
MergeTaskset			47	47	47	47	47	47
AssignTaskset			24	24	24	24	24	24
RemoveTaskset			49	49	49	49	49	49
TestSubTaskset			60	60	60	60	60	60
TestEquivalentTaskset			58	58	58	58	58	58
TickSchedule	SW	1	328	380	380	380	380	380
	NS		309	361	361	361	361	361
	KL	2	295	347	347	347	347	347
AdvanceSchedule	SW	1	262	286	286	286	286	286
	NS		243	267	267	267	267	267
	KL	2	229	253	253	253	253	253
StartSchedule			149	149	149	149	149	149
StopSchedule			93	93	93	93	93	93
GetScheduleStatus			154	154	154	154	154	154
GetScheduleValue			99	99	99	99	99	99
GetScheduleNext			61	61	61	61	61	61
SetScheduleNext			70	70	70	70	70	70
GetArrivalpointDelay			24	24	24	24	24	24
SetArrivalpointDelay			20	20	20	20	20	20
GetArrivalpointTasksetRef			29	29	29	29	29	29
GetArrivalpointNext			49	49	49	49	49	49
SetArrivalpointNext			48	48	48	48	48	48
TestArrivalpointWritable			44	44	44	44	44	44
GetExecutionTime			93	93	93	93	93	93
GetLargestExecutionTime			54	54	54	54	54	54
ResetLargestExecutionTime			50	50	50	50	50	50
GetStackOffset			23	23	23	23	23	23
Floating-point support			28	28	28	28	28	28
Interrupt support			235	235	235	235	235	235
Utility functions			67	67	67	67	67	67
Utility functions			23	23	23	23	23	23

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	257	390	551	264	396	578
	NS		356	491	643	363	497	670
	KL	2	194	331	483	201	337	510
TerminateTask	LExt	3	142	142	142	142	142	142
	H	5	199	199	199	199	199	199
ChainTask	SWL	1, 8	339	484	648	347	490	672
	SWH	1, 9	391	517	671	399	523	690
	NSL	8	476	616	769	483	622	793
	NSH	9	513	627	781	520	633	807
Schedule			241	241	300	241	241	300
GetTaskID			103	103	103	103	103	103
GetTaskState			279	279	279	273	273	273
EnableAllInterrupts			27	27	27	27	27	27
DisableAllInterrupts			25	25	25	25	25	25
ResumeAllInterrupts			76	76	76	76	76	76
SuspendAllInterrupts			34	34	34	34	34	34
ResumeOSInterrupts			95	95	95	95	95	95
SuspendOSInterrupts			64	64	64	64	64	64
GetResource	Task	7	671	671	571	671	671	571
	Combined	6	582	582	582	582	582	582
	CLEx	3	539	539	539	539	539	539
ReleaseResource	Task	7	493	493	493	493	493	493
	Combined	6	554	554	554	554	554	554
	CLEx	3	458	458	458	458	458	458
SetEvent	SW	1	n/a	n/a	n/a	373	373	603
	NS		n/a	n/a	n/a	452	452	673
	NS1i	10	n/a	n/a	n/a	266	n/a	n/a
	KL	2	n/a	n/a	n/a	312	312	535
	KL1i	2, 10	n/a	n/a	n/a	225	n/a	n/a
ClearEvent			n/a	n/a	n/a	148	148	148
GetEvent			n/a	n/a	n/a	214	214	214
WaitEvent	<default>		n/a	n/a	n/a	522	522	824
	fp	11	n/a	n/a	n/a	590	590	954
	1i	10	n/a	n/a	n/a	197	n/a	n/a
GetAlarmBase			177	177	177	177	177	177
GetAlarm			223	223	223	223	223	223
SetRelAlarm			307	307	307	307	307	307

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
SetAbsAlarm			344	344	344	344	344	344
CancelAlarm			185	185	185	185	185	185
InitCounter			213	213	213	213	213	213
GetCounterValue			222	222	222	222	222	222
osek_tick_alarm	<default>		138	138	138	138	138	138
	KL	2	78	78	78	78	78	78
osek_incr_counter			87	87	87	87	87	87
GetActiveApplicationMode		30	n/a	n/a	n/a	n/a	n/a	n/a
StartOS			164	164	164	164	164	164
ShutdownOS	NoHook	12	22	22	22	22	22	22
	Hook	13	32	32	32	32	32	32
InitCOM			6	6	6	6	6	6
CloseCOM			6	6	6	6	6	6
StartCOM			35	35	35	35	35	35
StopCOM			28	28	28	28	28	28
ReadFlag			21	21	21	21	21	21
ResetFlag			22	22	22	22	22	22
ReceiveMessage	CCCA	14	177	177	177	568	568	568
	CCCB	15	568	568	568	568	568	568
GetMessageResource			116	116	116	116	116	116
ReleaseMessageResource			115	115	115	115	115	115
GetMessageStatus			158	158	158	158	158	158
SendMessage	SW CCCA	1, 14	243	243	243	670	670	670
	SW CCCB	1, 15	624	624	624	670	670	670
	NS CCCA	14	243	243	243	670	670	670
	NS CCCB	15	624	624	624	670	670	670
	KL CCCA	2, 14	202	202	202	629	629	629
	KL CCCB	2, 15	583	583	583	629	629	629
main_dispatch	NoHook	12	294	294	405	294	294	405
	Hook	13	358	358	469	358	358	469
sub_dispatch	B1LF	19	17	17	17	17	17	17
	B1HI	20	118	118	118	118	118	118
	B1HF	21	126	126	126	126	126	126
	B2LI	22	n/a	117	203	n/a	117	203
	B2LF	23	n/a	125	211	n/a	125	211
	B2HI	24	n/a	223	487	n/a	223	487
	B2HF	25	n/a	231	495	n/a	231	495
	E1HI	26	n/a	n/a	n/a	522	522	768
	E1HF	27	n/a	n/a	n/a	530	530	776
	E2HI	28	n/a	n/a	n/a	n/a	n/a	768
	E2HF	29	n/a	n/a	n/a	n/a	n/a	776

