

embOS

Real-Time Operating System

embOS plug-in for IAR C-Spy Debugger

Document: UM01025

Software Version: 5.3

Revision: 0

Date: May 10, 2023



A product of SEGGER Microcontroller GmbH

www.segger.com

Disclaimer

The information written in this document is assumed to be accurate without guarantee. The information in this manual is subject to change for functional or performance improvements without notice. SEGGER Microcontroller GmbH (SEGGER) assumes no responsibility for any errors or omissions in this document. SEGGER disclaims any warranties or conditions, express, implied or statutory for the fitness of the product for a particular purpose. It is your sole responsibility to evaluate the fitness of the product for any specific use.

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.

© 2005-2023 SEGGER Microcontroller GmbH, Monheim am Rhein / 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

Ecolab-Allee 5
D-40789 Monheim am Rhein

Germany

Tel. +49 2173-99312-0
Fax. +49 2173-99312-28
E-mail: support@segger.com*
Internet: www.segger.com

*By sending us an email your (personal) data will automatically be processed. For further information please refer to our privacy policy which is available at <https://www.segger.com/legal/privacy-policy/>.

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: May 10, 2023

Software	Revision	Date	By	Description
5.3	0	230510	TS	New plug-in versions 6.10.5.3, 7.10.5.3, 7.50.5.3, 8.10.5.3, 8.30.5.3, 8.40.5.3, and 9.10.5.3.
5.2	0	220803	MM	New plug-in versions 6.10.5.2, 7.10.5.2, 7.50.5.2, 8.10.5.2, 8.30.5.2, and 9.10.5.2.
5.1	0	220718	MM	New plug-in versions 6.10.5.1, 7.10.5.1, 7.50.5.1, 8.10.5.1, 8.30.5.1, and 9.10.5.1.
5.0	0	220412	MM	New plug-in versions 6.10.5.0, 7.10.5.0, 7.50.5.0, 8.10.5.0, 8.30.5.0, and 9.10.5.0. Updated chapter "Getting Started" regarding the embOS-Ultra support.
4.0	0	220119	MM	New plug-in versions 6.10.4.0, 7.10.4.0, 7.50.4.0, 8.10.4.0, 8.30.4.0, and 9.10.4.0. Updated chapter "Getting Started" regarding the settings and readers-writer locks views.
3.9	0	211029	MM	New plug-in versions 6.10.3.9, 7.10.3.9, 7.50.3.9, 8.10.3.9, 8.30.3.9, and 9.10.3.9.
3.8	0	210622	MC	New plug-in versions 6.10.3.8, 7.10.3.8, 7.50.3.8, 8.10.3.8, 8.30.3.8, and 9.10.3.8.
3.7	0	210211	MC	New plug-in versions 6.10.3.7, 7.10.3.7, 7.50.3.7, 8.10.3.7, 8.30.3.7, and 9.10.3.7.
3.6	0	201030	MC	New plug-in versions 6.10.3.6, 7.10.3.6, 7.50.3.6, 8.10.3.6, and 8.30.3.6.
3.5	0	200603	MC	New plug-in versions 6.10.3.5, 7.10.3.5, 7.50.3.5, 8.10.3.5, and 8.30.3.5.
3.4	0	200325	MM	New plug-in versions 6.10.3.4, 7.10.3.4, 7.50.3.4, 8.10.3.4, and 8.30.3.4.
3.3	0	191205	MC	New plug-in version 8.30.3.3.
3.2	0	191106	MC	New plug-in versions 6.10.3.2, 7.10.3.2, 7.50.3.2, 8.10.3.2, and 8.30.3.2.
3.1	1	180924	MC	Updated to include most recent versions of IAR embedded workbench and their compatible plug-ins.
3.1	0	180503	MC	New plug-in versions 6.10.3.1, 7.10.3.1, 7.50.3.1, 8.10.3.1, and 8.30.3.1.
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.
GUIElement	Buttons, dialog boxes, menu names, menu commands.
Emphasis	Very important sections.

Table of contents

1	Introduction	8
1.1	embOS	9
1.2	IAR Embedded Workbench	9
1.3	embOS C-Spy Plug-in	9
1.3.1	Version Scheme and C-Spy Versions	9
1.4	Supported Embedded Workbench variants	9
2	Installation	11
2.1	Installation	12
2.2	Configuration	13
3	Getting Started	14
3.1	Overview	15
3.1.1	embOS-Ultra	15
3.2	Task List	16
3.3	Timers	18
3.4	Mailboxes	19
3.5	Queues	20
3.6	Mutexes	21
3.7	Semaphores	22
3.8	Readers-writer Locks	23
3.9	Memory Pools	24
3.10	Event Objects	25
3.11	Watchdogs	26
3.12	System Information	27
3.13	Settings	28
3.14	About	30
4	Support	31
4.1	Contacting support	32
4.1.1	Where can I find the license number?	32

Chapter 1

Introduction

1.1 embOS

Since 1992, embOS has been the preferred RTOS choice for engineers in the embedded market. It offers ease-of-use and guarantees 100% deterministic real-time operation for any embedded device. This real-time operating system is highly portable and fully source-compatible on all platforms, making it easy to port applications to different cores. Tasks can easily be created and safely communicate with each other, using communication mechanisms such as semaphores, mailboxes, and events.

