LiveDevices

Binding Manual: Hitachi H8S

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

Disclaimer

Trademarks

Contents

1 About this Manual

1.1 Purpose and Audience

This document provides port specific information for the Hitachi/H8S implementation of Realogy Real-Time Architect.

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

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

1.2 Document Conventions

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

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

1.3 Overview of Manual

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

2 Toolchain Issues

This section details important things you need to know about SSX5 and your toolchain. A port of SSX5 is specific to both the target hardware *and* the compiler toolchain. You must make sure that you build your application with this toolchain. If you are interested in using a different version of the same toolchain then you should contact technical support to see whether this is possible.

2.1 Memory Model

SSX5 operates in the "Advanced" memory mode – this mode offers a single 24-bit address space for code and data.

2.2 Compiler

The following compiler was used to build SSX5.

Vendor	Hitachi
Compiler	ch38
Version	4.0.03

The following table details compulsory compiler options for application code.

Option	**Description**
`-cpu=2600a:24`	H8S/2600 CPU core operating in advanced memory mode with 24 bit address space.
`-op=1`	Optimize for size.
`-cm`	Optimize common sub-expressions.
`-C=A`	Compile via assembler. This is necessary to compile the FP context save and restores functions in `osfptgt.h` and `osfptgt.c`.
`-m`	Show all messages.

The following table details prohibited compiler options for application code.

Option	**Description**
`-REGP=3`	Change number of parameter registers from 2 to 3.

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

Option	**Description**
`-cpu=2600a:24`	H8S/2600 CPU core operating in advanced memory mode with 24 bit address space.
`-op=1`	Optimize for size.
`-cm`	Optimize common sub-expressions.
`-m`	Show all messages.

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

Option	Description
`-REGP=3`	Change number of parameter registers from 2 to 3.

The Hitachi compiler user to build SSX5 is included with Hitachi Embedded Workshop 2.1 which includes v5.0 of the Hitachi H8S Tools.

The Hitachi compiler assumes that the stack does not span a 64Kb boundary. Therefore care must be taken within the linker control file to ensure that the entire stack segment (“S”) is located within a 64Kb ‘page’.

The Hitachi compiler includes an option to expand the number of registers used for parameter passing from 2 to 3. This option must *not* be used when compiling RTA applications.

2.3 Assembler

The following assembler was used to build SSX5.

Vendor	Hitachi
Assembler	as38
Version	4.1

The following table details compulsory assembler options for application code.

Option	Description
`-cpu=2600a:24`	H8S/2600 CPU core operating in advanced memory mode with 24 bit address space.
`-op`	Optimize (branch displacement, index displacement, absolute address width).

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

The following table details compulsory assembler options for `osgen.src`.

Option	Description
`-cpu=2600a:24`	H8S/2600 CPU core operating in advanced memory mode with 24 bit address space.
`-op`	Optimize (branch displacement, index displacement, absolute address width).

2.4 Linker/Locator

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

Sections	ROM/RAM	Description
`os_pid`	ROM	SSX5 read-only data.
`os_pird`	ROM	SSX5 initialization data.
`os_intvec`	ROM	Vector table if generated by RTArchitect.
`os_pir`	RAM	SSX5 initialized data.
`os_pur`	RAM	SSX5 uninitialized data.

The following table details the compiler run-time library functions required by SSX5.

C Library Functions	Description
`$MVN$3`	C Structure copy.
`setjmp/longjmp`	

Some RTA API calls require functions from the standard C library and so applications must be linked with it.

The Hitachi C compiler does *not* ship with a precompiled C library. Instead the Hitachi standard library generator must be used to create a custom C library containing the required functions. RTA only requires functions from the `RUNTIME` section.

2.5 Debugger

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

3 Target Hardware Issues

3.1 Interrupts

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

3.1.1 Interrupt Levels

Interrupts in SSX5 are allocated an Interrupt Priority Level (IPL), which is a processor independent abstraction of the interrupt priorities available on the target hardware. You can find out more about IPLs in the *RTA User Guide*. You can find out more about the hardware interrupt controller in the *H8S/2612 Series Hardware Manual*. The following table shows how SSX5 IPLs relate to interrupt priorities on the target hardware.

IPL Value	ILVL Value	Description
0	0	User level.
1-7	1-7	Category 1 and 2 interrupts.
8	-	NMI only (Category 1).

3.1.2 Interrupt Vectors

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

Vector	Legality
0x0, 0x4	The reset vectors are outside the control of RTA.
0x8-0x18 and 0x20-0x1fc	Category 1 and 2 interrupts at IPL 1-7.
0x1c	Category 1 interrupt at an IPL of 8.

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

Base Address	Notes
0x8	

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

3.1.4 Category 2 Handlers

Category 2 ISRs are provided with a C function context by SSX5, as SSX5 handles the interrupt context itself.

They are written using the OSEK standard `ISR()` macro as in the following example:

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

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

3.1.5 Vector Table Issues

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

Vector Location	Label
0xVVVV	`_os_wrapper_VVVV`
eg : 0x00E0	`_os_wrapper_00E0`

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

The build process will generate a vector table covering all Category 1 and 2 ISRs with the exception of the vector addresses in the range 0x0 to 0x4. The generated vector table will cover the range 0x8 to 0x1fc inclusive.

The “Power ON” and “Manual” reset vectors at addresses 0x0 and 0x4 respectively are not within the domain of RTA and must be programmed by the user. The entry point for the “Power ON” reset should be marked within the application code as follows:

```
#pragma entry PowerON_Reset
```

Defining the “Power ON” reset function in this manner ensures that the C compiler generates code to set the stack pointer as the first instruction in the function. The same effect can also be achieved using the `__entry` qualifier.

The “Power ON” and “Manual” reset vectors must be created within the application, for example:

```
#pragma section RESETVEC
typedef void (*interrupt_vector)(void);
const interrupt_vector reset_vector_table[] = {
        PowerON_Reset,
        0
};
```

The linker control file must ensure that the section (in this case `CRESETVEC`) is located at 0x0.

Important: When creating constant data sections in C, the Hitachi C compiler automatically prefixes all section names with “C”. Therefore the linker must locate the section `CRESETVEC` at address 0x0 rather than `RESETVEC`.

3.2 Register Settings

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

SSX5 does not reserve the use of any hardware registers.

3.3 Stack Usage

3.3.1 Number of Stacks

A single stack is used. The first argument to `StackFaultHook` is always 0 and `StackOffsetType` is a scalar, representing the number of bytes on the stack, with C type: `UInt16Type`

3.3.2 Stack Usage within API Calls

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

Standard

API max usage (bytes): 56

Timing

API max usage (bytes): 56

Extended

API max usage (bytes): 64

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

The stack pointer on the H8S must be kept even at all times.

4 Parameters of Implementation

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

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

Processor	**H8S/2612**
Clock speed (MHz)	18.432
Code memory	On-chip FLASH
Read-only data memory	On-chip FLASH
Read-write data memory	On-chip RAM

4.1 Functionality

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

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

4.2 Hardware Resources

4.2.1 ROM and RAM Overheads

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

Standard

Configuration		**Application Uses**					
Events		**No**			**Yes**		
Shared Task Priorities		**No**		**Yes**	**No**		**Yes**
Multiple Task Activations		**No**	**Yes**		**No**	**Yes**	
OS overhead	RAM	22	22	22	22	22	22
	ROM	156	156	156	156	156	156
COM overhead	RAM	2	2	2	2	2	2
	ROM	7	7	7	7	7	7

Timing

Configuration		**Application Uses**					
Events		**No**			**Yes**		
Shared Task Priorities		**No**		**Yes**	**No**		**Yes**
Multiple Task Activations		**No**	**Yes**		**No**	**Yes**	
OS overhead	RAM	34	34	34	34	34	34
	ROM	194	194	194	194	194	194
COM overhead	RAM	2	2	2	2	2	2
	ROM	7	7	7	7	7	7

Extended

Configuration		**Application Uses**					
Events		**No**			**Yes**		
Shared Task Priorities		**No**		**Yes**	**No**		**Yes**
Multiple Task Activations		**No**	**Yes**		**No**	**Yes**	
OS overhead	RAM	43	43	43	43	43	43
	ROM	230	230	230	230	230	230
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

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

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

The following tables give the ROM and/or RAM requirements (in bytes) for each OS object in SSX5. (The OSEK COM class was set to CCCA for systems without events, CCCB for systems with events. A default message of size 10 bytes was used both CCCA and CCCB. The CCCB message size includes queued messages.)