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
ErrorHook support		16	76	76	76	76	76	76
	ServiceID	17	81	81	81	81	81	81
	Parameters	18	120	120	120	120	120	120
validity_checks		3	61	61	61	61	61	61
Timing_dispatch		4	100	100	100	100	100	100
Timing_termination		4	180	180	180	180	180	180
ActivateTaskset	SW	1	427	511	701	413	532	755
	NS		508	592	782	494	613	836
	KL	2	359	443	633	345	464	687
ChainTaskset	SWL	1, 8	544	618	806	520	631	852
	SWH	1, 9	615	693	882	591	706	928
	NSL	8	644	718	906	620	731	952
	NSH	9	711	789	978	687	802	1024
GetTasksetRef			226	226	226	226	226	226
MergeTaskset			348	348	348	348	348	348
AssignTaskset			289	289	289	289	289	289
RemoveTaskset			350	350	350	350	350	350
TestSubTaskset			360	360	360	360	360	360
TestEquivalentTaskset			358	358	358	358	358	358
TickSchedule	SW	1	625	543	543	543	543	543
	NS		717	681	681	681	681	681
	KL	2	570	491	491	491	491	491
AdvanceSchedule	SW	1	554	473	473	473	473	473
	NS		656	618	618	618	618	618
	KL	2	517	428	428	428	428	428
StartSchedule			367	367	367	367	367	367
StopSchedule			226	226	226	226	226	226
GetScheduleStatus			293	293	293	293	293	293
GetScheduleValue			232	232	232	232	232	232
GetScheduleNext			199	199	199	199	199	199
SetScheduleNext			346	346	346	346	346	346
GetArrivalpointDelay			172	172	172	172	172	172
SetArrivalpointDelay			216	216	216	216	216	216
GetArrivalpointTasksetRef			223	223	223	223	223	223
GetArrivalpointNext			228	228	228	228	228	228
SetArrivalpointNext			402	402	402	402	402	402
TestArrivalpointWritable			191	191	191	191	191	191
GetExecutionTime			134	134	134	134	134	134
GetLargestExecutionTime			132	132	132	132	132	132
ResetLargestExecutionTime			127	127	127	127	127	127

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
GetStackOffset			23	23	23	23	23	23
Floating-point support			28	28	28	28	28	28
Interrupt support			235	235	235	235	235	235
Utility functions			67	67	67	67	67	67
Utility functions			23	23	23	23	23	23

Notes

Number	Note
1	Linked only if upward activations are allowed
2	Linked only if API is called within ISR
3	Present only in Extended OS status
4	Present only in Timing or Extended OS status
5	Linked only if there are heavyweight tasks in the system
6	Linked only if Resource is used by both tasks and ISRs
7	Linked only if Resource is used only by tasks
8	Linked only if Chaining task is Lightweight
9	Linked only if Chaining task is Heavyweight
10	Linked only if Idle task is the only extended task in the system
11	Linked only if calling Extended task uses floating-point
12	Linked only if neither Pre- nor Post-TaskHook is used
13	Linked only if Pre- or Post-TaskHook is used
14	Linked only if there are no flags, message queues, or message resources in the system, and COM status is not requested.
15	Linked only if there are any flags, message queues, or message resources in the system, or COM status is requested.
16	Linked only if `USEGETSERVICEID = FALSE` and `USEPARAMETERACCESS = FALSE`
17	Linked only if `USEGETSERVICEID = TRUE` and `USEPARAMETERACCESS = FALSE`
18	Linked only if `USEGETSERVICEID = TRUE` and `USEPARAMETERACCESS = TRUE`
19	Linked only for basic, single-activation, lightweight, floating-point tasks
20	Linked only for basic, single-activation, heavyweight, integer tasks
21	Linked only for basic, single-activation, heavyweight, floating-point tasks
22	Linked only for basic, multiple-activation, lightweight, integer tasks
23	Linked only for basic, multiple-activation, lightweight, floating-point tasks
24	Linked only for basic, multiple-activation, heavyweight, integer tasks
25	Linked only for basic, multiple-activation, heavyweight, floating-point tasks