1.2 IAR Embedded Workbench

IAR Embedded Workbench is a set of development tools for building and debugging embedded applications. It provides a completely integrated development environment that includes the C-SPY debugger which is used by the embOS C-Spy plug-in to retrieve all information from the target.

1.3 embOS C-Spy Plug-in

SEGGER's embOS C-Spy plug-in for IAR Embedded Workbench, or just "embOS plug-in", provides embOS-awareness during debug sessions. This enables you to inspect the state of several embOS primitives such as the task list, queues, mutexes, semaphores, readers-writer locks, mailboxes, software timers, memory pools, event objects, watchdogs, and major system variables.

1.3.1 Version Scheme and C-Spy Versions

The embOS plug-in is available in various versions. Each version was built for a specific C-Spy version. The specific C-Spy version a plug-in was built for can be determined by taking a look at the plug-in's version number. The version number has the form $x.x.y.y$.

x and x specify the major and minor C-Spy version the plug-in was built for. y and y specify the major and minor version of the embOS plug-in for a specific C-Spy version.

1.4 Supported Embedded Workbench variants

The following plug-ins are available and have been tested with the listed versions of IAR's Embedded Workbench:

embOS Port	IAR Embedded Workbench Version	Compatible Plug-In version
78K0	≤ 4.80 ≥ 4.81	6.10.5.3 7.10.5.3
8051	≤ 8.30 ≥ 9.10 and ≤ 9.30 ≥ 10.10 and < 10.40 ≥ 10.40	6.10.5.3 7.10.5.3 8.10.5.3 8.30.5.3
ARM7 / ARM9 / Cortex-A/R/M	≤ 6.70 ≥ 7.10 and ≤ 7.40 ≥ 7.50 and ≤ 7.80 ≥ 8.10 and ≤ 8.22 ≥ 8.30 and ≤ 8.32 ≥ 8.40 and ≤ 8.50	6.10.5.3 7.10.5.3 7.50.5.3 8.10.5.3 8.30.5.3 8.40.5.3
AVR	≤ 6.40 ≥ 6.50 and ≤ 6.80 ≥ 7.10	6.10.5.3 7.10.5.3 8.10.5.3

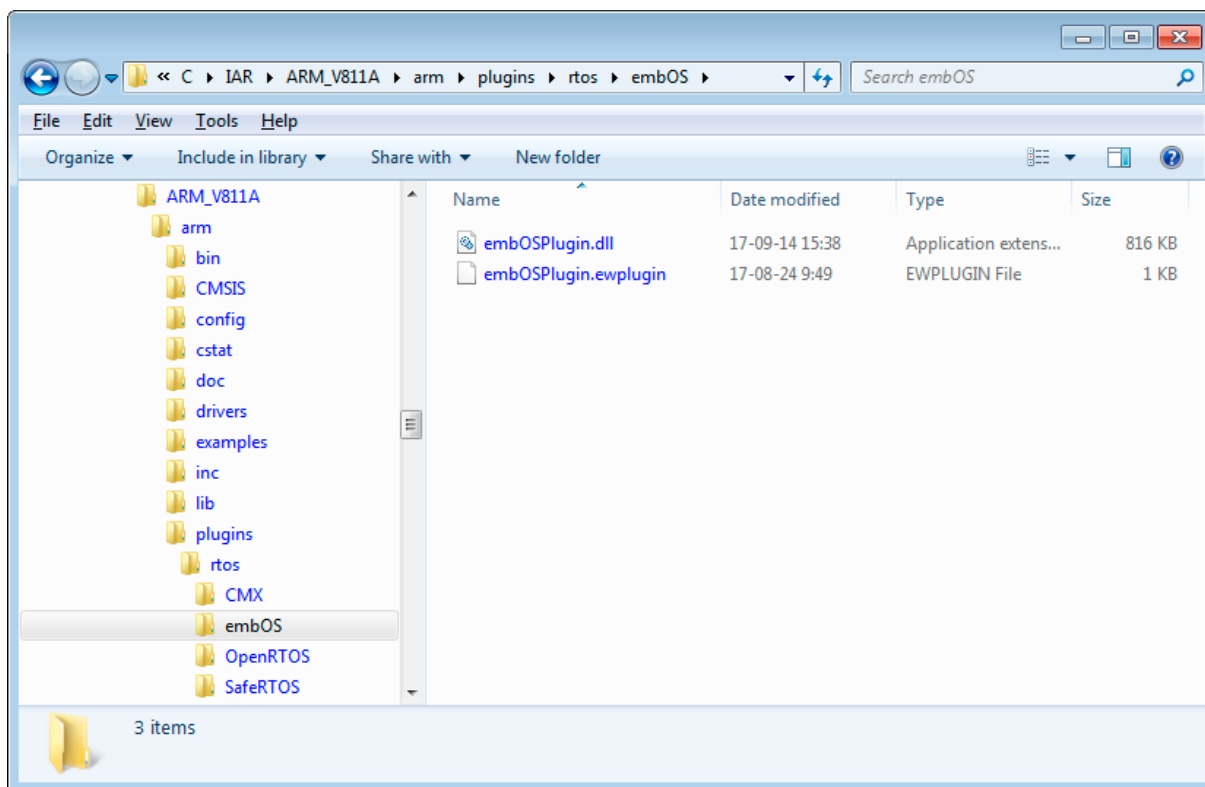
embOS Port	IAR Embedded Workbench Version	Compatible Plug-In version
AVR32	≤ 4.21 ≥ 4.30	6.10.5.3 7.10.5.3
Coldfire	Any	3.82.3.0
CR16C	≤ 3.20 ≥ 3.30	6.10.5.3 7.10.5.3
H8	Any	6.0.1.0
M16C	≤ 3.60 ≥ 3.70	6.10.5.3 7.10.5.3
M32C	Any	6.0.1.0
MSP430	≤ 5.60 ≥ 6.10 and ≤ 6.50 ≥ 7.10 and ≤ 7.12 ≥ 7.20	6.10.5.3 7.10.5.3 8.10.5.3 8.30.5.3
R32C	Any	6.10.5.3
RH850	≤ 1.30 = 1.40 = 2.10 ≥ 2.20	7.10.5.3 7.50.5.3 8.10.5.3 8.30.5.3
RL78	≤ 1.30 ≥ 1.40 and ≤ 2.21 = 3.10 ≥ 4.10	6.10.5.3 7.10.5.3 8.10.5.3 8.30.5.3
RX	≤ 2.50 ≥ 2.60 and ≤ 2.90 = 3.10 ≥ 4.10	6.10.5.3 7.10.5.3 8.10.5.3 8.30.5.3
SH	Any	6.10.5.3
STM8	≤ 1.42 ≥ 2.10 and ≤ 2.20 = 3.10 ≥ 3.11	6.10.5.3 7.10.5.3 8.10.5.3 8.30.5.3
V850	≤ 4.10 ≥ 4.20	6.10.5.3 7.10.5.3