Standard

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
BCC1 Lightweight task	RAM	0	0	0	0	0	0
	ROM	34	34	34	34	34	34
BCC1 Heavyweight task	RAM	4	4	4	4	4	4
	ROM	38	38	38	38	38	38
BCC2 task	RAM	n/a	6	8	n/a	6	8
	ROM	n/a	40	48	n/a	40	48
ECC1, Integer task	RAM	n/a	n/a	n/a	42	42	42
	ROM	n/a	n/a	n/a	54	54	54
ECC1, floating point task	RAM	n/a	n/a	n/a	50	50	50
	ROM	n/a	n/a	n/a	54	54	54
ECC2, Integer task	RAM	n/a	n/a	n/a	n/a	n/a	44
	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	52
	ROM	n/a	n/a	n/a	n/a	n/a	62
Category 2 ISR	RAM	0	0	0	0	0	0
	ROM	58	58	58	58	58	58
Category 2 ISR, floating point	RAM	8	8	8	8	8	8
	ROM	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**	
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	6	6	6	6	6	6
	ROM	40	40	40	40	40	40
Counter	RAM	2	2	2	2	2	2
	ROM	38	38	38	38	38	38
Message	RAM	11	11	11	31	31	31
	ROM	18	18	18	48	48	48
Flag	RAM	1	1	1	1	1	1
	ROM	1	1	1	1	1	1
Message resource	RAM	0	0	0	0	0	0
	ROM	20	20	20	20	20	20
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	10	0	10	10
Arrivalpoint (readonly)	RAM	0	0	0	0	0	0
	ROM	10	10	10	10	10	10
Arrivalpoint (writable)	RAM	10	10	10	10	10	10
	ROM	10	10	10	10	10	10
Schedule	RAM	10	10	10	10	10	10
	ROM	32	32	32	32	32	32
Taskset (readonly)	RAM	0	0	0	0	0	0
	ROM	4	4	4	4	4	4
Taskset (writable)	RAM	4	4	4	4	4	4
	ROM	4	4	4	4	4	4

Timing

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
BCC1 Lightweight task	RAM	6	6	6	6	6	6
	ROM	40	40	40	40	40	40
BCC1 Heavyweight task	RAM	10	10	10	10	10	10
	ROM	44	44	44	44	44	44
BCC2 task	RAM	n/a	12	14	n/a	12	14
	ROM	n/a	46	54	n/a	46	54
ECC1, Integer task	RAM	n/a	n/a	n/a	48	48	48
	ROM	n/a	n/a	n/a	60	60	60
ECC1, floating point task	RAM	n/a	n/a	n/a	56	56	56
	ROM	n/a	n/a	n/a	60	60	60
ECC2, Integer task	RAM	n/a	n/a	n/a	n/a	n/a	50
	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	58
	ROM	n/a	n/a	n/a	n/a	n/a	68
Category 2 ISR	RAM	6	6	6	6	6	6
	ROM	90	90	90	90	90	90
Category 2 ISR, floating point	RAM	14	14	14	14	14	14
	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	6	6	6	6	6	6
	ROM	40	40	40	40	40	40
Counter	RAM	2	2	2	2	2	2
	ROM	38	38	38	38	38	38
Message	RAM	11	11	11	31	31	31
	ROM	18	18	18	48	48	48
Flag	RAM	1	1	1	1	1	1
	ROM	1	1	1	1	1	1
Message resource	RAM	0	0	0	0	0	0
	ROM	20	20	20	20	20	20
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	10	0	10	10

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	10	10	10	10	10	10
Arrivalpoint (writable)	RAM	10	10	10	10	10	10
	ROM	10	10	10	10	10	10
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	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	7	7	7	7	7	7
	ROM	46	46	46	46	46	46
BCC1 Heavyweight task	RAM	12	12	12	12	12	12
	ROM	46	46	46	46	46	46
BCC2 task	RAM	n/a	14	16	n/a	14	16
	ROM	n/a	48	56	n/a	48	56
ECC1, Integer task	RAM	n/a	n/a	n/a	50	50	50
	ROM	n/a	n/a	n/a	62	62	62
ECC1, floating point task	RAM	n/a	n/a	n/a	58	58	58
	ROM	n/a	n/a	n/a	62	62	62
ECC2, Integer task	RAM	n/a	n/a	n/a	n/a	n/a	52
	ROM	n/a	n/a	n/a	n/a	n/a	70
ECC2, floating point task	RAM	n/a	n/a	n/a	n/a	n/a	60
	ROM	n/a	n/a	n/a	n/a	n/a	70
Category 2 ISR	RAM	7	7	7	7	7	7
	ROM	96	96	96	96	96	96
Category 2 ISR, floating point	RAM	15	15	15	15	15	15
	ROM	104	104	104	104	104	104
Resource	RAM	6	6	6	6	6	6
	ROM	28	28	28	28	28	28
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	6	6	6	6	6	6
	ROM	28	28	28	28	28	28
Alarm	RAM	6	6	6	6	6	6
	ROM	44	44	44	44	44	44
Counter	RAM	2	2	2	2	2	2
	ROM	42	42	42	42	42	42
Message	RAM	11	11	11	31	31	31
	ROM	22	22	22	52	52	52
Flag	RAM	1	1	1	1	1	1
	ROM	1	1	1	1	1	1
Message resource	RAM	6	6	6	6	6	6
	ROM	28	28	28	28	28	28
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	10	0	10	10
Arrivalpoint (readonly)	RAM	0	0	0	0	0	0
	ROM	16	16	16	16	16	16
Arrivalpoint (writable)	RAM	16	16	16	16	16	16
	ROM	16	16	16	16	16	16
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	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, that is each API call is placed into a separately linkable module. The following sections list the module sizes (in bytes) for each API call for each of the 3 SSX5 OS status types (standard, timing, and extended). In some cases there are multiple variants of particular API calls. This is because the offline configuration of SSX5 can determine when optimized versions of the API calls can be used and will select the smallest and fastest call. In these cases, modules sizes are given for each variant under the particular configuration of SSX5 for which the call is valid.

The call variants are as follows:

Variant	Description
1i	Idle task is only ECC task.
CCCA	OSEK COM class.
CCCB	OSEK COM class.
CLEx	Resource tests in Extended OS Status.
fp	ECC task uses floating point.
H	Used for heavyweight termination only.
Hook	Pre- and Post- Task hooks are used.
KL	API is called from OS level.
KL1i	API is called from OS level, idle task is only ECC task.
KL2	Activated taskset has one BCC2 task.
LExt	Used for lightweight termination in Extended Status.
No Params	`ErrorHook` uses `GetServiceID`, but does not use
No ServiceID	`ErrorHook` does not use `GetServiceID` or
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	166	238	318	178	250	372
	NS		140	212	292	152	224	342
	KL	2	82	160	224	94	172	276
TerminateTask	LExt	3	n/a	n/a	n/a	n/a	n/a	n/a
	H	5	26	26	26	26	26	26
ChainTask	SWL	1, 8	136	212	284	148	232	336
	SWH	1, 9	174	248	324	186	268	372
	NSL	8	136	212	284	148	232	336
	NSH	9	162	236	312	174	256	360