Number	Note
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 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. (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	158	291	495	161	214	416
	NS	142	294	480	142	205	408
	KL	91	222	405	93	144	352
TerminateTask	LExt	0	0	0	0	0	0
	H	299	303	317	299	285	299
ChainTask	SWL	538	671	970	640	682	1020
	SWH	646	738	1056	751	748	1093
	NSL	532	665	987	646	688	1026
	NSH	641	735	1068	748	746	1101
Schedule	SW	174	174	187	178	153	180
GetTaskID		125	125	127	122	124	122
GetTaskState		162	162	162	158	163	163
EnableAllInterrupts		38	38	38	41	39	39

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
DisableAllInterrupts		51	51	51	48	50	50
ResumeAllInterrupts		48	48	48	45	47	47
SuspendAllInterrupts		56	56	56	59	65	65
ResumeOSInterrupts		48	48	48	45	47	47
SuspendOSInterrupts		56	56	56	59	65	65
GetResource	Task	118	118	126	115	113	123
	Combined	197	197	197	200	182	182
	CLEx	n/a	n/a	n/a	n/a	n/a	n/a
ReleaseResource	Task	155	155	155	153	140	140
	Combined	206	206	206	205	187	187
	CLEx	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	n/a	n/a	n/a	228	237	242
	NS	n/a	n/a	n/a	236	228	247
	KL	n/a	n/a	n/a	196	174	192
ClearEvent		n/a	n/a	n/a	117	130	117
GetEvent		n/a	n/a	n/a	81	88	81
WaitEvent	<default>	n/a	n/a	n/a	862	806	990
	fp	n/a	n/a	n/a	898	834	1087
GetAlarmBase		158	150	150	167	159	159
GetAlarm		213	225	225	206	218	218
SetRelAlarm		236	230	230	245	239	239
SetAbsAlarm		242	235	235	233	226	226
CancelAlarm		153	154	154	160	161	161
InitCounter		149	149	150	157	158	157
GetCounterValue		152	152	156	161	165	161
osek_tick_alarm	<default>	204	200	200	193	189	189
	KL	132	135	135	144	147	147
osek_incr_counter		50	50	50	39	39	39
GetActiveApplicationMode		6	6	6	11	11	11
StartOS		433	433	433	434	424	424
ShutdownOS	NoHook	n/a	n/a	n/a	n/a	n/a	n/a
	Hook	109	109	109	113	115	115
InitCOM		26	26	26	29	30	30
CloseCOM		33	33	33	30	31	31
StartCOM		96	96	96	391	395	343
StopCOM		42	42	42	40	39	39
ReadFlag		n/a	n/a	n/a	16	16	16
ResetFlag		n/a	n/a	n/a	15	15	15
ReceiveMessage		148	148	148	634	584	584
GetMessageResource		n/a	n/a	n/a	255	247	238

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
ReleaseMessageResource		n/a	n/a	n/a	310	291	292
GetMessageStatus		n/a	n/a	n/a	156	170	156
SendMessage	SW	336	473	677	787	814	1016
	NS	324	480	666	750	839	1042
	KL	219	364	547	633	739	947
ActivateTaskset	SW	129	608	958	139	575	973
	NS	112	597	947	122	559	998
	KL	52	542	892	64	509	907
	SW2	129	608	958	139	575	973
	NS2	112	597	947	122	559	998
	KL2	52	542	892	64	509	907
ChainTaskset	SWL	530	995	1461	634	1031	1591
	SWH	620	1102	1628	760	1138	1696
	NSL	531	993	1462	635	1046	1592
	NSH	618	1104	1626	755	1136	1691
GetTasksetRef		126	126	126	127	131	131
MergeTaskset		155	155	155	156	155	155
AssignTaskset		65	65	65	65	65	65
RemoveTaskset		137	137	137	138	139	139
TestSubTaskset		143	143	143	144	142	142
TestEquivalentTaskset		147	147	147	148	146	146
TickSchedule	SW	437	1031	1381	554	1009	1391
	NS	415	1009	1359	532	990	1372
	KL	367	961	1311	483	939	1321
	SW2	437	1031	1381	554	958	1356
	NS2	415	1009	1359	532	939	1337
	KL2	367	961	1311	483	888	1286
AdvanceSchedule	SW	304	887	1237	382	906	1288
	NS	281	868	1218	359	887	1269
	KL	232	819	1169	309	837	1219
	SW2	304	887	1237	382	855	1253
	NS2	281	868	1218	359	836	1234
	KL2	232	819	1169	309	786	1184
StartSchedule		226	226	226	227	234	234
StopSchedule		199	199	199	200	188	188
GetScheduleStatus		210	210	210	211	200	200
GetScheduleValue		176	176	176	177	170	170
GetScheduleNext		133	133	133	134	142	142
SetScheduleNext		151	151	151	152	160	160
GetArrivalpointDelay		60	60	60	60	62	62

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
SetArrivalpointDelay		53	53	53	53	55	55
GetArrivalpointTasksetRef		104	104	104	105	107	107
GetArrivalpointNext		119	119	119	120	126	126
SetArrivalpointNext		135	135	135	136	141	141
TestArrivalpointWritable		69	69	69	69	77	77
GetExecutionTime		27	27	27	27	27	27
GetLargestExecutionTime		40	40	40	40	40	40
ResetLargestExecutionTime		37	37	37	36	37	37
GetStackOffset		66	70	70	70	66	66

