embOS

Real-Time Operating System

embOS plug-in for IAR C-Spy Debugger

Document: UM01085
Software Version: 6.3
Revision: 0
Date: July 18, 2024

A product of SEGGER Microcontroller GmbH

www.segger.com

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: July 18, 2024

Software	Revision	Date	By	Description
6.3	0	240718	MC	New plug-in versions 9.10.6.3, 9.30.6.3 and 9.40.6.3.
6.2	0	230622	MM	New plug-in versions 9.10.6.2, 9.30.6.2 and 9.40.6.2.
6.1	0	230505	MM/TS	Updated to latest version.
6.0	0	230315	MM	Initial manual version. New plug-in versions 9.10.6.0 and 9.30.6.0.

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.

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

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:

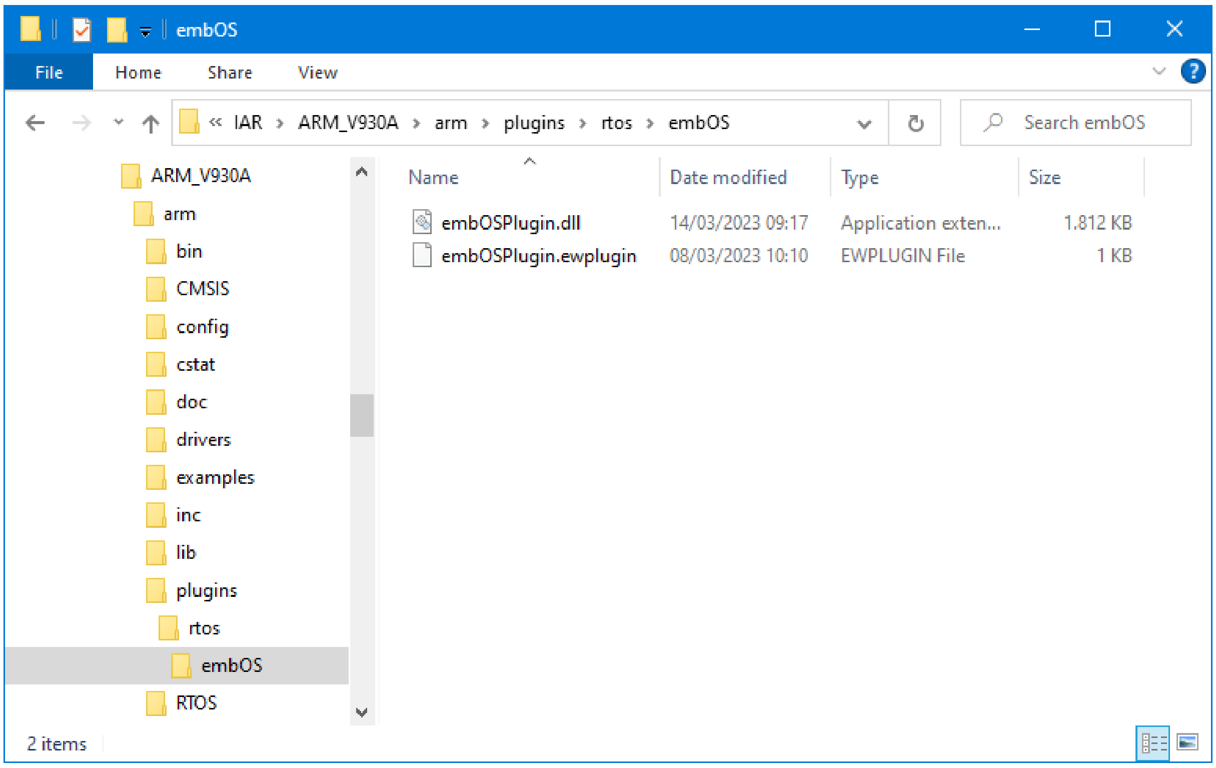
embOS Port	IAR Embedded Workbench Version	Compatible Plug-In version
ARM / Cortex-M	≥ 9.10 and ≤ 9.20 ≥ 9.30 and ≤ 9.32 ≥ 9.40 and ≤ 9.60	9.10.6.3 9.30.6.3 9.40.6.3
RL78	= 5.10	9.30.6.3
RX	= 5.10	9.40.6.3

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:



Name	Date modified	Type	Size
embOSPlugin.dll	14/03/2023 09:17	Application exten...	1.812 KB
embOSPlugin.ewplugin	08/03/2023 10:10	EWPLUGIN File	1 KB

