embOS
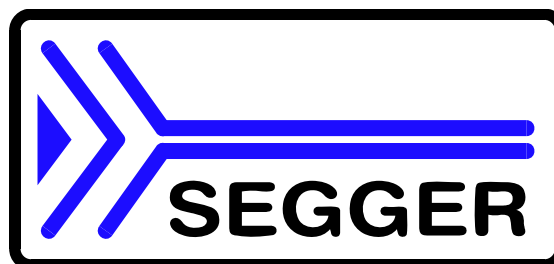# C-SPY Plug-in

**embOS plug-in for the
IAR C-SPY® Debugger**

**Version 3.82 + 6.0.5 + 6.0.6
Document UM01025
Revision 15**

Date: May 29, 2013

SEGGER

A product of SEGGER Microcontroller GmbH & Co. KG

Disclaimer

Specifications written in this document are believed to be accurate, but are not guaranteed to be entirely free of error. The information in this manual is subject to change for functional or performance improvements without notice. Please make sure your manual is the latest edition. While the information herein is assumed to be accurate, SEGGER MICROCONTROLLER GmbH & Co. KG (the manufacturer) assumes no responsibility for any errors or omissions. The manufacturer makes and you receive no warranties or conditions, express, implied, statutory or in any communication with you. The manufacturer specifically disclaims any implied warranty of merchantability or fitness for a particular purpose.

Copyright notice

Trademarks

Names mentioned in this manual may be trademarks of their respective companies. Brand and product names are trademarks or registered trademarks of their respective holders.

Contact address

SEGGER Microcontroller GmbH & Co. KG
In den Weiden 11
D-40721 Hilden
Germany
Tel.+49 2103-2878-0
Fax.+49 2103-2878-28
Email: support@segger.com
Internet: http://www.segger.com

Manual versions

Manual version	Date	By	Explanation
15.0	130527	AW	New software version number 6.0.6.10. List of supported CPUs corrected.
14.0	120321	AW	New software version number 6.0.6.9.
13.0	111118	AW	New software version 6.0.6.8, Task sensitivity for Cortex M4F with floating point unit.
12.0	110614	AW	New software version 6.0.6.5, Task sensitivity for RL78 added.
11.0	110408	AW	New software version 6.0.6.4, Cortex R4F support.
10.0	110323	TS	New embOS library mode DPL added.
9.0	110303	AW	New software version 6.0.6 listed.
8.0	101228	AW	New software version 3.82.4 listed.
7.0	100708	AW	New software version 3.82.3 and 6.0.5 listed.
6.0	100528	AW	New software version 6.0.4 listed. Description of task sensitivity added.
5.0	100408	AW	New software version 6.0.1 listed
4.0	091030	AW	Software version list updated.
3.0	091007	AW	Software version list updated.
2.0	081208	AW	Screenshot of installation folder updated.
1.0	070707	TW	Initial version for plugin V2.

Software versions

Software version	Date	By	Explanation
6.0.6.10 3.82.6	130529	AW	Modification required to show task sensitive call stack for embOS CortexM version 3.88b or above, because of modified stack layout. embOS version number shown in system information window for embOS versions V3.88b and above.
6.0.6.9	120321	AW	embOS timer list generation corrected. The previous version may have crashed the debugger when embOS timer were used in the application.
6.0.6.8	110614	AW	embOS timer list display corrected.
6.0.6.7	110614	AW	Full task sensitive call stack for Cortex M4F including floating point unit and registers.
6.0.6.6	110614	AW	Support for Cortex M4F added.
6.0.6.5	110614	AW	Full task sensitive call stack trace for RENESAS RL78 added.
6.0.6.4	110408	AW	New corrected plugin version for EWARM V6. Previous versions did not show the task sensitive call stack for ARM big endian targets.
6.0.6.3	110303	AW	New corrected plugin version for EWARM V6. Previous versions did not show the task sensitive call stack correctly.
3.82.5	110303	AW	New corrected plugin version for EWARM V5. Previous versions did not show the task sensitive call stack correctly.
6.0.6.2	110227	AW	RENESAS RL78 supported.
3.82.4	101228	AW	Plugin initialization modified to fix a problem that inhibited the start of the plugin. Under some circumstances, the OS_TASK structure could not be found.
6.0.5 + 3.82.3	100708	AW	Display errors under Windows 7 corrected. Debug and trace library mode now shown in the system information window. Several new error messages (OS_Status) in system information window added.
6.0.4	100528	AW	Big endian support for Renesas RX addded
6.0.3	100527	AW	Support for Renesas RX added for new IAR EW6
6.0.2	100413	AW	Support for SH2A added
6.0.1	100408	AW	New plugin for IAR EW 6
3.82.2	091030	AW	Task awareness for Renesas RX added. Task awareness for Renesas R32C added.