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	150	283	487	163	216	418
	NS	134	286	472	144	207	410
	KL	82	213	396	96	147	355
TerminateTask	LExt	0	0	0	0	0	0
	H	672	676	684	659	685	693
ChainTask	SWL	984	1155	1435	1085	1188	1478
	SWH	1073	1202	1493	1174	1233	1532
	NSL	990	1161	1464	1069	1182	1472
	NSH	1058	1189	1515	1161	1231	1540
Schedule	SW	161	161	188	157	160	173
GetTaskID		121	121	119	124	122	124
GetTaskState		152	152	152	162	167	167
EnableAllInterrupts		38	38	38	35	33	33
DisableAllInterrupts		45	45	45	48	50	50
ResumeAllInterrupts		42	42	42	45	47	47
SuspendAllInterrupts		56	56	56	53	59	59
ResumeOSInterrupts		42	42	42	45	47	47
SuspendOSInterrupts		56	56	56	53	59	59
GetResource	Task	112	112	120	115	113	123
	Combined	197	197	197	194	176	176
	CLEx	n/a	n/a	n/a	n/a	n/a	n/a
ReleaseResource	Task	148	148	148	150	141	141
	Combined	192	192	192	193	195	195
	CLEx	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	
SetEvent	SW	n/a	n/a	n/a	228	237	242
	NS	n/a	n/a	n/a	228	220	239
	KL	n/a	n/a	n/a	189	167	185
ClearEvent		n/a	n/a	n/a	122	109	122
GetEvent		n/a	n/a	n/a	82	75	82
WaitEvent	<default>	n/a	n/a	n/a	1123	1233	1401
	fp	n/a	n/a	n/a	1151	1269	1368
GetAlarmBase		164	156	156	155	147	147
GetAlarm		203	215	215	210	222	222
SetRelAlarm		242	236	236	233	227	227
SetAbsAlarm		230	223	223	239	232	232
CancelAlarm		157	158	158	150	151	151
InitCounter		155	155	154	147	146	147
GetCounterValue		162	162	158	153	149	153
osek_tick_alarm	<default>	190	186	186	201	197	197
	KL	141	144	144	129	132	132
osek_incr_counter		36	36	36	47	47	47
GetActiveApplicationMode		8	8	8	3	3	3
StartOS		859	859	859	858	854	854
ShutdownOS	NoHook	n/a	n/a	n/a	n/a	n/a	n/a
	Hook	109	109	109	105	107	107
InitCOM		26	26	26	23	24	24
CloseCOM		27	27	27	30	31	31
StartCOM		90	90	90	391	395	343
StopCOM		36	36	36	40	39	39
ReadFlag		n/a	n/a	n/a	15	15	15
ResetFlag		n/a	n/a	n/a	9	9	9
ReceiveMessage		150	150	150	634	584	584
GetMessageResource		n/a	n/a	n/a	264	238	247
ReleaseMessageResource		n/a	n/a	n/a	301	286	285
GetMessageStatus		n/a	n/a	n/a	169	155	169
SendMessage	SW	339	476	680	786	813	1015
	NS	327	483	669	749	838	1041
	KL	220	365	548	634	740	948
ActivateTaskset	SW	123	602	952	131	567	965
	NS	106	591	941	114	551	990
	KL	45	535	885	57	502	900
	SW2	123	602	952	131	567	965
	NS2	106	591	941	114	551	990
	KL2	45	535	885	57	502	900

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
ChainTaskset	SWL	986	1489	1936	1069	1536	2048
	SWH	1039	1568	2077	1181	1630	2142
	NSL	987	1477	1937	1070	1551	2049
	NSH	1037	1570	2065	1166	1619	2128
GetTasksetRef		132	132	132	131	135	135
MergeTaskset		144	144	144	143	142	142
AssignTaskset		66	66	66	66	66	66
RemoveTaskset		131	131	131	130	131	131
TestSubTaskset		147	147	147	146	144	144
TestEquivalentTaskset		139	139	139	138	136	136
TickSchedule	SW	396	982	1332	503	1040	1422
	NS	377	963	1313	484	1018	1400
	KL	326	912	1262	434	970	1352
	SW2	396	982	1332	503	989	1387
	NS2	377	963	1313	484	967	1365
	KL2	326	912	1262	434	919	1317
AdvanceSchedule	SW	294	877	1227	370	894	1276
	NS	271	858	1208	347	875	1257
	KL	221	808	1158	298	826	1208
	SW2	294	877	1227	370	843	1241
	NS2	271	858	1208	347	824	1222
	KL2	221	808	1158	298	775	1173
StartSchedule		238	238	238	237	230	230
StopSchedule		191	191	191	190	202	202
GetScheduleStatus		205	205	205	204	215	215
GetScheduleValue		175	175	175	174	181	181
GetScheduleNext		147	147	147	146	138	138
SetScheduleNext		168	168	168	167	159	159
GetArrivalpointDelay		66	66	66	66	68	68
SetArrivalpointDelay		55	55	55	55	57	57
GetArrivalpointTasksetRef		97	97	97	96	98	98
GetArrivalpointNext		122	122	122	121	127	127
SetArrivalpointNext		124	124	124	123	128	128
TestArrivalpointWritable		71	71	71	71	79	79
GetExecutionTime		206	206	206	179	205	205
GetLargestExecutionTime		92	92	92	101	92	92
ResetLargestExecutionTime		85	85	85	95	85	85
GetStackOffset		59	63	63	63	59	59