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

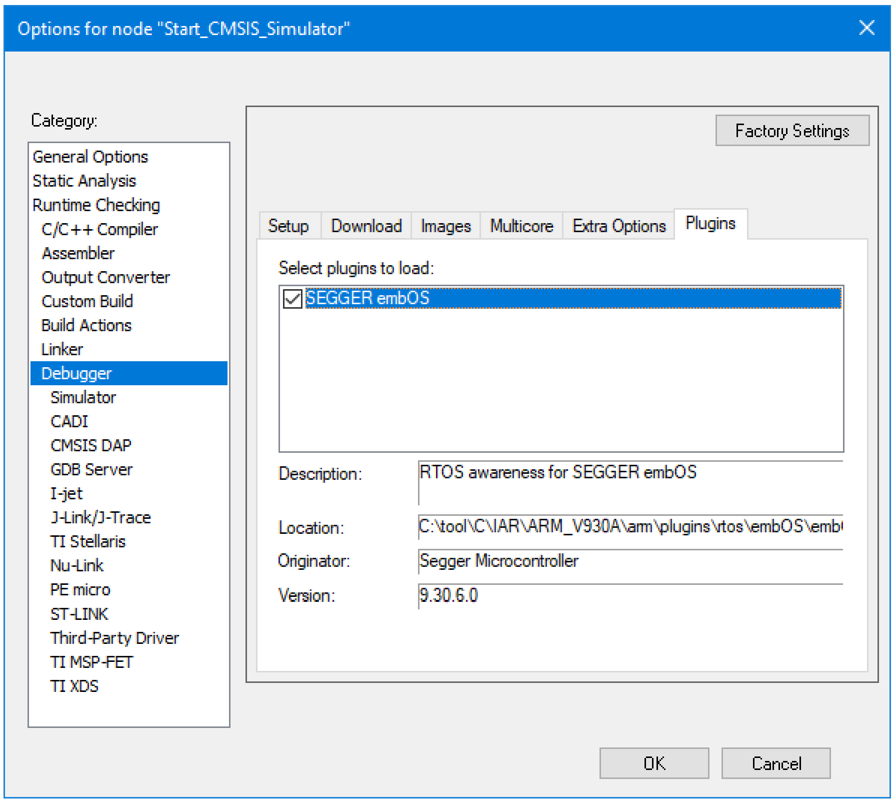
Before replacing any files already found in the `plugin\rtos\embOS` folder of the IAR Embedded Workbench, you may want to backup these files. You should also check the version number of the plug-in inside `embOSPlugin.ewplugin`. Therein, the version number is shown as the last entry and looks like follows:

```
<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:



Select plugins to load:

☑ SEGGER embOS

Description: RTOS awareness for SEGGER embOS

Location: C:\tool\C\IAR\ARM_V930A\arm\plugins\rtos\embOS\emb

Originator: Segger Microcontroller

Version: 9.30.6.0

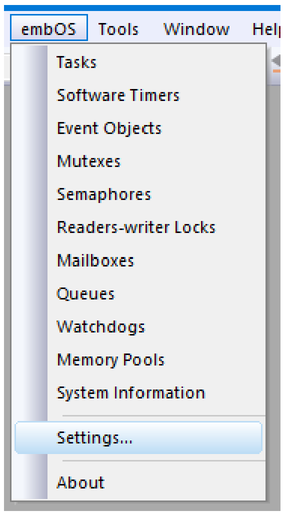
Chapter 3

Getting Started

3.1 Overview

3.1.1 embOS C-Spy Plug-in Menu

The the embOS C-Spy plug-in is accessible via the "embOS" menu item in the IAR Embedded Workbench menu bar. The "embOS" menu is only available if the plug-in is enabled in the project options.

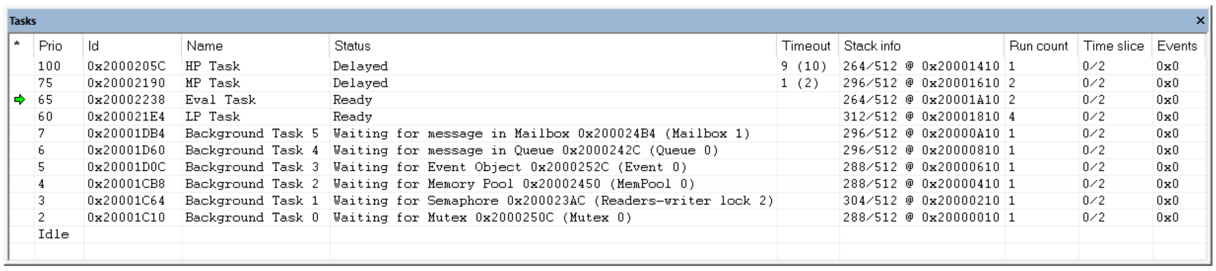


From the menu you may activate the individual windows that provide embOS related information. The sections below describe these individual windows. The amount of information available depends on the embOS build used during debugging. A Release build, for instance, will not show any information about semaphores, queues, event objects, or mailboxes.

3.1.2 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	0x2000205C	HP Task	Delayed	9 (10)	264/512 @ 0x20001410	1	0/2	0x0
	75	0x20002190	MP Task	Delayed	1 (2)	296/512 @ 0x20001610	2	0/2	0x0
➡	65	0x20002238	Eval Task	Ready		264/512 @ 0x20001A10	2	0/2	0x0
	60	0x200021E4	LP Task	Ready		312/512 @ 0x20001810	4	0/2	0x0
	7	0x20001DB4	Background Task 5	Waiting for message in Mailbox 0x200024B4 (Mailbox 1)		296/512 @ 0x20000A10	1	0/2	0x0
	6	0x20001D60	Background Task 4	Waiting for message in Queue 0x2000242C (Queue 0)		296/512 @ 0x20000810	1	0/2	0x0
	5	0x20001D0C	Background Task 3	Waiting for Event Object 0x2000252C (Event 0)		288/512 @ 0x20000610	1	0/2	0x0
	4	0x20001CB8	Background Task 2	Waiting for Memory Pool 0x20002450 (MemPool 0)		288/512 @ 0x20000410	1	0/2	0x0
	3	0x20001C64	Background Task 1	Waiting for Semaphore 0x200023AC (Readers-writer lock 2)		304/512 @ 0x20000210	1	0/2	0x0