Software version	Date	By	Explanation
3.82.1	091026	AW	Version number modified to corresponding embOS version. J-Link memory read error fixed.
3.82.0	091007	AW	Version number modified to corresponding embOS version. J-Link memory read error fixed.
3.80.3	090911	AW	Task awareness for M16C added.
3.80.2	090606	AW	Task state display corrected.
3.80.1	090505	AW	Task awareness for Cortex-M3 added.
3.60.e	081208	AW	Version number modified to corresponding embOS version.
2.00e	070707	AW	Initial version for IAR Embedded Workbench V5

Table of Contents

2

Chapter 1

Introduction

This chapter gives a short overview about the embOS C-Spy plug-in for IAR Embedded Workbench®.

1.1   Overview

1.1.1   embOS C-Spy Plug-in for IAR Embedded Workbench

SEGGER's embOS C-Spy plug-in for IAR Embedded Workbench provides embOS-awareness during debugging sessions. This enables you to inspect the state of several embOS primitives such as the task list, resource semaphores, mailboxes, embOS software timers and major system variables.

1.1.2   embOS

embOS is a real-time operating system for embedded applications designed to offer the benefits of a fully-fledged multitasking system at minimum cost. The kernel is fully interruptible and so efficient that embOS can be used in very time critical situations. The memory footprint in both RAM and ROM is so small that embOS can be used in single-chip applications, leaving maximum room for the user-program.

1.1.3   IAR Embedded Workbench

IAR Embedded Workbench is a set of development tools for building and debugging embedded applications using assembler, C and C++. It provides a completely integrated development environment that includes a project manager, editor, build tools and the C-SPY debugger. IAR Embedded Workbench supports a wide range of microcontrollers and cores from different chip manufacturers. It offers the same intuitive user interface regardless of which microcontroller you have chosen to work with—coupled with general and target-specific support for each chip.

1.2   Requirements

To use the embOS C-Spy plug-in you need a version of IAR Embedded Workbench installed and a debug target which uses embOS. Specifically:

- An embOS version 3.28 or higher is required for complete compatibility. Older embOS versions use different internal structures and the C-Spy plug-in is therefore of limited use with version prior to 3.28.
- An IAR Embedded Workbench IDE with a C-SPY debugger version 5 is required for the embOS plug-in V2 and the plug-in V3.
- An IAR Embedded Workbench IDE with a C-Spy debugger version V6 or higher is required for the plug-in V6.
  The version 6.0.6 of the embOS plugin is required for
  EWARM V6.10 or above
  EW430 V5.20 or above
  EWRX V2.20 or above
  EWRL78
  The version V6.0.5 of the plug-in is for all other IAR embedded Worbenches V6.

1.3   Supported CPUs

The embOS C-Spy plug-in works with 8-bit, 16-bit or 32-bit CPUs in little- or big-endian mode supported by embOS.