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	756	876	1060	766	787	1010
	NS	853	977	1159	873	872	1090
	KL	689	800	986	691	710	922
TerminateTask	LExt	754	800	807	774	770	782
	H	846	871	878	867	840	853
ChainTask	SWL	1674	1853	2135	1826	1846	2185
	SWH	1759	1949	2240	1907	1923	2212
	NSL	1790	2006	2281	1942	1982	2279
	NSH	1863	2038	2338	2045	2015	2337
Schedule	SW	266	266	286	269	263	283
GetTaskID		176	176	180	184	180	184
GetTaskState		786	753	746	779	746	746
EnableAllInterrupts		46	46	46	40	43	43
DisableAllInterrupts		53	53	53	59	56	56
ResumeAllInterrupts		57	57	57	62	60	60
SuspendAllInterrupts		64	64	64	68	61	61
ResumeOSInterrupts		57	57	57	62	60	60
SuspendOSInterrupts		64	64	64	68	61	61
GetResource	Task	1576	1558	836	1840	1687	965
	Combined	760	742	742	942	865	865
	CLEx	845	799	799	949	950	950
ReleaseResource	Task	753	735	735	852	864	864
	Combined	748	730	730	836	859	859
	CLEx	722	704	704	860	833	833
SetEvent	SW	n/a	n/a	n/a	861	814	829
	NS	n/a	n/a	n/a	909	858	878
	KL	n/a	n/a	n/a	804	750	764
ClearEvent		n/a	n/a	n/a	165	186	165
GetEvent		n/a	n/a	n/a	679	652	647
WaitEvent	<default>	n/a	n/a	n/a	1357	1417	1493
	fp	n/a	n/a	n/a	1385	1453	1521
GetAlarmBase		596	574	574	588	566	564
GetAlarm		641	610	610	649	618	625
SetRelAlarm		735	714	714	727	706	703
SetAbsAlarm		707	685	685	717	695	693
CancelAlarm		591	569	569	585	563	561
InitCounter		551	530	526	540	523	519
GetCounterValue		546	525	526	539	517	518

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
osek_tick_alarm	<default>	241	241	241	253	253	253
	KL	144	141	141	132	129	132
osek_incr_counter		36	36	36	47	47	47
GetActiveApplicationMode		8	8	8	3	3	3
StartOS		909	901	901	907	899	899
ShutdownOS	NoHook	n/a	n/a	n/a	n/a	n/a	n/a
	Hook	115	114	114	112	111	111
InitCOM		26	27	27	24	23	23
CloseCOM		27	28	28	31	30	30
StartCOM		104	104	103	353	351	405
StopCOM		48	47	47	51	50	50
ReadFlag		n/a	n/a	n/a	50	52	52
ResetFlag		n/a	n/a	n/a	48	52	52
ReceiveMessage		451	437	437	887	923	923
GetMessageResource		n/a	n/a	n/a	1347	1257	1246
ReleaseMessageResource		n/a	n/a	n/a	1226	1251	1233
GetMessageStatus		n/a	n/a	n/a	396	384	381
SendMessage	SW	1262	1359	1543	1695	1671	1924
	NS	1355	1472	1654	1810	1764	2012
	KL	1126	1245	1431	1577	1530	1793
ActivateTaskset	SW	828	1291	1577	820	1208	1610
	NS	914	1333	1740	899	1286	1664
	KL	760	1190	1582	745	1117	1510
	SW2	828	1291	1577	820	1208	1610
	NS2	914	1333	1740	899	1286	1664
	KL2	760	1190	1582	745	1117	1510
ChainTaskset	SWL	1834	2335	2861	1952	2340	2924
	SWH	1932	2446	3006	2051	2392	3011
	NSL	1904	2456	2818	2031	2387	2934
	NSH	1988	2533	3043	2116	2526	2996
GetTasksetRef		741	708	710	733	700	700
MergeTaskset		388	388	386	397	397	397
AssignTaskset		268	268	264	264	264	264
RemoveTaskset		374	374	377	378	378	378
TestSubTaskset		394	394	390	388	388	388
TestEquivalentTaskset		386	386	382	392	392	392
TickSchedule	SW	662	1790	2182	1298	1821	2195
	NS	755	1879	2271	1374	1910	2284
	KL	559	1708	2100	1223	1739	2113
	SW2	662	1790	2182	1298	1719	2112
	NS2	755	1879	2271	1374	1808	2201
	KL2	559	1708	2100	1223	1637	2030

