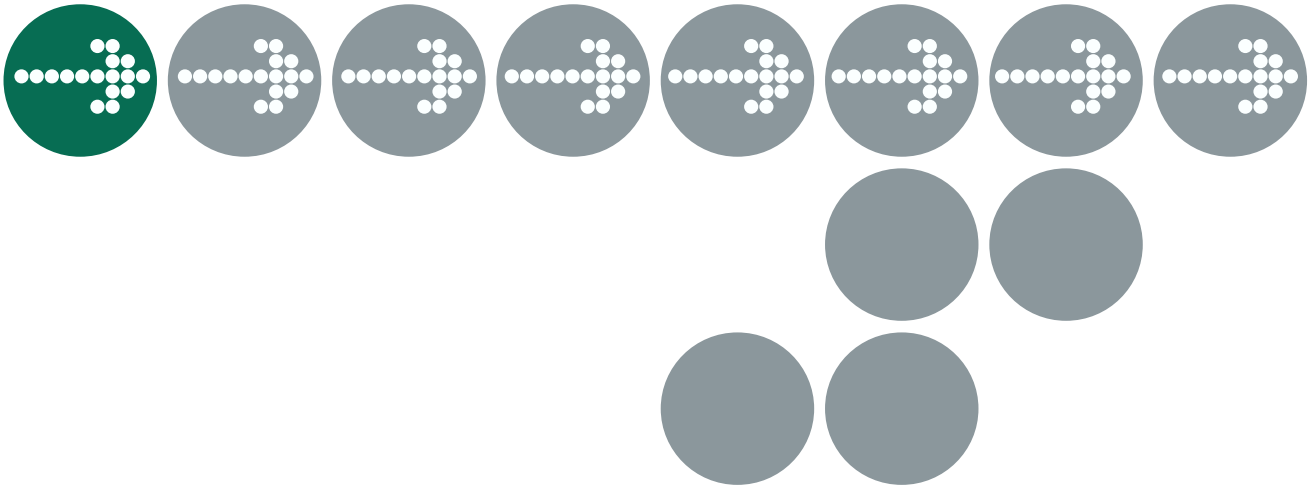
LiveDevices

Binding Manual: Blackfin®/ADI



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 États-Unis
SILIC 307
94588 Rungis Cedex

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

www.etas.fr

Korea

ETAS Korea Co. Ltd.
3F, Samseung Bldg. 61-1
Yangjae-dong, Seocho-gu
Seoul

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

www.etas.co.kr

Great Britain

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

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

www.etas-uk.net

Copyright Notice

Version: RM00059-002

Disclaimer

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

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

Trademarks

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 Blackfin®/ADI 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.

The Blackfin/ADI port of RTA currently supports the following Blackfin variants: ADSP-BF531, ADSP-BF532, ADSP-BF533 and ADSP-DM102

2.1 Compiler

SSX5 was built using the following compiler:

Vendor	Analog Devices
Compiler	VisualDSP++™ v3.1 C/C++ Compiler Blackfin
Version	6.3.0

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

Option	Description
`-c`	Do not link
`-proc ADSP-xxxxx`	Generate code for Blackfin variant ADSP-xxxxx, where xxxxx is one of the following list: BF531, BF532, BF533 or DM102

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

Option	Description
`-g`	Add debug information

2.1.1 CPU_TYPE environment variable

To support the range of Blackfin CPU types when building an application within the RTA environment a DOS environment variable `CPU_TYPE` is used. `CPU_TYPE` should be set to be equal to a CPU type from the following list: ADSP-BF531, ADSP-BF532, ADSP-BF533 and ADSP-DM102. The `-proc` command line option utilizes the `CPU_TYPE` variable when compiling and assembling source files; this is demonstrated in the SSX5 example application.

2.2 Assembler

SSX5 was built using the following assembler:

Vendor	Analog Devices
Assembler	VisualDSP++ v3.1 Blackfin Assembler
Version	2.2.4.1

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

Option	Description
`-proc ADSP-xxxxx`	Generate code for Blackfin variant ADSP-xxxxx, where xxxxx is one of the following list: BF531, BF532, BF533 or DM102

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

2.3 Linker/Locator

The compulsory linker/locator options for an SSX5 application are shown in the following table:

Option	Description
`-proc ADSP-xxxxx`	Link application for Blackfin variant ADSP-xxxxx, where xxxxx is one of the following list: BF531, BF532, BF533 or DM102

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
`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`	VisualDSP++ v3.1 library `setjmp` routine
`longjmp`	VisualDSP++ v3.1 library `longjmp` routine
___memcpy	VisualDSP++ v3.1 library `memcpy` routine

2.4 Debugger

Information about ORTI for RTA can be found in the *RTA ORTI Guide*

At the time of writing, we were not aware of any debuggers for the Analog Devices Blackfin with support for ORTI.

If you are using an ORTI version 2.0 aware debugger on this platform you can use the “Unknown ORTI debugger” option in RTArchitect to generate an ORTI output file. The ORTI generated will not have been tested on the debugger and, therefore, is not guaranteed to work.

Please contact LiveDevices if you have any questions about ORTI support in RTA.

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

In SSX5 interrupts are allocated an Interrupt Priority Level (IPL). This is a processor independent abstraction of the interrupt priorities that are available on the target hardware. You can find out more about IPLs in the *RTA User Guide*. The hardware interrupt controller is explained in the *The Analog Blackfin Processor Hardware reference manual relating to the specifc chip variant*.

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

IPL Value	`IMASK` Binary	Description
0	11xx xxxx xxx1 1111	User level
1	110x xxxx xxx1 1111	General Interrupt 13
2	1100 xxxx xxx1 1111	General Interrupt 12
3	1100 0xxx xxx1 1111	General Interrupt 11
4	1100 00xx xxx1 1111	General Interrupt 10
5	1100 000x xxx1 1111	General Interrupt 9
6	1100 0000 xxx1 1111	General Interrupt 8
7	1100 0000 0xx1 1111	General Interrupt 7
8	1100 0000 00x1 1111	Core Timer
9	1100 0000 0001 1111	Hardware Error
10	1100 0000 0001 1111	Exception
11	1100 0000 0001 1111	Non-Maskable interrupt
Notes:	x is either 0 or 1 depending on whether the level is used by the application or not respectively SSX5 does not make use of the Blackfin core user mode and IPL 0 does not correspond to this. IPL 0 is the lowest priority (`IVG15`) in the Blackfin core supervisor mode. `IVG14` is reserved for use by SSX5 The lower 5 bits of `IMASK` are always set to 1 by the hardware	

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 Offset	Legality
0 and 1	Reset vector is outside of the scope of SSX5
2 and 3	Category 1 interrupts only
4	CPU Reserved
5 to 13	Category 1 and 2 interrupts
14 and 15	Reserved by SSX5

The valid base addresses for the vector table are:

Base Address	Notes
0xFFE02000	The entries to the Event Vector Table are entered as core Memory Mapped Registers (MMRs) `EVT0` to `EVT15`, which are initialised in `StartOS`. There is no scope for changing the base address

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 Analog Devices C compiler can generate appropriate interrupt handling code for a C function decorated with the `EX_REENTRANT_HANDLER()` 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 vector table is generated within `osgen.asm`.

Note that a generated vector table omits the reset vector entry. If you choose to provide your own vector table, it must contain an entry for each interrupt handler, including the Category 2 interrupt handlers in SSX5.

The following table shows the syntax for labels attached to SSX5 Category 2 interrupt handlers (VV represents the vector offset).

Vector Offset	Label
5 to 13	os_ist_VV
eg :	e.g. os_ist_11

3.1.6 Operating Mode

The processor is exclusively run in supervisor mode and the user should never enter user mode. Manipulation of `IMASK` is performed by SSX5 and users should not attempt to access it directly.

3.1.7 Manual Vector Table Generation

The Blackfin's vector table is a set of registers that form part of a core peripheral. Therefore, the vectors cannot be initialized in the conventional way of placing values in ROM. Instead, vector addresses are located in MMRs `EVT0` to `EVT15`. When manual vector table generation is enabled, SSX5 calls a function, with prototype `void os_min_vec_init(void)`, which must be implemented by the user. This function is responsible for ensuring vector registers are initialized, including the reserved vector `EVT14` that must be initialized to `os_IST_exit`. The example code below initializes the vector registers for an application with one Category 1 interrupt at vector `EVT8` and one Category 2 interrupt at `EVT11`.

```
#include <stddef.h>
#include <ccblkfn.h>

typedef void (*os_vector)(void);

extern void os_IST_exit(void);
extern void os_ist_11(void);
extern void category_1_ist(void);

void os_min_vec_init(void)
{
   *((os_vector *)EVT5) = NULL;
   *((os_vector *)EVT6) = NULL;
   *((os_vector *)EVT7) = NULL;
   *((os_vector *)EVT8) = category_1_ist;
   *((os_vector *)EVT9) = NULL;
   *((os_vector *)EVT10) = NULL;
   *((os_vector *)EVT11) = os_ist_11;
   *((os_vector *)EVT12) = NULL;
   *((os_vector *)EVT13) = NULL;
   *((os_vector *)EVT14) = os_IST_exit;
}
```

3.2 Register Settings

SSX5 does not require the initialization of registers before calling `StartOS()`.