Due to limited testing, support can only be granted for the CPUs listed below:

- Any ARM7 / ARM9 CPU
- Any ARM Cortex M0 CPU
- Any ARM Cortex M3 CPU
- Any ARM Cortex M4/M4F CPU
- Atmel AVR / ATMega
- National CR16C
- Renesas H8/H8S
- Renesas M16C
- Renesas M16C80
- Renesas M32C
- Renesas R32C
- Renesas R8C
- Renesas RL78
- Renesas RX
- Renesas SH2A
- Renesas (NEC) V850 / V850E / V850ES / V850E2
- Renesas (NEC) 78/K0 / 78/K0R
- TI MSP430 / MSP430x

The task sensitive source window, call stack and register window is supported for the following CPUs:

- Any ARM7 / ARM9 CPU
- Any ARM Cortex M0 CPU
- Any ARM Cortex M3 CPU
- Any ARM Cortex M4/M4F CPU
- Renesas M16C
- Renesas R32C
- Renesas RL78
- Renesas RX
- Renesas SH2A

Others will follow.

Chapter 2

Installation

This chapter describes the installation steps required to use the embOS C-Spy plug-in.

2.1   Installation Procedure

Normally there is no installation required, because the IAR Embedded Workbench comes with the plug-in already installed. If for some reason you want to uptade the plug-in, you have to replace two files.

The installation procedure is very straightforward because it only requires you to copy the contents of the embOS C-Spy plug-in package into the IAR CPU specific plug-in folder for `rtos` plug-ins. The directory structure may look like this:
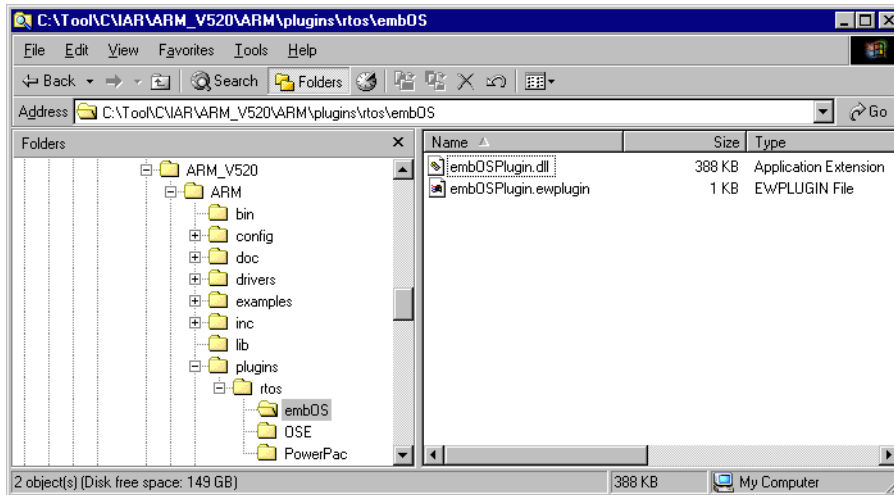
If not already delivered with the IAR Embedded Workbench IDE, create a directory `embOS` below the CPU specific `plugin\rtos\` folder and copy the files from the embOS folder which comes with the plugin into that folder in your IAR installation directory. Then restart the IAR Embedded Workbench IDE.