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
AdvanceSchedule	SW	545	1650	2042	1188	1683	2036
	NS	602	1721	2113	1312	1754	2160
	KL	453	1590	1982	1115	1623	1963
	SW2	545	1650	2042	1188	1581	1953
	NS2	602	1721	2113	1312	1652	2077
	KL2	453	1590	1982	1115	1521	1880
StartSchedule		416	416	416	410	409	409
StopSchedule		305	305	305	288	299	299
GetScheduleStatus		313	313	313	309	305	305
GetScheduleValue		289	289	289	274	281	281
GetScheduleNext		255	255	255	244	247	247
SetScheduleNext		404	404	404	407	393	393
GetArrivalpointDelay		164	163	163	156	155	155
SetArrivalpointDelay		194	190	190	189	185	185
GetArrivalpointTasksetRef		224	218	218	229	223	223
GetArrivalpointNext		241	246	246	236	241	241
SetArrivalpointNext		441	444	444	450	453	453
TestArrivalpointWritable		152	160	160	147	155	155
GetExecutionTime		235	264	264	231	260	260
GetLargestExecutionTime		613	590	590	610	587	587
ResetLargestExecutionTime		605	581	581	602	578	578
GetStackOffset		59	63	63	67	63	67

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	38	38	38	41	41	41
	Cat 2	106	106	111	115	115	110

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	41	41	41	38	38	38
	Cat 2	352	352	384	386	386	354

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	41	41	41	38	38	38
	Cat 2	384	384	352	349	349	381

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()`).

SSX5 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 SSX5 switching contexts measured.

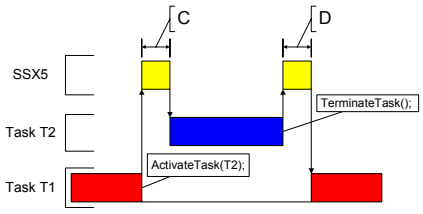


Figure 1: Task Activates a Higher Priority Task which Terminates Normally

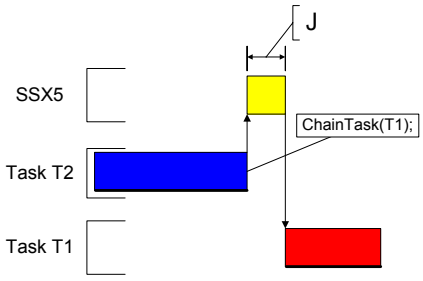


Figure 2: Task Chaining

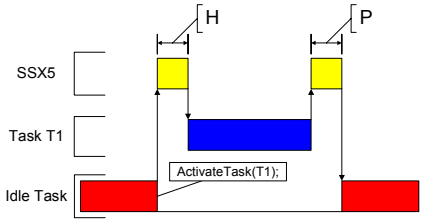


Figure 3: Task Activation from Idle Task

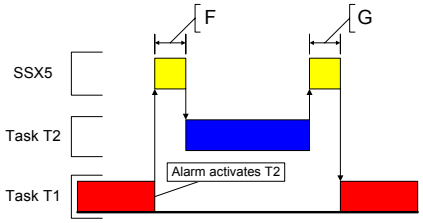


Figure 4: Task Activation from an Alarm

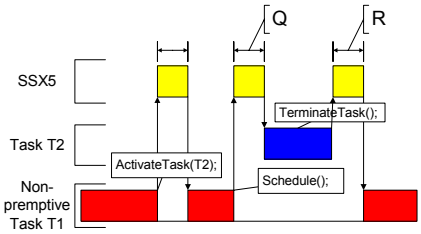


Figure 5: Non-Preemptive Task Calls Schedule()

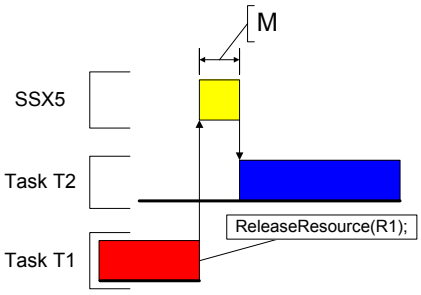


Figure 6: Blocked Task Activated by ReleaseResource()

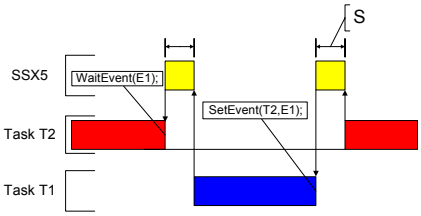


Figure 7: Waiting Task Activated by SetEvent()

