μC/OS-II
Kernel Awareness for C-SPY

User Guide

μC/OS-II: V2.81
μC/OS-II Kernel Awareness Plug-in in C-SPY: V2.10
IAR Embedded Workbench: 4.6

www.Micrium.com
Contents

CONTENTS ..............................................................................................................................................................2

INTRODUCTION ..........................................................................................................................................................3

μC/OS-II Kernel Awareness in C-SPY ........................................................................................................................3
μC/OS-II V2.81 ......................................................................................................................................................3
IAR Embedded Workbench and C-SPY Debugger ..................................................................................................3

QUICK START ........................................................................................................................................................4

INSTALLATION ........................................................................................................................................................5

Requirements ......................................................................................................................................................5
Setup .................................................................................................................................................................5
Configuring the Simulator ................................................................................................................................5

USING THE PLUG-IN ...............................................................................................................................................7

Starting the Plug-In ...............................................................................................................................................7
Updating Data ..................................................................................................................................................7
List Window Controls ...................................................................................................................................8
Window Management ...................................................................................................................................8
Disabled Features ...........................................................................................................................................8
Kernel Object Names .......................................................................................................................................9

REFERENCE ..........................................................................................................................................................10

Display Conventions .....................................................................................................................................10
Status ..............................................................................................................................................................10
Task List ..........................................................................................................................................................13
Timer List ........................................................................................................................................................17
Semaphore List ...............................................................................................................................................19
Mutex List .......................................................................................................................................................20
Mailbox List ...................................................................................................................................................21
Queue List .......................................................................................................................................................22
Event Flag Groups .........................................................................................................................................23
Memory Partitions .........................................................................................................................................24
Config. Constants ........................................................................................................................................25
Options ..........................................................................................................................................................26
About ............................................................................................................................................................27

BIBLIOGRAPHY ................................................................................................................................................28

CONTACTS ..........................................................................................................................................................28
Introduction

This User Guide describes how to use the µC/OS-II kernel awareness capabilities with the IAR Embedded Workbench C-SPY Debugger.

- The Quick Start section helps you start using it right away.
- The Installation section explains the installation process in detail.
- The Reference section provides a comprehensive description of all the features.

µC/OS-II Kernel Awareness in C-SPY

The µC/OS-II Kernel Awareness is added to the C-SPY Debugger as a Plug-in which is automatically loaded when the Debugger is started. The Plug-in can be enabled or disabled in the Project's Options under the Debugger's Plugins tab. Version 2.10 of the µC/OS-II Kernel Awareness Plug-in is compatible with version 4.6 (and higher) of the IAR Embedded Workbench.

Kernel Awareness allows you to display µC/OS-II's internal data structures in a convenient series of Windows integrated with the C-SPY Debugger within the IAR Embedded Workbench. This provides you with information about each of the active tasks in the target application, about each semaphore, mutex, mailbox, queue and event flag group along with a list of all the tasks waiting on these kernel objects, and more. This can become very useful to the embedded developer when testing and debugging applications.

µC/OS-II V2.81

Although previous versions of µC/OS-II provided many features to support kernel awareness, as of µC/OS-II V2.6, a name can be assigned to each kernel object, such as a task, a semaphore, a mutex, a mailbox, a queue, an event flag group, a memory partition and a timer. A kernel aware debugger can thus display the name of these objects, and allow you to quickly see information about these objects. Also, V2.6 allows the debugger to obtain the configuration of an application. V2.81 integrates timer management.

IAR Embedded Workbench and C-SPY Debugger

The IAR Embedded Workbench is a powerful Integrated Development Environment that allows you to develop and manage a complete embedded application project for a variety of target processors in a convenient Windows interface. This IDE is the framework where all necessary tools are integrated: a C/EC++ compiler, an assembler, a linker, an editor, a project manager, and the C-SPY™ Debugger.

The IAR C-SPY Debugger is a high-level language debugger for embedded applications. It is designed for use with the IAR compilers and assemblers, and is completely integrated in the IAR Embedded Workbench IDE, providing seamless switching between development and debugging. Some C-SPY Debuggers are available in a simulator, emulator, and ROM-monitor versions. The simulator version simulates the functions of the target processor entirely in software. The Emulator version provides control over an in-circuit emulator, which is connected to the host computer. The ROM-monitor version provides a low-cost solution to real-time debugging.

C-SPY can be extended with Plug-ins to provide Kernel Awareness capabilities during debugging. This is what the µC/OS-II Kernel Awareness Plug-in for C-SPY provides.
Quick Start

These are the minimum steps necessary to start using the µC/OS-II Kernel Awareness Plug-in with the IAR Embedded Workbench.

1. **Run Setup.exe**
   - Follow the setup instructions to install the µC/OS-II KA Plug-in.

2. **Start the IAR Embedded Workbench**

3. **Open your Project**
   - The project should be for a functional µC/OS-II application.

4. **Start the Debugger**
   - The µC/OS-II menu will appear in the menu bar, unless there were compiler/linker errors.
   - Make sure the application runs for a few hundred clock cycles to allow µC/OS-II to be initialized.

5. **Setup the Simulator (if applicable)**
   - If you are running the target using a Simulator, you need to simulate the clock tick interrupt so that µC/OS-II will function in a useful way. That is, if there are no clock ticks, then µC/OS-II won't work properly.

   For the ARM processor, for example:
   a. Open **Interrupts...** from the **Low level** menu
   b. Select Interrupt "IRQ 1 0x18 CPSR.I"
   c. Set Repeat Interval = 2500, Latency = 50, Probability = 100, Variance = 2
   d. Click **Install**

   This must be done every time the Debugger is started, but fortunately you can use a C-SPY macro to automate this step. (See **Installation** section for more details.)

6. **Open the µC/OS-II kernel awareness Status window**
   - Note: This is the first item in the µC/OS-II menu.
   - Verify that µC/OS-II is detected.
   - Click **Update All** to make the information reflect the current state of µC/OS-II.

7. **Open other µC/OS-II kernel awareness windows**
   - There is a specific window for each type of µC/OS-II kernel object, as well as windows for Configuration Constants, Options and About information.
   - µC/OS-II windows are managed the same way as other C-SPY Debugger windows in the IDE workspace. You can also use the features in the **Window** menu to select or organize the windows.