2.2 Configuration

By default, the embOS C-Spy plug-in is not loaded during debugging. For each project configuration you have to explicitly enable the plug-in in the debugger section of the project options:
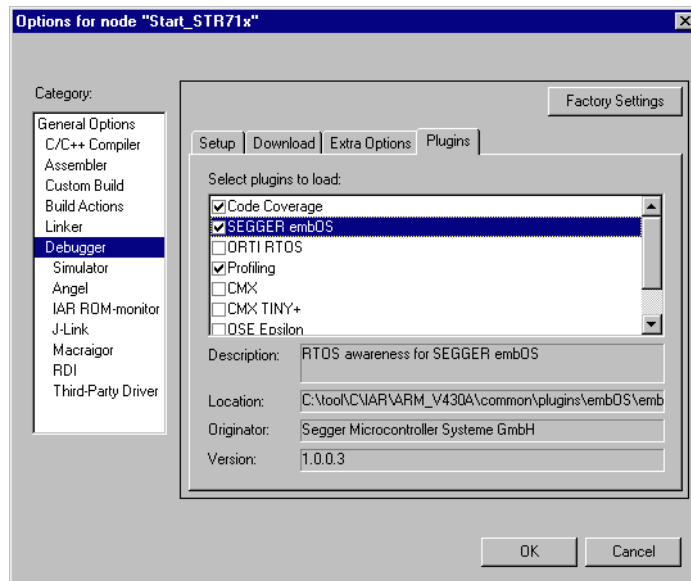
The embOS C-Spy plug-in is now available in debugging sessions and may be accessed from the main menu.
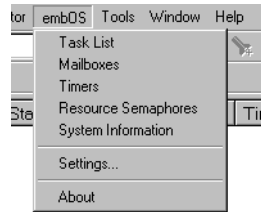
Chapter 3

Getting started

This chapter describes the embOS C-Spy plug-in and its capabilities in greater detail.

3.1   Overview

During your debugging session, the embOS C-Spy plug-in is accessible from the IAR
Embedded Workbench IDE **main** menu. Note that if you are not running a debugging
session, there is no **embOS menu** item available.

From the menu you may activate the individual windows that provide embOS related
information. The sections below describe these individual windows. The amount of
information available depends on the embOS build used during debugging. If a cer-
tain part is not available, the respective menu item is either greyed out or the win-
dow column shows a **N/A**.

3.2   Task list

The **Task List** window lists all current embOS tasks. It retrieves its information directly from the embOS task list.

The individual columns are described below:

Column	Description
*	A green arrow points at the currently active embOS task.
Prio	The priority of the task.
Id	The task control block address that uniquely identifies a task.
Name	If available, the task name is shown here.
Status	The task status as a short text.
Timeout	If a task is delayed, this column shows the timeout value and in parentheses the point in time when the delay will be finished.
Stack Info	If available, this column shows the amount of used stack space, and the available stack space, as well as the value of the current stack bottom pointer.
Run count	The number of task activations.
Time slice	If round robin scheduling is available, this column shows the number of remaining time slices and the number of time slice reloads.
Events	The event mask of a task.

Table 3.1: Task list window items

3.2.1   Task sensitivity

The **Source Code** window, the **Disassembly** window, the **Register** window, and the **Call Stack** window of the C-Spy debugger are task sensitive since version 3.62 of the embOS C-Spy plug-in for several CPUs. This means that they show the position in the code, the general-purpose registers and the call stack of the selected task. By default, the selected task is always the running task, which is the normal behavior of a debugger that the user expects.

You can examine a particular thread by double-clicking on the corresponding row in the window. The selected task will be underlayed in yellow. The C-Spy Debugger rebuilds the call stack and the preserved general-purpose registers of a suspended task. Refer to *State of suspended tasks* on page 14 for detailed information about which information are available for the different task states.

Every time the CPU is started or when the Idle-row of the task window is double clicked, the selected task is switched back to this default.

3.2.1.1   State of suspended tasks

Blocked tasks (suspended by cooperative task switch)

Tasks which have given up execution voluntarily by calling a blocking function, such as `OS_Delay()` or `OS_Wait_...()`. In this case, there was no need for the OS to save the scratch registers (in case of ARM R0-R3, R12).
The **Register** window will show "`----------`" for the content of these registers.