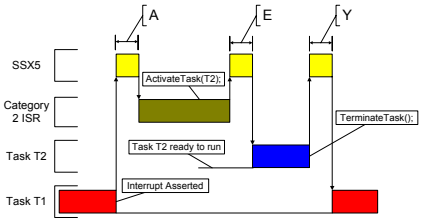


Figure 8: Category 2 ISR Activates a Higher Priority Task

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	150	270	427	148	269	426
Figure 1: D	Heavy, Basic/Extended	308	436	534	407	393	525
ChainTask	Light, Basic	382	565	879	372	563	907
Figure 2: J	Heavy, Basic/Extended	838	1117	1541	936	1074	1550
Pre-emption	Light, Basic	305	489	845	314	490	840
Figure 1: C	Heavy, Basic/Extended	479	617	976	602	629	1002
From idle task	Light, Basic	299	483	839	314	490	840
Figure 3: H	Heavy, Basic/Extended	473	611	970	602	629	1002
Triggered by alarm	Light, Basic	512	691	1055	530	701	1043
Figure 4: F	Heavy, Basic/Extended	694	827	1178	810	832	1213
Schedule	Light, Basic	282	326	483	283	312	485
Figure 5: Q	Heavy, Basic/Extended	456	454	614	571	529	722
Release resource	Light, Basic	343	387	528	344	369	510
Figure 6: M	Heavy, Basic/Extended	517	515	659	632	586	747
SetEvent							
Figure 7: S	Heavy, Extended	n/a	n/a	n/a	983	1001	1388
From category 2 ISR	Light, Basic	236	280	420	233	277	419
Figure 8: E	Heavy, Basic/Extended	410	408	551	521	494	656

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	546	667	755	521	692	780
Figure 1: D	Heavy, Basic/Extended	684	766	917	738	774	886
ChainTask	Light, Basic	868	1058	1361	861	1056	1389
Figure 2: J	Heavy, Basic/Extended	1699	1938	2406	1735	1928	2375

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations	Task Attributes	No	Yes		No	Yes	
Pre-emption	Light, Basic	608	798	1143	641	771	1110
Figure 1: C	Heavy, Basic/Extended	749	925	1265	883	932	1257
From idle task	Light, Basic	614	804	1149	641	771	1110
Figure 3: H	Heavy, Basic/Extended	755	931	1271	883	932	1257
Triggered by alarm	Light, Basic	825	1010	1363	865	990	1321
Figure 4: F	Heavy, Basic/Extended	974	1145	1477	1099	1143	1476
Schedule	Light, Basic	579	629	791	604	617	763
Figure 5: Q	Heavy, Basic/Extended	720	756	913	846	856	985
Release resource	Light, Basic	655	705	835	680	659	789
Figure 6: M	Heavy, Basic/Extended	796	832	957	922	898	1011
SetEvent							
Figure 7: S	Heavy, Extended	n/a	n/a	n/a	1234	1238	1665
From category 2 ISR	Light, Basic	935	985	1114	935	985	1116
Figure 8: E	Heavy, Basic/Extended	1076	1112	1236	1177	1224	1338

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	740	906	993	758	872	964
Figure 1: D	Heavy, Basic/Extended	856	960	1110	953	924	1041
ChainTask	Light, Basic	1620	1818	2128	1671	1764	2163
Figure 2: J	Heavy, Basic/Extended	2629	2946	3404	2771	2855	3282
Pre-emption	Light, Basic	1252	1427	1756	1286	1383	1786
Figure 1: C	Heavy, Basic/Extended	1394	1556	1879	1529	1546	1914
From idle task	Light, Basic	1258	1433	1762	1280	1377	1780
Figure 3: H	Heavy, Basic/Extended	1400	1562	1885	1523	1540	1908
Triggered by alarm	Light, Basic	1520	1694	2031	1545	1641	2053
Figure 4: F	Heavy, Basic/Extended	1670	1831	2146	1796	1812	2173

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations	Task Attributes	No	Yes		No	Yes	
Schedule	Light, Basic	677	727	879	686	693	871
Figure 5: Q	Heavy, Basic/Extended	819	856	1002	929	934	1092
Release resource	Light, Basic	1281	1312	1442	1403	1419	1575
Figure 6: M	Heavy, Basic/Extended	1423	1441	1565	1646	1660	1796
SetEvent							
Figure 7: S	Heavy, Extended	n/a	n/a	n/a	1930	1859	2284
From category 2 ISR	Light, Basic	961	1066	1197	1004	1007	1162
Figure 8: E	Heavy, Basic/Extended	1103	1195	1320	1247	1248	1383

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. RTArchitect 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		45	45	51	45	45	51
BCC1 lightweight, floating-point		57	57	63	57	57	63
BCC1 heavyweight, integer		75	75	81	75	75	81
BCC1 heavyweight, floating-point		75	75	81	75	75	81
BCC2 lightweight, integer		n/a	57	67	n/a	57	67
BCC2 lightweight, floating-point		n/a	57	67	n/a	57	67
BCC2 heavyweight, integer		n/a	75	87	n/a	75	87
BCC2 heavyweight, floating-point		n/a	75	87	n/a	75	87
ECC1 heavyweight, integer		n/a	n/a	n/a	86	86	92