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

Register	Notes
IMASK	Used to set the OS prirority level
EVT14	reserved for use by SSX5
EVT15	SSX5 does not use the Blackfin Core user mode
RETI	SSX5 uses the RETI register in its API calls. All Category 1 interurpt handlers should be reentrant i.e. use the EX_REENTRANT_HANDLER function modifier not the EX_INTERRUPT_HANDLER function modifier

3.3 Stack Usage

3.3.1 Number of Stacks

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

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

3.3.2 Stack Usage within API Calls

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

Standard

API max usage (bytes): 68

Timing

API max usage (bytes): 68

Extended

API max usage (bytes): 120

To determine the correct stack usage for tasks that use other library code, you may need to contact the vendor to find out more about library call stack usage.

3.3.3 Stack section

As SSX5 runs the processor exclusively in supervisor mode only the supervisor stack (sysstack) is used in applications. To support ECC tasks in SSX5 the definition of the sysstack section in the linker command file must declare a label at the top of stack called _ldf_sysstack_end. A suggested

method of this is shown in **Code Example 3:2**, which can also be found in the example application linker command file `blackfin.ldf`.

```
sysstack
{
   ldf_sysstack_space = .;
   ldf_sysstack_end
         = ldf_sysstack_space
           + MEMORY_SIZEOF(MEM_SYSSTACK);
   _ldf_sysstack_end
         = ldf_sysstack_space
           + MEMORY_SIZEOF(MEM_SYSSTACK);
} >MEM_SYSSTACK
```

Code Example 3:2 – Marking the top of the supervisor stack

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 six 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	**ADSP-BF532**
Clock speed (MHz)	196.608
Code memory	Internal SRAM
Read-only data memory	Internal SRAM
Read-write data memory	Internal SRAM

It should be noted that the timing of code execution depends upon a number of factors such as the alignment of code and data objects, stack alignment when an interrupt fires, and the relative location of data objects between two consecutive data accesses. As execution times therefore depend upon configuration the timing information in this section should be used solely as a guideline.

4.1 Functionality

The OSEK Operating System Specification specifies four conformance classes. These attributes apply to *systems* built with OSEK OS objects. The following table specifies the number of OSEK OS and COM objects supported per conformance class.

Configuration	**Application Uses**					
Events	**No**			**Yes**		
Shared Task Priorities	**No**		**Yes**	**No**		**Yes**
Multiple Task Activations	**No**	**Yes**		**No**	**Yes**	
Maximum number of tasks	32	32	32	32	32	32
Maximum number of not suspended tasks	32	32	32	32	32	32
Maximum number of priorities	32	32	32	32	32	32
Number of tasks per priority (for BCC2 and ECC2)	n/a	32	32	n/a	32	32
Upper limit for number of basic task activations per task priority	1	255	255	1	255	255
Maximum number of events per task	0	0	0	32	32	32
Limits for the number of alarm objects (per system / per task)	not limited by SSX5					
Limits for the number of standard resources (per system)	255	255	255	255	255	255

Configuration	Application Uses					
Events	No			Yes		
Shared Task Priorities	No		Yes	No		Yes
Multiple Task Activations	No	Yes		No	Yes	
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)	4294967295					

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		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
OS overhead	RAM	28	28	28	28	28	28
	ROM	166	166	166	166	166	166
COM overhead	RAM	8	8	8	8	8	8
	ROM	16	16	16	16	16	16

Timing

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
OS overhead	RAM	48	48	48	48	48	48
	ROM	238	238	238	238	238	238
COM overhead	RAM	8	8	8	8	8	8
	ROM	16	16	16	16	16	16

Extended

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
OS overhead	RAM	66	66	66	66	66	66
	ROM	284	284	284	284	284	284
COM overhead	RAM	8	8	8	8	8	8
	ROM	16	16	16	16	16	16

4.2.2 ROM and RAM for OSEK OS Objects

In addition to the base OS overhead, detailed in Section 4.2.1, each OSEK OS object requires ROM and/or RAM. 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	36	36	36	36	36	36
BCC1 Heavyweight task	RAM	4	4	4	4	4	4
	ROM	40	40	40	40	40	40
BCC2 task	RAM	n/a	8	10	n/a	8	10
	ROM	n/a	44	52	n/a	44	52

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
ECC1, Integer task	RAM	n/a	n/a	n/a	176	176	176
	ROM	n/a	n/a	n/a	60	60	60
ECC1, floating-point task	RAM	n/a	n/a	n/a	178	178	178
	ROM	n/a	n/a	n/a	60	60	60
ECC2, Integer task	RAM	n/a	n/a	n/a	n/a	n/a	178
	ROM	n/a	n/a	n/a	n/a	n/a	68
ECC2, floating-point task	RAM	n/a	n/a	n/a	n/a	n/a	180
	ROM	n/a	n/a	n/a	n/a	n/a	68
Category 2 ISR	RAM	0	0	0	0	0	0
	ROM	76	76	76	76	76	76
Category 2 ISR, floating-point	RAM	1	1	1	1	1	1
	ROM	98	98	98	98	98	98
Resource	RAM	0	0	0	0	0	0
	ROM	20	20	20	20	20	20
Internal resource	RAM	0	0	0	0	0	0
	ROM	0	0	0	0	0	0
Linked resource	RAM	0	0	0	0	0	0
	ROM	20	20	20	20	20	20
Alarm	RAM	12	12	12	12	12	12
	ROM	42	42	42	42	42	42
Counter	RAM	4	4	4	4	4	4
	ROM	48	48	48	48	48	48
Message	RAM	11	11	11	51	51	51
	ROM	20	20	20	56	56	56
Flag	RAM	4	4	4	4	4	4
	ROM	1	1	1	1	1	1
Message resource	RAM	0	0	0	0	0	0
	ROM	20	20	20	20	20	20
Event	RAM	0	0	0	0	0	0
	ROM	4	4	4	4	4	4
Priority level	RAM	0	0	6	0	6	6
	ROM	0	0	12	0	12	12
Arrivalpoint (readonly)	RAM	0	0	0	0	0	0
	ROM	12	12	12	12	12	12
Arrivalpoint (writable)	RAM	12	12	12	12	12	12
	ROM	12	12	12	12	12	12
Schedule	RAM	16	16	16	16	16	16
	ROM	36	36	36	36	36	36
Taskset (readonly)	RAM	0	0	0	0	0	0
	ROM	4	4	4	4	4	4

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
Taskset (writable)	RAM	4	4	4	4	4	4
	ROM	4	4	4	4	4	4

Timing

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
BCC1 Lightweight task	RAM	12	12	12	12	12	12
	ROM	48	48	48	48	48	48
BCC1 Heavyweight task	RAM	16	16	16	16	16	16
	ROM	52	52	52	52	52	52
BCC2 task	RAM	n/a	20	22	n/a	20	22
	ROM	n/a	56	64	n/a	56	64
ECC1, Integer task	RAM	n/a	n/a	n/a	188	188	188
	ROM	n/a	n/a	n/a	72	72	72
ECC1, floating-point task	RAM	n/a	n/a	n/a	190	190	190
	ROM	n/a	n/a	n/a	72	72	72
ECC2, Integer task	RAM	n/a	n/a	n/a	n/a	n/a	190
	ROM	n/a	n/a	n/a	n/a	n/a	80
ECC2, floating-point task	RAM	n/a	n/a	n/a	n/a	n/a	192
	ROM	n/a	n/a	n/a	n/a	n/a	80
Category 2 ISR	RAM	12	12	12	12	12	12
	ROM	148	148	148	148	148	148
Category 2 ISR, floating-point	RAM	14	14	14	14	14	14
	ROM	156	156	156	156	156	156
Resource	RAM	0	0	0	0	0	0
	ROM	20	20	20	20	20	20
Internal resource	RAM	0	0	0	0	0	0
	ROM	0	0	0	0	0	0
Linked resource	RAM	0	0	0	0	0	0
	ROM	20	20	20	20	20	20
Alarm	RAM	12	12	12	12	12	12
	ROM	42	42	42	42	42	42
Counter	RAM	4	4	4	4	4	4
	ROM	48	48	48	48	48	48
Message	RAM	11	11	11	51	51	51
	ROM	20	20	20	56	56	56