	2	0x20001C10	Background Task 0	Waiting for Mutex 0x2000250C (Mutex 0)		288/512 @ 0x20000010	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, the Stack window and the Call Stack window of the C-Spy debugger are task-sensitive for several CPUs. This means that they show the position in the code, the general-purpose registers, the stack content 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 highlighted in yellow. The C-Spy Debugger rebuilds the call stack and the preserved general-purpose registers of a suspended task.

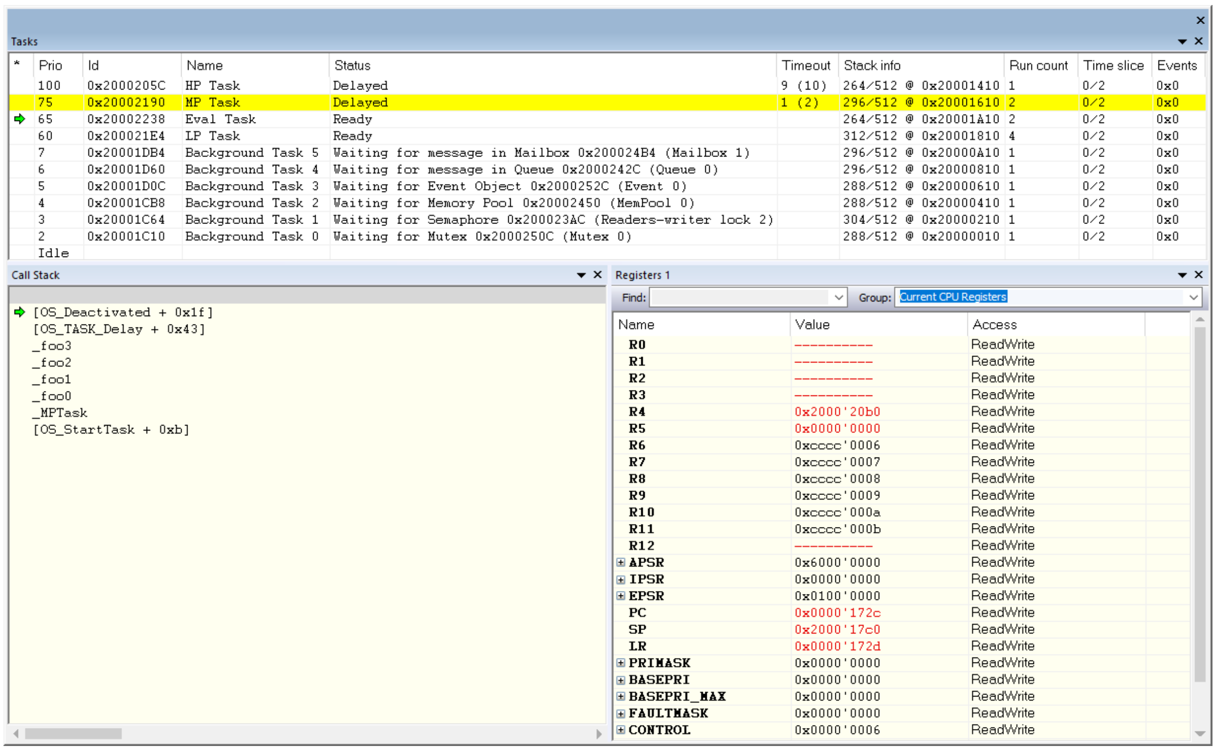
Every time the CPU is started and halted again or when the Idle-row of the task window is double-clicked, the context of the currently active task is shown again.

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

- ARM7/ARM9, Cortex-A/R/M
- Renesas RL78

3.2.1.1 Temporary Registers

When selecting a suspended task to show its context, it can be that the content of some registers is not shown in the register window. Hyphens are displayed instead of the register content as shown in the screenshot below. This is because temporary registers might not be available on the task stack of some suspended tasks.



Tasks

*	Prio	Id	Name	Status	Timeout	Stack info	Run count	Time slice	Events
	100	0x2000205C	HP Task	Delayed	9 (10)	264/512 @ 0x20001410	1	0/2	0x0
	75	0x20002190	MP Task	Delayed	1 (2)	296/512 @ 0x20001610	2	0/2	0x0
➪	65	0x20002238	Eval Task	Ready		264/512 @ 0x20001A10	2	0/2	0x0
	60	0x200021E4	LP Task	Ready		312/512 @ 0x20001810	4	0/2	0x0
	7	0x20001DB4	Background Task 5	Waiting for message in Mailbox 0x200024B4 (Mailbox 1)		296/512 @ 0x20000A10	1	0/2	0x0
	6	0x20001D60	Background Task 4	Waiting for message in Queue 0x2000242C (Queue 0)		296/512 @ 0x20000810	1	0/2	0x0
	5	0x20001D0C	Background Task 3	Waiting for Event Object 0x2000252C (Event 0)		288/512 @ 0x20000610	1	0/2	0x0
	4	0x20001CB8	Background Task 2	Waiting for Memory Pool 0x20002450 (MemPool 0)		288/512 @ 0x20000410	1	0/2	0x0
	3	0x20001C64	Background Task 1	Waiting for Semaphore 0x200023AC (Readers-writer lock 2)		304/512 @ 0x20000210	1	0/2	0x0
	2	0x20001C10	Background Task 0	Waiting for Mutex 0x2000250C (Mutex 0)		288/512 @ 0x20000010	1	0/2	0x0
	Idle								

Call Stack