   See **Using the Plug-in** and **Reference** sections for details.
Installation

Requirements
µC/OS-II kernel awareness requires the following:

Software
IAR Embedded Workbench IDE: V4.6 (or higher) for your target processor
µC/OS-II: V2.81 (or higher) with a port for your target processor

Processors
µC/OS-II kernel awareness should work with any processor as long it is supported by C-SPY.

Environment
Windows 95, Windows 98, Windows NT, Window 2000, or Windows XP

Setup
Run Setup.exe and follow the instructions.

If you have multiple installations of the IAR Embedded Workbench, the Plug-in must be installed on each one for which you wish to use the µC/OS-II KA Plug-in. It must also be installed on each product family (of a EW installation) for which you wish to use the µC/OS-II KA Plug-in.

The installation process will copy the Plug-in DLL under the plugins directory of each selected IAR Embedded Workbench installation path.

Configuring the Simulator
If you are running your application using a Simulator, you need to simulate 'clock tick' interrupts so that µC/OS-II will function in a useful way. That is, if there are no clock ticks, the µC/OS-II tasks won't be able to delay or timeout.

To avoid configuring the clock tick interrupts every time the Debugger is started (as described in Quick Start section), you can create a C-SPY macro to perform this task automatically.

1. Create a macro file, which you can name "irq-clock.mac", or edit an existing macro file for your target, and add this to it:

   ```
   execUserSetup()
   {
   __orderInterrupt("0x18",10000,2500,2,50,100);
   }
   ```

   You can set the interrupt parameters that are appropriate for your context. In the example above:
   • 0x18 is the interrupt vector (used for the ARM processor)
   • 10000 is the Activation Time
• 2500 is the Repeat Interval. A smaller interval may cause high CPU usage, and even prevent user code from being executed since most of the time would be spent in the OS. The right balance depends on the performance of your computer. A value of 10000 works well on a Pentium IV running at 1GHz.
• 2 is the Variance
• 50 is the Latency
• 100 is the Probability
(See the IAR Embedded Workbench IDE User Guide for more information)

2. Place the macro file in your project directory (or in a more central location for all your projects).

3. In your IAR EW project, under Project .. Options .. Debugger .. Setup, enable Use macro file and click the browse button to select your macro file. This causes the macro file to execute automatically every time the Debugger is started. This should be indicated by an entry in the Debug Log window stating "Loaded macro file: <path>\irq-clock.mac"
Using the Plug-in

This section describes how to use the features of the µC/OS-II Kernel Awareness Plug-in. The
detailed description for each window is found in the Reference section.

Starting the Plug-in

In the IAR Embedded Workbench, open your project containing a functional µC/OS-II application.
When you start the Debugger (i.e. Project .. Debug), the µC/OS-II menu will appear in the menu
bar, unless there were compiler/linker errors.

You can now open any of the µC/OS-II windows from the µC/OS-II menu.

You should first check the status of µC/OS-II using the Status window, which will indicate if the
application is using µC/OS-II and if µC/OS-II is running. See Status on page 10 for details.

Typically, the debugger will initially break upon entering the main() function of the application. At
this point, µC/OS-II is not initialized, so its status should be "Not Running" but you will be able to
see its version. Also, you can see the application's configuration constants in the Config. Constants
window, which are useful if you need to diagnose how µC/OS-II is configured.

Updating Data

The kernel data will not be available until µC/OS-II has had time to initialize it. In other words,
µC/OS-II's data structures will be initialized only when OSInit() is executed. Because of this, the
information windows will be empty until data is available. Depending on your application, it may
take a few hundred clock cycles to have all the data.

You can monitor the changes in kernel data in different ways:

1. Set breakpoints in the application. If the Auto Update option is enabled (see Options on page
   26 for details), the kernel data will be re-read each time a breakpoint is reached, and will be
   used to refresh the contents of the windows.

2. Force a Break while the application is running (i.e. Debug .. Break), causing the same effect
   as a normal breakpoint.

3. Click Update All in the Status window while the application is running, or select Update
   from the context menu of any of the List windows.

Each time, the contents of the open windows change to reflect the current data in µC/OS-II. For
example, you will see the Time (ticks) value increment in the Status window.
List Window Controls

Most µC/OS-II windows are List windows showing columns of information for each item in the list. These windows have the following special features:

AutoFit Columns
The width of each column is automatically adjusted to fit its content whenever the window is refreshed. The column's width is set to fit the text in the largest item. This feature can be disabled from the context menu.

Sort
List items can be sorted by clicking on the header of the column to use as sorting criteria. Clicking again toggles the sort order between ascending and descending.

Context Menu
Right-clicking anywhere in the window opens the context menu. Every list window has the following commands in the context menu:

- **AutoFit Columns**: Enables/Disables *AutoFit Columns* feature for this window.
- **Refresh**: Redraws the window contents (using AutoFit if needed) with the currently known target data (i.e. target data is not re-read from the target).
- **Update**: Re-reads all the data from the target, and force an update of all the µC/OS-II windows. This is the same as *Update All* in the *Status* window.

Additional commands may be appended to the end of the menu for specific windows. At this time, only the Task List window has extra context menu commands (See *Task List* on page 12).

Context menu option selections are lost when the window is closed. These are restored to defaults when the window is opened again. However, the options are preserved when the debugger is stopped while the window is open.

Window Management

When the Debugger is re-started, the µC/OS-II windows that were open when it was stopped are restored, and their settings such as sorting and AutoFit are restored.

Disabled features

Most of the features of µC/OS-II can be disabled using Configuration Constants. If major features (Semaphores, Mutexes, Mailboxes, Queues, Event Flag Groups, Memory Partitions, Timers) are disabled, their corresponding window will show a message like this one for the Semaphore List:

```
Semaphore functionality disabled.
To enable, set OS_SEM_EN to 1 in OS_CFG.H
```

Other features can be disabled which will cause some kernel data to be unavailable. In such cases, the information will be left blank, 'n/a' or '?'. For example, if OS_TASK_STAT_EN is 0 then CPU Usage will remain n/a in the Status window. Also, most Stack statistics will not be available in the Task List window if OS_TASK_PROFILE_EN or OS_TASK_CREATE_EXT_EN are 0.
Kernel Object Names

As of V2.6, µC/OS-II allows you to assign names to kernel objects to help identify them in Kernel Awareness tools:

There are five(5) types of kernel object that can be named:

<table>
<thead>
<tr>
<th>Kernel Object Type</th>
<th>Name-Size Constant</th>
<th>Function used to set the name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td>OS_TASK_NAME_SIZE</td>
<td>OSTaskNameSet()</td>
</tr>
<tr>
<td>Event (Sem., Mutex, Mailbox, Queue)</td>
<td>OS_EVENT_NAME_SIZE</td>
<td>OSFlagNameSet()</td>
</tr>
<tr>
<td>Event Flag Group</td>
<td>OS_FLAG_NAME_SIZE</td>
<td>OSFlagNameSet()</td>
</tr>
<tr>
<td>Memory Partition</td>
<td>OS_MEM_NAME_SIZE</td>
<td>OSMemNameSet()</td>
</tr>
<tr>
<td>Timer</td>
<td>OS_TMR_CFG_NAME_SIZE</td>
<td>OSTmrStart()</td>
</tr>
</tbody>
</table>

You can assign a name to an object if its name-size configuration constant is greater than 0. This constant actually establishes the number of characters allowed for the object names and must account for a NUL-terminated ASCII string. Object names are assigned to each object after the object is created by calling its corresponding OS..NameSet() function (refer to the µC/OS-II release notes for details).

Task names are shown in the Name column of the Task List window and in the Tasks Waiting column of the Semaphore, Mutex, Mailbox, Queue List windows, as well as in the Event Flag Group window. Event names are used in the Name column of their corresponding window and in the Waiting On column of the Task List window.
Reference

This section contains the description of each µC/OS-II Kernel Awareness window with a definition for each item of information.

Display Conventions

Address values are expressed in hexadecimal format. The width of an address value depends on the target processor's addressing capabilities (2, 3 or 4 bytes), which is indicated by OSPtrSize.

Numerical values are displayed with a right-justified alignment, while text values are displayed with left-justification.

Screen output may differ from what is seen here, depending on the system and its configuration.

Status

The Status window shows general information concerning µC/OS-II and contains general controls.

Information

Status

The status of µC/OS-II can be one of the following:

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not Detected</td>
<td>µC/OS-II code not present in target application</td>
</tr>
<tr>
<td>Debug Disabled</td>
<td>OSDiagEn = 0. Debug Mode must be enabled for kernel awareness</td>
</tr>
<tr>
<td>Not Running</td>
<td>µC/OS-II has not started running (Not enough clock cycles occurred to be initialized)</td>
</tr>
<tr>
<td>Running</td>
<td>µC/OS-II is running</td>
</tr>
</tbody>
</table>
**Version**
Current version of µC/OS-II in target application. If µC/OS-II is not detected or debug mode is disabled, then the version shows V?.??. Version 2.62 and above is necessary for Kernel Awareness to be functional.

**Statistics: Ready / Not Ready**
Based on the value of OSStatRdy, indicating if the Statistics task is ready. This is only relevant if OS_TASK_STAT_EN is set.

**CPU Usage**
Percentage of CPU used. (OSCPUUsage)

**Tasks**
Total number of tasks running, i.e. includes system tasks. (OSTaskCtr)

**Idle Counter**
Idle counter. (OSIdleCtr)
This counter is reset by the **Reset Counters** feature.

**Context Switches**
Number of context switches. (OSCtxSwCtr)
This counter is reset by the **Reset Counters** feature.

**Nesting – Interrupt**
Interrupt nesting level. (OSIntNesting)

**Nesting - Multitask Lock**
Multitasking lock nesting level. (OSLockNesting)

**Step Mode**
Indicates the state of the tick step feature:

<table>
<thead>
<tr>
<th>Value</th>
<th>OSTickStepState</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disabled</td>
<td>0</td>
<td>Stepping is disabled; Tick runs as normal</td>
</tr>
<tr>
<td>Waiting</td>
<td>1</td>
<td>Waiting for µC/OS-View to set OSTickStepState to _ONCE</td>
</tr>
<tr>
<td>Stepped</td>
<td>2</td>
<td>Process tick once and wait for next command from µC/OS-View</td>
</tr>
<tr>
<td>Unknown</td>
<td>any other value</td>
<td>Non-supported value</td>
</tr>
</tbody>
</table>

**Time (ticks)**
Current value of system time (in ticks). (OSTime)

**Timer Time**
Current value of timer time. (OSTmrTime)
It increments every OS_TICKS_PER_SEC / OS_TMR_CFG_TICKS_PER_SEC.

**Used Timers**
Number of timers used. (OSTmrUsed)

**Free Timers**
Number of timers in the list of unused timers. (OSTmrFree)
**Controls**

*Update All*
Updates all the µC/OS-II data (by reading from the target) and forces a refresh of all the windows to show the new data. This is automatically done when the target stops at a breakpoint, but it is also permitted to perform an update while the target is running, although you must be aware that the data may be in transition while it is being read, and may produce some inconsistencies.

*Reset Counters*
Resets (to 0) the global Context Switches counter (OSCtxSwCtr) and the global Idle counter (OSIdleCtr), as well as the Context Switches counter (.OSTCBCtxSwCtr) of every task.
Task List

The Task List window shows information about each task running, as well as its Stack information.