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
Flag	RAM	4	4	4	4	4	4
	ROM	1	1	1	1	1	1
Message resource	RAM	0	0	0	0	0	0
	ROM	20	20	20	20	20	20
Event	RAM	0	0	0	0	0	0
	ROM	4	4	4	4	4	4
Priority level	RAM	0	0	6	0	6	6
	ROM	0	0	12	0	12	12
Arrivalpoint (readonly)	RAM	0	0	0	0	0	0
	ROM	12	12	12	12	12	12
Arrivalpoint (writable)	RAM	12	12	12	12	12	12
	ROM	12	12	12	12	12	12
Schedule	RAM	16	16	16	16	16	16
	ROM	36	36	36	36	36	36
Taskset (readonly)	RAM	0	0	0	0	0	0
	ROM	4	4	4	4	4	4
Taskset (writable)	RAM	4	4	4	4	4	4
	ROM	4	4	4	4	4	4

Extended

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
BCC1 Lightweight task	RAM	16	16	16	16	16	16
	ROM	60	60	60	60	60	60
BCC1 Heavyweight task	RAM	20	20	20	20	20	20
	ROM	60	60	60	60	60	60
BCC2 task	RAM	n/a	24	26	n/a	24	26
	ROM	n/a	64	72	n/a	64	72
ECC1, Integer task	RAM	n/a	n/a	n/a	192	192	192
	ROM	n/a	n/a	n/a	80	80	80
ECC1, floating-point task	RAM	n/a	n/a	n/a	194	194	194
	ROM	n/a	n/a	n/a	80	80	80
ECC2, Integer task	RAM	n/a	n/a	n/a	n/a	n/a	194
	ROM	n/a	n/a	n/a	n/a	n/a	88
ECC2, floating-point task	RAM	n/a	n/a	n/a	n/a	n/a	196
	ROM	n/a	n/a	n/a	n/a	n/a	88

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
Category 2 ISR	RAM	16	16	16	16	16	16
	ROM	160	160	160	160	160	160
Category 2 ISR, floating-point	RAM	18	18	18	18	18	18
	ROM	168	168	168	168	168	168
Resource	RAM	8	8	8	8	8	8
	ROM	28	28	28	28	28	28
Internal resource	RAM	0	0	0	0	0	0
	ROM	0	0	0	0	0	0
Linked resource	RAM	8	8	8	8	8	8
	ROM	28	28	28	28	28	28
Alarm	RAM	12	12	12	12	12	12
	ROM	46	46	46	46	46	46
Counter	RAM	4	4	4	4	4	4
	ROM	52	52	52	52	52	52
Message	RAM	11	11	11	51	51	51
	ROM	24	24	24	60	60	60
Flag	RAM	4	4	4	4	4	4
	ROM	1	1	1	1	1	1
Message resource	RAM	8	8	8	8	8	8
	ROM	28	28	28	28	28	28
Event	RAM	0	0	0	0	0	0
	ROM	4	4	4	4	4	4
Priority level	RAM	0	0	6	0	6	6
	ROM	0	0	12	0	12	12
Arrivalpoint (readonly)	RAM	0	0	0	0	0	0
	ROM	20	20	20	20	20	20
Arrivalpoint (writable)	RAM	20	20	20	20	20	20
	ROM	20	20	20	20	20	20
Schedule	RAM	20	20	20	20	20	20
	ROM	44	44	44	44	44	44
Taskset (readonly)	RAM	0	0	0	0	0	0
	ROM	4	4	4	4	4	4
Taskset (writable)	RAM	4	4	4	4	4	4
	ROM	4	4	4	4	4	4

4.2.3 Size of Linkable Modules

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	146	186	226	162	202	288
	NS		114	150	190	126	166	256
	KL	2	76	114	154	90	130	216
TerminateTask	LExt	3	n/a	n/a	n/a	n/a	n/a	n/a
	H	5	30	30	30	30	30	30
ChainTask	SWL	1, 8	126	188	248	138	204	290
	SWH	1, 9	184	224	264	192	240	328
	NSL	8	126	188	248	138	204	290
	NSH	9	168	208	248	176	224	312
Schedule			120	120	130	120	120	130
GetTaskID			42	42	42	42	42	42
GetTaskState			90	90	90	112	112	112
EnableAllInterrupts			18	18	18	18	18	18
DisableAllInterrupts			22	22	22	22	22	22
ResumeAllInterrupts			36	36	36	36	36	36
SuspendAllInterrupts			42	42	42	42	42	42
ResumeOSInterrupts			40	40	40	40	40	40
SuspendOSInterrupts			60	60	60	60	60	60
GetResource	Task	7	26	26	30	26	26	30
	Combined	6	78	78	78	78	78	78
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
ReleaseResource	Task	7	92	92	92	92	92	92
	Combined	6	190	190	190	190	190	190
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	1	n/a	n/a	n/a	144	144	230
	NS		n/a	n/a	n/a	96	96	186
	NS1i	10	n/a	n/a	n/a	52	n/a	n/a
	KL	2	n/a	n/a	n/a	66	66	168
	KL1i	2, 10	n/a	n/a	n/a	26	n/a	n/a
ClearEvent			n/a	n/a	n/a	56	56	56
GetEvent			n/a	n/a	n/a	14	14	14
WaitEvent	<default>		n/a	n/a	n/a	280	280	514
	fp	11	n/a	n/a	n/a	310	310	562
	1i	10	n/a	n/a	n/a	22	n/a	n/a
GetAlarmBase			70	70	70	70	70	70
GetAlarm			102	102	102	102	102	102
SetRelAlarm			122	122	122	122	122	122
SetAbsAlarm			142	142	142	142	142	142
CancelAlarm			92	92	92	92	92	92

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
InitCounter			52	52	52	52	52	52
GetCounterValue			58	58	58	58	58	58
osek_tick_alarm	<default>		74	74	74	74	74	74
	KL	2	40	40	40	40	40	40
osek_incr_counter			28	28	28	28	28	28
GetActiveApplicationMode		30	n/a	n/a	n/a	n/a	n/a	n/a
StartOS			226	226	226	226	226	226
ShutdownOS	NoHook	12	48	48	48	48	48	48
	Hook	13	60	60	60	60	60	60
InitCOM			4	4	4	4	4	4
CloseCOM			4	4	4	4	4	4
StartCOM			30	30	30	30	30	30
StopCOM			28	28	28	28	28	28
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	68	68	68	206	206	206
	CCCB	15	206	206	206	206	206	206
GetMessageResource			54	54	54	54	54	54
ReleaseMessageResource			48	48	48	48	48	48
GetMessageStatus			54	54	54	54	54	54
SendMessage	SW CCCA	1, 14	104	104	104	240	240	240
	SW CCCB	1, 15	228	228	228	240	240	240
	NS CCCA	14	104	104	104	240	240	240
	NS CCCB	15	228	228	228	240	240	240
	KL CCCA	2, 14	74	74	74	212	212	212
	KL CCCB	2, 15	200	200	200	212	212	212
main_dispatch	NoHook	12	154	154	192	154	154	192
	Hook	13	206	206	244	206	206	244
sub_dispatch	B1LF	19	32	32	32	32	32	32
	B1HI	20	112	112	112	112	112	112
	B1HF	21	106	106	106	106	106	106
	B2LI	22	n/a	80	106	n/a	80	106
	B2LF	23	n/a	88	114	n/a	88	114
	B2HI	24	n/a	154	236	n/a	154	236
	B2HF	25	n/a	162	230	n/a	162	230
	E1HI	26	n/a	n/a	n/a	304	304	380
	E1HF	27	n/a	n/a	n/a	312	312	388
	E2HI	28	n/a	n/a	n/a	n/a	n/a	380
	E2HF	29	n/a	n/a	n/a	n/a	n/a	388
ErrorHook support		16	46	46	46	46	46	46