Configuration			**Application Uses**					
Events			**No**			**Yes**		
Shared Task Priorities			**No**		**Yes**	**No**		**Yes**
Multiple Task Activations			**No**	**Yes**		**No**	**Yes**	
Schedule			118	118	152	118	118	152
GetTaskID			44	44	44	44	44	44
GetTaskState			96	96	96	132	132	132
EnableAllInterrupts			24	24	24	24	24	24
DisableAllInterrupts			14	14	14	14	14	14
ResumeAllInterrupts			38	38	38	38	38	38
SuspendAllInterrupts			30	30	30	30	30	30
ResumeOSInterrupts			38	38	38	38	38	38
SuspendOSInterrupts			46	46	46	46	46	46
GetResource	Task	7	52	52	58	52	52	58
	Combined	6	116	116	116	116	116	116
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
ReleaseResource	Task	7	86	86	86	86	86	86
	Combined	6	86	86	86	86	86	86
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	1	n/a	n/a	n/a	172	172	324
	NS		n/a	n/a	n/a	140	140	298
	NS1i	10	n/a	n/a	n/a	68	n/a	n/a
	KL	2	n/a	n/a	n/a	110	110	260
	KL1i	2, 10	n/a	n/a	n/a	28	n/a	n/a
ClearEvent			n/a	n/a	n/a	54	54	54
GetEvent			n/a	n/a	n/a	64	64	64
WaitEvent	<default>		n/a	n/a	n/a	294	294	542
	fp	11	n/a	n/a	n/a	330	330	620
	1i	10	n/a	n/a	n/a	50	n/a	n/a
GetAlarmBase			62	62	62	62	62	62
GetAlarm			118	118	118	118	118	118
SetRelAlarm			130	130	130	130	130	130
SetAbsAlarm			154	154	154	154	154	154
CancelAlarm			94	94	94	94	94	94
InitCounter			76	76	76	76	76	76
GetCounterValue			90	90	90	90	90	90
osek_tick_alarm	<default>		84	84	84	84	84	84
	KL	2	44	44	44	44	44	44
osek_incr_counter			52	52	52	52	52	52
GetActiveApplicationMode		31	n/a	n/a	n/a	n/a	n/a	n/a
StartOS			142	142	142	142	142	142
ShutdownOS	NoHook	12	34	34	34	34	34	34
	Hook	13	40	40	40	40	40	40
InitCOM			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**	
CloseCOM			4	4	4	4	4	4
StartCOM			20	20	20	20	20	20
StopCOM			16	16	16	16	16	16
ReadFlag		31	n/a	n/a	n/a	n/a	n/a	n/a
ResetFlag		31	n/a	n/a	n/a	n/a	n/a	n/a
ReceiveMessage	CCCA	14	78	78	78	260	260	260
	CCCB	15	260	260	260	260	260	260
GetMessageResource			50	50	50	50	50	50
ReleaseMessageResource			40	40	40	40	40	40
GetMessageStatus			60	60	60	60	60	60
SendMessage	SW CCCA	1, 14	96	96	96	286	286	286
	SW CCCB	1, 15	270	270	270	286	286	286
	NS CCCA	14	96	96	96	286	286	286
	NS CCCB	15	270	270	270	286	286	286
	KL CCCA	2, 14	62	62	62	250	250	250
	KL CCCB	2, 15	234	234	234	250	250	250
main_dispatch	NoHook	12	154	154	222	154	154	222
	Hook	13	198	198	266	198	198	266
sub_dispatch	B1LI	19	16	16	16	16	16	16
	B1LF	20	24	24	24	24	24	24
	B1HI	21	116	116	116	116	116	116
	B1HF	22	124	124	124	124	124	124
	B2LI	23	n/a	94	132	n/a	94	132
	B2LF	24	n/a	102	140	n/a	102	140
	B2HI	25	n/a	180	314	n/a	180	314
	B2HF	26	n/a	188	322	n/a	188	322
	E1HI	27	n/a	n/a	n/a	440	440	584
	E1HF	28	n/a	n/a	n/a	448	448	592
	E2HI	29	n/a	n/a	n/a	n/a	n/a	584
	E2HF	30	n/a	n/a	n/a	n/a	n/a	592
CAT2_wrapper			60	60	60	60	60	60
hook_support	No ServiceID	16	36	36	36	36	36	36
	No Parameters	17	46	46	46	46	46	46
		18	74	74	74	74	74	74
fp_support			154	154	154	154	154	154
utility_functions	common		0	0	0	0	0	0
	optional	32	26	26	26	26	26	26
	optional	32	66	66	66	66	66	66
validity_checks		3	n/a	n/a	n/a	n/a	n/a	n/a
Timing_dispatch		4	n/a	n/a	n/a	n/a	n/a	n/a

Configuration			**Application Uses**					
Events			**No**			**Yes**		
Shared Task Priorities			**No**		**Yes**	**No**		**Yes**
Multiple Task Activations			**No**	**Yes**		**No**	**Yes**	
Timing_termination		4	n/a	n/a	n/a	n/a	n/a	n/a
ActivateTaskset	SW	1	94	160	276	108	188	328
	NS		68	142	256	82	170	296
	KL	2	34	104	210	48	132	252
ChainTaskset	SWL	1, 8	74	130	230	74	144	264
	SWH	1, 9	132	202	306	132	216	334
	NSL	8	74	130	230	74	144	264
	NSH	9	120	190	294	120	204	322
GetTasksetRef			12	12	12	12	12	12
MergeTaskset			62	62	62	62	62	62
AssignTaskset			12	12	12	12	12	12
RemoveTaskset			64	64	64	64	64	64
TestSubTaskset			78	78	78	78	78	78
TestEquivalentTaskset			72	72	72	72	72	72
TickSchedule	SW	1	210	188	188	188	188	188
	NS		184	158	158	158	158	158
	KL	2	142	132	132	132	132	132
AdvanceSchedule	SW	1	176	158	158	158	158	158
	NS		146	128	128	128	128	128
	KL	2	114	90	90	90	90	90
StartSchedule			102	102	102	102	102	102
StopSchedule			72	72	72	72	72	72
GetScheduleStatus			108	108	108	108	108	108
GetScheduleValue			82	82	82	82	82	82
GetScheduleNext			16	16	16	16	16	16
SetScheduleNext			12	12	12	12	12	12
GetArrivalpointDelay			10	10	10	10	10	10
SetArrivalpointDelay			8	8	8	8	8	8
GetArrivalpointTasksetRef			8	8	8	8	8	8
GetArrivalpointNext			14	14	14	14	14	14
SetArrivalpointNext			10	10	10	10	10	10
TestArrivalpointWritable			42	42	42	42	42	42
GetExecutionTime			4	4	4	4	4	4
GetLargestExecutionTime			12	12	12	12	12	12
ResetLargestExecutionTime			4	4	4	4	4	4
GetStackOffset			24	24	24	24	24	24

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	166	238	318	178	250	372
	NS		140	212	292	152	224	342
	KL	2	82	160	224	94	172	276
TerminateTask	LExt	3	n/a	n/a	n/a	n/a	n/a	n/a
	H	5	26	26	26	26	26	26
ChainTask	SWL	1, 8	136	212	284	148	232	336
	SWH	1, 9	174	248	324	186	268	372
	NSL	8	136	212	284	148	232	336
	NSH	9	162	236	312	174	256	360
Schedule			136	136	170	136	136	170
GetTaskID			44	44	44	44	44	44
GetTaskState			96	96	96	132	132	132
EnableAllInterrupts			24	24	24	24	24	24
DisableAllInterrupts			14	14	14	14	14	14
ResumeAllInterrupts			38	38	38	38	38	38
SuspendAllInterrupts			30	30	30	30	30	30
ResumeOSInterrupts			38	38	38	38	38	38
SuspendOSInterrupts			46	46	46	46	46	46
GetResource	Task	7	52	52	58	52	52	58
	Combined	6	116	116	116	116	116	116
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
ReleaseResource	Task	7	104	104	104	104	104	104
	Combined	6	104	104	104	104	104	104
	CLEx	3	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	1	n/a	n/a	n/a	172	172	324
	NS		n/a	n/a	n/a	140	140	298
	NS1i	10	n/a	n/a	n/a	68	n/a	n/a
	KL	2	n/a	n/a	n/a	110	110	260
	KL1i	2, 10	n/a	n/a	n/a	28	n/a	n/a
ClearEvent			n/a	n/a	n/a	54	54	54
GetEvent			n/a	n/a	n/a	64	64	64
WaitEvent	<default>		n/a	n/a	n/a	294	294	542
	fp	11	n/a	n/a	n/a	330	330	620
	1i	10	n/a	n/a	n/a	50	n/a	n/a
GetAlarmBase			62	62	62	62	62	62
GetAlarm			118	118	118	118	118	118
SetRelAlarm			130	130	130	130	130	130
SetAbsAlarm			154	154	154	154	154	154

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
CancelAlarm			94	94	94	94	94	94
InitCounter			76	76	76	76	76	76
GetCounterValue			90	90	90	90	90	90
osek_tick_alarm	<default>		84	84	84	84	84	84
	KL	2	44	44	44	44	44	44
osek_incr_counter			52	52	52	52	52	52
GetActiveApplicationMode		31	n/a	n/a	n/a	n/a	n/a	n/a
StartOS			190	190	190	190	190	190
ShutdownOS	NoHook	12	34	34	34	34	34	34
	Hook	13	40	40	40	40	40	40
InitCOM			4	4	4	4	4	4
CloseCOM			4	4	4	4	4	4
StartCOM			20	20	20	20	20	20
StopCOM			16	16	16	16	16	16
ReadFlag		31	n/a	n/a	n/a	n/a	n/a	n/a
ResetFlag		31	n/a	n/a	n/a	n/a	n/a	n/a
ReceiveMessage	CCCA	14	78	78	78	260	260	260
	CCCB	15	260	260	260	260	260	260
GetMessageResource			50	50	50	50	50	50
ReleaseMessageResource			40	40	40	40	40	40
GetMessageStatus			60	60	60	60	60	60
SendMessage	SW CCCA	1, 14	96	96	96	286	286	286
	SW CCCB	1, 15	270	270	270	286	286	286
	NS CCCA	14	96	96	96	286	286	286
	NS CCCB	15	270	270	270	286	286	286
	KL CCCA	2, 14	62	62	62	250	250	250
	KL CCCB	2, 15	234	234	234	250	250	250
main_dispatch	NoHook	12	260	260	328	260	260	328
	Hook	13	304	304	372	304	304	372
sub_dispatch	B1LI	19	4	4	4	4	4	4
	B1LF	20	12	12	12	12	12	12
	B1HI	21	116	116	116	116	116	116
	B1HF	22	124	124	124	124	124	124
	B2LI	23	n/a	76	112	n/a	76	112
	B2LF	24	n/a	84	120	n/a	84	120
	B2HI	25	n/a	158	290	n/a	158	290
	B2HF	26	n/a	166	298	n/a	166	298
	E1HI	27	n/a	n/a	n/a	454	454	586
	E1HF	28	n/a	n/a	n/a	462	462	594
	E2HI	29	n/a	n/a	n/a	n/a	n/a	586
	E2HF	30	n/a	n/a	n/a	n/a	n/a	594

