

# embOS

Real-Time Operating System

embOS plug-in for IAR C-Spy Debugger

Document: UM01025

Software Version: 3.0

Revision: 0

Date: September 18, 2017



A product of SEGGER Microcontroller GmbH & Co. KG

[www.segger.com](http://www.segger.com)

## 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 (SEGGER) assumes no responsibility for any errors or omissions. SEGGER makes and you receive no warranties or conditions, express, implied, statutory or in any communication with you. SEGGER specifically disclaims any implied warranty of merchantability or fitness for a particular purpose.

## Copyright notice

You may not extract portions of this manual or modify the PDF file in any way without the prior written permission of SEGGER. The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such a license.

© 2005-2017 SEGGER Microcontroller GmbH & Co. KG, Hilden / Germany

## 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  
E-mail:       support@segger.com  
Internet:     www.segger.com

## Manual versions

This manual describes the current software version. If you find an error in the manual or a problem in the software, please inform us and we will try to assist you as soon as possible. Contact us for further information on topics or functions that are not yet documented.

Print date: September 18, 2017

Software	Revision	Date	By	Description
3.0	0	170915	MM	New plug-in versions 6.10.3.0, 7.10.3.0, 7.50.3.0, and 8.10.3.0.



# About this document

---

## Assumptions

This document assumes that you already have a solid knowledge of the following:

- The software tools used for building your application (assembler, linker, C compiler).
- The C programming language.
- The target processor.
- DOS command line.

If you feel that your knowledge of C is not sufficient, we recommend *The C Programming Language* by Kernighan and Richie (ISBN 0-13-1103628), which describes the standard in C programming and, in newer editions, also covers the ANSI C standard.

## How to use this manual

This manual explains all the functions and macros that the product offers. It assumes you have a working knowledge of the C language. Knowledge of assembly programming is not required.

## Typographic conventions for syntax

This manual uses the following typographic conventions:

Style	Used for
Body	Body text.
Keyword	Text that you enter at the command prompt or that appears on the display (that is system functions, file- or pathnames).
Parameter	Parameters in API functions.
Sample	Sample code in program examples.
Sample comment	Comments in program examples.
Reference	Reference to chapters, sections, tables and figures or other documents.
GUI Element	Buttons, dialog boxes, menu names, menu commands.
Emphasis	Very important sections.



# Table of contents

---

1	Introduction .....	8
1.1	Overview .....	9
1.2	Supported CPUs .....	10
2	Installation .....	11
2.1	Installation Procedure .....	12
2.2	Configuration .....	13
3	Getting Started .....	14
3.1	Overview .....	15
3.2	Task List .....	16
3.3	Timers .....	18
3.4	Mailboxes .....	19
3.5	Queues .....	20
3.6	Resource Semaphores .....	21
3.7	Counting Semaphores .....	22
3.8	Memory Pools .....	23
3.9	Event Objects .....	24
3.10	Watchdogs .....	25
3.11	System Information .....	26
3.12	Settings .....	27
3.13	About .....	28
4	Support .....	29
4.1	Contacting Support .....	30

# 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 debug sessions. This enables you to inspect the state of several embOS primitives such as the task list, queues, counting semaphores, resource semaphores, mailboxes, software timers, memory pools, event objects, watchdogs 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 Supported CPUs

The embOS C-Spy plug-in works with 8-bit, 16-bit or 32-bit CPUs in little- or big-endian mode. To use the embOS C-Spy plug-in you need a version of IAR Embedded Workbench installed and a debug target which uses embOS.