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
	ServiceID	17	58	58	58	58	58	58
	Parameters	18	82	82	82	82	82	82
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	106	160	218	114	192	250
	NS		52	124	178	66	152	196
	KL	2	26	90	148	40	118	180
ChainTaskset	SWL	1, 8	56	122	164	56	140	190
	SWH	1, 9	116	184	226	116	206	244
	NSL	8	56	122	164	56	140	190
	NSH	9	96	168	210	96	190	228
GetTasksetRef			12	12	12	12	12	12
MergeTaskset			46	46	46	46	46	46
AssignTaskset			12	12	12	12	12	12
RemoveTaskset			48	48	48	48	48	48
TestSubTaskset			60	60	60	60	60	60
TestEquivalentTaskset			58	58	58	58	58	58
TickSchedule	SW	1	152	152	152	152	152	152
	NS		114	106	106	106	106	106
	KL	2	80	76	76	76	76	76
AdvanceSchedule	SW	1	164	162	162	162	162	162
	NS		120	116	116	116	116	116
	KL	2	102	90	90	90	90	90
StartSchedule			94	94	94	94	94	94
StopSchedule			76	76	76	76	76	76
GetScheduleStatus			96	96	96	96	96	96
GetScheduleValue			86	86	86	86	86	86
GetScheduleNext			14	14	14	14	14	14
SetScheduleNext			10	10	10	10	10	10
GetArrivalpointDelay			12	12	12	12	12	12
SetArrivalpointDelay			8	8	8	8	8	8
GetArrivalpointTasksetRef			12	12	12	12	12	12
GetArrivalpointNext			12	12	12	12	12	12
SetArrivalpointNext			8	8	8	8	8	8
TestArrivalpointWritable			38	38	38	38	38	38
GetExecutionTime			4	4	4	4	4	4
GetLargestExecutionTime			10	10	10	10	10	10
ResetLargestExecutionTime			4	4	4	4	4	4
GetStackOffset			18	18	18	18	18	18

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	146	186	226	162	202	288
	NS		114	150	190	126	166	256
	KL	2	76	114	154	90	130	216
TerminateTask	LExt	3	n/a	n/a	n/a	n/a	n/a	n/a
	H	5	30	30	30	30	30	30
ChainTask	SWL	1, 8	126	188	248	138	204	290
	SWH	1, 9	184	224	264	192	240	328
	NSL	8	126	188	248	138	204	290
	NSH	9	168	208	248	176	224	312
Schedule			148	148	158	148	148	158
GetTaskID			42	42	42	42	42	42
GetTaskState			90	90	90	112	112	112
EnableAllInterrupts			18	18	18	18	18	18
DisableAllInterrupts			22	22	22	22	22	22
ResumeAllInterrupts			36	36	36	36	36	36
SuspendAllInterrupts			42	42	42	42	42	42
ResumeOSInterrupts			40	40	40	40	40	40
SuspendOSInterrupts			60	60	60	60	60	60
GetResource	Task	7	26	26	30	26	26	30
	Combined	6	78	78	78	78	78	78
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
ReleaseResource	Task	7	120	120	120	120	120	120
	Combined	6	256	256	256	256	256	256
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	1	n/a	n/a	n/a	144	144	230
	NS		n/a	n/a	n/a	96	96	186
	NS1i	10	n/a	n/a	n/a	52	n/a	n/a
	KL	2	n/a	n/a	n/a	66	66	168
	KL1i	2, 10	n/a	n/a	n/a	26	n/a	n/a
ClearEvent			n/a	n/a	n/a	56	56	56
GetEvent			n/a	n/a	n/a	14	14	14
WaitEvent	<default>		n/a	n/a	n/a	366	366	580
	fp	11	n/a	n/a	n/a	394	394	640
	1i	10	n/a	n/a	n/a	122	n/a	n/a
GetAlarmBase			70	70	70	70	70	70
GetAlarm			102	102	102	102	102	102
SetRelAlarm			122	122	122	122	122	122

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
SetAbsAlarm			142	142	142	142	142	142
CancelAlarm			92	92	92	92	92	92
InitCounter			52	52	52	52	52	52
GetCounterValue			58	58	58	58	58	58
osek_tick_alarm	<default>		74	74	74	74	74	74
	KL	2	40	40	40	40	40	40
osek_incr_counter			28	28	28	28	28	28
GetActiveApplicationMode		30	n/a	n/a	n/a	n/a	n/a	n/a
StartOS			288	288	288	288	288	288
ShutdownOS	NoHook	12	48	48	48	48	48	48
	Hook	13	60	60	60	60	60	60
InitCOM			4	4	4	4	4	4
CloseCOM			4	4	4	4	4	4
StartCOM			30	30	30	30	30	30
StopCOM			28	28	28	28	28	28
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	68	68	68	206	206	206
	CCCB	15	206	206	206	206	206	206
GetMessageResource			54	54	54	54	54	54
ReleaseMessageResource			48	48	48	48	48	48
GetMessageStatus			54	54	54	54	54	54
SendMessage	SW CCCA	1, 14	104	104	104	240	240	240
	SW CCCB	1, 15	228	228	228	240	240	240
	NS CCCA	14	104	104	104	240	240	240
	NS CCCB	15	228	228	228	240	240	240
	KL CCCA	2, 14	74	74	74	212	212	212
	KL CCCB	2, 15	200	200	200	212	212	212
main_dispatch	NoHook	12	216	216	252	216	216	252
	Hook	13	270	270	308	270	270	308
sub_dispatch	B1LF	19	22	22	22	22	22	22
	B1HI	20	108	108	108	108	108	108
	B1HF	21	110	110	110	110	110	110
	B2LI	22	n/a	58	84	n/a	58	84
	B2LF	23	n/a	66	92	n/a	66	92
	B2HI	24	n/a	160	212	n/a	160	212
	B2HF	25	n/a	142	206	n/a	142	206
	E1HI	26	n/a	n/a	n/a	376	376	438
	E1HF	27	n/a	n/a	n/a	370	370	446
	E2HI	28	n/a	n/a	n/a	n/a	n/a	438

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
	E2HF	29	n/a	n/a	n/a	n/a	n/a	446
ErrorHook support		16	46	46	46	46	46	46
	ServiceID	17	58	58	58	58	58	58
	Parameters	18	82	82	82	82	82	82
validity_checks		3	n/a	n/a	n/a	n/a	n/a	n/a
Timing_dispatch		4	92	92	92	92	92	92
Timing_termination		4	104	104	104	104	104	104
ActivateTaskset	SW	1	106	160	218	114	192	250
	NS		52	124	178	66	152	196
	KL	2	26	90	148	40	118	180
ChainTaskset	SWL	1, 8	56	122	164	56	140	190
	SWH	1, 9	116	184	226	116	206	244
	NSL	8	56	122	164	56	140	190
	NSH	9	96	168	210	96	190	228
GetTasksetRef			12	12	12	12	12	12
MergeTaskset			46	46	46	46	46	46
AssignTaskset			12	12	12	12	12	12
RemoveTaskset			48	48	48	48	48	48
TestSubTaskset			60	60	60	60	60	60
TestEquivalentTaskset			58	58	58	58	58	58
TickSchedule	SW	1	152	152	152	152	152	152
	NS		114	106	106	106	106	106
	KL	2	80	76	76	76	76	76
AdvanceSchedule	SW	1	164	162	162	162	162	162
	NS		120	116	116	116	116	116
	KL	2	102	90	90	90	90	90
StartSchedule			94	94	94	94	94	94
StopSchedule			76	76	76	76	76	76
GetScheduleStatus			96	96	96	96	96	96
GetScheduleValue			86	86	86	86	86	86
GetScheduleNext			14	14	14	14	14	14
SetScheduleNext			10	10	10	10	10	10
GetArrivalpointDelay			12	12	12	12	12	12
SetArrivalpointDelay			8	8	8	8	8	8
GetArrivalpointTasksetRef			12	12	12	12	12	12
GetArrivalpointNext			12	12	12	12	12	12
SetArrivalpointNext			8	8	8	8	8	8
TestArrivalpointWritable			38	38	38	38	38	38
GetExecutionTime			120	120	120	120	120	120
GetLargestExecutionTime			16	16	16	16	16	16

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
ResetLargestExecutionTime			14	14	14	14	14	14
GetStackOffset			18	18	18	18	18	18

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	260	310	350	278	326	412
	NS		304	366	406	322	382	468
	KL	2	220	266	308	238	282	370
TerminateTask	LExt	3	166	166	166	166	166	166
	H	5	194	194	194	194	194	194
ChainTask	SWL	1, 8	324	388	428	338	404	474
	SWH	1, 9	368	448	488	386	460	536
	NSL	8	416	466	506	430	482	552
	NSH	9	418	490	530	436	502	578
Schedule			272	272	290	272	272	290
GetTaskID			62	62	62	62	62	62
GetTaskState			270	270	270	284	284	284
EnableAllInterrupts			34	34	34	34	34	34
DisableAllInterrupts			38	38	38	38	38	38
ResumeAllInterrupts			122	122	122	122	122	122
SuspendAllInterrupts			60	60	60	60	60	60
ResumeOSInterrupts			116	116	116	116	116	116
SuspendOSInterrupts			76	76	76	76	76	76
GetResource	Task	7	414	414	366	414	414	366
	Combined	6	418	418	418	418	418	418
	CLEx	3	366	366	366	366	366	366
ReleaseResource	Task	7	378	378	378	378	378	378
	Combined	6	496	496	496	496	496	496
	CLEx	3	334	334	334	334	334	334
SetEvent	SW	1	n/a	n/a	n/a	324	324	412
	NS		n/a	n/a	n/a	376	376	466
	NS1i	10	n/a	n/a	n/a	234	n/a	n/a
	KL	2	n/a	n/a	n/a	282	282	374
	KL1i	2, 10	n/a	n/a	n/a	218	n/a	n/a