Configuration			**Application Uses**					
Events			**No**			**Yes**		
Shared Task Priorities			**No**		**Yes**	**No**		**Yes**
Multiple Task Activations			**No**	**Yes**		**No**	**Yes**	
CAT2_wrapper			206	206	206	206	206	206
hook_support	No ServiceID	16	36	36	36	36	36	36
	No Parameters	17	46	46	46	46	46	46
		18	74	74	74	74	74	74
fp_support			154	154	154	154	154	154
utility_functions	common		0	0	0	0	0	0
	optional	32	26	26	26	26	26	26
	optional	32	66	66	66	66	66	66
validity_checks		3	n/a	n/a	n/a	n/a	n/a	n/a
Timing_dispatch		4	84	84	84	84	84	84
Timing_termination		4	132	132	132	132	132	132
ActivateTaskset	SW	1	94	160	276	108	188	328
	NS		68	142	256	82	170	296
	KL	2	34	104	210	48	132	252
ChainTaskset	SWL	1, 8	74	130	230	74	144	264
	SWH	1, 9	132	202	306	132	216	334
	NSL	8	74	130	230	74	144	264
	NSH	9	120	190	294	120	204	322
GetTasksetRef			12	12	12	12	12	12
MergeTaskset			62	62	62	62	62	62
AssignTaskset			12	12	12	12	12	12
RemoveTaskset			64	64	64	64	64	64
TestSubTaskset			78	78	78	78	78	78
TestEquivalentTaskset			72	72	72	72	72	72
TickSchedule	SW	1	210	188	188	188	188	188
	NS		184	158	158	158	158	158
	KL	2	142	132	132	132	132	132
AdvanceSchedule	SW	1	176	158	158	158	158	158
	NS		146	128	128	128	128	128
	KL	2	114	90	90	90	90	90
StartSchedule			102	102	102	102	102	102
StopSchedule			72	72	72	72	72	72
GetScheduleStatus			108	108	108	108	108	108
GetScheduleValue			82	82	82	82	82	82
GetScheduleNext			16	16	16	16	16	16
SetScheduleNext			12	12	12	12	12	12
GetArrivalpointDelay			10	10	10	10	10	10
SetArrivalpointDelay			8	8	8	8	8	8
GetArrivalpointTasksetRef			8	8	8	8	8	8
GetArrivalpointNext			14	14	14	14	14	14

Configuration			Application Uses					
Events			No			Yes		
Shared Task Priorities			No		Yes	No		Yes
Multiple Task Activations			No	Yes		No	Yes	
SetArrivalpointNext			10	10	10	10	10	10
TestArrivalpointWritable			42	42	42	42	42	42
GetExecutionTime			108	108	108	108	108	108
GetLargestExecutionTime			18	18	18	18	18	18
ResetLargestExecutionTime			18	18	18	18	18	18
GetStackOffset			24	24	24	24	24	24

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	266	346	444	284	358	494
	NS		310	410	514	330	422	564
	KL	2	174	252	320	192	264	394
TerminateTask	LExt	3	114	114	114	114	114	114
	H	5	154	154	154	154	154	154
ChainTask	SWL	1, 8	294	380	452	314	398	510
	SWH	1, 9	342	428	508	362	444	564
	NSL	8	372	452	534	390	468	590
	NSH	9	412	490	578	430	504	630
Schedule			262	262	298	262	262	298
GetTaskID			60	60	60	60	60	60
GetTaskState			232	232	232	246	246	246
EnableAllInterrupts			40	40	40	40	40	40
DisableAllInterrupts			30	30	30	30	30	30
ResumeAllInterrupts			92	92	92	92	92	92
SuspendAllInterrupts			46	46	46	46	46	46
ResumeOSInterrupts			92	92	92	92	92	92
SuspendOSInterrupts			62	62	62	62	62	62
GetResource	Task	7	428	428	396	428	428	396
	Combined	6	440	440	440	440	440	440
	CLEx	3	354	354	354	354	354	354
ReleaseResource	Task	7	382	382	382	382	382	382
	Combined	6	382	382	382	382	382	382
	CLEx	3	336	336	336	336	336	336

Configuration			**Application Uses**					
Events			**No**			**Yes**		
Shared Task Priorities			**No**		**Yes**	**No**		**Yes**
Multiple Task Activations			**No**	**Yes**		**No**	**Yes**	
SetEvent	SW	1	n/a	n/a	n/a	332	332	486
	NS		n/a	n/a	n/a	390	390	548
	NS1i	10	n/a	n/a	n/a	228	n/a	n/a
	KL	2	n/a	n/a	n/a	246	246	396
	KL1i	2, 10	n/a	n/a	n/a	170	n/a	n/a
ClearEvent			n/a	n/a	n/a	184	184	184
GetEvent			n/a	n/a	n/a	246	246	246
WaitEvent	<default>		n/a	n/a	n/a	404	404	660
	fp	11	n/a	n/a	n/a	440	440	736
	1i	10	n/a	n/a	n/a	162	n/a	n/a
GetAlarmBase			176	176	176	176	176	176
GetAlarm			170	170	170	170	170	170
SetRelAlarm			218	218	218	218	218	218
SetAbsAlarm			246	246	246	246	246	246
CancelAlarm			154	154	154	154	154	154
InitCounter			170	170	170	170	170	170
GetCounterValue			200	200	200	200	200	200
osek_tick_alarm	<default>		108	108	108	108	108	108
	KL	2	44	44	44	44	44	44
osek_incr_counter			52	52	52	52	52	52
GetActiveApplicationMode		31	n/a	n/a	n/a	n/a	n/a	n/a
StartOS			206	206	206	206	206	206
ShutdownOS	NoHook	12	42	42	42	42	42	42
	Hook	13	48	48	48	48	48	48
InitCOM			4	4	4	4	4	4
CloseCOM			4	4	4	4	4	4
StartCOM			36	36	36	36	36	36
StopCOM			42	42	42	42	42	42
ReadFlag			28	28	28	28	28	28
ResetFlag			26	26	26	26	26	26
ReceiveMessage	CCCA	14	156	156	156	334	334	334
	CCCB	15	334	334	334	334	334	334
GetMessageResource			88	88	88	88	88	88
ReleaseMessageResource			88	88	88	88	88	88
GetMessageStatus			100	100	100	100	100	100
SendMessage	SW CCCA	1, 14	194	194	194	380	380	380
	SW CCCB	1, 15	364	364	364	380	380	380
	NS CCCA	14	194	194	194	380	380	380
	NS CCCB	15	364	364	364	380	380	380
	KL CCCA	2, 14	134	134	134	320	320	320
	KL CCCB	2, 15	304	304	304	320	320	320