```
➪ [OS_Deactivated + 0x1f]
  [OS_TASK_Delay + 0x43]
  _foo3
  _foo2
  _foo1
  _foo0
  _MPTask
  [OS_StartTask + 0xb]
```

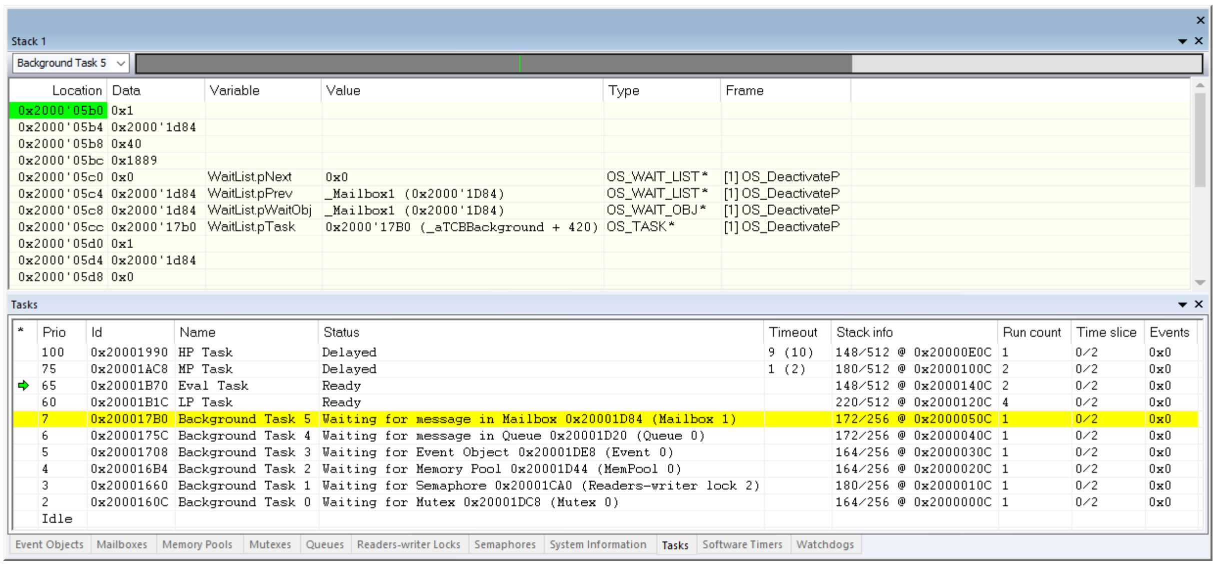
Registers 1

Find: Group: Current CPU Registers

Name	Value	Access
R0	----------	ReadWrite
R1	----------	ReadWrite
R2	----------	ReadWrite
R3	----------	ReadWrite
R4	0x2000'20b0	ReadWrite
R5	0x0000'0000	ReadWrite
R6	0xcccc'0006	ReadWrite
R7	0xcccc'0007	ReadWrite
R8	0xcccc'0008	ReadWrite
R9	0xcccc'0009	ReadWrite
R10	0xcccc'000a	ReadWrite
R11	0xcccc'000b	ReadWrite
R12	----------	ReadWrite
⊞ APSR	0x6000'0000	ReadWrite
⊞ IPSR	0x0000'0000	ReadWrite
⊞ EPSR	0x0100'0000	ReadWrite
PC	0x0000'172c	ReadWrite
SP	0x2000'17c0	ReadWrite
LR	0x0000'172d	ReadWrite
⊞ PRIMASK	0x0000'0000	ReadWrite
⊞ BASEPRI	0x0000'0000	ReadWrite
⊞ BASEPRI_MAX	0x0000'0000	ReadWrite
⊞ FAULTMASK	0x0000'0000	ReadWrite
⊞ CONTROL	0x0000'0006	ReadWrite

3.2.1.2 Stack Information

IAR Embedded Workbench's Stack window shows the content of stacks. When the debug session is halted, the Stack window shows the default CSTACK. When double-clicking on a task, the selected task's stack is shown instead. This applies also to the currently running task. The default stack can be shown again by double-clicking the Idle row in the task list.



Stack 1

Background Task 5

Location	Data	Variable	Value	Type	Frame
0x2000'05b0	0x1				
0x2000'05b4	0x2000'1d84				
0x2000'05b8	0x40				
0x2000'05bc	0x1889				
0x2000'05c0	0x0	WaitList.pNext	0x0	OS_WAIT_LIST*	[1] OS_DeactivateP
0x2000'05c4	0x2000'1d84	WaitList.pPrev	_Mailbox1 (0x2000'1D84)	OS_WAIT_LIST*	[1] OS_DeactivateP
0x2000'05c8	0x2000'1d84	WaitList.pWaitObj	_Mailbox1 (0x2000'1D84)	OS_WAIT_OBJ*	[1] OS_DeactivateP
0x2000'05cc	0x2000'17b0	WaitList.pTask	0x2000'17B0 (_aTCBBackground + 420)	OS_TASK*	[1] OS_DeactivateP
0x2000'05d0	0x1				
0x2000'05d4	0x2000'1d84				
0x2000'05d8	0x0				