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
ClearEvent			n/a	n/a	n/a	176	176	176
GetEvent			n/a	n/a	n/a	208	208	208
WaitEvent	<default>		n/a	n/a	n/a	510	510	724
	fp	11	n/a	n/a	n/a	538	538	784
	1i	10	n/a	n/a	n/a	274	n/a	n/a
GetAlarmBase			164	164	164	164	164	164
GetAlarm			180	180	180	180	180	180
SetRelAlarm			260	260	260	260	260	260
SetAbsAlarm			282	282	282	282	282	282
CancelAlarm			174	174	174	174	174	174
InitCounter			220	220	220	220	220	220
GetCounterValue			190	190	190	190	190	190
osek_tick_alarm	<default>		116	116	116	116	116	116
	KL	2	40	40	40	40	40	40
osek_incr_counter			28	28	28	28	28	28
GetActiveApplicationMode		30	n/a	n/a	n/a	n/a	n/a	n/a
StartOS			312	312	312	312	312	312
ShutdownOS	NoHook	12	60	60	60	60	60	60
	Hook	13	72	72	72	72	72	72
InitCOM			4	4	4	4	4	4
CloseCOM			4	4	4	4	4	4
StartCOM			54	54	54	54	54	54
StopCOM			52	52	52	52	52	52
ReadFlag			34	34	34	34	34	34
ResetFlag			36	36	36	36	36	36
ReceiveMessage	CCCA	14	194	194	194	310	310	310
	CCCB	15	310	310	310	310	310	310
GetMessageResource			122	122	122	122	122	122
ReleaseMessageResource			124	124	124	124	124	124
GetMessageStatus			120	120	120	120	120	120
SendMessage	SW CCCA	1, 14	216	216	216	364	364	364
	SW CCCB	1, 15	352	352	352	364	364	364
	NS CCCA	14	216	216	216	364	364	364
	NS CCCB	15	352	352	352	364	364	364
	KL CCCA	2, 14	186	186	186	316	316	316
	KL CCCB	2, 15	304	304	304	316	316	316
main_dispatch	NoHook	12	216	216	252	216	216	252
	Hook	13	270	270	308	270	270	308
sub_dispatch	B1LF	19	22	22	22	22	22	22
	B1HI	20	108	108	108	108	108	108

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
	B1HF	21	110	110	110	110	110	110
	B2LI	22	n/a	58	84	n/a	58	84
	B2LF	23	n/a	66	92	n/a	66	92
	B2HI	24	n/a	160	212	n/a	160	212
	B2HF	25	n/a	142	206	n/a	142	206
	E1HI	26	n/a	n/a	n/a	376	376	438
	E1HF	27	n/a	n/a	n/a	370	370	446
	E2HI	28	n/a	n/a	n/a	n/a	n/a	438
	E2HF	29	n/a	n/a	n/a	n/a	n/a	446
ErrorHook support		16	144	144	144	144	144	144
	ServiceID	17	156	156	156	156	156	156
	Parameters	18	192	192	192	192	192	192
validity_checks		3	38	38	38	38	38	38
Timing_dispatch		4	92	92	92	92	92	92
Timing_termination		4	104	104	104	104	104	104
ActivateTaskset	SW	1	344	422	466	378	466	538
	NS		408	486	530	442	530	602
	KL	2	308	386	440	336	428	504
ChainTaskset	SWL	1, 8	434	492	536	446	522	578
	SWH	1, 9	484	566	612	496	600	654
	NSL	8	496	572	616	516	608	664
	NSH	9	516	604	650	536	638	696
GetTasksetRef			172	172	172	172	172	172
MergeTaskset			322	322	322	322	322	322
AssignTaskset			268	268	268	268	268	268
RemoveTaskset			324	324	324	324	324	324
TestSubTaskset			304	304	304	304	304	304
TestEquivalentTaskset			302	302	302	302	302	302
TickSchedule	SW	1	384	298	298	298	298	298
	NS		422	348	348	348	348	348
	KL	2	314	226	226	226	226	226
AdvanceSchedule	SW	1	366	288	288	288	288	288
	NS		414	344	344	344	344	344
	KL	2	366	278	278	278	278	278
StartSchedule			242	242	242	242	242	242
StopSchedule			180	180	180	180	180	180
GetScheduleStatus			220	220	220	220	220	220
GetScheduleValue			200	200	200	200	200	200
GetScheduleNext			124	124	124	124	124	124
SetScheduleNext			226	226	226	226	226	226

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
GetArrivalpointDelay			176	176	176	176	176	176
SetArrivalpointDelay			188	188	188	188	188	188
GetArrivalpointTasksetRef			174	174	174	174	174	174
GetArrivalpointNext			176	176	176	176	176	176
SetArrivalpointNext			266	266	266	266	266	266
TestArrivalpointWritable			202	202	202	202	202	202
GetExecutionTime			176	176	176	176	176	176
GetLargestExecutionTime			140	140	140	140	140	140
ResetLargestExecutionTime			144	144	144	144	144	144
GetStackOffset			18	18	18	18	18	18

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

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

4.2.4 Reserved Hardware Resources

No timer units are reserved by SSX5.

SSX5 reserves the general interrupt IVG14. As the Blackfin core user mode is not used IVG15 cannot trigger as application code runs at this level.

4.3 Performance

The collection of performance data for the Blackfin®/ADI port of SSX5 was achieved using a timer running 2 times slower than the CPU clock speed. The figures in this section, therefore, have an uncertainty level of up to 2 CPU cycles. The actual times are between 0 and 2 cycles shorter than those reported in the remainder of this section.

4.3.1 Execution Times for 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	33	43	54	35	52	72
	NS	25	36	46	28	45	64
	KL	20	25	37	23	35	56
TerminateTask	LExt	0	0	0	0	0	0

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
	H	107	107	114	107	107	114
ChainTask	SWL	151	169	188	172	194	223
	SWH	225	230	248	245	256	285
	NSL	151	169	188	172	194	224
	NSH	223	228	247	244	254	284
Schedule	SW	37	36	45	36	37	45
GetTaskID		14	13	13	13	14	14
GetTaskState		37	37	37	45	44	44
EnableAllInterrupts		10	10	10	10	10	10
DisableAllInterrupts		12	11	11	12	11	11
ResumeAllInterrupts		13	13	13	13	13	13
SuspendAllInterrupts		14	14	14	14	14	14
ResumeOSInterrupts		13	13	13	13	13	13
SuspendOSInterrupts		14	14	14	14	14	14
GetResource	Task	15	15	16	15	15	16
	Combined	29	28	28	29	28	28
	CLEx	n/a	n/a	n/a	n/a	n/a	n/a
ReleaseResource	Task	32	32	32	32	32	32
	Combined	52	52	52	52	52	52
	CLEx	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	n/a	n/a	n/a	41	41	45
	NS	n/a	n/a	n/a	37	37	43
	KL	n/a	n/a	n/a	28	28	31
ClearEvent		n/a	n/a	n/a	24	24	24
GetEvent		n/a	n/a	n/a	16	16	16
WaitEvent	<default>	n/a	n/a	n/a	294	294	331
	fp	n/a	n/a	n/a	303	303	332
GetAlarmBase		52	52	52	52	52	52
GetAlarm		32	32	32	32	32	32
SetRelAlarm		42	42	42	42	42	42
SetAbsAlarm		34	35	35	34	35	35
CancelAlarm		23	23	23	23	23	23
InitCounter		27	27	27	27	27	27
GetCounterValue		26	26	26	26	26	26
osek_tick_alarm	<default>	28	28	28	28	28	28
	KL	21	21	21	21	21	21
osek_incr_counter		8	8	8	8	8	8
GetActiveApplicationMode		5	5	5	5	5	5
StartOS		511	510	510	510	509	510
ShutdownOS	NoHook	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	
	Hook	23	22	22	23	22	22
InitCOM		7	6	6	7	6	6
CloseCOM		7	6	6	6	6	6
StartCOM		21	21	21	60	61	61
StopCOM		14	14	14	14	14	14
ReadFlag		n/a	n/a	n/a	7	7	7
ResetFlag		n/a	n/a	n/a	6	6	6
ReceiveMessage		24	24	24	104	104	104
GetMessageResource		n/a	n/a	n/a	34	33	33
ReleaseMessageResource		n/a	n/a	n/a	51	52	51
GetMessageStatus		n/a	n/a	n/a	24	24	24
SendMessage	SW	67	76	87	148	161	181
	NS	58	70	80	141	153	172
	KL	46	51	63	128	136	156
ActivateTaskset	SW	31	345	362	33	348	398
	NS	19	333	350	22	337	386
	KL	10	323	340	13	325	373
	SW2	31	345	362	33	348	398
	NS2	19	333	350	22	337	386
	KL2	10	323	340	13	325	373
ChainTaskset	SWL	147	460	483	166	477	538
	SWH	210	534	555	229	550	612
	NSL	147	460	483	166	477	538
	NSH	209	532	554	228	549	610
GetTasksetRef		12	12	12	12	12	12
MergeTaskset		22	23	23	23	22	22
AssignTaskset		12	12	12	12	12	12
RemoveTaskset		23	23	23	23	23	23
TestSubTaskset		25	25	25	25	25	25
TestEquivalentTaskset		25	25	25	25	25	25
TickSchedule	SW	52	372	389	63	377	422
	NS	38	362	379	52	367	413
	KL	28	354	371	44	359	404
	SW2	52	373	389	63	376	423
	NS2	37	363	379	52	366	413
	KL2	28	354	371	44	357	405
AdvanceSchedule	SW	47	366	383	56	370	416
	NS	36	356	373	46	361	406
	KL	24	349	366	39	353	398
	SW2	47	366	383	56	369	416