<table>
<thead>
<tr>
<th>Name</th>
<th>Ref</th>
<th>Prio</th>
<th>State</th>
<th>Dly</th>
<th>Waiting On</th>
<th>Brk</th>
<th>Cnt</th>
<th>Src</th>
<th>Sbrk</th>
<th>Start P</th>
<th>Memo</th>
<th>Curr</th>
<th>Max</th>
<th>Cur</th>
<th>Bits</th>
<th>Starts</th>
<th>Ends</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start Task</td>
<td>5</td>
<td>5</td>
<td>Dly</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00100100</td>
<td>21</td>
<td>15</td>
<td>100</td>
<td>100</td>
<td>512</td>
<td>00100004</td>
<td>00100004</td>
</tr>
<tr>
<td>Task D</td>
<td>5</td>
<td>5</td>
<td>Dly</td>
<td>55</td>
<td></td>
<td>111</td>
<td>00101018</td>
<td>21</td>
<td>17</td>
<td>105</td>
<td>92</td>
<td>513</td>
<td>00101004</td>
<td>00101004</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Serial #1</td>
<td>5</td>
<td>10</td>
<td>Dly</td>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00100354</td>
<td>23</td>
<td>23</td>
<td>124</td>
<td>112</td>
<td>512</td>
<td>00100004</td>
<td>00100004</td>
</tr>
<tr>
<td>Serial #2</td>
<td>4</td>
<td>13</td>
<td>Dly</td>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00100594</td>
<td>24</td>
<td>24</td>
<td>124</td>
<td>112</td>
<td>513</td>
<td>00100004</td>
<td>00100004</td>
</tr>
<tr>
<td>Display</td>
<td>5</td>
<td>14</td>
<td>Dly</td>
<td>0</td>
<td>0 Bem. #2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0010079C</td>
<td>20</td>
<td>20</td>
<td>104</td>
<td>104</td>
<td>512</td>
<td>00100004</td>
<td>00100004</td>
</tr>
<tr>
<td>Midi Control</td>
<td>5</td>
<td>16</td>
<td>Dly</td>
<td>15</td>
<td></td>
<td>370</td>
<td>001009AC</td>
<td>17</td>
<td>17</td>
<td>88</td>
<td>88</td>
<td>512</td>
<td>00100004</td>
<td>00100004</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network</td>
<td>7</td>
<td>18</td>
<td>Dly</td>
<td>5</td>
<td></td>
<td>185</td>
<td>0010CB4C</td>
<td>17</td>
<td>17</td>
<td>88</td>
<td>88</td>
<td>512</td>
<td>00100004</td>
<td>00100004</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Keyboard</td>
<td>8</td>
<td>30</td>
<td>Q</td>
<td>0</td>
<td>Queue #1</td>
<td>463</td>
<td>0010CF7C</td>
<td>24</td>
<td>24</td>
<td>136</td>
<td>136</td>
<td>512</td>
<td>00100004</td>
<td>00100004</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analog In</td>
<td>9</td>
<td>33</td>
<td>Q</td>
<td>0</td>
<td>Queue #2</td>
<td>555</td>
<td>0010CF7C</td>
<td>24</td>
<td>24</td>
<td>136</td>
<td>136</td>
<td>512</td>
<td>00100004</td>
<td>00100004</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Discrete I/O</td>
<td>10</td>
<td>34</td>
<td>Dly</td>
<td>78</td>
<td></td>
<td>69</td>
<td>00101108</td>
<td>17</td>
<td>17</td>
<td>92</td>
<td>92</td>
<td>512</td>
<td>00100104</td>
<td>00100104</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User #1</td>
<td>11</td>
<td>35</td>
<td>Dly</td>
<td>78</td>
<td></td>
<td>63</td>
<td>00101138</td>
<td>28</td>
<td>28</td>
<td>148</td>
<td>148</td>
<td>512</td>
<td>00100104</td>
<td>00100104</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modbus</td>
<td>12</td>
<td>28</td>
<td>Flag</td>
<td>78</td>
<td>Flag #1</td>
<td>63</td>
<td>0010157D</td>
<td>28</td>
<td>28</td>
<td>148</td>
<td>148</td>
<td>512</td>
<td>00101504</td>
<td>00101504</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G75</td>
<td>13</td>
<td>30</td>
<td>Flag</td>
<td>65</td>
<td>Flag #1</td>
<td>56</td>
<td>00101779</td>
<td>24</td>
<td>24</td>
<td>148</td>
<td>148</td>
<td>512</td>
<td>00101504</td>
<td>00101504</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task C</td>
<td>14</td>
<td>32</td>
<td>Muxex</td>
<td>0</td>
<td>Muxex #1</td>
<td>84</td>
<td>00101598</td>
<td>24</td>
<td>24</td>
<td>148</td>
<td>148</td>
<td>512</td>
<td>00101504</td>
<td>00101504</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task E</td>
<td>15</td>
<td>36</td>
<td>Dly</td>
<td>83</td>
<td></td>
<td>74</td>
<td>00101B4C</td>
<td>17</td>
<td>17</td>
<td>88</td>
<td>88</td>
<td>512</td>
<td>00101004</td>
<td>00101004</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task F</td>
<td>17</td>
<td>38</td>
<td>Mbox</td>
<td>0</td>
<td>Mbox #1</td>
<td>113</td>
<td>00101F78</td>
<td>27</td>
<td>27</td>
<td>140</td>
<td>140</td>
<td>512</td>
<td>00102004</td>
<td>00102004</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>uC/OS-II Stat</td>
<td>1</td>
<td>62</td>
<td>Ready</td>
<td>0</td>
<td></td>
<td>663</td>
<td>0010263D</td>
<td>22</td>
<td>22</td>
<td>156</td>
<td>156</td>
<td>512</td>
<td>00102500</td>
<td>00102500</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>uC/OS-II Idle</td>
<td>0</td>
<td>63</td>
<td>Ready</td>
<td>0</td>
<td></td>
<td>1613</td>
<td>00102854</td>
<td>25</td>
<td>25</td>
<td>132</td>
<td>132</td>
<td>72</td>
<td>0010010C</td>
<td>0010010C</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Information

Current Task
The first column indicates the currently running task, with a ‘>’ symbol.

Name
Name of the task. If it has no name, it is set to '?'. See Kernel Object Names on page 9.

Ref
Index of the task in OSTCBtbl[]. This also corresponds to the order in which the tasks were created.

Prio
Priority assigned to each task. The default sort criteria for the Task List is the Priority in ascending order. (.OSTCBPrio) Priority values range from 0 (highest) to 63 (lowest).
**State**

State of the task. The possible state values are:

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
<th>Value of .OSTCBStat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ready</td>
<td>Ready to run</td>
<td>OS_STAT_RDY</td>
</tr>
<tr>
<td>Dly</td>
<td>Waiting for time to expire</td>
<td>OS_STAT_RDY</td>
</tr>
<tr>
<td></td>
<td>but .OSTCBDly is non-zero</td>
<td></td>
</tr>
<tr>
<td>Sem</td>
<td>Waiting on a semaphore</td>
<td>OS_STAT_SEM</td>
</tr>
<tr>
<td>Mutex</td>
<td>Waiting on a mutual exclusion semaphore</td>
<td>OS_STAT_MUTEX</td>
</tr>
<tr>
<td>Flag</td>
<td>Waiting on an event flag group</td>
<td>OS_STAT_FLAG</td>
</tr>
<tr>
<td>Mbox</td>
<td>Waiting for a message at a mailbox</td>
<td>OS_STAT_MBOX</td>
</tr>
<tr>
<td>Q</td>
<td>Waiting for a message at a queue</td>
<td>OS_STAT_Q</td>
</tr>
<tr>
<td>Sem+Suspended</td>
<td>Waiting on a semaphore and task is also suspended</td>
<td>OS_STAT_SEM + OS_STAT_SUSPEND</td>
</tr>
<tr>
<td>Mutex+Suspended</td>
<td>Waiting on a mutual exclusion semaphore and the task is also suspended</td>
<td>OS_STAT_MUTEX + OS_STAT_SUSPEND</td>
</tr>
<tr>
<td>Flag+Suspended</td>
<td>Waiting on an event flag group and the task is also suspended</td>
<td>OS_STAT_FLAG + OS_STAT_SUSPEND</td>
</tr>
<tr>
<td>Mbox+Suspended</td>
<td>Waiting for a message at a mailbox and the task is also suspended</td>
<td>OS_STAT_MBOX + OS_STAT_SUSPEND</td>
</tr>
<tr>
<td>Q+Suspended</td>
<td>Waiting for a message at a queue and the task is also suspended</td>
<td>OS_STAT_Q + OS_STAT_SUSPEND</td>
</tr>
</tbody>
</table>

**Dly**

Amount of time (in ticks) the task has been delayed (if the State column indicates 'Dly') or, the amount of time left that the task will be waiting for either the semaphore, the mutex, the event flag group, the mailbox or the queue (if the State column indicates an object type). The value is 0 if the task will wait forever for one of the objects. (.OSTCBDly)

**Waiting On**

Name of the object (if any) for which the task is waiting. This can be either an Event Flag Group or an Event (Semaphore, Mutex, Mailbox, or Queue).

**Msg**

Message received from OSMboxPost() or OSQPost(). This pointer is shown in hexadecimal format. Typically, it will be empty unless you single-step through the code and a 'Post' call deposits a message either to the mailbox or a queue that the task is waiting for. (.OSTCBMsg)

**Ctx Sw**

Number of times the task was 'switched-in'. This counter can be reset to 0 by selecting Reset Counters from the context menu, or by clicking the Reset Counters button in the Status window. This counter is only available if you set the configuration constant OS_TASK_PROFILE_EN to 1 which should be done when you are using the kernel awareness feature of µC/OS-II. (.OSTCBCtxSwCtr)

**Stk Ptr**

Current value of the task’s stack pointer (in hexadecimal).
### Notes on Stack Statistics:

- The following Stack-related information fields can be disabled with the **Stack Stats** feature in the context menu or in the **Options** window.
- Most of these values assume that tasks were created with `OSTaskCreateExt()`, specifying `OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR` for the `opt` argument. Also, you need to enable the Statistics task (set `OS_TASK_STAT_EN` to 1 in `OS_CFG.H`), and enable stack checking by the Statistics task (set `OS_TASK_STAT_STK_CHK_EN` to 1).

---

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| Max%  | Maximum stack space used by the task expressed as a percentage. For example, a value of 47% means that, during execution of the task, the total stack space used never exceeded 47%. This value is reset to 0 by the **Reset StkUsed** feature of the context menu.  
\[
\text{Max} = \frac{\text{.OSTCBStkUsed}}{\text{.OSTCBStkSize} \times \text{OSStkWidth}}
\] |

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| Cur%  | Current stack usage of the task expressed as a percentage. For example, a value of 39% means that the stack pointer is currently located 39% into the stack.  
\[
\text{Cur} = \frac{\text{abs(.OSTCBStkPtr} - \text{.OSTCBStkBase})}{\text{.OSTCBStkSize} \times \text{OSStkWidth}}
\] |

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max</td>
<td>Maximum stack space used by the task (in bytes). This value is reset to 0 by the <strong>Reset StkUsed</strong> feature of the context menu. (.OSTCBStkUsed)</td>
</tr>
</tbody>
</table>
| Cur   | Current stack usage of the task (in bytes).  
\[
\text{Cur} = \text{abs(.OSTCBStkPtr} - \text{.OSTCBStkBase})
\] |

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| Size  | Number of bytes allocated for the task stack.  
\[
\text{Size} = \text{.OSTCBStkSize} \times \text{OSStkWidth}
\] |

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Starts @</td>
<td>Address of the beginning of the stack. If the stack, on the processor you are using, grows downwards (i.e. <code>OS_STK_GROWTH</code> set to 1 in <code>OS_CPU.H</code>) then this indicates the highest address that the stack pointer can take, otherwise (i.e. <code>OS_STK_GROWTH</code> set to 0), this indicates the lowest address the stack pointer can take. (.OSTCBStkBase)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ends @</td>
<td>Address of the end of the stack. If the stack, on the processor you are using, grows downwards (i.e. <code>OS_STK_GROWTH</code> set to 1 in <code>OS_CPU.H</code>) then this indicates the lowest address that the stack pointer can take, otherwise (i.e. <code>OS_STK_GROWTH</code> set to 0), this indicates the highest address the stack pointer can take.</td>
</tr>
</tbody>
</table>
Controls

In addition to the standard context menu features (described in Using the Plug-in section), the following features are available in the Task List context menu:

- **StackStats**: Shows/Hides the Stack-related information fields (except Stk Ptr). This is the same as in the Options window.

- **Reset Counters**: This has the same effect as the Reset Counters button of the Status window.

- **Reset StkUsed**: Resets (to 0) the Stack Used counter (.OSTCBStkUsed) of every task, represented in the Max% and Max columns.
## Timer List

The Timer window shows information about the timers (OS_TMR) in the pool of timers. It is based on a representation of the timer manager 'wheel' (OSTmrWheelTbl[]).