The following plug-ins are available and may be used with the listed versions of IAR's Embedded Workbench:

embOS Port	IAR Embedded Workbench Version	Compatible Plug-In version
<b>78K0</b>	≤ 4.80 ≥ 4.81	6.10.3.0 7.10.3.0
<b>8051</b>	≤ 8.30 ≥ 9.10 and ≤ 9.30 ≥ 10.10	6.10.3.0 7.10.3.0 8.10.3.0
<b>ARM7/9 Cortex-M</b>	≤ 6.70 ≥ 7.10 and ≤ 7.40 ≥ 7.50 and ≤ 7.80 ≥ 8.10	6.10.3.0 7.10.3.0 7.50.3.0 8.10.3.0
<b>AVR</b>	≤ 6.40 ≥ 6.50 and ≤ 6.80 ≥ 7.10	6.10.3.0 7.10.3.0 8.10.3.0
<b>AVR32</b>	≤ 4.21 ≥ 4.30	6.10.3.0 7.10.3.0
<b>Coldfire</b>	Any	3.82.3.0
<b>CR16C</b>	≤ 3.20 ≥ 3.30	6.10.3.0 7.10.3.0
<b>H8 M32C</b>	Any	6.0.1.0
<b>M16C</b>	≤ 3.60 ≥ 3.70	6.10.3.0 7.10.3.0
<b>MSP430</b>	≤ 5.60 ≥ 6.10 and ≤ 6.50 ≥ 7.10	6.10.3.0 7.10.3.0 8.10.3.0
<b>R32C SH</b>	Any	6.10.3.0
<b>RH850</b>	≤ 1.30 ≥ 1.40	7.10.3.0 7.50.3.0
<b>RL78</b>	≤ 1.30 ≥ 1.40	6.10.3.0 7.10.3.0
<b>RX</b>	≤ 2.50 ≥ 2.60 and ≤ 2.90 ≥ 3.10	6.10.3.0 7.10.3.0 8.10.3.0
<b>STM8</b>	≤ 1.42 ≥ 2.10	6.10.3.0 7.10.3.0
<b>V850</b>	≤ 4.10 ≥ 4.20	6.10.3.0 7.10.3.0