*	Prio	Id	Name	Status	Timeout	Stack Info	Run count	Time slice	Events
	113	0x209FF8	IP_RxTask	Waiting (event object)		240 / 512 @ 0x2096F0	48	0 / 2	0x0
	112	0x209FB4	IP_Task		11 (14190)	492 / 768 @ 0x2093F0	1339	0 / 2	0x0
➡	107	0x209F70	IP_WebServer	Ready		288 / 8192 @ 0x206130	1102	0 / 2	0x0
	106	0x209C20	IP_WebserverChild	Waiting (event object)		1848 / 2400 @ 0x208130	45	0 / 2	0x0
	25	0x20A03C	BlinkTask	Ready		96 / 128 @ 0x209D28	1365	0 / 2	0x0
	Idle								

Register — CPU Registers
```
R0        = ----------
R1        = ----------
R2        = ----------
R3        = ----------
R4        = 0x00000000
R5        = 0x0020861C
R6        = 0x00000000
R7        = 0x00000200
R8        = 0xCCCC0008
R9        = 0xCCCC0009
R10       = 0xCCCC000A
R11       = 0xCCCC000B
R12       = ----------
R13 (SP)  = 0x002085E0
R14 (LR)  = 0x00007F03
CPSR      = 0x0000003F
SPSR      = 0xFFFFFFFF
PC        = 0x00007F02
R8_fiq    = 0x00000000
R9_fiq    = 0x00000000
```

Call Stack
```
➡ OS_Deactivated ( )
  OS_DeactivateP ( 0x0020861C, 'H' (0x48) )
  OS_EVENT_Wait ( 0x0020861C )
  IP_OS_WaitItemTimed ( 0x00204E90, 0 )
  sbwait ( 0x00204E90, 0 )
  soreceive ( 0x00204E64, _LocaleC_isalpha(int) (0x0), 0x2
  t_recv ( 2117220, 0x208860 "GET /favicon.ico HTTP/1.1□
  _Recv ( 0x208860 "GET /favicon.ico HTTP/1.1□□Host: 19
  _Read ( 0x00208718 )
  _ReadLine ( 0x00208718 )
  _Process ( 0x00208700 )
  IP_WEBS_Process ( 0x0000992D, 0x00009945, 0x00204E
  _WebServerChildTask ( 0x00204E64 )
  [OS_ReturnFromTask + 0]
```

Tasks waiting for first activation

These basically fall into the same category as blocked tasks, the call stack and registers look similar to the following screenshots. Similarly, temporary registers are unknown. The **Call Stack** shows a single entry **OS_StartTask**. **Run count** is 0.

*	Prio	Id	Name	Status	Timeout	Stack Info	Run count	Time slice	Events
	113	0x209FF8	IP_RxTask	Waiting (event object)		240 / 512 @ 0x2096F0	1	0 / 2	0x0
	112	0x209FB4	IP_Task		11 (16)	240 / 768 @ 0x2093F0	1	0 / 2	0x0
➡	100	0x209F70	MainTask	Ready		288 / 8192 @ 0x206130	3	0 / 2	0x0
	25	0x20A03C	BlinkTask	Ready		44 / 128 @ 0x209D28	0	0 / 2	0x0
	Idle								

Register — CPU Registers
```
R0        = ----------
R1        = ----------
R2        = ----------
R3        = ----------
R4        = 0xCCCC0004
R5        = 0xCCCC0005
R6        = 0xCCCC0006
R7        = 0xCCCC0007
R8        = 0xCCCC0008
R9        = 0xCCCC0009
R10       = 0xCCCC000A
R11       = 0xCCCC000B
R12       = ----------
R13 (SP)  = 0x00209DA4
R14 (LR)  = 0x00010824
CPSR      = 0x0000001F
SPSR      = 0xFFFFFFFF
PC        = 0x00010824
R8_fiq    = 0x00000000
R9_fiq    = 0x00000000
```

Call Stack
```
➡ [OS_StartTask + 0]
```

Interrupted tasks

Tasks which have been interrupted and preempted, typically by a task with higher priority becoming ready. In this case, the OS saved all registers, including the scratch registers (in case of ARM R0-R3, R12). The **Register** window shows the values of all registers, including the scratch registers.