Configuration / Events / Shared Task Priorities / Multiple Task Activations		Application Uses					
		No			Yes		
		No		Yes	No		Yes
		No	Yes		No	Yes	
	NS2	36	356	373	46	359	407
	KL2	24	349	366	39	351	399
StartSchedule		46	46	46	47	46	46
StopSchedule		35	35	35	35	35	35
GetScheduleStatus		31	31	31	31	31	31
GetScheduleValue		40	40	40	40	40	40
GetScheduleNext		14	14	14	14	14	14
SetScheduleNext		16	15	15	15	16	16
GetArrivalpointDelay		13	13	13	13	13	13
SetArrivalpointDelay		14	13	13	13	14	14
GetArrivalpointTasksetRef		11	12	12	12	11	11
GetArrivalpointNext		12	12	12	12	12	12
SetArrivalpointNext		13	14	14	14	13	13
TestArrivalpointWritable		14	14	14	14	14	14
GetExecutionTime		8	8	8	8	8	8
GetLargestExecutionTime		12	12	12	12	12	12
ResetLargestExecutionTime		8	8	8	8	8	8
GetStackOffset		10	10	10	10	10	10

Timing

Configuration / Events / Shared Task Priorities / Multiple Task Activations		Application Uses					
		No			Yes		
		No		Yes	No		Yes
		No	Yes		No	Yes	
Service	Variant						
ActivateTask	SW	33	43	54	35	52	72
	NS	25	36	46	29	45	64
	KL	20	25	37	23	35	56
TerminateTask	LExt	0	0	0	0	0	0
	H	200	200	208	200	200	208
ChainTask	SWL	258	274	296	284	306	337
	SWH	321	326	347	348	358	390
	NSL	258	274	296	284	306	338
	NSH	320	328	349	350	361	393
Schedule	SW	37	36	45	37	37	45
GetTaskID		14	13	13	14	14	14
GetTaskState		37	37	37	44	44	44
EnableAllInterrupts		10	10	10	10	10	10

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
DisableAllInterrupts		12	11	11	11	11	11
ResumeAllInterrupts		13	13	13	13	13	13
SuspendAllInterrupts		14	14	14	14	14	14
ResumeOSInterrupts		13	13	13	13	13	13
SuspendOSInterrupts		14	14	14	14	14	14
GetResource	Task	15	15	16	15	15	16
	Combined	29	28	28	28	28	28
	CLEx	n/a	n/a	n/a	n/a	n/a	n/a
ReleaseResource	Task	32	32	32	32	32	32
	Combined	57	57	57	57	57	57
	CLEx	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	n/a	n/a	n/a	41	41	45
	NS	n/a	n/a	n/a	37	37	43
	KL	n/a	n/a	n/a	28	28	31
ClearEvent		n/a	n/a	n/a	24	24	24
GetEvent		n/a	n/a	n/a	16	16	16
WaitEvent	<default>	n/a	n/a	n/a	395	403	427
	fp	n/a	n/a	n/a	409	409	433
GetAlarmBase		52	52	52	52	52	52
GetAlarm		32	32	32	32	32	32
SetRelAlarm		42	42	42	42	42	42
SetAbsAlarm		34	35	35	35	35	35
CancelAlarm		23	23	23	23	23	23
InitCounter		27	27	27	27	27	27
GetCounterValue		26	26	26	26	26	26
osek_tick_alarm	<default>	28	28	28	28	28	28
	KL	21	21	21	21	21	21
osek_incr_counter		8	8	8	8	8	8
GetActiveApplicationMode		5	5	5	5	5	5
StartOS		1179	1177	1177	1177	1177	1177
ShutdownOS	NoHook	n/a	n/a	n/a	n/a	n/a	n/a
	Hook	23	22	22	22	22	22
InitCOM		7	6	6	6	6	6
CloseCOM		7	6	6	6	6	6
StartCOM		21	21	21	61	61	61
StopCOM		14	14	14	14	14	14
ReadFlag		n/a	n/a	n/a	7	7	7
ResetFlag		n/a	n/a	n/a	6	6	6
ReceiveMessage		24	24	24	104	104	104
GetMessageResource		n/a	n/a	n/a	33	33	33

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	56	55	56
GetMessageStatus		n/a	n/a	n/a	24	24	24
SendMessage	SW	67	76	87	144	161	181
	NS	58	70	80	137	153	172
	KL	46	51	63	123	136	156
ActivateTaskset	SW	31	345	362	33	348	398
	NS	19	333	350	21	337	386
	KL	10	323	340	12	325	373
	SW2	31	345	362	33	348	398
	NS2	19	333	350	21	337	386
	KL2	10	323	340	12	325	373
ChainTaskset	SWL	254	565	590	282	588	652
	SWH	310	630	654	336	653	717
	NSL	254	565	590	278	588	652
	NSH	309	632	656	335	652	719
GetTasksetRef		12	12	12	12	12	12
MergeTaskset		22	23	23	22	22	22
AssignTaskset		12	12	12	12	12	12
RemoveTaskset		23	23	23	23	23	23
TestSubTaskset		25	25	25	25	25	25
TestEquivalentTaskset		25	25	25	25	25	25
TickSchedule	SW	52	372	389	62	377	422
	NS	38	362	379	53	367	413
	KL	28	354	371	44	359	404
	SW2	52	373	389	62	376	423
	NS2	37	363	379	53	366	413
	KL2	28	354	371	44	357	405
AdvanceSchedule	SW	47	366	383	56	370	416
	NS	36	356	373	46	361	406
	KL	24	349	366	38	353	398
	SW2	47	366	383	56	369	416
	NS2	36	356	373	46	359	407
	KL2	24	349	366	38	351	399
StartSchedule		46	46	46	46	46	46
StopSchedule		35	35	35	35	35	35
GetScheduleStatus		31	31	31	31	31	31
GetScheduleValue		40	40	40	40	40	40
GetScheduleNext		14	14	14	14	14	14
SetScheduleNext		16	15	15	16	16	16
GetArrivalpointDelay		13	13	13	13	13	13

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
SetArrivalpointDelay		14	13	13	14	14	14
GetArrivalpointTasksetRef		11	12	12	11	11	11
GetArrivalpointNext		12	12	12	12	12	12
SetArrivalpointNext		13	14	14	13	13	13
TestArrivalpointWritable		14	14	14	14	14	14
GetExecutionTime		50	49	49	49	49	49
GetLargestExecutionTime		19	19	19	19	19	19
ResetLargestExecutionTime		14	14	14	14	14	14
GetStackOffset		10	10	10	10	10	10

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	125	135	143	127	144	162
	NS	129	143	151	131	152	170
	KL	109	119	128	112	129	146
TerminateTask	LExt	197	198	202	198	196	201
	H	245	246	251	246	245	250
ChainTask	SWL	387	405	429	415	435	469
	SWH	442	451	476	470	485	517
	NSL	396	413	437	424	447	478
	NSH	455	464	489	483	498	530
Schedule	SW	70	71	77	71	69	77
GetTaskID		18	18	18	18	17	17
GetTaskState		136	136	136	139	140	140
EnableAllInterrupts		13	13	13	13	13	13
DisableAllInterrupts		15	15	15	14	15	15
ResumeAllInterrupts		21	21	21	21	21	21
SuspendAllInterrupts		17	17	17	17	17	17
ResumeOSInterrupts		21	21	21	21	21	21
SuspendOSInterrupts		17	17	17	17	17	17
GetResource	Task	234	234	129	251	251	145
	Combined	122	122	122	137	138	138
	CLEx	137	137	137	153	153	153
ReleaseResource	Task	125	125	125	142	142	142