# Chapter 2

## Installation

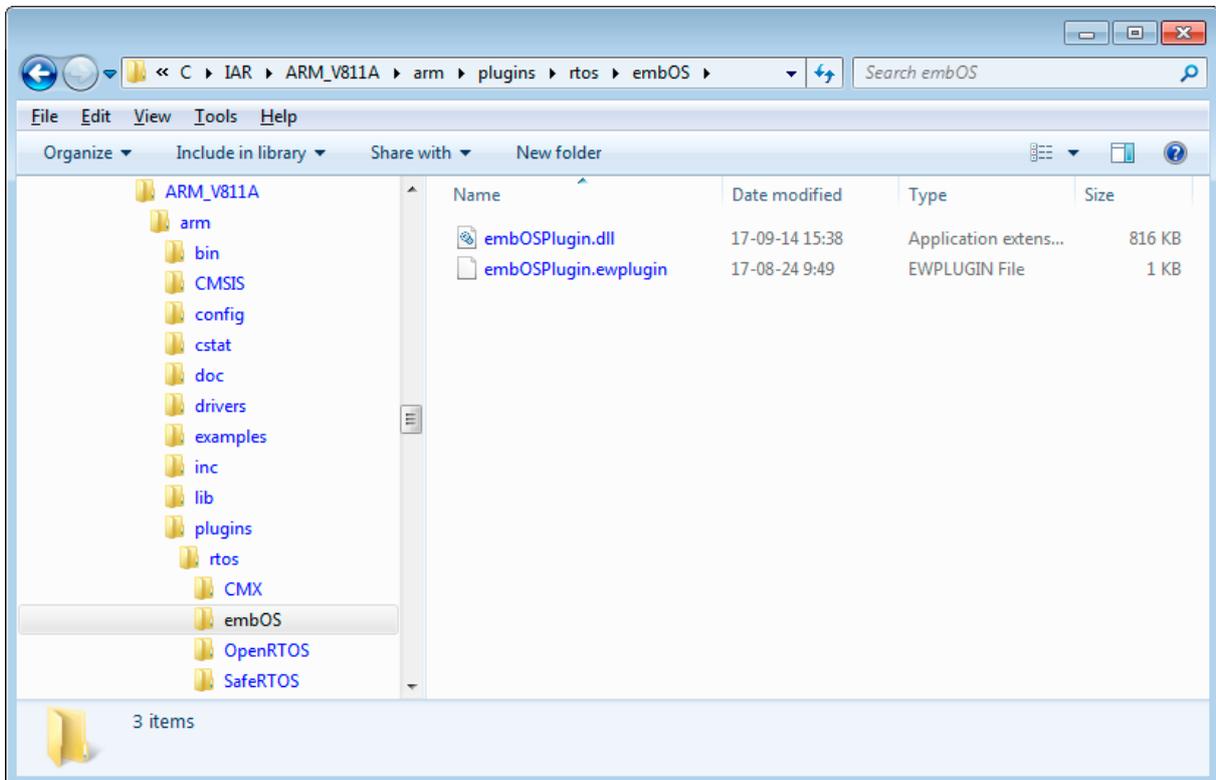
---

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 update 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.

### Note

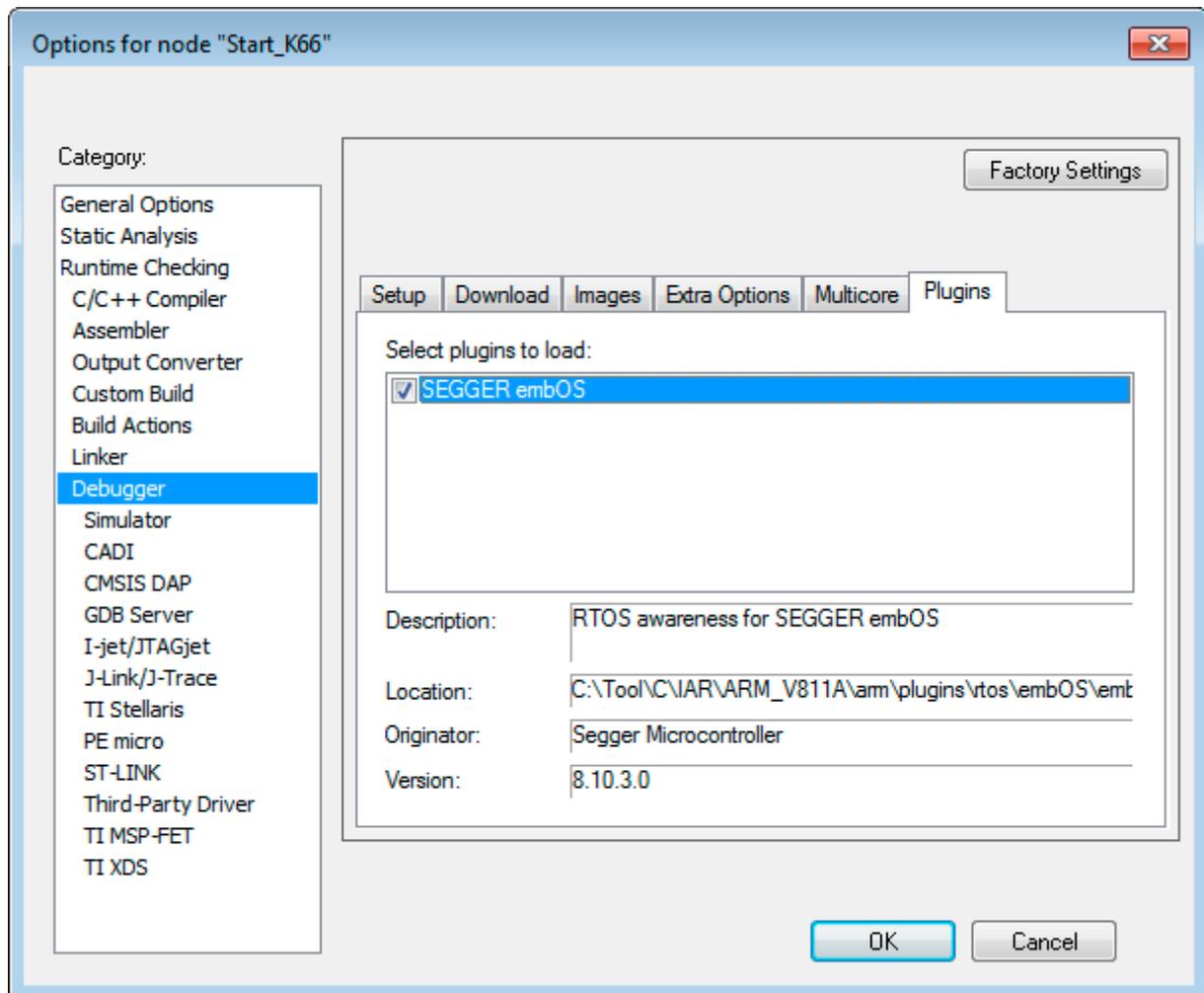
Before replacing any files already found in the `plugin\rtos\embOS` folder of the IAR embedded workbench, you should make a backup copy of these files. You should also check the version number of the plugin. To check the version number, you may open the file `embOSPlugin.ewplugin` which is an xml text file. The version number is shown as last entry and looks like follows:

```
<version>8.10.3.0</version>
```