*	Prio	Id	Name	Status	Timeout	Stack Info	Run count	Time slice	Events
	113	0x209FF8	IP_RxTask	Waiting (event object)		240 / 512 @ 0x2096F0	48	0 / 2	0x0
	112	0x209FB4	IP_Task		11 (14190)	492 / 768 @ 0x2093F0	1339	0 / 2	0x0
→	107	0x209F70	IP_WebServer	Ready		288 / 8192 @ 0x206130	1102	0 / 2	0x0
	106	0x209C20	IP_WebserverChild	Waiting (event object)		1848 / 2400 @ 0x208130	45	0 / 2	0x0
	25	0x20A03C	BlinkTask	Ready		96 / 128 @ 0x209D28	1365	0 / 2	0x0
			Idle						

Register

CPU Registers

```
R0        = 0x00700001
R1        = 0x00000000
R2        = 0x00000013
R3        = 0x00000000
R4        = 0x00000000
R5        = 0xCCCC0005
R6        = 0xCCCC0006
R7        = 0xCCCC0007
R8        = 0xCCCC0008
R9        = 0xCCCC0009
R10       = 0xCCCC000A
R11       = 0xCCCC000B
R12       = 0x000135E0
R13 (SP)  = 0x00209D90
R14 (LR)  = 0x00009FC9
⊞ CPSR    = 0x6000003F
⊞ SPSR    = 0xFFFFFFFF
PC        = 0x0001134E
R8_fiq    = 0x00000000
R9_fiq    = 0x00000000
```

Call Stack

```
↱ BSP_ClrLED(int)
⇨ BSP_ToggleLED ( 0 )
  _ToggleLED ( )
  _BlinkTask ( )
  [OS_ReturnFromTask + 0]
```

3.2.1.2   Call stack with embOS libraries

All embOS libraries are built with full optimization. Therefore it may happen that not all function calls are shown in the call stack in detail. The additional embOS library *dpl.a is built with low optimization. It may be used for application development instead of the Debug and Profiling library.

Using a library built with low optimization level gives the ability to see the complete detailed call stack.

*	Prio	Id	Name	Status	Timeout	Stack Info	Run count	Time slice	Events
	100	0x20000510	HP Task	Delay	19 (3675)	144 / 512 @ 0x2000003C	147	0 / 2	0x0
→	50	0x20000558	LP Task	Waiting (semaphore zero)		144 / 512 @ 0x2000023C	147	0 / 2	0x0
			Idle						

Call stack with DP library   Call stack with DPL library

Call Stack

```
⇨ [OS_DelayUntil + 0x75]
  HPTask ( )
  [OS_StartTask + 0x7]
```

Call Stack

```
⇨ [OS_Deactivated + 0x15]
  [OS_DelayUntil + 0x51]
  [OS_Delay + 0xb]
  HPTask ( )
  [OS_StartTask + 0x7]
```

3.3   Mailboxes

A mailbox is a buffer that is managed by the real-time operating system. The buffer behaves like a normal buffer; you can put something (called a message) in and retrieve it later. This window shows the mailboxes and provides information about the number of messages, waiting tasks etc.

Column	Description
Id	The mailbox address.
Messages	The number of messages in a mailbox and the maximum number of messages the mailbox can hold.
Message size	The size of an individual message in bytes.
pBuffer	The message buffer address.
Waiting tasks	The list of tasks that are waiting for a mailbox, that is their address and name.

Table 3.2: Mailboxes window items

3.4   Timers

A software timer is an object that calls a user-specified routine after a specified delay. This window provides information about active software timers.
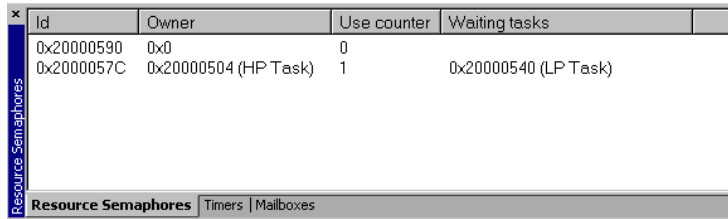
Column	Description
Id	The timer's address.
Hook	The function (address and name) that is called after the timeout.
Time	The time delay and the point in time, when the timer finishes waiting.
Period	The time period the timer runs.
Active	Shows whether the timer is active (running) or not.