Chapter 2

Installation

2.1 Installation

Typically, there is no installation required since the IAR Embedded Workbench comes with the plug-in already pre-installed. In case you want to update the plug-in to a more recent version, however, you would need to replace two files that are located within the Embedded Workbench installation directory with the respective files from the embOS C-Spy plug-in package. The Embedded Workbench installation directory should resemble the following structure:



If appropriate folders do not yet exist with your installation, you should create a directory called `embOS` within the CPU specific folder `plugin\rtos\`, and subsequently copy the files from the embOS C-Spy plug-in package into that folder. Note that IAR Embedded Workbench must not be running during the update process.

Note

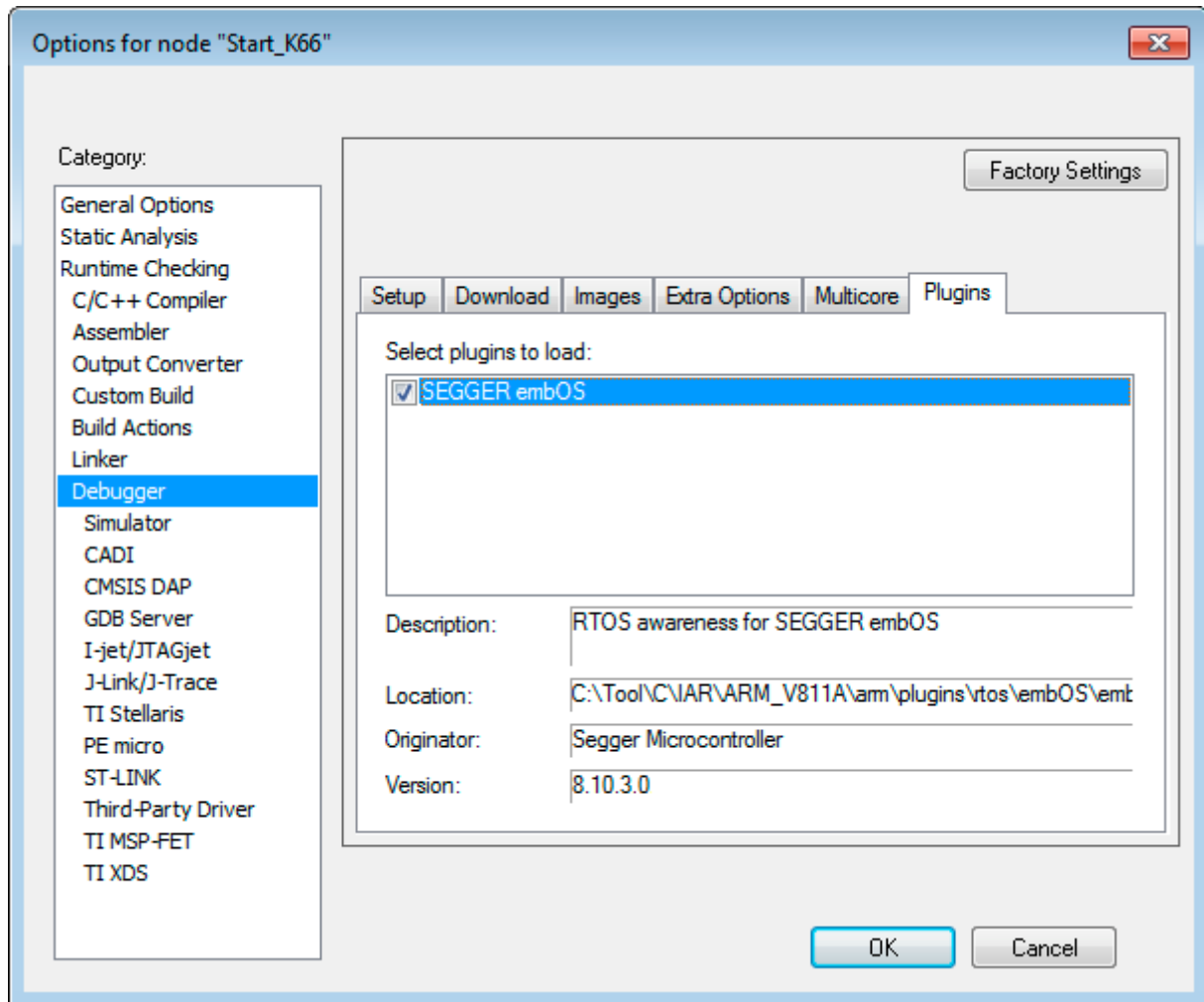
Before replacing any files already found in the `plugin\rtos\embOS` folder of the IAR Embedded Workbench, you may want to backup these files. You should also check the version number of the plug-in inside `embOSPlugin.ewplugin`. Therein, the version number is shown as the last entry and looks like follows:

```
<version>9.30.6.0</version>
```

It is not recommended to replace the plug-in with a plug-in which was built for another C-Spy version. The first two parts of the version number, here `9.30`, indicate the C-Spy version the plug-in was built for. For more information about the version scheme, please refer to *Version Scheme and C-Spy Versions* on page 9.

2.2 Configuration

By default, embOS start projects have the embOS C-Spy plug-in enabled upon project load. The plug-in may be explicitly disabled, individually for each project configuration, in the debugger section of the project options:

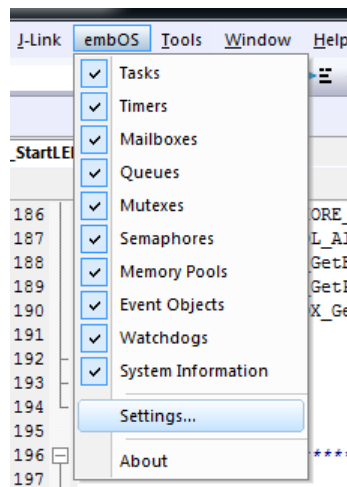


Chapter 3

Getting Started

