µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio version 1.1.0 Release Notes

Introduction

This document contains the release notes for µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio version 1.1.0. It describes the release in detail and provides latest information that supplements the main documentation.

This release includes support for the processors listed in the next section, below. Users of previous releases should check the "Version Compatibility" section, below, for pertinent instructions on modifying existing applications for this new release.

Support and Assistance

There are several options for contacting support:

- Submit your questions online at http://www.analog.com/support
- E-mail your Processor and DSP software and development tools questions from within CrossCore Embedded Studio.

  Go to "Help->E-mail Support…". This will create a new e-mail addressed to processor.tools.support@analog.com, and will automatically attach your CrossCore Embedded Studio version information (ProductInfo.html).

- E-mail your Processors and DSP applications and processor questions to:
  - processor.support@analog.com OR
  - processor.china@analog.com (Greater China support)
- Post your questions in the Processors and DSP online technical support community in Engineer Zone at http://ez.analog.com/community/dsp

Update Highlights

The focus of µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio version 1.1.0 is to support the release of ADSP-BF70x processor family.

The version of the Micriµm's kernel included with µC/OS-III™ Real-Time Kernel for CCES version 1.1.0 has been upgraded to v 3.04.04.

Micriµm's documentation has migrated to their website at address http://doc.micrium.com instead of being included with µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio. Any documentation specific to Analog Devices processors can still be found within this product.

ADSP-BF70x processor support

The µC/OS-III support has been extended to support the following processors

- ADSP-BF700, ADSP-BF701, ADSP-BF702, ADSP-BF703, ADSP-BF704, ADSP-BF705, ADSP-BF706, ADSP-BF707

To find more information about these processors go to http://www.analog.com/BF70x

New add-ins to support ADSP-BF70x silicon revision 0.0 Utility ROM

ADSP-BF70x contains a Utility ROM designed to reduce the amount of code and data that is required to be linked in L1 and L2 RAM memory and such leaving more space available for application code and data. The functionality included in ADSP-BF70x silicon revision 0.0 in L2 ROM space is:

- Commonly-accessed run-time library support, including DSP library functions and maths emulation.
- FFT twiddle tables
- Device drivers.
- Micrium µC/OS-III Real-Time Kernel version 3.03.01.

Although support for using the L2 ROM is enabled by default when building applications for ADSP-BF70x parts, to use µC/OS-III in ROM the Micrium uC/OS-III product must be installed and the appropriate add-ins must be added to the application.

µC/OS-III is highly configurable at build time so, in order to place it in ROM, some options which are normally available to customers are fixed. To increase the number of applications that are able to use µC/OS-III in ROM, the ROM itself contains two separate configurations which correspond to the new uCOS-III ROM add-ins:

- Configuration1 which performs error checking and contains the statistics task, and is more suitable for debugging applications and application development
- Configuration2 which is more streamlined and might be more suitable for the release of applications.

Dedicated ISR stack support in Blackfin processors

µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio version 1.1.0 includes the check box "Enable dedicated ISR stack" in the µC/OS-III Application General Options which allows you to select whether interrupt service routines (ISR) should use a separate stack instead of the running task stack. Enabling this setting uses a stack of size "ISR stack size". Although this setting is available for all applications, the option is ignored for any non-Blackfin processors since SHARC does not support dedicated ISR stacks at this point.

Setting dedicated ISR stacks might reduce the footprint of an application since task stacks no longer need to be large enough to accommodate the stack usage of all nested ISRs.

Improvements on out-of-the-box performance due to user interface fix

Due to a user interface issue, un-checking the µC/CPU check box Enable ‘interrupts disabled’ time measurement did not result on the appropriate macro being correctly generated. This issue has been fixed in µC/CPU version 1.1.0. This produces a significant performance improvement in many API calls so it should be noticeable in a great number of applications. The performance improvement that can be seen in the figures provided with this product is mainly due to this change.

Improved Mutex priority inheritance

Micriµm changes to the kernel have improved the robustness of the priority inheritance feature of µC/OS-III. The enhancements are:

- Deleting and Pend aborting mutexes are now guaranteed not to cause priority inversion.
- Deleting a task will now automatically release all the mutexes owned by that task.
- Priority inheritance is correctly and deterministically applied in the case of intentional deadlocks. (Two or more tasks pending on mutexes owned by each other.)
- Breaking a deadlock using OSMutexPendAbort() or OSMutexDel() won't create priority inversion even if other mutexes are still actively owned. Please note that while the kernel will correctly prevent priority inversion, breaking a deadlock using OSMutexPendAbort() is NOT a recommended coding practice.

Micriµm software versions

µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio version 1.1.0 is based on Micriµm's µC/OS-III™ Real-Time Kernel version 3.04.04

There are several CrossCore Embedded Studio products based on Micriµm's software which share common add-ins. To ensure that the same version of these add-ins is used by all the products that require them, the common add-ins are installed in a location which is distinct from the µC/OS-III install folder. The common add-ins are

- µC/CPU which is installed in %COMMONPROGRAMFILES%\Analog Devices\uC-CPU v1.1.0. This installation includes µC/CPU version 1.30.01.
- µC/LIB which is installed in %COMMONPROGRAMFILES%\Analog Devices\uC-LIB v1.1.0. This installation includes µC/LIB version 1.38.00.

Version Compatibility

µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio version 1.1.0 requires CrossCore® Embedded Studio version 1.1.0 or later.

µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio version 1.1.0 requires µC/CPU version 1.1.0 or later. This is because the kernel as shipped by Micrium requires at least µC/CPU version 1.30.00.