Configuration	Application Uses					
Events	No			Yes		
Shared Task Priorities	No		Yes	No		Yes
Multiple Task Activations	No	Yes		No	Yes	
ECC1 heavyweight, floating-point	n/a	n/a	n/a	86	86	92
ECC2 heavyweight, integer	n/a	n/a	n/a	n/a	n/a	98
ECC2 heavyweight, floating-point	n/a	n/a	n/a	n/a	n/a	98
Pre- and/or Post-Task hooks used						
Task type						
BCC1 lightweight, integer	51	51	51	51	51	51
BCC1 lightweight, floating-point	63	63	63	63	63	63
BCC1 heavyweight, integer	81	81	81	81	81	81
BCC1 heavyweight, floating-point	81	81	81	81	81	81
BCC2 lightweight, integer	n/a	63	67	n/a	63	67
BCC2 lightweight, floating-point	n/a	63	67	n/a	63	67
BCC2 heavyweight, integer	n/a	81	87	n/a	81	87
BCC2 heavyweight, floating-point	n/a	81	87	n/a	81	87
ECC1 heavyweight, integer	n/a	n/a	n/a	92	92	92
ECC1 heavyweight, floating-point	n/a	n/a	n/a	92	92	92
ECC2 heavyweight, integer	n/a	n/a	n/a	n/a	n/a	98
ECC2 heavyweight, floating-point	n/a	n/a	n/a	n/a	n/a	98

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	65	65	71	65	65	71
BCC1 lightweight, floating-point	71	71	77	71	71	77
BCC1 heavyweight, integer	93	93	99	93	93	99
BCC1 heavyweight, floating-point	93	93	99	93	93	99
BCC2 lightweight, integer	n/a	77	87	n/a	77	87
BCC2 lightweight, floating-point	n/a	77	87	n/a	77	87
BCC2 heavyweight, integer	n/a	95	107	n/a	95	107
BCC2 heavyweight, floating-point	n/a	95	107	n/a	95	107
ECC1 heavyweight, integer	n/a	n/a	n/a	106	106	112
ECC1 heavyweight, floating-point	n/a	n/a	n/a	106	106	112
ECC2 heavyweight, integer	n/a	n/a	n/a	n/a	n/a	118
ECC2 heavyweight, floating-point	n/a	n/a	n/a	n/a	n/a	118

Configuration / Events / Shared Task Priorities / Multiple Task Activations		Application Uses					
		No			Yes		
		No		Yes	No		Yes
		No	Yes		No	Yes	
Pre- and/or Post-Task hooks used							
Task type							
BCC1 lightweight, integer		71	71	71	71	71	71
BCC1 lightweight, floating-point		77	77	77	77	77	77
BCC1 heavyweight, integer		99	99	99	99	99	99
BCC1 heavyweight, floating-point		99	99	99	99	99	99
BCC2 lightweight, integer		n/a	83	87	n/a	83	87
BCC2 lightweight, floating-point		n/a	83	87	n/a	83	87
BCC2 heavyweight, integer		n/a	101	107	n/a	101	107
BCC2 heavyweight, floating-point		n/a	101	107	n/a	101	107
ECC1 heavyweight, integer		n/a	n/a	n/a	112	112	112
ECC1 heavyweight, floating-point		n/a	n/a	n/a	112	112	112
ECC2 heavyweight, integer		n/a	n/a	n/a	n/a	n/a	118
ECC2 heavyweight, floating-point		n/a	n/a	n/a	n/a	n/a	118

Extended

Configuration / Events / Shared Task Priorities / Multiple Task Activations		Application Uses					
		No			Yes		
		No		Yes	No		Yes
		No	Yes		No	Yes	
Pre- and Post-Task hooks not used							
Task type							
BCC1 lightweight, integer		65	65	71	65	65	71
BCC1 lightweight, floating-point		71	71	77	71	71	77
BCC1 heavyweight, integer		93	93	99	93	93	99
BCC1 heavyweight, floating-point		93	93	99	93	93	99
BCC2 lightweight, integer		n/a	77	87	n/a	77	87
BCC2 lightweight, floating-point		n/a	77	87	n/a	77	87
BCC2 heavyweight, integer		n/a	95	107	n/a	95	107
BCC2 heavyweight, floating-point		n/a	95	107	n/a	95	107
ECC1 heavyweight, integer		n/a	n/a	n/a	106	106	112
ECC1 heavyweight, floating-point		n/a	n/a	n/a	106	106	112
ECC2 heavyweight, integer		n/a	n/a	n/a	n/a	n/a	118
ECC2 heavyweight, floating-point		n/a	n/a	n/a	n/a	n/a	118
Pre- and/or Post-Task hooks used							
Task type							
BCC1 lightweight, integer		71	71	71	71	71	71
BCC1 lightweight, floating-point		77	77	77	77	77	77

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
BCC1 heavyweight, integer		99	99	99	99	99	99
BCC1 heavyweight, floating-point		99	99	99	99	99	99
BCC2 lightweight, integer		n/a	83	87	n/a	83	87
BCC2 lightweight, floating-point		n/a	83	87	n/a	83	87
BCC2 heavyweight, integer		n/a	101	107	n/a	101	107
BCC2 heavyweight, floating-point		n/a	101	107	n/a	101	107
ECC1 heavyweight, integer		n/a	n/a	n/a	112	112	112
ECC1 heavyweight, floating-point		n/a	n/a	n/a	112	112	112
ECC2 heavyweight, integer		n/a	n/a	n/a	n/a	n/a	118
ECC2 heavyweight, floating-point		n/a	n/a	n/a	n/a	n/a	118

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