Table 3.3: Timers window items

3.5   Resource semaphores

Resource semaphores are used to manage resources by avoiding conflicts caused by simultaneous use of a resource. This window provides information about available resources.
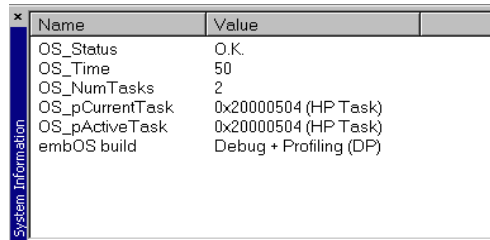

Column	Description
Id	The resource semaphore address.
Owner	The address and name of the owner task.
Use counter	Counts the number of semaphore uses.
Waiting tasks	Lists the tasks (address and name) that are waiting at the sema-phore.

Table 3.4: Resource Semaphores window items

3.6   System information

A running embOS contains a number of system variables that are available for inspection. This window lists the most important ones.

| Name	Value	
OS_Status	O.K.	
OS_Time	50	
OS_NumTasks	2	
OS_pCurrentTask	0x20000504 (HP Task)	
OS_pActiveTask	0x20000504 (HP Task)	
embOS build	Debug + Profiling (DP)	

3.7   Settings

To be safe, the embOS C-Spy plug-in imposes certain limits on the amount of infor-
mation   retrieved from the target, to avoid endless requests in case of false values in
the target memory. This dialog box allows you to tweak these limits in a certain
range, for example if your task names are no longer than 32 characters you may set
the **Maximum string length** to 32, or if they are longer than the default you may
increase that value.
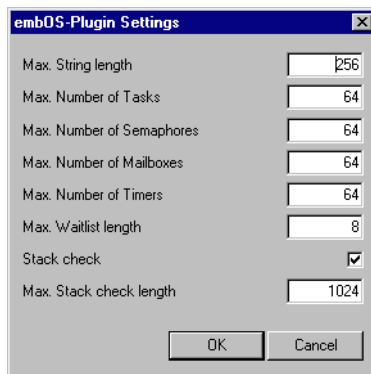
After changing settings and clicking the **OK** button, your changes are applied imme-
diately and should become noticeable after the next window update, for example
when hitting the next breakpoint. However, the settings are restored to their default
values on plug-in reload.

3.8   About

Finally, the **About** dialog box contains the embOS C-Spy plug-in version number and the date of compilation.
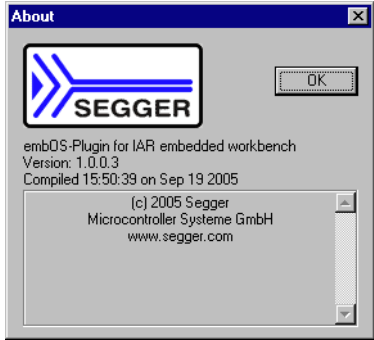
Chapter 4

Support

This chapter contains information about contacting support and what information to provide.

4.1   Contacting Support

We work hard to avoid as much software defects as possible. However, if you encoun-
ter an error in our software, you may contact our support at `support@segger.com`.
We will try to correct any malfunction as soon as possible. To do this, we need all rel-
evant information. Please try to provide us with at least the following information:

• IAR Embedded Workbench IDE & C-SPY debugger versions.
• Information about the target CPU.
• embOS C-Spy plug-in version number.
• A detailled description of the problem and how to reproduce it.
• If possible send us a project that triggers the problem.