The first number is the major version number. Never replace a plugin with a different major version number. You may replace the plugin in the embedded workbench when the last number (revision) in the files found in the embedded workbench is lower than the number of the files delivered with `embOS`.

## 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

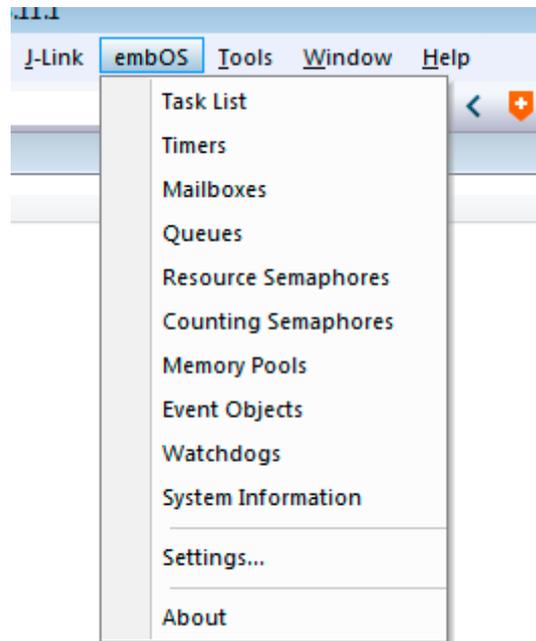
## Getting Started

---

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. A Release build, for instance, won't show any information about counting semaphores, queues, event objects and mailboxes.

## 3.2 Task List

* Prio	Id	Name	Status	Timeout	Stack Info	Run count	Time slice	Events
100	0x1FFF0878	HP Task	Waiting for space in Queue 0x1FFF0A08 (MyQueue)	2500 (16500)	176 / 512 @ 0x1FFF0078	8	0 / 2	0x0
75	0x1FFF0930	MP Task1	Waiting for C-Semaphore 0x1FFF0A58 (MyCSema)		152 / 512 @ 0x1FFF0478	1	2 / 2	0x0
75	0x1FFF08D4	MP Task0	Ready		168 / 512 @ 0x1FFF0278	9	0 / 2	0x0
50	0x1FFF098C	LP Task	Waiting for message in Mailbox 0x1FFF09E8 (MyMailbox)		160 / 512 @ 0x1FFF0678	1	0 / 2	0x0
	Idle							