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
	Combined	139	139	139	155	155	155
	CLEx	126	126	126	142	142	142
SetEvent	SW	n/a	n/a	n/a	139	139	140
	NS	n/a	n/a	n/a	151	151	151
	KL	n/a	n/a	n/a	123	123	124
ClearEvent		n/a	n/a	n/a	50	50	50
GetEvent		n/a	n/a	n/a	103	103	103
WaitEvent	<default>	n/a	n/a	n/a	464	453	480
	fp	n/a	n/a	n/a	473	472	489
GetAlarmBase		117	117	117	117	117	117
GetAlarm		96	96	96	96	96	96
SetRelAlarm		124	124	124	124	124	124
SetAbsAlarm		104	104	104	104	104	104
CancelAlarm		89	89	89	89	89	89
InitCounter		161	162	162	161	161	162
GetCounterValue		88	88	88	88	88	88
osek_tick_alarm	<default>	44	44	44	45	44	44
	KL	21	21	21	21	21	21
osek_incr_counter		8	8	8	8	8	8
GetActiveApplicationMode		5	5	5	5	5	5
StartOS		1331	1332	1333	1332	1333	1333
ShutdownOS	NoHook	n/a	n/a	n/a	n/a	n/a	n/a
	Hook	25	25	25	24	25	25
InitCOM		7	7	7	6	7	7
CloseCOM		7	7	7	7	6	6
StartCOM		28	27	27	66	66	65
StopCOM		15	15	15	15	15	15
ReadFlag		n/a	n/a	n/a	16	15	15
ResetFlag		n/a	n/a	n/a	14	15	15
ReceiveMessage		81	81	81	160	160	160
GetMessageResource		n/a	n/a	n/a	210	210	210
ReleaseMessageResource		n/a	n/a	n/a	198	197	197
GetMessageStatus		n/a	n/a	n/a	63	63	63
SendMessage	SW	219	231	237	291	307	325
	NS	223	238	245	299	320	338
	KL	186	195	205	265	282	299
ActivateTaskset	SW	430	761	775	435	766	802
	NS	434	765	763	439	754	805
	KL	417	749	763	422	753	788
	SW2	430	761	775	435	766	802

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
	NS2	434	765	763	439	754	805
	KL2	417	749	763	422	753	788
ChainTaskset	SWL	691	1022	1049	718	1033	1096
	SWH	745	1082	1091	772	1094	1157
	NSL	717	1022	1044	744	1043	1107
	NSH	772	1081	1102	784	1104	1184
GetTasksetRef		104	104	104	104	103	104
MergeTaskset		77	77	77	77	78	78
AssignTaskset		63	63	63	63	62	63
RemoveTaskset		78	78	78	78	78	78
TestSubTaskset		72	72	72	72	73	72
TestEquivalentTaskset		72	72	72	72	73	72
TickSchedule	SW	96	817	832	491	823	856
	NS	106	829	842	502	833	866
	KL	72	795	809	468	800	833
	SW2	96	817	832	491	822	857
	NS2	106	829	842	502	832	867
	KL2	72	795	809	468	799	834
AdvanceSchedule	SW	88	811	825	485	817	850
	NS	100	826	839	499	830	863
	KL	74	796	811	471	803	836
	SW2	88	811	825	485	816	851
	NS2	100	826	839	499	829	864
	KL2	74	796	811	471	802	837
StartSchedule		75	75	75	75	76	76
StopSchedule		60	60	60	60	60	60
GetScheduleStatus		56	56	56	56	56	56
GetScheduleValue		66	66	66	66	66	66
GetScheduleNext		29	29	29	29	28	28
SetScheduleNext		44	44	44	44	43	43
GetArrivalpointDelay		35	35	35	35	35	35
SetArrivalpointDelay		49	48	49	48	48	48
GetArrivalpointTasksetRef		35	34	35	34	35	35
GetArrivalpointNext		35	35	35	35	35	35
SetArrivalpointNext		64	64	64	64	64	64
TestArrivalpointWritable		40	40	40	40	40	40
GetExecutionTime		63	64	64	64	63	63
GetLargestExecutionTime		92	92	92	92	92	92
ResetLargestExecutionTime		88	88	88	88	88	88
GetStackOffset		10	10	10	10	10	10

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	80	80	80	80	80	80
	Cat 2	156	156	156	156	156	156

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	80	80	80	80	80	80
	Cat 2	284	280	280	280	280	280

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	80	80	80	80	80	80
	Cat 2	280	284	284	284	284	284

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.

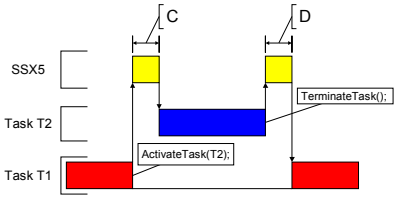


Figure 1: Task Activates a Higher Priority Task which Terminates Normally

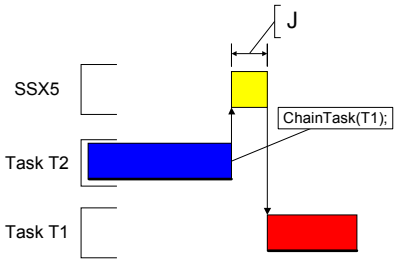


Figure 2: Task Chaining

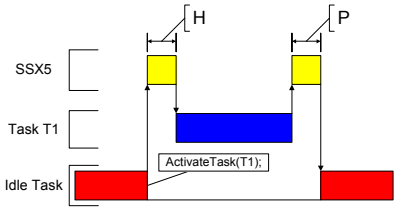


Figure 3: Task Activation from Idle Task

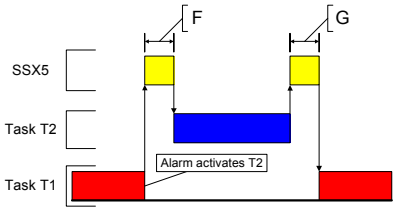


Figure 4: Task Activation from an Alarm

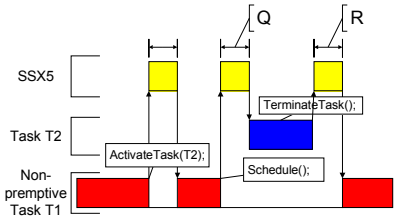


Figure 5: Non-Premptive Task Calls Schedule()

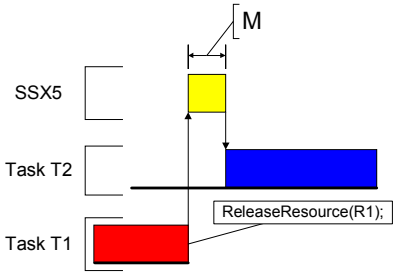


Figure 6: Blocked Task Activated by ReleaseResource()

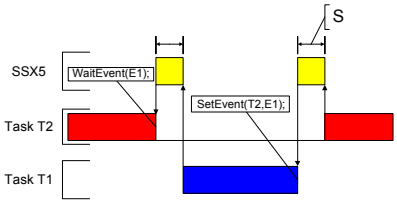


Figure 7: Waiting Task Activated by SetEvent()

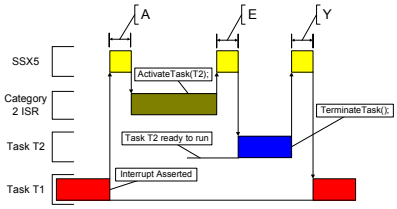


Figure 8: Category 2 ISR Activates a Higher Priority Task

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	36	55	70	36	55	69
Figure 1: D	Heavy, Basic/Extended	108	117	133	125	124	136
ChainTask	Light, Basic	91	116	139	94	115	144
Figure 2: J	Heavy, Basic/Extended	290	310	348	308	317	360
Pre-emption	Light, Basic	57	76	101	59	76	110
Figure 1: C	Heavy, Basic/Extended	140	153	175	162	178	213
From idle task	Light, Basic	57	76	101	59	76	110
Figure 3: H	Heavy, Basic/Extended	140	153	175	162	178	213
Triggered by alarm	Light, Basic	90	109	134	92	109	143
Figure 4: F	Heavy, Basic/Extended	173	186	208	195	212	246
Schedule	Light, Basic	52	61	86	52	61	86
Figure 5: Q	Heavy, Basic/Extended	135	138	160	155	155	179
Release resource	Light, Basic	58	67	84	58	67	84
Figure 6: M	Heavy, Basic/Extended	141	144	158	160	160	177
SetEvent							
Figure 7: S	Heavy, Extended	n/a	n/a	n/a	339	339	407
From category 2 ISR	Light, Basic	63	72	89	63	72	89
Figure 8: E	Heavy, Basic/Extended	146	149	163	166	166	183