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 semaphores, queues, event objects, or mailboxes.

3.1.1 embOS-Ultra

Since the plug-in version x.x.5.0, the plug-in versions 8.10.x.x, 8.30.x.x and 9.10.x.x support both, the regular embOS and embOS-Ultra.

Please note that some information might be displayed differently with embOS-Ultra. Time related values are converted from cycles into milliseconds, but timeouts only show the point in time the timeout expires and not the remaining time. This is because the system time stored internally is not up to date when the debug session is halted, as it is only updated by embOS when it is required by the application. Therefore, the system time displayed in the `System Information` window also does not display the current system time, but the last time the system time was updated.

3.2 Task List

Tasks									
*	Prio	Id	Name	Status	Timeout	Stack Info	Run count	Time slice	Events
	100	0x20001AF0	HP Task	Delayed	5 (210)	132 / 512 @ 0x2000103C	21	0 / 2	0x0
	75	0x20001B4C	MP Task	Delayed	1 (206)	164 / 512 @ 0x2000123C	202	0 / 2	0x0
➔	65	0x20001C04	Eval Task	Ready		148 / 512 @ 0x2000163C	4	0 / 2	0x0
	50	0x20001BA8	LP Task	Ready		220 / 512 @ 0x2000143C	192	0 / 2	0x0
	6	0x20000FE0	Background Task 5	Waiting for message in Mailbox 0x20001E40 (Mailbox 1)		164 / 256 @ 0x20000514	1	0 / 2	0x0
	5	0x20000F84	Background Task 4	Waiting for message in Queue 0x20001D90 (Queue 0)		164 / 256 @ 0x20000414	1	0 / 2	0x0
	4	0x20000F28	Background Task 3	Waiting for Event Object 0x20001EB0 (Event 0)		156 / 256 @ 0x20000314	1	0 / 2	0x0
	3	0x20000ECC	Background Task 2	Waiting for Memory Pool 0x20001C60 (MemPool 0)		156 / 256 @ 0x20000214	1	0 / 2	0x0
	2	0x20000E70	Background Task 1	Waiting for Semaphore 0x20001ED8 (Semaphore 0)		156 / 256 @ 0x20000114	1	0 / 2	0x0
	1	0x20000E14	Background Task 0	Waiting for Mutex 0x20001E9C (Mutex 0)		156 / 256 @ 0x20000014	1	0 / 2	0x0
		Idle							