<table>
<thead>
<tr>
<th>Spoke#</th>
<th>#Timers</th>
<th>Timer Name</th>
<th>Match</th>
<th>Option</th>
<th>Delay</th>
<th>Period</th>
<th>Callback</th>
<th>CallbackArg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>Timer 01</td>
<td>497</td>
<td>Periodic</td>
<td>0</td>
<td>52</td>
<td>000003F41</td>
<td>00000001</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Timer 03</td>
<td>497</td>
<td>Periodic</td>
<td>0</td>
<td>124</td>
<td>000003F41</td>
<td>00000003</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Timer 07</td>
<td>497</td>
<td>Periodic</td>
<td>0</td>
<td>240</td>
<td>000003F41</td>
<td>00000007</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Timer 15</td>
<td>497</td>
<td>Periodic</td>
<td>0</td>
<td>496</td>
<td>000003F41</td>
<td>0000000F</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>Timer 02</td>
<td>466</td>
<td>Periodic</td>
<td>0</td>
<td>31</td>
<td>000003F41</td>
<td>00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Timer 03</td>
<td>466</td>
<td>Periodic</td>
<td>0</td>
<td>98</td>
<td>000003F41</td>
<td>00000002</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Timer 04</td>
<td>466</td>
<td>Periodic</td>
<td>0</td>
<td>195</td>
<td>000003F41</td>
<td>00000004</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Timer 14</td>
<td>466</td>
<td>Periodic</td>
<td>0</td>
<td>465</td>
<td>000003F41</td>
<td>0000000E</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>Timer 12</td>
<td>603</td>
<td>Periodic</td>
<td>0</td>
<td>394</td>
<td>000003F41</td>
<td>0000000D</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>Timer 12</td>
<td>807</td>
<td>Periodic</td>
<td>0</td>
<td>408</td>
<td>000003F41</td>
<td>0000000C</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>Timer 11</td>
<td>745</td>
<td>Periodic</td>
<td>0</td>
<td>372</td>
<td>000003F41</td>
<td>0000000B</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>Timer 10</td>
<td>631</td>
<td>Periodic</td>
<td>0</td>
<td>541</td>
<td>000003F41</td>
<td>0000000A</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>Timer 06</td>
<td>652</td>
<td>Periodic</td>
<td>0</td>
<td>217</td>
<td>000003F41</td>
<td>00000006</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>Timer 09</td>
<td>621</td>
<td>Periodic</td>
<td>0</td>
<td>316</td>
<td>000003F41</td>
<td>00000009</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>Timer 05</td>
<td>555</td>
<td>Periodic</td>
<td>0</td>
<td>186</td>
<td>000003F41</td>
<td>00000005</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Timer 08</td>
<td>555</td>
<td>Periodic</td>
<td>0</td>
<td>279</td>
<td>000003F41</td>
<td>00000008</td>
</tr>
</tbody>
</table>

### Information

**Spoke#**

Spoke number in the wheel. (0 to OS_TMR_CFG_WHEEL_SIZE-1)

**#Timers**

Number of timers in this Spoke. (.OSTmrEntries of OS_TMR_WHEEL)

**Timer Name**

Name of the timer. If it has no name, it is set to "?". See Kernel Object Names on page 9.

**Match**

Value used by the timer manager to determine if a timer has expired. A timer expires when OSTmrTime == .OSTmrMatch
**Option**
Type of timer.

<table>
<thead>
<tr>
<th>Description</th>
<th>Value of <code>OSTmrOpt</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>One-Shot</td>
<td><code>OS_TMR_OPT_ONE_SHOT</code></td>
</tr>
<tr>
<td>Periodic</td>
<td><code>OS_TMR_OPT_PERIODIC</code></td>
</tr>
</tbody>
</table>

**Delay**
Time before the first signaling of a periodic timer (in timer time).

**Period**
Period with which the timer will repeat (in timer time).

**Callback**
Pointer to the function to call when timer expires.

**CallbackArg**
Argument to pass to the callback function when the timer expires.

**Controls**
There are no special controls for this window in addition to the standard list window controls. See *List Window Controls* on page 8 for details.
Semaphore List

The Semaphore window shows information about OS_EVENT structures that were created as semaphores.

<table>
<thead>
<tr>
<th>Name</th>
<th>Ref</th>
<th>Count</th>
<th>Tasks Waiting</th>
<th>OS_EVENT @</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sem. #1</td>
<td>0</td>
<td>1</td>
<td></td>
<td>00102110</td>
</tr>
<tr>
<td>Sem. #2</td>
<td>1</td>
<td>0</td>
<td>14-Display</td>
<td>001021D4</td>
</tr>
</tbody>
</table>

**Information**

**Name**
Name of the semaphore. If it has no name, it is set to "?". See *Kernel Object Names* on page 9.

**Ref**
Index of the semaphore structure in OSEventTbl[]. This also corresponds to the order in which the events were created.

**Count**
Value of the semaphore interpreted from .OSEventCnt.

**Tasks Waiting**
Tasks waiting on the semaphore. A task is represented by its Priority followed by its Name (if any). If more than one task is waiting on the same semaphore, additional rows are added for the same semaphore with duplicate information in the common columns.

**OS_EVENT @**
Address of the OS_EVENT structure.

**Controls**
There are no special controls for this window in addition to the standard list window controls. See *List Window Controls* on page 8 for details.
**Mutex List**

The Mutex window shows information about OS_EVENT structures that were created as Mutual exclusion semaphores.

<table>
<thead>
<tr>
<th>Name</th>
<th>Ref</th>
<th>PIP:Owner</th>
<th>Tasks Waiting</th>
<th>OS_EVENT @</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutex #1</td>
<td>6</td>
<td>8:34</td>
<td>Task C</td>
<td>001022D8</td>
</tr>
<tr>
<td>Mutex #2</td>
<td>7</td>
<td>9:255</td>
<td></td>
<td>0010230C</td>
</tr>
</tbody>
</table>

**Information**

**Name**
Name of the mutex. If it has no name, it is set to '?'. See *Kernel Object Names* on page 9.

**Ref**
Index of the mutex structure in OSEventTbl[]. This also corresponds to the order in which the events were created.

**PIP-Owner (high-byte - low-byte)**
The first value is the PIP (Priority Inheritance Priority) of the mutex, interpreted from the 'upper' eight bits of .OSEventCnt. The second value is the priority of the task that owns the mutex or 255 if the mutex is available (i.e. not owned), interpreted from the 'lower' eight bits of .OSEventCnt.

**Tasks Waiting**
Tasks waiting on the mutex. A task is represented by its Priority followed by its Name (if any). If more than one task is waiting on the same mutex, additional rows are added for the same mutex with duplicate information in the common columns.

**OS_EVENT @**
Address of the OS_EVENT structure.

**Controls**
There are no special controls for this window in addition to the standard list window controls. See *List Window Controls* on page 8 for details.
Mailbox List

The Mailbox window shows information about OS_EVENT structures that were created as mailboxes.