Column	Description
<b>*</b>	A green arrow points at the currently active embOS task.
<b>Prio</b>	The priority of the task.
<b>Id</b>	The task control block address that uniquely identifies a task.
<b>Name</b>	If available, the task name is shown here.
<b>Status</b>	The task status as a short text. The name in the parentheses belongs to the object (semaphore, mailbox, etc.) the task is waiting for.
<b>Timeout</b>	If a task is delayed, this column shows the timeout value and in parentheses the point in time when the delay will be finished.
<b>Stack Info</b>	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.
<b>Run count</b>	The number of task activations.
<b>Time slice</b>	If round robin scheduling is available, this column shows the number of remaining time slices and the number of time slice reloads.
<b>Events</b>	The event mask of a task.

### 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 17 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.

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

- ARM7/9
- ARM Cortex-M0/M3/M4/M4F/M7
- Renesas M16C
- Renesas R32C
- Renesas RL78
- Renesas RX
- Renesas SH2A

### 3.2.1.1 State of suspended tasks

#### Blocked tasks

Tasks which have given up execution voluntarily by calling a blocking function, such as `OS_Delay()` or `OS_Wait_...()`. In this case, the OS saved all registers which can be viewed in the Register window. It might be that scratch registers are not saved and thus not displayed in the Register window.

The screenshot shows the Task List window with the following data:

* Prio	Id	Name	Status	Timeout	Stack Info	Run count	Time slice	Events
100	0x1FFF0878	HP Task	Waiting for space in Queue 0x1FFF0A08 (MyQueue)	549 (39000)	176 / 512 @ 0x1FFF0078	18	0 / 2	0x0
75	0x1FFF0930	MP Task1	Waiting for C-Semaphore 0x1FFF0A58 (MyCSema)		152 / 512 @ 0x1FFF0478	1	2 / 2	0x0
75	0x1FFF08D4	MP Task0	Delay	49 (38500)	168 / 512 @ 0x1FFF0278	22	0 / 2	0x0
50	0x1FFF098C	LP Task	Waiting for message in Mailbox 0x1FFF09E8 (MyMailbox)		160 / 512 @ 0x1FFF0678	1	0 / 2	0x0

The Registers 1 window shows the following register values:

Current CPU Registers	Value	Access
R0	0x1FFF0000	ReadWrite
R1	0xE000ED04	ReadWrite
R2	0x10000000	ReadWrite
R3	0x00000001	ReadWrite
R4	0x1FFF0000	ReadWrite
R5	0x00000000	ReadWrite
R6	0x00000080	ReadWrite
R7	0xCCCC0007	ReadWrite
R8	0xCCCC0008	ReadWrite
R9	0xCCCC0009	ReadWrite
R10	0xCCCC000A	ReadWrite
R11	0xCCCC000B	ReadWrite
R12	0xCDCDCDCD	ReadWrite
APSR	0x00000000	ReadWrite
IPSR	0x00000000	ReadWrite
EPSR	0x01000000	ReadWrite
PC	0x00002016	ReadWrite
SP	0x1FFF0630	ReadWrite
LR	0x00002017	ReadWrite
PRIMASK	0x00000000	ReadWrite

#### 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.

The screenshot shows the Task List window with the following data:

* Prio	Id	Name	Status	Timeout	Stack Info	Run count	Time slice	Events
100	0x1FFF0878	HP Task	Delay	500 (500)	160 / 512 @ 0x1FFF0078	1	0 / 2	0x0
75	0x1FFF0930	MP Task1	Ready		88 / 512 @ 0x1FFF0478	1	2 / 2	0x0
75	0x1FFF08D4	MP Task0	Ready		88 / 512 @ 0x1FFF0278	0	0 / 2	0x0
50	0x1FFF098C	LP Task	Ready		88 / 512 @ 0x1FFF0678	0	0 / 2	0x0

The Registers 1 window shows the following register values:

Current CPU Registers	Value	Access
R0	0xCDCDCDCD	ReadWrite
R1	0xCCCC0001	ReadWrite
R2	0xCCCC0002	ReadWrite
R3	0xCCCC0003	ReadWrite
R4	0xCCCC0004	ReadWrite
R5	0xCCCC0005	ReadWrite
R6	0xCCCC0006	ReadWrite
R7	0xCCCC0007	ReadWrite
R8	0xCCCC0008	ReadWrite
R9	0xCCCC0009	ReadWrite
R10	0xCCCC000A	ReadWrite
R11	0xCCCC000B	ReadWrite
R12	0xCDCDCDCD	ReadWrite
APSR	0x00000000	ReadWrite
IPSR	0x00000000	ReadWrite
EPSR	0x01000000	ReadWrite
PC	0x000025D5	ReadWrite
SP	0x1FFF0470	ReadWrite
LR	0xCDCDCDCD	ReadWrite
PRIMASK	0x00000000	ReadWrite

## 3.3 Timers

Timers						
Id	Name	Hook	Timeout	Period	Active	
0x1FFF0A30	MyTimer	0x46C8 (Timer50)	50 (550)	50	1	

Column	Description
Id	The timer's address.
Name	If available, the respective object identifier is shown here.
Hook	The function (address and name) that is called after the timeout.
Timeout	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.

## 3.4 Mailboxes

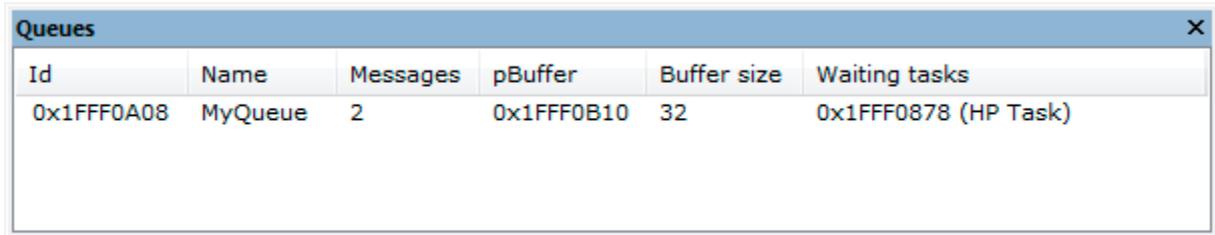
This view displays information in debug builds of embOS only.

Mailboxes					
Id	Name	Messages	Message size	pBuffer	Waiting tasks
0x1FFF09E8	MyMailbox	0 / 2	8	0x1FFF0B00	0x1FFF098C (LP Task)

Column	Description
Id	The mailbox address.
Name	If available, the respective object identifier is shown here.
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 (address and, if available, name). Only those tasks that are displayed in the task list window are shown here.

## 3.5 Queues

With embOS V4.38 and subsequent versions, this view displays information in debug builds of embOS only.



Id	Name	Messages	pBuffer	Buffer size	Waiting tasks
0x1FFF0A08	MyQueue	2	0x1FFF0B10	32	0x1FFF0878 (HP Task)

Column	Description
Id	The queue address.
Name	If available, the respective object identifier is shown here.
Messages	The number of messages in a queue.
pBuffer	Address of the buffer which contains the messages.
Buffer size	The message buffer address.
Waiting tasks	The list of tasks that are waiting for a mailbox (address and, if available, name). Only those tasks that are displayed in the task list window are shown here.

## 3.6 Resource Semaphores

Resource Semaphores				
Id	Name	Owner	Use counter	Waiting tasks
0x1FFF0A44	MyRSema	0x1FFF0878 (HP Task)	1	0x1FFF08D4 (MP Task0)

Column	Description
Id	The resource semaphore address.
Name	If available, the respective object identifier is shown here.
Owner	The address and name of the owner task.
Use counter	Counts the number of semaphore uses.
Waiting tasks	The list of tasks that are waiting for a semaphore (address and, if available, name). Only those tasks that are displayed in the task list window are shown here.