Column	Description
*	A green arrow points at the task that is currently executed.
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. If the task is waiting for an OS object (e.g. semaphore), the object's type and control block address is given and, in paranthesis, the object's identifier (if any).
Timeout	If a task is blocked with timeout, this column shows the remaining timeout value in system ticks and, in parenthesis, the system time at which the timeout will expire. With embOS-Ultra only the system time in milliseconds at which the timeout will expire is shown.
Stack Info	If available, this column shows the maximum used amount of stack, the total stack size, and the stack's base address which uniquely identifies a task stack.
Run count	The number of times a task has been activated by the scheduler.
Time slice	If round-robin scheduling is available, this column shows the number of currently remaining time slices and the time slice reload value.
Events	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/ARM9, Cortex-A/R/M
- 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 IAR C-Spy Debugger interface. The **Tasks** window displays a list of tasks, with task 75 (MP Task) highlighted in yellow, indicating it is the current task. The task is in a **Delayed** state. The **Call Stack** window shows the current call stack, with the top entry being `[OS_Deactivated + 0x23]`. The **Registers** window shows the current CPU registers, with the **Value** column displaying hexadecimal values and the **Access** column showing the access type (Read/Write/Read Only).

Prio	Id	Name	Status	Timeout	Stack Info	Run count	Time slice	Events
100	0x20001AF0	HP Task	Delayed	10 (20)	132 / 512 @ 0x2000103C	2	0 / 2	0x0
75	0x20001B4C	MP Task	Delayed	1 (11)	164 / 512 @ 0x2000123C	11	0 / 2	0x0
65	0x20001C04	Eval Task	Ready		132 / 512 @ 0x2000163C	1	0 / 2	0x0
50	0x20001BA8	LP Task	Ready		220 / 512 @ 0x2000143C	2	0 / 2	0x0
6	0x20000FE0	Background Task 5	Waiting for message in Mailbox 0x20001E40 (Mailbox 1)		164 / 256 @ 0x20000514	1	0 / 2	0x0
5	0x20000F84	Background Task 4	Waiting for message in Queue 0x20001D90 (Queue 0)		164 / 256 @ 0x20000414	1	0 / 2	0x0
4	0x20000F28	Background Task 3	Waiting for Event Object 0x20001EB0 (Event 0)		156 / 256 @ 0x20000314	1	0 / 2	0x0
3	0x20000ECC	Background Task 2	Waiting for Memory Pool 0x20001C60 (MemPool 0)		156 / 256 @ 0x20000214	1	0 / 2	0x0
2	0x20000E70	Background Task 1	Waiting for Semaphore 0x20001ED8 (Semaphore 0)		156 / 256 @ 0x20000114	1	0 / 2	0x0
1	0x20000E14	Background Task 0	Waiting for Mutex 0x20001E9C (Mutex 0)		156 / 256 @ 0x20000014	1	0 / 2	0x0
		Idle						

Name	Value	Access
R0	0x20001c88	ReadWrite
R1	0xe000ed04	ReadWrite
R2	0x10000000	ReadWrite
R3	0x00000001	ReadWrite
R4	0x20001c88	ReadWrite
R5	0x00000000	ReadWrite
R6	0x00000006	ReadWrite
R7	0x00000007	ReadWrite
R8	0x00000008	ReadWrite
R9	0x00000009	ReadWrite
R10	0x0000000a	ReadWrite
R11	0x0000000b	ReadWrite
R12	0xcdcdcdcd	ReadWrite
APSR	0x60000000	ReadWrite
IPSR	0x00000000	ReadWrite
EPSR	0x01000000	ReadWrite
PC	0x08001cf4	ReadWrite
SP	0x200013e8	ReadWrite
LR	0x08001cf5	ReadWrite
PRIMASK	0x00000000	ReadWrite
BASEPRI	0x00000000	ReadWrite
BASEPRI_MAX	0x00000000	ReadWrite
FAULTMASK	0x00000000	ReadWrite
CONTROL	0x00000002	ReadWrite
CYCLECOUNTER	1710066	Read Only
CCTIMER1	1710066	ReadWrite
CCTIMER2	1710066	ReadWrite
CCSTEP	1694520	Read Only

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 IAR C-Spy Debugger interface. The **Tasks** window displays a list of tasks, with task 65 (Eval Task) highlighted in yellow, indicating it is the current task. The task is in a **Ready** state. The **Call Stack** window shows the current call stack, with the top entry being `[OS_StartTask + 0]`. The **Registers** window shows the current CPU registers, with the **Value** column displaying hexadecimal values and the **Access** column showing the access type (Read/Write/Read Only).