Configuration			**Application Uses**					
Events			**No**			**Yes**		
Shared Task Priorities			**No**		**Yes**	**No**		**Yes**
Multiple Task Activations			**No**	**Yes**		**No**	**Yes**	
main_dispatch	NoHook	12	260	260	328	260	260	328
	Hook	13	304	304	372	304	304	372
sub_dispatch	B1LI	19	4	4	4	4	4	4
	B1LF	20	12	12	12	12	12	12
	B1HI	21	118	118	118	118	118	118
	B1HF	22	126	126	126	126	126	126
	B2LI	23	n/a	76	112	n/a	76	112
	B2LF	24	n/a	84	120	n/a	84	120
	B2HI	25	n/a	160	292	n/a	160	292
	B2HF	26	n/a	168	300	n/a	168	300
	E1HI	27	n/a	n/a	n/a	458	458	590
	E1HF	28	n/a	n/a	n/a	466	466	598
	E2HI	29	n/a	n/a	n/a	n/a	n/a	590
	E2HF	30	n/a	n/a	n/a	n/a	n/a	598
CAT2_wrapper			206	206	206	206	206	206
hook_support	No ServiceID	16	120	120	120	120	120	120
	No Parameters	17	130	130	130	130	130	130
		18	190	190	190	190	190	190
fp_support			154	154	154	154	154	154
utility_functions	common		0	0	0	0	0	0
	optional	32	26	26	26	26	26	26
	optional	32	66	66	66	66	66	66
validity_checks		3	32	32	32	32	32	32
Timing_dispatch		4	84	84	84	84	84	84
Timing_termination		4	132	132	132	132	132	132
ActivateTaskset	SW	1	372	476	610	412	516	650
	NS		400	488	596	428	528	666
	KL	2	260	342	464	296	382	534
ChainTaskset	SWL	1, 8	438	532	626	458	558	678
	SWH	1, 9	498	586	706	514	614	754
	NSL	8	494	582	686	514	604	742
	NSH	9	542	634	752	558	662	812
GetTasksetRef			132	132	132	132	132	132
MergeTaskset			290	290	290	290	290	290
AssignTaskset			204	204	204	204	204	204
RemoveTaskset			292	292	292	292	292	292
TestSubTaskset			284	284	284	284	284	284
TestEquivalentTaskset			274	274	274	274	274	274
TickSchedule	SW	1	416	302	302	302	302	302
	NS		482	410	410	410	410	410
	KL	2	350	232	232	232	232	232

Configuration			**Application Uses**					
Events			**No**			**Yes**		
Shared Task Priorities			**No**		**Yes**	**No**		**Yes**
Multiple Task Activations			**No**	**Yes**		**No**	**Yes**	
AdvanceSchedule	SW	1	398	280	280	280	280	280
	NS		464	390	390	390	390	390
	KL	2	332	222	222	222	222	222
StartSchedule			236	236	236	236	236	236
StopSchedule			178	178	178	178	178	178
GetScheduleStatus			224	224	224	224	224	224
GetScheduleValue			172	172	172	172	172	172
GetScheduleNext			102	102	102	102	102	102
SetScheduleNext			174	174	174	174	174	174
GetArrivalpointDelay			126	126	126	126	126	126
SetArrivalpointDelay			164	164	164	164	164	164
GetArrivalpointTasksetRef			124	124	124	124	124	124
GetArrivalpointNext			130	130	130	130	130	130
SetArrivalpointNext			208	208	208	208	208	208
TestArrivalpointWritable			150	150	150	150	150	150
GetExecutionTime			146	146	146	146	146	146
GetLargestExecutionTime			98	98	98	98	98	98
ResetLargestExecutionTime			92	92	92	92	92	92
GetStackOffset			24	24	24	24	24	24

Notes

Number	Note
1	Linked only if upward activations are allowed.
2	Linked only if API is called within ISR.
3	Present only in Extended OS status.
4	Present only in Timing or Extended OS status.
5	Linked only if there are heavyweight tasks in the system.
6	Linked only if Resource is used by both tasks and ISRs.
7	Linked only if Resource is used only by tasks.
8	Linked only if Chaining task is Lightweight.
9	Linked only if Chaining task is Heavyweight.
10	Linked only if Idle task is the only extended task in the system.
11	Linked only if calling Extended task uses floating point.
12	Linked only if neither Pre- nor Post-TaskHook is used.
13	Linked only if Pre- or Post-TaskHook is used.
14	Linked only if there are no flags, message queues, or message resources in the system, and COM status is not requested.
15	Linked only if there are any flags, message queues, or message resources in the system, or COM status is requested.

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

4.2.4 Reserved Hardware Resources

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

4.3 Performance

4.3.1 Execution Times for SSX5 API Calls

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

Standard

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
Service	Variant						
ActivateTask	SW	116	161	216	124	152	240
	NS	99	144	199	107	135	218
	KL	71	122	174	79	112	192
TerminateTask	Lext	0	0	0	0	0	0
	H	195	197	200	195	197	201
ChainTask	SWL	317	371	492	402	440	573
	SWH	406	460	577	491	524	658
	NSL	317	371	492	402	440	573
	NSH	398	452	569	483	516	650
Schedule	SW	99	99	118	99	99	118
GetTaskID		47	47	47	47	47	47
GetTaskState		91	91	91	118	118	118
EnableAllInterrupts		22	22	22	22	22	22
DisableAllInterrupts		17	17	17	17	17	17
ResumeAllInterrupts		32	32	32	32	32	32
SuspendAllInterrupts		29	29	29	29	29	29
ResumeOSInterrupts		32	32	32	32	32	32
SuspendOSInterrupts		29	29	29	29	29	29
GetResource	Task	67	67	70	67	67	70
	Combined	72	72	72	72	72	72
	CLEx	n/a	n/a	n/a	n/a	n/a	n/a
ReleaseResource	Task	77	77	77	77	77	77
	Combined	101	101	101	101	101	101
	CLEx	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	n/a	n/a	n/a	124	124	127
	NS	n/a	n/a	n/a	115	115	127
	KL	n/a	n/a	n/a	101	101	108
ClearEvent		n/a	n/a	n/a	47	47	47
GetEvent		n/a	n/a	n/a	63	63	63
WaitEvent	<default>	n/a	n/a	n/a	529	531	623
	fp	n/a	n/a	n/a	540	542	643
GetAlarmBase		132	132	132	132	132	132
GetAlarm		93	93	93	93	93	93
SetRelAlarm		99	99	99	99	99	99
SetAbsAlarm		97	97	97	97	97	97
CancelAlarm		68	68	68	68	68	68
InitCounter		68	68	68	68	68	68

Configuration		**Application Uses**					
Events		**No**			**Yes**		
Shared Task Priorities		**No**		**Yes**	**No**		**Yes**
Multiple Task Activations		**No**	**Yes**		**No**	**Yes**	
GetCounterValue		79	79	79	79	79	79
osek_tick_alarm	<default>	75	75	75	75	75	75
	KL	50	50	50	50	50	50
osek_incr_counter		15	15	15	15	15	15
GetActiveApplicationMode		9	9	9	9	9	9
StartOS		902	902	902	902	902	902
ShutdownOS	NoHook	n/a	n/a	n/a	n/a	n/a	n/a
	Hook	32	32	32	32	32	32
InitCOM		11	11	11	11	11	11
CloseCOM		11	11	11	11	11	11
StartCOM		41	41	41	178	178	178
StopCOM		19	19	19	19	19	19
ReadFlag		n/a	n/a	n/a	14	14	14
ResetFlag		n/a	n/a	n/a	11	11	11
ReceiveMessage		63	63	63	245	245	245
GetMessageResource		n/a	n/a	n/a	133	133	133
ReleaseMessageResource		n/a	n/a	n/a	133	133	133
GetMessageStatus		n/a	n/a	n/a	67	67	67
SendMessage	SW	188	233	288	362	390	478
	NS	171	216	271	345	373	456
	KL	121	172	224	294	327	407
ActivateTaskset	SW	80	385	631	93	387	686
	NS	63	374	618	72	376	664
	KL	41	350	590	50	352	637
	SW2	80	385	631	93	387	686
	NS2	63	374	618	72	376	664
	KL2	41	350	590	50	352	637
ChainTaskset	SWL	309	603	874	386	665	1010
	SWH	403	699	959	480	761	1061
	NSL	309	603	874	386	665	1010
	NSH	395	691	951	472	753	1053
GetTasksetRef		28	28	28	28	28	28
MergeTaskset		67	67	67	67	67	67
AssignTaskset		25	25	25	25	25	25
RemoveTaskset		68	68	68	68	68	68
TestSubTaskset		88	88	88	88	88	88
TestEquivalentTaskset		81	81	81	81	81	81
TickSchedule	SW	178	500	740	200	518	803
	NS	161	481	721	181	499	784
	KL	133	465	705	165	483	768

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
TickSchedule	SW2	178	500	740	200	502	787
	NS2	161	481	721	181	483	768
	KL2	133	465	705	165	467	752
AdvanceSchedule	SW	147	471	711	171	489	774
	NS	122	452	692	152	470	755
	KL	107	428	668	128	446	731
	SW2	147	471	711	171	473	758
	NS2	122	452	692	152	454	739
	KL2	107	428	668	128	430	715
StartSchedule		100	100	100	100	100	100
StopSchedule		78	78	78	78	78	78
GetScheduleStatus		93	93	93	93	93	93
GetScheduleValue		83	83	83	83	83	83
GetScheduleNext		29	29	29	29	29	29
SetScheduleNext		28	28	28	28	28	28
GetArrivalpointDelay		22	22	22	22	22	22
SetArrivalpointDelay		21	21	21	21	21	21
GetArrivalpointTasksetRef		21	21	21	21	21	21
GetArrivalpointNext		26	26	26	26	26	26
SetArrivalpointNext		25	25	25	25	25	25
TestArrivalpointWritable		45	45	45	45	45	45
GetExecutionTime		11	11	11	11	11	11
GetLargestExecutionTime		28	28	28	28	28	28
ResetLargestExecutionTime		17	17	17	17	17	17
GetStackOffset		29	29	29	29	29	29