<table>
<thead>
<tr>
<th>Name</th>
<th>Ref</th>
<th>Msg</th>
<th>Tasks Waiting</th>
<th>OS_EVENT @</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mailbox #1</td>
<td>2</td>
<td>C0000000</td>
<td>38-Task F</td>
<td>00102208</td>
</tr>
<tr>
<td>Mailbox #2</td>
<td>3</td>
<td>C0000001</td>
<td></td>
<td>0010223C</td>
</tr>
</tbody>
</table>

Information

Name
Name of the mailbox. If it has no name, it is set to '?'. See Kernel Object Names on page 9.

Ref
Index of the mailbox structure in OSEventTbl[]. This also corresponds to the order in which the events were created.

Msg
Current contents of the mailbox. This value is the pointer (.OSEventPtr) to the message.

Tasks Waiting
Tasks waiting on the mailbox. A task is represented by its Priority followed by its Name (if any). If more than one task is waiting on the same mailbox, additional rows are added for the same mailbox with duplicate information in the common columns.

OS_EVENT @
Address of the OS_EVENT structure.

Controls
There are no special controls for this window in addition to the standard list window controls. See List Window Controls on page 8 for details.
Queue List

The Queue window shows information about OS_EVENT structures that were created as message queues.

<table>
<thead>
<tr>
<th>Name</th>
<th>Ref</th>
<th>Entries</th>
<th>Size</th>
<th>Next_Msg</th>
<th>Tasks Waiting</th>
<th>OS_EVENT @</th>
<th>OS_Q @</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue #1</td>
<td>4</td>
<td>0</td>
<td>10</td>
<td>00102068</td>
<td>20-Keyboard</td>
<td>00102270</td>
<td>00103598</td>
</tr>
<tr>
<td>Queue #2</td>
<td>5</td>
<td>0</td>
<td>20</td>
<td>00102050</td>
<td>22-Analog In</td>
<td>001022A4</td>
<td>001035B4</td>
</tr>
</tbody>
</table>

Information

Name
Name of the message queue. If it has no name, it is set to "?”. See Kernel Object Names on page 9.

Ref
Index of the message queue in OSEventTbl[]. This also corresponds to the order in which the events were created.

Entries
Number of messages currently in the message queue. (.OSQEntries in OS_Q of the queue)

Size
Maximum number of entries allowed in the message queue. (.OSQSize in OS_Q of the queue)

Next_Msg
Pointer to the next message available from the queue. Note that if there are no messages in the queue (.OSQEntries is 0), then this value is meaningless because it contains whatever was in that message queue position. (.OSQOut in OS_Q of the queue)

Tasks Waiting
Tasks waiting on the message queue. A task is represented by its Priority followed by its Name (if any). If more than one task is waiting on the same queue, additional rows are added for the same queue with duplicate information in the common columns.

OS_EVENT @
Address of the OS_EVENT structure.

OS_Q @
Address of the OS_Q structure of the queue. (OSEventPtr)

Controls
There are no special controls for this window in addition to the standard list window controls. See List Window Controls on page 8 for details.
Event Flag Groups

The Event Flag Groups window shows information about the `OS_FLAG_GRP` structures.

<table>
<thead>
<tr>
<th>Name</th>
<th>Ref</th>
<th>Flags</th>
<th><code>OS_FLAG_GRP</code> @</th>
<th>Tasks Waiting</th>
<th>Wait Type</th>
<th>Waiting for Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flag #1</td>
<td>0</td>
<td>0000F000</td>
<td>001C23&amp;8</td>
<td>28-Mcdbus</td>
<td>All Set</td>
<td>0000FF00</td>
</tr>
<tr>
<td>Flag #1</td>
<td>0</td>
<td>0000F000</td>
<td>001C23&amp;8</td>
<td>30-5FS</td>
<td>All Set</td>
<td>00000FF0</td>
</tr>
<tr>
<td>Flag #2</td>
<td>1</td>
<td>0000FF00</td>
<td>001C23D4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Information**

**Name**
Name of the event flag group. If it has no name, it is set to '?' . See *Kernel Object Names* on page 9.

**Ref**
Index of the event flag group in `OSFlagTbl[]`. This also corresponds to the order in which the event flag groups were created.

**Flags**
Current value stored in the event flag group. The number flag bits used to store flags depends on the definition of `OS_FLAGS` (in `OS_CFG.H`). It could be 8, 16 or 32 bits wide.

**`OS_FLAG_GRP` @**
Address of the `OS_FLAG_GRP` structure.

**Tasks Waiting**
Tasks waiting on the event flag group. A task is represented by its Priority followed by its Name (if any). If more than one task is waiting on the same event, additional rows are added for the same flag with duplicate information in the common columns (as seen in the example above).

**Wait Type**
Condition for which a task will wait. The possible values are:

<table>
<thead>
<tr>
<th>Wait Type</th>
<th>Description</th>
<th>Value of <code>.OSFlagNodeWaitType</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Any Set</td>
<td>A task will wait for ANY of the bits specified in Waiting For Flags to be SET in Flags.</td>
<td><code>OS_FLAG_WAIT_SET_ANY/OR</code></td>
</tr>
<tr>
<td>All Set</td>
<td>A task will wait for ALL the bits specified in Waiting For Flags to be SET in Flags.</td>
<td><code>OS_FLAG_WAIT_SET_ALL/AND</code></td>
</tr>
<tr>
<td>Any Clr</td>
<td>A task will wait for ANY of the bits specified in Waiting For Flags to be CLEARED in Flags.</td>
<td><code>OS_FLAG_WAIT_CLR_ANY/OR</code></td>
</tr>
<tr>
<td>All Clr</td>
<td>A task will wait for ALL the bits specified in Waiting For Flags to be CLEARED in Flags.</td>
<td><code>OS_FLAG_WAIT_CLR_ALL/AND</code></td>
</tr>
<tr>
<td>+ Consume</td>
<td>Flag will be consumed if condition is satisfied.</td>
<td><code>+ OS_FLAG_CONSUME</code></td>
</tr>
</tbody>
</table>

**Waiting for Flags**
Bits that a task will wait for to be set (or cleared) in the event flag group, depending on *Wait Type*.

**Controls**
There are no special controls for this window in addition to the standard list window controls. See *List Window Controls* on page 8 for details.
Memory Partitions

The Memory Partitions window shows information about the OS_MEM structures.