Prio	Id	Name	Status	Timeout	Stack Info	Run count	Time slice	Events
100	0x20001AF0	HP Task	Delayed	10 (20)	132 / 512 @ 0x2000103C	2	0 / 2	0x0
75	0x20001B4C	MP Task	Delayed	1 (11)	164 / 512 @ 0x2000123C	11	0 / 2	0x0
65	0x20001C04	Eval Task	Ready		92 / 512 @ 0x2000163C	0	0 / 2	0x0
50	0x20001BA8	LP Task	Ready		220 / 512 @ 0x2000143C	2	0 / 2	0x0
6	0x20000FE0	Background Task 5	Waiting for message in Mailbox 0x20001E40 (Mailbox 1)		164 / 256 @ 0x20000514	1	0 / 2	0x0
5	0x20000F84	Background Task 4	Waiting for message in Queue 0x20001D90 (Queue 0)		164 / 256 @ 0x20000414	1	0 / 2	0x0
4	0x20000F28	Background Task 3	Waiting for Event Object 0x20001EB0 (Event 0)		156 / 256 @ 0x20000314	1	0 / 2	0x0
3	0x20000ECC	Background Task 2	Waiting for Memory Pool 0x20001C60 (MemPool 0)		156 / 256 @ 0x20000214	1	0 / 2	0x0
2	0x20000E70	Background Task 1	Waiting for Semaphore 0x20001ED8 (Semaphore 0)		156 / 256 @ 0x20000114	1	0 / 2	0x0
1	0x20000E14	Background Task 0	Waiting for Mutex 0x20001E9C (Mutex 0)		156 / 256 @ 0x20000014	1	0 / 2	0x0
		Idle						

Name	Value	Access
R0	0xcdcdcdcd	ReadWrite
R1	0x00000001	ReadWrite
R2	0x00000002	ReadWrite
R3	0x00000003	ReadWrite
R4	0x00000004	ReadWrite
R5	0x00000005	ReadWrite
R6	0x00000006	ReadWrite
R7	0x00000007	ReadWrite
R8	0x00000008	ReadWrite
R9	0x00000009	ReadWrite
R10	0x0000000a	ReadWrite
R11	0x0000000b	ReadWrite
R12	0xcdcdcdcd	ReadWrite
APSR	0x20000000	ReadWrite
IPSR	0x00000000	ReadWrite
EPSR	0x01000000	ReadWrite
PC	0x080021ed	ReadWrite
SP	0x20001830	ReadWrite
LR	0xcdcdcdcd	ReadWrite
PRIMASK	0x00000000	ReadWrite
BASEPRI	0x00000000	ReadWrite
BASEPRI_MAX	0x00000000	ReadWrite
FAULTMASK	0x00000000	ReadWrite
CONTROL	0x00000002	ReadWrite
CYCLECOUNTER	1707834	Read Only
CCTIMER1	1707834	ReadWrite
CCTIMER2	1707834	ReadWrite
CCSTEP	1707834	Read Only

3.3 Timers

Timers					
Id	Name	Hook	Timeout	Period	Active
0x20001E88	TimerShort	0x800403A (_TimerShort_Callback)	15 (220)	20	1
0x20001E74	TimerLong	0x8004024 (_TimerLong_Callback)	195 (400)	200	1

Column	Description
Id	The timer control block address that uniquely identifies a timer.
Name	If available, the respective object identifier is shown here.
Hook	The function (address and name) that is called after the timeout.
Timeout	This column shows the remaining timer period in system ticks and, in parenthesis, the system time at which the timer will expire. With em-bOS-Ultra only the system time in milliseconds at which the timer will expire is shown.
Period	The timer's periodicity in system ticks.
Active	Indicates whether the timer is currently 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
0x20001E40	Mailbox1	0 / 8	8	0x20001D10	0x2000FE0 (Background Task 5)
0x20001E24	Mailbox0	2 / 8	8	0x20001CD0	

Column	Description
Id	The mailbox control block address that uniquely identifies a mailbox.
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 the mailbox (address and, if available, name). Only those tasks that are displayed in the task list window may be shown here.

3.5 Queues

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

Queues					
Id	Name	Messages	pBuffer	Buffer size	Waiting tasks
0x20001D90	Queue 0	0	0x20001A90	96	0x20000F84 (Background Task 4)

Column	Description
Id	The queue control block address that uniquely identifies a queue.
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 size of the message buffer.
Waiting tasks	The list of tasks that are waiting for the queue (address and, if available, name). Only those tasks that are displayed in the task list window may be shown here.