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	116	161	216	124	152	240
	NS	99	144	199	107	135	218
	KL	71	122	174	79	112	192
TerminateTask	Lext	0	0	0	0	0	0
	H	395	397	400	395	397	401

Configuration		**Application Uses**					
Events		**No**			**Yes**		
Shared Task Priorities		**No**		**Yes**	**No**		**Yes**
Multiple Task Activations		**No**	**Yes**		**No**	**Yes**	
ChainTask	SWL	555	607	728	640	678	811
	SWH	637	689	806	722	755	889
	NSL	555	607	728	640	678	811
	NSH	618	670	787	703	736	870
Schedule	SW	99	99	118	99	99	118
GetTaskID		47	47	47	47	47	47
GetTaskState		91	91	91	118	118	118
EnableAllInterrupts		22	22	22	22	22	22
DisableAllInterrupts		17	17	17	17	17	17
ResumeAllInterrupts		32	32	32	32	32	32
SuspendAllInterrupts		29	29	29	29	29	29
ResumeOSInterrupts		32	32	32	32	32	32
SuspendOSInterrupts		29	29	29	29	29	29
GetResource	Task	67	67	70	67	67	70
	Combined	72	72	72	72	72	72
	CLEx	n/a	n/a	n/a	n/a	n/a	n/a
ReleaseResource	Task	77	77	77	77	77	77
	Combined	114	114	114	114	114	114
	CLEx	n/a	n/a	n/a	n/a	n/a	n/a
SetEvent	SW	n/a	n/a	n/a	124	124	127
	NS	n/a	n/a	n/a	115	115	127
	KL	n/a	n/a	n/a	101	101	108
ClearEvent		n/a	n/a	n/a	47	47	47
GetEvent		n/a	n/a	n/a	63	63	63
WaitEvent	<default>	n/a	n/a	n/a	734	736	823
	fp	n/a	n/a	n/a	745	747	843
GetAlarmBase		132	132	132	132	132	132
GetAlarm		93	93	93	93	93	93
SetRelAlarm		99	99	99	99	99	99
SetAbsAlarm		97	97	97	97	97	97
CancelAlarm		68	68	68	68	68	68
InitCounter		68	68	68	68	68	68
GetCounterValue		79	79	79	79	79	79
osek_tick_alarm	<default>	75	75	75	75	75	75
	KL	50	50	50	50	50	50
osek_incr_counter		15	15	15	15	15	15
GetActiveApplicationMode		9	9	9	9	9	9
StartOS		1717	1717	1717	1717	1717	1717
ShutdownOS	NoHook	n/a	n/a	n/a	n/a	n/a	n/a
	Hook	32	32	32	32	32	32

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
InitCOM		11	11	11	11	11	11
CloseCOM		11	11	11	11	11	11
StartCOM		41	41	41	178	178	178
StopCOM		19	19	19	19	19	19
ReadFlag		n/a	n/a	n/a	14	14	14
ResetFlag		n/a	n/a	n/a	11	11	11
ReceiveMessage		63	63	63	245	245	245
GetMessageResource		n/a	n/a	n/a	133	133	133
ReleaseMessageResource		n/a	n/a	n/a	146	146	146
GetMessageStatus		n/a	n/a	n/a	67	67	67
SendMessage	SW	188	233	288	362	390	478
	NS	171	216	271	345	373	456
	KL	121	172	224	294	327	407
ActivateTaskset	SW	80	385	631	93	387	686
	NS	63	374	618	72	376	664
	KL	41	350	590	50	352	637
	SW2	80	385	631	93	387	686
	NS2	63	374	618	72	376	664
	KL2	41	350	590	50	352	637
ChainTaskset	SWL	536	839	1110	613	903	1248
	SWH	634	928	1188	700	992	1292
	NSL	547	839	1110	613	903	1248
	NSH	615	909	1169	692	973	1273
GetTasksetRef		28	28	28	28	28	28
MergeTaskset		67	67	67	67	67	67
AssignTaskset		25	25	25	25	25	25
RemoveTaskset		68	68	68	68	68	68
TestSubTaskset		88	88	88	88	88	88
TestEquivalentTaskset		81	81	81	81	81	81
TickSchedule	SW	178	500	740	200	518	803
	NS	161	481	721	181	499	784
	KL	133	465	705	165	483	768
	SW2	178	500	740	200	502	787
	NS2	161	481	721	181	483	768
	KL2	133	465	705	165	467	752
AdvanceSchedule	SW	147	471	711	171	489	774
	NS	122	452	692	152	470	755
	KL	107	428	668	128	446	731
	SW2	147	471	711	171	473	758
	NS2	122	452	692	152	454	739
	KL2	107	428	668	128	430	715

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
StartSchedule		100	100	100	100	100	100
StopSchedule		78	78	78	78	78	78
GetScheduleStatus		93	93	93	93	93	93
GetScheduleValue		83	83	83	83	83	83
GetScheduleNext		29	29	29	29	29	29
SetScheduleNext		28	28	28	28	28	28
GetArrivalpointDelay		22	22	22	22	22	22
SetArrivalpointDelay		21	21	21	21	21	21
GetArrivalpointTasksetRef		21	21	21	21	21	21
GetArrivalpointNext		26	26	26	26	26	26
SetArrivalpointNext		25	25	25	25	25	25
TestArrivalpointWritable		45	45	45	45	45	45
GetExecutionTime		102	102	102	102	102	102
GetLargestExecutionTime		32	32	32	32	32	32
ResetLargestExecutionTime		29	29	29	29	29	29
GetStackOffset		29	29	29	29	29	29

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	283	328	403	294	319	425
	NS	313	368	444	324	362	466
	KL	239	291	341	250	282	381
TerminateTask	Lext	409	411	412	409	411	412
	H	481	483	484	481	483	484
ChainTask	SWL	784	840	956	872	902	1043
	SWH	860	916	1040	948	978	1130
	NSL	827	883	1006	915	945	1093
	NSH	895	951	1073	983	1013	1160
Schedule	SW	151	151	166	151	151	166
GetTaskID		57	57	57	57	57	57
GetTaskState		273	273	273	283	283	283
EnableAllInterrupts		32	32	32	32	32	32
DisableAllInterrupts		27	27	27	27	27	27
ResumeAllInterrupts		54	54	54	54	54	54

Configuration		**Application Uses**					
Events		**No**			**Yes**		
Shared Task Priorities		**No**		**Yes**	**No**		**Yes**
Multiple Task Activations		**No**	**Yes**		**No**	**Yes**	
SuspendAllInterrupts		39	39	39	39	39	39
ResumeOSInterrupts		54	54	54	54	54	54
SuspendOSInterrupts		39	39	39	39	39	39
GetResource	Task	512	512	322	542	542	352
	Combined	275	275	275	305	305	305
	CLEx	296	296	296	326	326	326
ReleaseResource	Task	289	289	289	319	319	319
	Combined	287	287	287	317	317	317
	CLEx	261	261	261	291	291	291
SetEvent	SW	n/a	n/a	n/a	298	298	299
	NS	n/a	n/a	n/a	325	325	328
	KL	n/a	n/a	n/a	279	279	280
ClearEvent		n/a	n/a	n/a	96	96	96
GetEvent		n/a	n/a	n/a	273	273	273
WaitEvent	<default>	n/a	n/a	n/a	796	798	873
	fp	n/a	n/a	n/a	807	809	884
GetAlarmBase		274	274	274	274	274	274
GetAlarm		211	211	211	211	211	211
SetRelAlarm		240	240	240	240	240	240
SetAbsAlarm		238	238	238	238	238	238
CancelAlarm		186	186	186	186	186	186
InitCounter		185	185	185	185	185	185
GetCounterValue		199	199	199	199	199	199
osek_tick_alarm	<default>	92	92	92	92	92	92
	KL	50	50	50	50	50	50
osek_incr_counter		15	15	15	15	15	15
GetActiveApplicationMode		9	9	9	9	9	9
StartOS		1823	1823	1823	1823	1823	1823
ShutdownOS	NoHook	n/a	n/a	n/a	n/a	n/a	n/a
	Hook	37	37	37	37	37	37
InitCOM		11	11	11	11	11	11
CloseCOM		11	11	11	11	11	11
StartCOM		56	56	56	193	193	193
StopCOM		34	34	34	34	34	34
ReadFlag		n/a	n/a	n/a	37	37	37
ResetFlag		n/a	n/a	n/a	28	28	28
ReceiveMessage		170	170	170	343	343	343
GetMessageResource		n/a	n/a	n/a	483	483	483
ReleaseMessageResource		n/a	n/a	n/a	468	468	468
GetMessageStatus		n/a	n/a	n/a	150	150	150