## 3.7 Counting Semaphores

This view displays information in debug builds of embOS only.

Counting Semaphores			
Id	Name	Count	Waiting tasks
0x1FFF0A58	MyCSema	0	0x1FFF0930 (MP Task1)

Column	Description
Id	The counting semaphore address.
Name	If available, the respective object identifier is shown here.
Count	Counts how often this semaphore can be claimed until it blocks.
Waiting tasks	The list of tasks that are waiting for a semaphore (address and, if available, name). Only those tasks that are displayed in the task list window are shown here.

## 3.8 Memory Pools

Memory Pools						
Id	Name	Total blocks	Block size	Max. usage	pPool	Waiting tasks
0x1FFF0A68	MyMemPool	7 / 8	16	1	0x1FFF0B30	

Column	Description
Id	The memory pool address.
Name	If available, the respective object identifier is shown here.
Total blocks	Shows the available blocks and the maximal number of blocks.
Block size	Shows the size of a single memory block.
Max. usage	Shows the maximal count of blocks which were simultaneously allocated.
pPool	The address of the memory pool buffer.
Waiting tasks	The list of tasks that are waiting for a memory pool (address and, if available, name). Only those tasks that are displayed in the task list window are shown here.

## 3.9 Event Objects

This view displays information in debug builds of embOS only. This view displays information with embOS V4.38 and subsequent versions only.

Event Objects						
Id	Name	Signaled	Reset Mode	Mask Mode	Waiting tasks	
0x1FFF0A8C	MyEvent	0x0	Semiauto	Or logic		

Column	Description
Id	The event object address.
Name	If available, the respective object identifier is shown here.
Signaled	The hexadecimal value of the bit mask containing the signaled event bits.
Reset Mode	The event objects reset mode.
Mask Mode	The current mask mode indicating whether Or or And logic is used to check whether a task shall resume.
Waiting tasks	The list of tasks that are waiting for an event object (address and, if available, name). Only those tasks that are displayed in the task list window are shown here.

## 3.10 Watchdogs

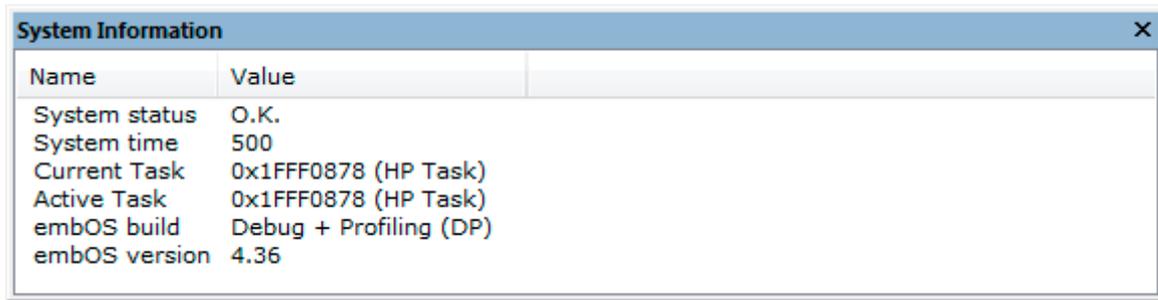
This view displays information with embOS V4.38 and subsequent versions only.

Watchdogs			
Id	Name	Timeout	Period
0x1FFF0C78	MyWatchdog	7815 (17000)	10000

Column	Description
Id	The watchdog address.
Name	If available, the respective object identifier is shown here.
Timeout	The remaining time (and the system time in parentheses) until the watchdog has to be fed.
Period	The period in which the watchdog has to be fed.