3.6 Mutexes

Mutexes				
Id	Name	Owner	Use counter	Waiting tasks
0x20001E9C	Mutex 0	0x20001B4C (MP Task)	202	0x20000E14 (Background Task 0)

Column	Description
Id	The mutex control block address that uniquely identifies a mutex.
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 the semaphore (address and, if available, name). Only those tasks that are displayed in the task list window may be shown here.

3.7 Semaphores

This view displays information in debug builds of embOS only.

Semaphores			
Id	Name	Count	Waiting tasks
0x20001ED8	Semaphore 0	0	0x20000E70 (Background Task 1)

Column	Description
Id	The semaphore control block address that uniquely identifies a semaphore.
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 the semaphore (address and, if available, name). Only those tasks that are displayed in the task list window may be shown here.

3.8 Readers-writer Locks

This view displays information in debug builds of embOS only.

Readers-writer Locks				
Id	Name	Status	Max. number of tokens	Tokens left
0x20001DBC	Readers-writer lock 3	Locked	4	0
0x20001D90	Readers-writer lock 2	Locked	3	0
0x20001C5C	Readers-writer lock 1	Unlocked	2	1

Column	Description
Id	The readers-writer lock control block address that uniquely identifies a readers-writer lock.
Name	If available, the respective object identifier is shown here.
Status	If all tokens are taken the readers-writer lock is locked. Otherwise it is unlocked.
Max. number of tokens	The maximum numbers of token which were defined when the readers-writer lock was created.
Tokens left	The number of available tokens.

3.9 Memory Pools

Memory Pools						
Id	Name	Total blocks	Block size	Max. usage	pPool	Waiting tasks
0x20001C60	MemPool 0	0 / 3	4	3	0x20001F68	0x20000ECC (Background Task 2)

Column	Description
Id	The memory pool control block address that uniquely identifies a memory pool.
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 free blocks in the memory pool (address and, if available, name). Only those tasks that are displayed in the task list window may be shown here.

3.10 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
0x20001EB0	Event 0	0x0	Semiauto	Or logic	0x20000F28 (Background Task 3)

Column	Description
Id	The event object control block address that uniquely identifies an event object.
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 may be shown here.

3.11 Watchdogs

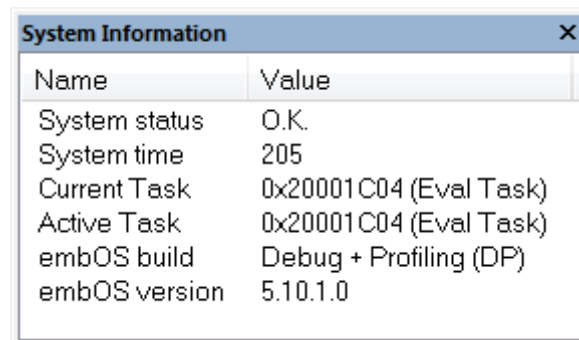
This view displays information with embOS V4.38 and subsequent versions only.

Watchdogs			
Id	Name	Timeout	Period
0x20001FC8	WatchdogEval	1000 (1205)	1000
0x20001FB0	WatchdogLP	705 (910)	750
0x20001F98	WatchdogMP	500 (705)	500
0x20001F80	WatchdogHP	245 (450)	250

Column	Description
Id	The watchdog control block address that uniquely identifies a watchdog.
Name	If available, the respective object identifier is shown here.
Timeout	This column shows the remaining time in system ticks and, in parenthesis, the system time until the watchdog needs to be fed. With embOS-Ultra only the system time in milliseconds until the watchdog needs to be fed is shown.
Period	The period in which the watchdog has to be fed.

3.12 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	205
Current Task	0x20001C04 (Eval Task)
Active Task	0x20001C04 (Eval Task)
embOS build	Debug + Profiling (DP)
embOS version	5.10.1.0

3.13 Settings

To avoid endless requests in case of erroneous data in target memory, the embOS C-Spy plug-in imposes several limits on the amount of information retrieved from the target. It also configures an entry point for the plug-in at which it will start reading data from the target to avoid accessing invalid data and/or uninitialized memory, e.g. when the debug session is halted during start-up.