Configuration		**Application Uses**					
Events		**No**			**Yes**		
Shared Task Priorities		**No**		**Yes**	**No**		**Yes**
Multiple Task Activations		**No**	**Yes**		**No**	**Yes**	
SendMessage	SW	462	507	582	627	652	758
	NS	492	547	623	657	695	799
	KL	393	445	495	554	586	685
ActivateTaskset	SW	898	1394	2037	931	1390	1256
	NS	610	835	1174	532	831	1227
	KL	444	764	1111	467	760	1193
	SW2	898	1394	2037	931	1390	1256
	NS2	610	835	1174	532	831	1227
	KL2	444	764	1111	467	760	1193
ChainTaskset	SWL	1446	1780	2073	1537	1838	2179
	SWH	1518	1837	2213	1605	1896	2282
	NSL	1480	1810	2109	1571	1866	2248
	NSH	1544	1865	2304	1631	1924	2447
GetTasksetRef		219	219	219	219	219	219
MergeTaskset		169	169	169	169	169	169
AssignTaskset		142	142	142	142	142	142
RemoveTaskset		170	170	170	170	170	170
TestSubTaskset		180	180	180	180	180	180
TestEquivalentTaskset		167	167	167	167	167	167
TickSchedule	SW	276	940	1287	643	966	1399
	NS	318	1000	1347	703	1026	1459
	KL	239	901	1248	604	927	1360
	SW2	276	940	1287	643	936	1369
	NS2	318	1000	1347	703	996	1429
	KL2	239	901	1248	604	897	1330
AdvanceSchedule	SW	247	913	1260	616	939	1372
	NS	289	970	1317	673	996	1429
	KL	210	884	1231	587	910	1343
	SW2	247	913	1260	616	909	1342
	NS2	289	970	1317	673	966	1399
	KL2	210	884	1231	587	880	1313
StartSchedule		141	141	141	141	141	141
StopSchedule		109	109	109	109	109	109
GetScheduleStatus		135	135	135	135	135	135
GetScheduleValue		121	121	121	121	121	121
GetScheduleNext		85	85	85	85	85	85
SetScheduleNext		113	113	113	113	113	113
GetArrivalpointDelay		94	94	94	94	94	94
SetArrivalpointDelay		95	95	95	95	95	95
GetArrivalpointTasksetRef		93	93	93	93	93	93

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
GetArrivalpointNext		98	98	98	98	98	98
SetArrivalpointNext		127	127	127	127	127	127
TestArrivalpointWritable		107	107	107	107	107	107
GetExecutionTime		118	118	118	118	118	118
GetLargestExecutionTime		200	200	200	200	200	200
ResetLargestExecutionTime		190	190	190	190	190	190
GetStackOffset		29	29	29	29	29	29

4.3.2 OS Start-up Time

OS start-up time is the time from the entry to the `StartOS` function to the execution of the first instruction in a user task (including the idle task) without any hook routines being called. This time is always application dependant, since `StartOS` may activate any number of tasks and start any number of user specified alarms.

4.3.3 Interrupt Latencies

The interrupt latency is the time between an interrupt request being recognized by the target hardware and the execution of the first instruction of the user provided handler function. The following table gives interrupt latencies in CPU cycles.

Standard

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations		No	Yes		No	Yes	
Operation	ISR Category						
ISR Latency	Cat 1	22	22	22	22	22	22
	Cat 2	52	52	52	52	52	52

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	22	22	22	22	22	22
	Cat 2	178	178	178	178	178	178

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	22	22	22	22	22	22
	Cat 2	178	178	178	178	178	178

4.3.4 Task Switching Times

The task switching time is the time between the last instruction of the previous task and the first instruction of the next task. The switching time is different for different switching contexts (e.g. an `ActivateTask` versus a `ChainTask`).

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

The task switching time is the time between the last instruction of the previous task and the first instruction of the next task. The switching time is different for different switching. Figures 1 through 8 show the SSX5 switching contexts measured.

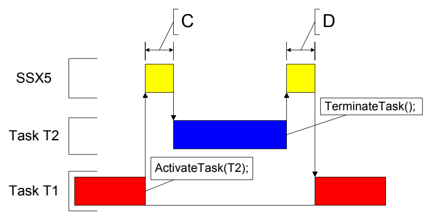

Figure 1: Task Activates a Higher Priority Task which Terminates Normally

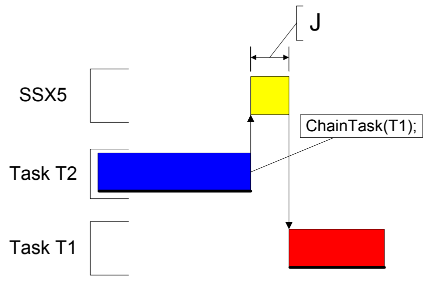

Figure 2: Task Chaining

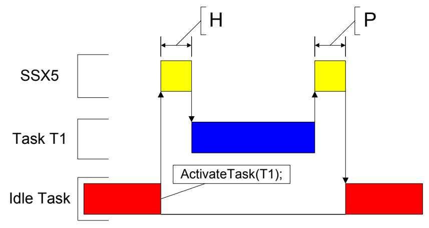

Figure 3: Task Activation from Idle Task

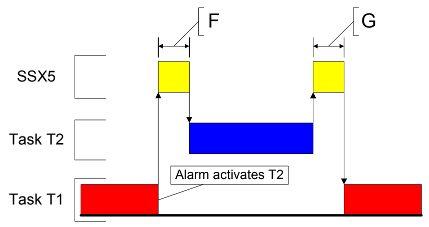

Figure 4: Task Activation from an Alarm

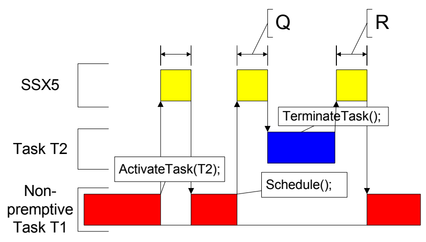

Figure 5: Non-Premptive Task Calls Schedule()

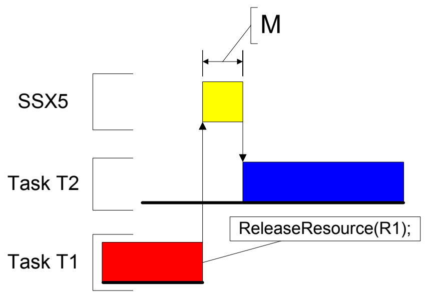

Figure 6: Blocked Task Activated by ReleaseResource()

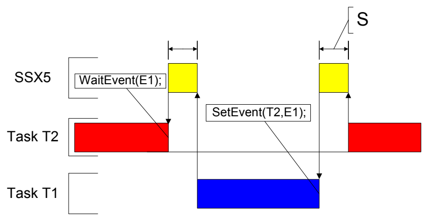

Figure 7: Waiting Task Activated by SetEvent()

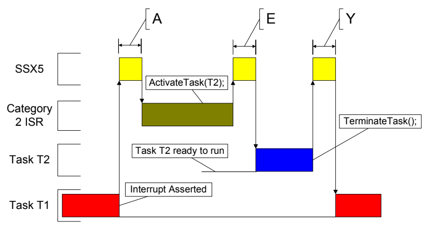

Figure 8: Category 2 ISR Activates a Higher Priority Task

Standard

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations	Task Attributes	No	Yes		No	Yes	
Normal termination	Light, Basic	96	133	170	95	132	170
Figure 1: D	Heavy, Basic/Extended	193	220	262	243	245	286
ChainTask	Light, Basic	242	309	429	249	318	455
Figure 2: J	Heavy, Basic/Extended	546	640	798	604	670	849
Pre-emption	Light, Basic	209	273	390	217	273	422
Figure 1: C	Heavy, Basic/Extended	314	365	483	399	425	570
From idle task	Light, Basic	208	272	389	216	272	421
Figure 3: H	Heavy, Basic/Extended	313	364	482	398	424	569
Triggered by alarm	Light, Basic	300	364	481	308	364	513
Figure 4: F	Heavy, Basic/Extended	405	456	574	490	516	661
Schedule	Light, Basic	177	198	279	177	198	279
Figure 5: Q	Heavy, Basic/Extended	282	290	372	359	359	436
Release resource	Light, Basic	191	212	276	191	212	276
Figure 6: M	Heavy, Basic/Extended	296	304	369	373	373	433
SetEvent							
Figure 7: S	Heavy, Extended	n/a	n/a	n/a	635	635	827
From category 2 ISR	Light, Basic	158	179	243	158	179	243
Figure 8: E	Heavy, Basic/Extended	263	271	336	340	340	400