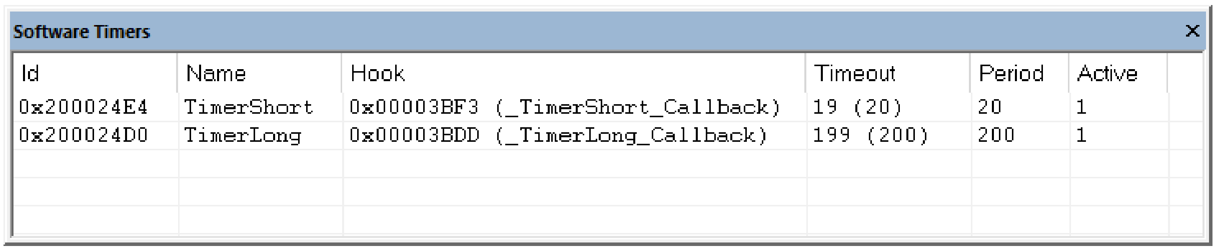
Tasks

*	Prio	Id	Name	Status	Timeout	Stack info	Run count	Time slice	Events
	100	0x20001990	HP Task	Delayed	9 (10)	148/512 @ 0x20000E0C	1	0/2	0x0
	75	0x20001AC8	MP Task	Delayed	1 (2)	180/512 @ 0x2000100C	2	0/2	0x0
➪	65	0x20001B70	Eval Task	Ready		148/512 @ 0x2000140C	2	0/2	0x0
	60	0x20001B1C	LP Task	Ready		220/512 @ 0x2000120C	4	0/2	0x0
	7	0x200017B0	Background Task 5	Waiting for message in Mailbox 0x20001D84 (Mailbox 1)		172/256 @ 0x2000050C	1	0/2	0x0
	6	0x2000175C	Background Task 4	Waiting for message in Queue 0x20001D20 (Queue 0)		172/256 @ 0x2000040C	1	0/2	0x0
	5	0x20001708	Background Task 3	Waiting for Event Object 0x20001DE8 (Event 0)		164/256 @ 0x2000030C	1	0/2	0x0
	4	0x200016B4	Background Task 2	Waiting for Memory Pool 0x20001D44 (MemPool 0)		164/256 @ 0x2000020C	1	0/2	0x0
	3	0x20001660	Background Task 1	Waiting for Semaphore 0x20001CA0 (Readers-writer lock 2)		180/256 @ 0x2000010C	1	0/2	0x0
	2	0x2000160C	Background Task 0	Waiting for Mutex 0x20001DC8 (Mutex 0)		164/256 @ 0x2000000C	1	0/2	0x0
	Idle								

The Stack window shows the name of the corresponding task in the upper left corner. Next to it, a stack bar displays in dark gray how much of the stack was already used, and with a green marker where the task's stack pointer is currently pointing to. Below the task name and stack bar the content of the stack is shown. If the task's used stack space cannot be determined, because the stack check limit specified in the settings it too low, only the green marker for the current stack pointer without the dark gray field is shown in the bar.

Information about the task stacks is shown only if the required information is available, i.e. an embOS library mode with stack check is used, and the stack check is enabled in the plug-in settings.

3.3 Software Timers