## 3.11 System Information

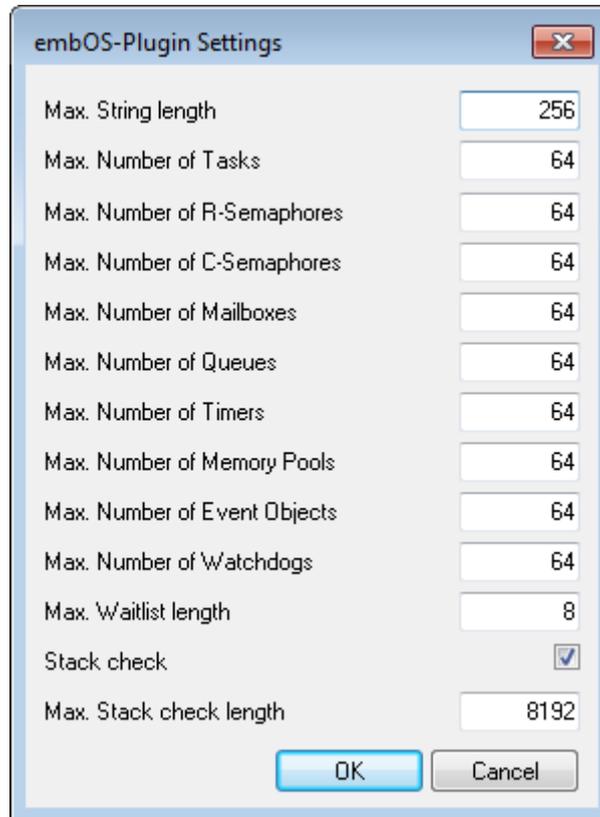
A running embOS contains a number of system variables that are available for inspection. This window lists the most important ones.



Name	Value
System status	O.K.
System time	500
Current Task	0x1FFF0878 (HP Task)
Active Task	0x1FFF0878 (HP Task)
embOS build	Debug + Profiling (DP)
embOS version	4.36

## 3.12 Settings

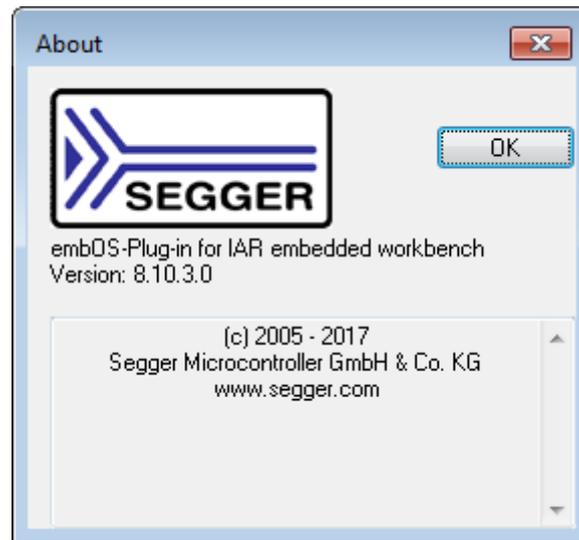
in order to avoid endless requests in case of erroneous data in the target memory, the embOS C-Spy plug-in imposes certain limits on the amount of information retrieved from the target. This dialog box allows you to tweak these limits within a certain range. For example, if your task names are no longer than 32 characters, you might want to configure the `Max. string length` to 32. Alternatively, if task names are longer than the default value, you may increase that value according to your needs.



When clicking the `OK` button, changes to the settings are applied immediately and become visible in the plug-in windows upon the next window update, e.g. when hitting a breakpoint. However, the settings are restored to their default values on plug-in reload.

## 3.13 About

The `About` dialog box contains the embOS C-Spy plug-in version number.



# 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 encounter an error in our software, you may contact our support at [support\\_embos@segger.com](mailto:support_embos@segger.com). We will try to correct any malfunction as soon as possible. To do this, we need all relevant 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 detailed description of the problem and how to reproduce it.
- If possible send us a project that triggers the problem.