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	303	335	373	302	334	373
Figure 1: D	Heavy, Basic/Extended	393	415	456	440	442	478
ChainTask	Light, Basic	494	547	667	501	556	693
Figure 2: J	Heavy, Basic/Extended	1002	1077	1234	1046	1109	1283
Pre-emption	Light, Basic	351	401	518	359	401	550
Figure 1: C	Heavy, Basic/Extended	442	491	609	527	553	698
From idle task	Light, Basic	350	400	517	358	400	549
Figure 3: H	Heavy, Basic/Extended	441	490	608	526	552	697
Triggered by alarm	Light, Basic	442	492	609	450	492	641
Figure 4: F	Heavy, Basic/Extended	533	582	700	618	644	789
Schedule	Light, Basic	319	326	407	319	326	407
Figure 5: Q	Heavy, Basic/Extended	410	416	498	487	487	564

Configuration		Application Uses					
Events		No			Yes		
Shared Task Priorities		No		Yes	No		Yes
Multiple Task Activations	Task Attributes	No	Yes		No	Yes	
Release resource	Light, Basic	341	348	412	341	348	412
Figure 6: M	Heavy, Basic/Extended	432	438	503	509	509	569
SetEvent							
Figure 7: S	Heavy, Extended	n/a	n/a	n/a	746	746	938
From category 2 ISR	Light, Basic	492	499	563	492	499	563
Figure 8: E	Heavy, Basic/Extended	583	589	654	660	660	720

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	395	427	463	394	426	462
Figure 1: D	Heavy, Basic/Extended	479	501	540	526	528	561
ChainTask	Light, Basic	722	779	894	732	778	920
Figure 2: J	Heavy, Basic/Extended	1310	1389	1551	1368	1416	1600
Pre-emption	Light, Basic	512	562	701	523	562	732
Figure 1: C	Heavy, Basic/Extended	604	653	793	692	715	881
From idle task	Light, Basic	511	561	700	522	561	731
Figure 3: H	Heavy, Basic/Extended	603	652	792	691	714	880
Triggered by alarm	Light, Basic	620	670	809	631	670	840
Figure 4: F	Heavy, Basic/Extended	712	761	901	800	823	989
Schedule	Light, Basic	360	367	446	360	367	446
Figure 5: Q	Heavy, Basic/Extended	452	458	538	529	529	604
Release resource	Light, Basic	528	535	599	558	565	629
Figure 6: M	Heavy, Basic/Extended	620	626	691	727	727	787
SetEvent							
Figure 7: S	Heavy, Extended	n/a	n/a	n/a	931	931	1115
From category 2 ISR	Light, Basic	508	515	579	508	515	579
Figure 8: E	Heavy, Basic/Extended	600	606	671	677	677	737

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. This causes run-time contexts of mutually exclusive tasks to be effectively overlaid. RTArchitect can calculate the worst-case stack requirement for the entire application based on the declared stack usage, priorities and resource occupation of individual tasks.

The size of the 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		48	48	48	48	48	48
BCC1 lightweight, floating point		52	52	52	52	52	52
BCC1 heavyweight, integer		94	94	94	94	94	94
BCC1 heavyweight, floating point		94	94	94	94	94	94
BCC2 lightweight, integer		n/a	56	64	n/a	56	64
BCC2 lightweight, floating point		n/a	56	64	n/a	56	64
BCC2 heavyweight, integer		n/a	98	106	n/a	98	106
BCC2 heavyweight, floating point		n/a	98	106	n/a	98	106
ECC1 heavyweight, integer		n/a	n/a	n/a	140	140	140
ECC1 heavyweight, floating point		n/a	n/a	n/a	140	140	140
ECC2 heavyweight, integer		n/a	n/a	n/a	n/a	n/a	142
ECC2 heavyweight, floating point		n/a	n/a	n/a	n/a	n/a	142
Pre- and/or Post-Task hooks used							
Task type							
BCC1 lightweight, integer		48	48	48	48	48	48
BCC1 lightweight, floating point		52	52	52	52	52	52
BCC1 heavyweight, integer		94	94	94	94	94	94
BCC1 heavyweight, floating point		94	94	94	94	94	94
BCC2 lightweight, integer		n/a	56	64	n/a	56	64
BCC2 lightweight, floating point		n/a	56	64	n/a	56	64
BCC2 heavyweight, integer		n/a	98	106	n/a	98	106
BCC2 heavyweight, floating point		n/a	98	106	n/a	98	106
ECC1 heavyweight, integer		n/a	n/a	n/a	140	140	140
ECC1 heavyweight, floating point		n/a	n/a	n/a	140	140	140
ECC2 heavyweight, integer		n/a	n/a	n/a	n/a	n/a	142
ECC2 heavyweight, floating point		n/a	n/a	n/a	n/a	n/a	142

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	80	80	80	80	80	80
BCC1 lightweight, floating point	84	84	84	84	84	84
BCC1 heavyweight, integer	122	122	122	122	122	122
BCC1 heavyweight, floating point	122	122	122	122	122	122
BCC2 lightweight, integer	n/a	88	96	n/a	88	96
BCC2 lightweight, floating point	n/a	88	96	n/a	88	96
BCC2 heavyweight, integer	n/a	126	134	n/a	126	134
BCC2 heavyweight, floating point	n/a	126	134	n/a	126	134
ECC1 heavyweight, integer	n/a	n/a	n/a	168	168	168
ECC1 heavyweight, floating point	n/a	n/a	n/a	168	168	168
ECC2 heavyweight, integer	n/a	n/a	n/a	n/a	n/a	172
ECC2 heavyweight, floating point	n/a	n/a	n/a	n/a	n/a	172
Pre- and/or Post-Task hooks used						
Task type						
BCC1 lightweight, integer	80	80	80	80	80	80
BCC1 lightweight, floating point	84	84	84	84	84	84
BCC1 heavyweight, integer	122	122	122	122	122	122
BCC1 heavyweight, floating point	122	122	122	122	122	122
BCC2 lightweight, integer	n/a	88	96	n/a	88	96
BCC2 lightweight, floating point	n/a	88	96	n/a	88	96
BCC2 heavyweight, integer	n/a	126	134	n/a	126	134
BCC2 heavyweight, floating point	n/a	126	134	n/a	126	134
ECC1 heavyweight, integer	n/a	n/a	n/a	168	168	168
ECC1 heavyweight, floating point	n/a	n/a	n/a	168	168	168
ECC2 heavyweight, integer	n/a	n/a	n/a	n/a	n/a	172
ECC2 heavyweight, floating point	n/a	n/a	n/a	n/a	n/a	172

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		80	80	80	80	80	80
BCC1 lightweight, floating point		84	84	84	84	84	84
BCC1 heavyweight, integer		122	122	122	122	122	122
BCC1 heavyweight, floating point		122	122	122	122	122	122
BCC2 lightweight, integer		n/a	88	96	n/a	88	96
BCC2 lightweight, floating point		n/a	88	96	n/a	88	96
BCC2 heavyweight, integer		n/a	126	134	n/a	126	134
BCC2 heavyweight, floating point		n/a	126	134	n/a	126	134
ECC1 heavyweight, integer		n/a	n/a	n/a	164	164	164
ECC1 heavyweight, floating point		n/a	n/a	n/a	164	164	164
ECC2 heavyweight, integer		n/a	n/a	n/a	n/a	n/a	168
ECC2 heavyweight, floating point		n/a	n/a	n/a	n/a	n/a	168
Pre- and/or Post-Task hooks used							
Task type							
BCC1 lightweight, integer		80	80	80	80	80	80
BCC1 lightweight, floating point		84	84	84	84	84	84
BCC1 heavyweight, integer		122	122	122	122	122	122
BCC1 heavyweight, floating point		122	122	122	122	122	122
BCC2 lightweight, integer		n/a	88	96	n/a	88	96
BCC2 lightweight, floating point		n/a	88	96	n/a	88	96
BCC2 heavyweight, integer		n/a	126	134	n/a	126	134
BCC2 heavyweight, floating point		n/a	126	134	n/a	126	134
ECC1 heavyweight, integer		n/a	n/a	n/a	164	164	164
ECC1 heavyweight, floating point		n/a	n/a	n/a	164	164	164
ECC2 heavyweight, integer		n/a	n/a	n/a	n/a	n/a	168
ECC2 heavyweight, floating point		n/a	n/a	n/a	n/a	n/a	168

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