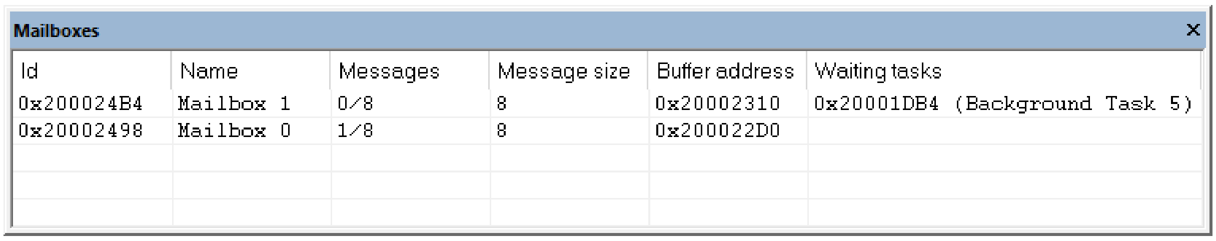

Software Timers

Id	Name	Hook	Timeout	Period	Active
0x200024E4	TimerShort	0x00003BF3 (_TimerShort_Callback)	19 (20)	20	1
0x200024D0	TimerLong	0x00003BDD (_TimerLong_Callback)	199 (200)	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 embOS-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.

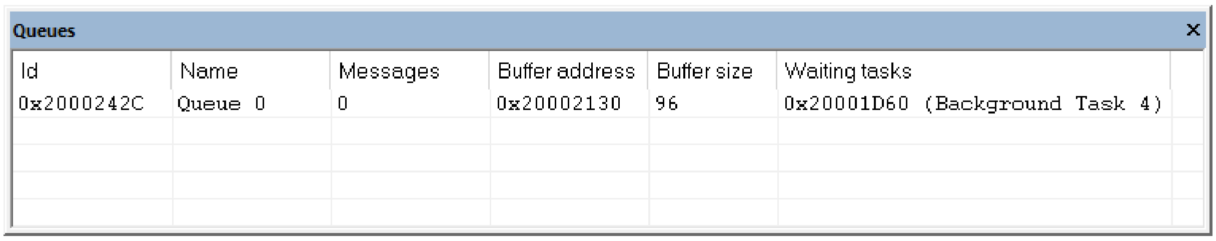


Id	Name	Messages	Message size	Buffer address	Waiting tasks
0x200024B4	Mailbox 1	0/8	8	0x20002310	0x20001DB4 (Background Task 5)
0x20002498	Mailbox 0	1/8	8	0x200022D0	

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.
Buffer address	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.

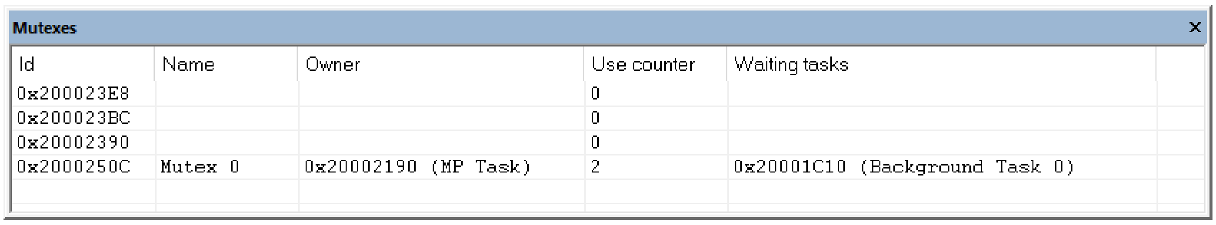


Queues

Id	Name	Messages	Buffer address	Buffer size	Waiting tasks
0x2000242C	Queue 0	0	0x20002130	96	0x20001D60 (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.
Buffer address	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



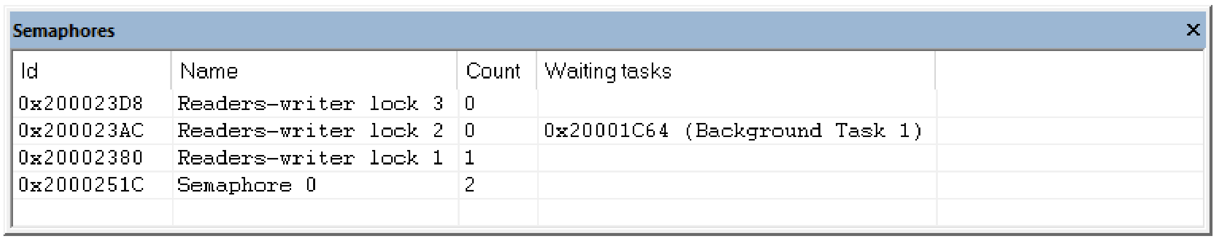
Mutexes

Id	Name	Owner	Use counter	Waiting tasks
0x200023E8			0	
0x200023BC			0	
0x20002390			0	
0x2000250C	Mutex 0	0x20002190 (MP Task)	2	0x20001C10 (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.



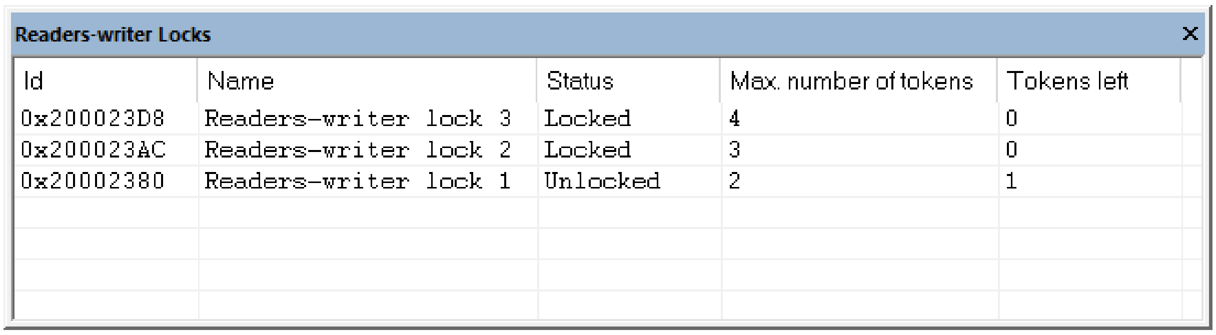
Semaphores

Id	Name	Count	Waiting tasks
0x200023D8	Readers-writer lock 3	0	
0x200023AC	Readers-writer lock 2	0	0x20001C64 (Background Task 1)
0x20002380	Readers-writer lock 1	1	
0x2000251C	Semaphore 0	2	

Column	Description
Id	The semaphore control block address that uniquely identifies a sema-phore.
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.

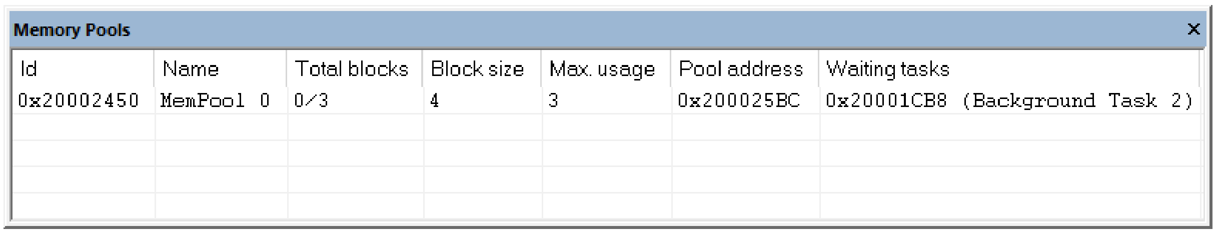


Readers-writer Locks

Id	Name	Status	Max. number of tokens	Tokens left
0x200023D8	Readers-writer lock 3	Locked	4	0
0x200023AC	Readers-writer lock 2	Locked	3	0
0x20002380	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