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	139	156	170	139	154	169
Figure 1: D	Heavy, Basic/Extended	202	213	227	217	216	231
ChainTask	Light, Basic	203	222	245	204	220	250
Figure 2: J	Heavy, Basic/Extended	482	498	535	505	509	557
Pre-emption	Light, Basic	122	135	161	124	135	171
Figure 1: C	Heavy, Basic/Extended	200	212	236	228	244	281
From idle task	Light, Basic	122	135	161	124	135	171
Figure 3: H	Heavy, Basic/Extended	200	212	236	228	244	281
Triggered by alarm	Light, Basic	155	169	194	157	169	204
Figure 4: F	Heavy, Basic/Extended	233	246	269	261	278	314
Schedule	Light, Basic	117	121	146	117	121	147

Configuration / Events / Shared Task Priorities / Multiple Task Activations	Task Attributes	Application Uses: No / No / No	Application Uses: No / No / Yes	Application Uses: No / Yes / Yes	Application Uses: Yes / No / No	Application Uses: Yes / No / Yes	Application Uses: Yes / Yes / Yes
Figure 5: Q	Heavy, Basic/Extended	195	198	221	221	221	247
Release resource	Light, Basic	127	131	148	127	131	149
Figure 6: M	Heavy, Basic/Extended	205	208	223	231	231	249
SetEvent							
Figure 7: S	Heavy, Extended	n/a	n/a	n/a	409	407	458
From category 2 ISR	Light, Basic	243	244	262	240	244	263
Figure 8: E	Heavy, Basic/Extended	317	317	333	340	340	359

Extended

Configuration / Events / Shared Task Priorities / Multiple Task Activations	Task Attributes	Application Uses: No / No / No	Application Uses: No / No / Yes	Application Uses: No / Yes / Yes	Application Uses: Yes / No / No	Application Uses: Yes / No / Yes	Application Uses: Yes / Yes / Yes
Normal termination	Light, Basic	194	210	223	195	210	222
Figure 1: D	Heavy, Basic/Extended	246	258	270	263	263	275
ChainTask	Light, Basic	330	351	375	334	350	381
Figure 2: J	Heavy, Basic/Extended	646	669	707	669	683	726
Pre-emption	Light, Basic	208	223	250	212	224	260
Figure 1: C	Heavy, Basic/Extended	287	300	326	315	333	370
From idle task	Light, Basic	208	223	250	212	224	260
Figure 3: H	Heavy, Basic/Extended	287	300	326	315	333	370
Triggered by alarm	Light, Basic	258	272	300	262	274	309
Figure 4: F	Heavy, Basic/Extended	337	349	376	365	383	420
Schedule	Light, Basic	145	150	176	148	150	176
Figure 5: Q	Heavy, Basic/Extended	224	227	252	251	250	277
Release resource	Light, Basic	207	210	230	225	228	246
Figure 6: M	Heavy, Basic/Extended	286	287	306	328	328	347
SetEvent							
Figure 7: S	Heavy, Extended	n/a	n/a	n/a	532	531	582
From category 2 ISR	Light, Basic	233	238	257	236	238	257
Figure 8: E	Heavy, Basic/Extended	308	311	329	335	335	354

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		204	204	220	204	204	220
BCC1 lightweight, floating-point		220	220	236	220	220	236
BCC1 heavyweight, integer		392	392	408	392	392	408
BCC1 heavyweight, floating-point		392	392	408	392	392	408
BCC2 lightweight, integer		n/a	224	240	n/a	224	240
BCC2 lightweight, floating-point		n/a	224	240	n/a	224	240
BCC2 heavyweight, integer		n/a	400	408	n/a	400	408
BCC2 heavyweight, floating-point		n/a	400	408	n/a	400	408
ECC1 heavyweight, integer		n/a	n/a	n/a	444	444	460
ECC1 heavyweight, floating-point		n/a	n/a	n/a	444	444	460
ECC2 heavyweight, integer		n/a	n/a	n/a	n/a	n/a	460
ECC2 heavyweight, floating-point		n/a	n/a	n/a	n/a	n/a	460
Pre- and/or Post-Task hooks used							
Task type							
BCC1 lightweight, integer		220	220	220	220	220	220
BCC1 lightweight, floating-point		236	236	236	236	236	236
BCC1 heavyweight, integer		408	408	408	408	408	408
BCC1 heavyweight, floating-point		408	408	408	408	408	408
BCC2 lightweight, integer		n/a	240	240	n/a	240	240
BCC2 lightweight, floating-point		n/a	240	240	n/a	240	240
BCC2 heavyweight, integer		n/a	416	408	n/a	416	408
BCC2 heavyweight, floating-point		n/a	416	408	n/a	416	408
ECC1 heavyweight, integer		n/a	n/a	n/a	460	460	460
ECC1 heavyweight, floating-point		n/a	n/a	n/a	460	460	460
ECC2 heavyweight, integer		n/a	n/a	n/a	n/a	n/a	460
ECC2 heavyweight, floating-point		n/a	n/a	n/a	n/a	n/a	460

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	232	232	248	232	232	248
BCC1 lightweight, floating-point	248	248	264	248	248	264
BCC1 heavyweight, integer	416	416	432	416	416	432
BCC1 heavyweight, floating-point	416	416	432	416	416	432
BCC2 lightweight, integer	n/a	248	264	n/a	248	264
BCC2 lightweight, floating-point	n/a	248	264	n/a	248	264
BCC2 heavyweight, integer	n/a	424	436	n/a	424	436
BCC2 heavyweight, floating-point	n/a	424	436	n/a	424	436
ECC1 heavyweight, integer	n/a	n/a	n/a	464	464	480
ECC1 heavyweight, floating-point	n/a	n/a	n/a	464	464	480
ECC2 heavyweight, integer	n/a	n/a	n/a	n/a	n/a	484
ECC2 heavyweight, floating-point	n/a	n/a	n/a	n/a	n/a	484
Pre- and/or Post-Task hooks used						
Task type						
BCC1 lightweight, integer	248	248	248	248	248	248
BCC1 lightweight, floating-point	264	264	264	264	264	264
BCC1 heavyweight, integer	432	432	432	432	432	432
BCC1 heavyweight, floating-point	432	432	432	432	432	432
BCC2 lightweight, integer	n/a	264	264	n/a	264	264
BCC2 lightweight, floating-point	n/a	264	264	n/a	264	264
BCC2 heavyweight, integer	n/a	440	436	n/a	440	436
BCC2 heavyweight, floating-point	n/a	440	436	n/a	440	436
ECC1 heavyweight, integer	n/a	n/a	n/a	480	480	480
ECC1 heavyweight, floating-point	n/a	n/a	n/a	480	480	480
ECC2 heavyweight, integer	n/a	n/a	n/a	n/a	n/a	484
ECC2 heavyweight, floating-point	n/a	n/a	n/a	n/a	n/a	484

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	232	232	248	232	232	248
BCC1 lightweight, floating-point	248	248	264	248	248	264
BCC1 heavyweight, integer	416	416	432	416	416	432
BCC1 heavyweight, floating-point	416	416	432	416	416	432
BCC2 lightweight, integer	n/a	248	264	n/a	248	264
BCC2 lightweight, floating-point	n/a	248	264	n/a	248	264
BCC2 heavyweight, integer	n/a	424	436	n/a	424	436
BCC2 heavyweight, floating-point	n/a	424	436	n/a	424	436
ECC1 heavyweight, integer	n/a	n/a	n/a	480	480	496
ECC1 heavyweight, floating-point	n/a	n/a	n/a	480	480	496
ECC2 heavyweight, integer	n/a	n/a	n/a	n/a	n/a	500
ECC2 heavyweight, floating-point	n/a	n/a	n/a	n/a	n/a	500
Pre- and/or Post-Task hooks used						
Task type						
BCC1 lightweight, integer	248	248	248	248	248	248
BCC1 lightweight, floating-point	264	264	264	264	264	264
BCC1 heavyweight, integer	432	432	432	432	432	432
BCC1 heavyweight, floating-point	432	432	432	432	432	432
BCC2 lightweight, integer	n/a	264	264	n/a	264	264
BCC2 lightweight, floating-point	n/a	264	264	n/a	264	264
BCC2 heavyweight, integer	n/a	440	436	n/a	440	436
BCC2 heavyweight, floating-point	n/a	440	436	n/a	440	436
ECC1 heavyweight, integer	n/a	n/a	n/a	496	496	496
ECC1 heavyweight, floating-point	n/a	n/a	n/a	496	496	496
ECC2 heavyweight, integer	n/a	n/a	n/a	n/a	n/a	500
ECC2 heavyweight, floating-point	n/a	n/a	n/a	n/a	n/a	500

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