### Table

<table>
<thead>
<tr>
<th>Name</th>
<th>Ref</th>
<th>Avail%</th>
<th>Used%</th>
<th>#Blk Avail</th>
<th>#Blk Used</th>
<th>#Blk Max</th>
<th>Blk Size</th>
<th>OS_MEM @</th>
<th>Starts @</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition #1</td>
<td>0</td>
<td>50%</td>
<td>10%</td>
<td>9</td>
<td>1</td>
<td>10</td>
<td>8</td>
<td>00103890</td>
<td>00102068</td>
</tr>
<tr>
<td>Partition #2</td>
<td>1</td>
<td>100%</td>
<td>0%</td>
<td>8</td>
<td>0</td>
<td>8</td>
<td>12</td>
<td>001038C4</td>
<td>00102138</td>
</tr>
</tbody>
</table>

### Information

**Name**
Name of the memory partition. If it has no name, it is set to '?'. See Kernel Object Names on page 9.

**Ref**
Index of the memory partition in OSMemTbl[]. This also corresponds to the order in which the memory partitions were created.

**Avail%**
Available memory as a percentage of the memory partition's size.

**Used%**
Used memory as a percentage of the memory partition's size.

**#Blk Avail**
Number of memory blocks available in the memory partition. (.OSMemNFree)

**#Blk Used**
Number of memory blocks in use in the memory partition. (.OSMemNBlks - OSMemNFree)

**#Blk Max**
Number of memory blocks allocated in the memory partition when it was created. (.OSMemNBlks)

**Blk Size**
Size (in bytes) of each memory block. (.OSMemBlkSize)

**OS_MEM @**
Address of the memory partition.

**Starts @**
Address of the beginning of the memory partition (.OSMemAddr) This address is typically the base of the storage area and is the lowest address of the memory partition. Memory blocks in the memory partition have an address between Starts@ and Starts@ + (#BlkMax * BlkSize).

### Controls
There are no special controls for this window in addition to the standard list window controls. See List Window Controls on page 8 for details.
Config. Constants

This window lists the µC/OS-II configuration constants in the target application.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Endianness</td>
<td>Little</td>
</tr>
<tr>
<td>OSDataSize</td>
<td>6296</td>
</tr>
<tr>
<td>OSDbugEn</td>
<td>1</td>
</tr>
<tr>
<td>OSEventEn</td>
<td>1</td>
</tr>
<tr>
<td>OSEventMax</td>
<td>10</td>
</tr>
<tr>
<td>OSEventNameSize</td>
<td>32</td>
</tr>
<tr>
<td>OSEventSize</td>
<td>52</td>
</tr>
<tr>
<td>OSEventTblSize</td>
<td>520</td>
</tr>
<tr>
<td>OSFlagEn</td>
<td>1</td>
</tr>
<tr>
<td>OSFlagGrpSize</td>
<td>44</td>
</tr>
<tr>
<td>OSFlagMax</td>
<td>5</td>
</tr>
<tr>
<td>OSFlagNameSize</td>
<td>32</td>
</tr>
<tr>
<td>OSFlagNodeSize</td>
<td>24</td>
</tr>
<tr>
<td>OSFlagWidth</td>
<td>4</td>
</tr>
<tr>
<td>OSLowestPrio</td>
<td>63</td>
</tr>
<tr>
<td>OSMsgxEn</td>
<td>1</td>
</tr>
<tr>
<td>OSMemEn</td>
<td>1</td>
</tr>
<tr>
<td>OSMemMax</td>
<td>5</td>
</tr>
<tr>
<td>OSMemNameSize</td>
<td>32</td>
</tr>
<tr>
<td>OSMemSize</td>
<td>52</td>
</tr>
<tr>
<td>OSMemTblSize</td>
<td>260</td>
</tr>
<tr>
<td>OSMutexEn</td>
<td>1</td>
</tr>
<tr>
<td>OSPtrSize</td>
<td>4</td>
</tr>
<tr>
<td>OSQEn</td>
<td>1</td>
</tr>
</tbody>
</table>

Information

A description of the Configuration Constants can be found in OS_DEBUG.C or with their equivalent definitions in OS_CORE.C.

Controls

There are no special controls for this window in addition to the standard list window controls. See List Window Controls on page 8 for details.
**Options**

This window controls the general optional features of the µC/OS-II Kernel Awareness Plug-in. More specific options can also be found in the context menu of List windows.

![µC/OS-II Plugin Options](image)

**Controls**

*Auto Update*

When enabled, the target data is re-read each time a breakpoint is reached, and is used to refresh the contents of the windows. When disabled, the user must manually update the data, using the **Update** command of the context menu in List windows, or the **Update All** button in the *Status* window.

*Stack Stats*

Shows/Hides the Stack-related information fields in the Task List window. These are Max%, Cur%, Max, Cur, Size, Starts@, and Ends@. This feature is also accessible through the context menu of the Task List window.

*Font: Fixed / Proportional*

Selects which type of font to use in List windows. The Plug-in uses either the Fixed or Proportional font of the Embedded Workbench's Common Fonts. The Common Fonts are configured under **Tools .. Options .. Common Fonts**. Font changes only take effect the next time a window is re-opened, or when a new font is applied through **Tools .. Options .. Common Fonts**. A smaller font means you can have smaller windows, which is important on limited screen space. The following fonts are recommended:

- **Fixed**: Courier New 9pt
- **Proportional**: MS Sans Serif 8pt

*Apply*

Click **Apply** to apply the changes.
About

This window shows information about the µC/OS-II Kernel Awareness Plug-in.

Information

Version
Current version of the µC/OS-II Kernel Awareness Plug-in.

Contact
How to contact the Micriµm.
Bibliography

µC/OS-II, The Real-Time Kernel, 2nd Edition
Jean J. Labrosse
R&D Technical Books, 2002
ISBN 1-57820-103-9

Contacts

Micrium, Inc.
949 Crestview Circle
Weston, FL 33327-1848
954-217-2036
954-217-2037 (FAX)
e-mail: Jean.Labrosse@Micrium.com
WEB: www.Micrium.com

IAR Systems, Inc.
Century Plaza
1065 E. Hillsdale Blvd
Foster City, CA 94404
USA
+1 650 287 4250
+1 650 287 4253 (FAX)
WEB: http://www.IAR.com
e-mail: info@IAR.com

CMP Books, Inc.
1601 W. 23rd St., Suite 200
Lawrence, KS 66046-9950
USA
+1 785 841 1631
+1 785 841 2624 (FAX)
WEB: http://www.rdbooks.com
e-mail: rdorders@rdbooks.com