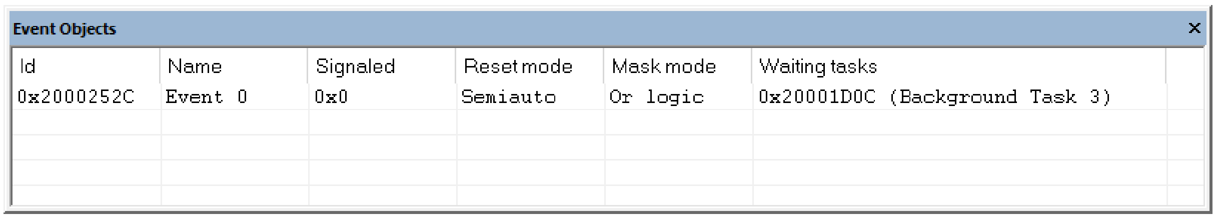

Memory Pools

Id	Name	Total blocks	Block size	Max. usage	Pool address	Waiting tasks
0x20002450	MemPool 0	0/3	4	3	0x200025BC	0x20001CB8 (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.
Pool address	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.



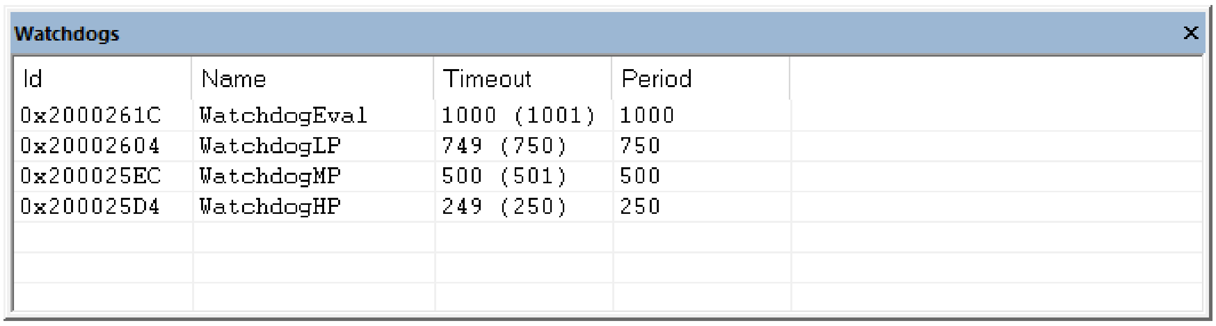
Event Objects

Id	Name	Signaled	Reset mode	Mask mode	Waiting tasks
0x2000252C	Event 0	0x0	Semiauto	Or logic	0x20001D0C (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.

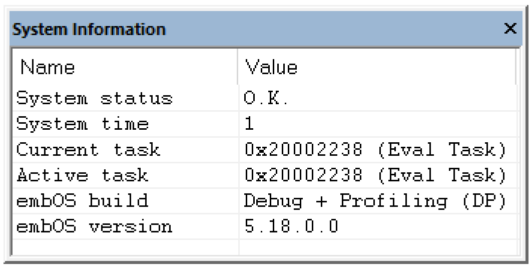


Id	Name	Timeout	Period
0x2000261C	WatchdogEval	1000 (1001)	1000
0x20002604	WatchdogLP	749 (750)	750
0x200025EC	WatchdogMP	500 (501)	500
0x200025D4	WatchdogHP	249 (250)	250

Column	Description
Id	The watchdog control block address that uniquely identifies a watch-dog.
Name	If available, the respective object identifier is shown here.
Timeout	This column shows the remaining time in system ticks and, in paren-thesis, the system time until the watchdog needs to be fed. With em-bOS-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.

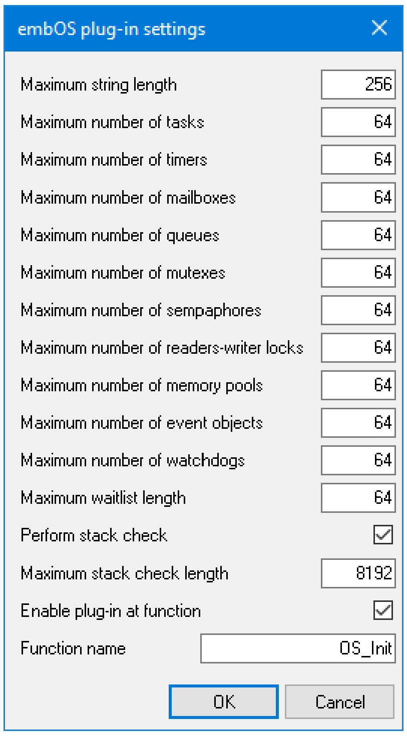


Name	Value
System status	O.K.
System time	1
Current task	0x20002238 (Eval Task)
Active task	0x20002238 (Eval Task)
embOS build	Debug + Profiling (DP)
embOS version	5.18.0.0

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:



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 sempaphores	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	64
Perform stack check	✓
Maximum stack check length	8192
Enable plug-in at function	✓
Function name	OS_Init

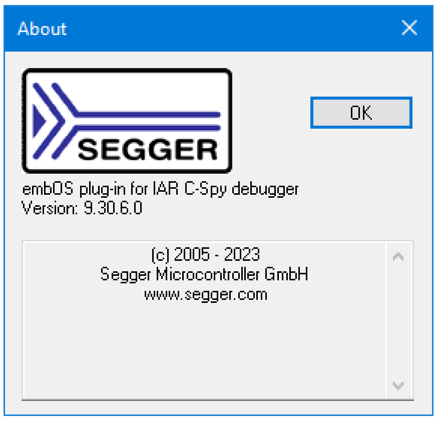
Setting	Permissible values	Description
Maximum string length	1 to 1024	Maximum number of characters to read for each string (e.g. task names).
Maximum number of tasks	1 to 1024	Maximum number of tasks to display in the Task List.