The settings dialog allows to configure these limits and the entry point for the plug-in:

embOS plug-in settings	
Maximum string length	256
Maximum number of tasks	64
Maximum number of timers	64
Maximum number of mailboxes	64
Maximum number of queues	64
Maximum number of mutexes	64
Maximum number of semaphores	64
Maximum number of readers-writer locks	64
Maximum number of memory pools	64
Maximum number of event objects	64
Maximum number of watchdogs	64
Maximum waitlist length	8
Perform stack check	<input checked="" type="checkbox"/>
Maximum stack check length	8192
Enable plug-in at function	<input checked="" type="checkbox"/>
Function name	OS_Init
<input type="button" value="OK"/> <input type="button" value="Cancel"/>	

Setting	Permissible values	Description
Maximum string length	1 to 1024	Maximum number of characters to read for each string (e.g. task names).
Maximum number of tasks	1 to 256	Maximum number of tasks to display in the Task List.
Maximum number of timers	1 to 256	Maximum number of timers to display in the Timers view.
Maximum number of mailboxes	1 to 256	Maximum number of mailboxes to display in the Mailboxes view.
Maximum number of queues	1 to 256	Maximum number of queues to display in the Queues view.
Maximum number of mutexes	1 to 256	Maximum number of mutexes to display in the Mutexes view.
Maximum number of semaphores	1 to 256	Maximum number of semaphores to display in the Semaphores view.
Maximum number of readers-writer locks	1 to 256	Maximum number of readers-writer locks to display in the Readers-writer Lock view.
Maximum number of memory pools	1 to 256	Maximum number of memory pools to display in the Memory Pools view.
Maximum number of event objects	1 to 256	Maximum number of event objects to display in the Event Objects view.
Maximum number of watchdogs	1 to 256	Maximum number of watchdogs to display in the Watchdogs view.
Maximum waitlist length	1 to 64	Maximum number of waiting tasks to display in the Mutexes, Semaphores, Mailboxes, Queues, Memory Pools, and Event Objects views.
Perform stack check	n/a	Enables/disables the calculation and display of stack usage information in the Task List.
Maximum stack check length	1 to 65,536	Maximum number of bytes used to calculate and display in the stack usage information in the Task List.
Enable plug-in at function	n/a	Enables/disables an entry point for the plug-in. If checked, the plug-in will become active when the target executes the specified function. If unchecked, the plug-in will become active with the start of the debug session.
Function name	n/a	Name of the function with a length of up to 127 characters to be used as an entry point for the plug-in.

When clicking the **OK** button, all entries are checked for valid values. If valid, the settings are applied immediately.

The plug-in settings for plug-in versions 3.9 and older are stored inside the Windows registry at `HKEY_CURRENT_USER\Software\SEGGER\embOS plug-in for IAR C-Spy 6 and 7` for plug-in versions up to 6.x and 7.x, at `HKEY_CURRENT_USER\Software\SEGGER\embOS plug-in for IAR C-Spy 8` for plug-in version 8.x, and at `HKEY_CURRENT_USER\Software\SEGGER\embOS plug-in for IAR C-Spy 9` for plug-in version 9.x.

The plug-in settings for plug-in versions 4.0 and newer are stored inside the Windows registry at `HKEY_CURRENT_USER\Software\SEGGER\embOS plug-in for IAR C-Spy 6 and 7 (V2)` for plug-in versions up to 6.x and 7.x, at `HKEY_CURRENT_USER\Software\SEGGER\embOS plug-in for IAR C-Spy 8 (V2)` for plug-in version 8.x, and at `HKEY_CURRENT_USER\Software\SEGGER\embOS plug-in for IAR C-Spy 9 (V2)` for plug-in version 9.x.

3.14 About

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



Chapter 4

Support

4.1 Contacting support

If you need help or if any problem occurs the following describes how to contact the embOS support.

If you are a registered embOS user there are different ways to contact the embOS support:

1. You can create a support ticket via email to ticket_embos@segger.com.*
2. You can create a support ticket at segger.com/ticket.*
3. You can send an email to support_embos@segger.com.*

Please include the following information in the email or ticket:

- Which embOS do you use? (Core, compiler).
- The embOS version.
- Your embOS license number.
- If you are unsure about the above information you can also use the name of the embOS zip file (which contains the above information).
- A detailed description of the problem.
- Optionally a project with which we can reproduce the problem.

Note

Even without a valid license, feel free to contact our support e.g. in case of questions during your evaluation of embOS or for hobbyist purposes.

Please also take a few moments to help us improve our services by providing a short feedback once your support case has been solved.

4.1.1 Where can I find the license number?

The license number is part of the shipped zip file name. For example `embOS_CortexM_GC-C_SRC_V5.10.2.0_OS-01234_C1010320_200305.zip` where OS-01234 is the license number.

The license number is also part of every *.c- and *.h-file header. For example, if you open RTOS.h you should find the license number as with the example below:

```
-----
Licensing information
Licensor:                SEGGER Microcontroller GmbH
Licensed to:             Customer name
Licensed SEGGER software: embOS
License number:          OS-01234
License model:           SSL
Licensed product:        -
Licensed platform:       Cortex-M, GCC
Licensed number of seats: 1
-----
Support and Update Agreement (SUA)
SUA period:              2020-03-05 - 2021-03-05
Contact to extend SUA:   sales@segger.com
----- END-OF-HEADER -----
File      : RTOS.h
Purpose   : Include file for the OS,
            to be included in every C-module accessing OS-routines
```

*By sending us an email your (personal) data will automatically be processed. For further information please refer to our privacy policy which is available at <https://www.segger.com/legal/privacy-policy/>.