Maximum number of timers	1 to 1024	Maximum number of timers to display in the Timers view.
Maximum number of mailboxes	1 to 1024	Maximum number of mailboxes to display in the Mailboxes view.
Maximum number of queues	1 to 1024	Maximum number of queues to display in the Queues view.
Maximum number of mutexes	1 to 1024	Maximum number of mutexes to display in the Mu-texes view.
Maximum number of semaphores	1 to 1024	Maximum number of semaphores to display in the Semaphores view.
Maximum number of readers-writer locks	1 to 1024	Maximum number of readers-writer locks to display in the Readers-writer Lock view.
Maximum number of memory pools	1 to 1024	Maximum number of memory pools to display in the Memory Pools view.
Maximum number of event objects	1 to 1024	Maximum number of event objects to display in the Event Objects view.
Maximum number of watchdogs	1 to 1024	Maximum number of watchdogs to display in the Watchdogs view.
Maximum waitlist length	1 to 1024	Maximum number of waiting tasks to display in the Mutexes, Semaphores, Mailboxes, Queues, Memory Pools, and Event Objects views.
Perform stack check	☐/☑	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	☐/☑	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	Existing Function	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. However, the entry point function can only be verified during an active debug session. Its validity will be checked as soon as the debug session starts.

The plug-in settings are stored separately for each project in the project’s *.dnx file.

3.14 About

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



embOS plug-in for IAR C-Spy debugger
Version: 9.30.6.0

(c) 2005 - 2023
Segger Microcontroller GmbH
www.segger.com

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_GCC_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/.