Software requirements

In order to be able to add µC/OS-III Real-Time Kernel for CrossCore® Embedded Studio version 1.1.0 to a project you need to use CrossCore Embedded Studio version 1.1.0 or later.

µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio Software Anomalies

This section enumerates the most significant anomalies which relate to µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio. For a comprehensive list of all the public software anomalies visit http://www.analog.com/SoftwareAnomalies

Anomalies fixed in version 1.1.0

Anomalies fixed by the Micriµm upgrade to version 3.04.04

- OSMutexDel(): Deleting a mutex not owned by any tasks with the option OS_OPT_DEL_NO_PEND would cause a null pointer access. This issue was introduced in version 3.04.03.
- os.h, os_tmr.c: Fixed a few local include directives which were using brackets <> instead of quotes ""
- Minor corrections to some function comment headers
- os.h: Removed double "??" characters in a comment that could be confounded for a trigraph by some preprocessors.
- OSFlagPendAbort() will now, as documented, abort the highest pending task when called with the OS_OPT_PEND_ABORT_1 option.
- The declaration of OSSemDel() is now correctly preprocessed out when OS_CFG_SEM_DEL_EN is disabled (MISRA compliance)
- Minor performance improvement to the tick task by removing an unneeded check.
- Minor improvements to MISRA compliance
- Various corrections to the comments.
- os_core.c, os_task.c: Deleting a task pending on an object without timeout or deleting the object itself may have caused a null pointer access. This issue only affected version 3.04.00 and later of the kernel.
- os_mutex.c: Aborting a mutex pend in certain conditions left the mutex owner at an elevated priority.
- Various typos and formatting fixes.

Known issues with µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio 1.1.0

These are the currently known problems which affect µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio.

UCOS3-307: Upgrading µC/OS-III add-in to 1.1.0 results in warnings

CCES produces some unexpected warnings when upgrading the µCOS-III component to 1.1.0.
The following warnings can be ignored safely and the resulting project should build without issues.
Property "uccpu-ucos3-compatibility" required by Add-in "..." doesn't exist
Property "uclib-ucos3-compatibility" required by Add-in "..." doesn't exist

The root cause for this issue is a CrossCore Embedded Studio problem which has been recorded as CCES-7883.

UCOS3-336: The user interfaces of µC/LIB and µC/CPU in ROM report an error because they cannot find the help

Due to an error in the user interface that displays the µC-CPU and µC-LIB configurations, selecting the tabs in a project that uses µC/OS-III in ROM results the following errors on the console:

Unable to determine HelpContextId for com.analog.micrium.uclib

Unable to determine HelpContextId for com.analog.micrium.uccpu

Further to this, clicking on the question mark in the configuration window does not bring up any help.

These errors do not affect applications or any code generation and can be safely ignored. The help can still be found in the CrossCore Embedded Studio Help menu.

µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio version 1.0.2 Release Notes

Introduction

This document contains the release notes for µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio version 1.0.2. It describes the release in detail and provides latest information that supplements the main documentation.

This release includes support for the processors listed in the next section, below. Users of previous releases should check the "Version Compatibility" section, below, for pertinent instructions on modifying existing applications for this new release.

For product support assistance, please contact our Processor Tools Support Team at <processor.tools.support@analog.com>.

Update Highlights

The focus of µC/OS-III™ Real-Time Kernel for CCES version 1.0.2 is better integration with CrossCore Embedded Studio rather than new features. These are the most notable changes:

- RTOS Status window better integrated with CCES 1.0.2
  CrossCore Embedded Studio provides a framework for RTOS developers to create a Status window. µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio version 1.0.2 uses this framework to ensure that you do not get multiple versions of similar windows when you have other RTOS installed. Instead, CCES detects which supported RTOS your application is running and it displays the appropriate information.

- Support for ADSP-BF60x revision 0.1
  Prior revisions of µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio used libraries which applied all known workarounds for the anomalies that applied all silicon revisions of ADSP-BF60x. Version 1.0.2 uses a new set of libraries which only contains the anomalies which affect silicon revision 0.1

Supported Processors

µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio 1.0.2 supports the same processors as version 1.0.1. These supported processors are:

Blackfin

- ADSP-BF504, ADSP-BF504F, ADSP-BF506F
- ADSP-BF512, ADSP-BF514, ADSP-BF516, ADSP-BF518
- ADSP-BF522, ADSP-BF524, ADSP-BF526, ADSP-BF523, ADSP-BF525, ADSP-BF527
- ADSP-BF531, ADSP-BF532, ADSP-BF533, ADSP-BF534, ADSP-BF536, ADSP-BF537, ADSP-BF538, ADSP-BF539
- ADSP-BF542, ADSP-BF542M, ADSPBF547, ADSP-BF547M, ADSP-BF548, ADSP-BF548M, ADSP-BF549, ADSP-BF549M
- ADSP-BF544, ADSP-BF544M
- ADSP-BF561
- ADSP-BF592-A
- ADSP-BF606, ADSP-BF607, ADSP-BF608, ADSP-BF609

SHARC

- ADSP-21160, ADSP-21161
- ADSP-21261, ADSP-21262, ADSP-21266
- ADSP-21362, ADSP-21363, ADSP-21364, ADSP-21365, ADSP-21366, ADSP-21367, ADSP-21368, ADSP-21369
- ADSP-21371, ADSP-21375
- ADSP-21467, ADSP-21469
- ADSP-21477, ADSP-21478, ADSP-21479
- ADSP-21483, ADSP-21486, ADSP-21487, ADSP-21488, ADSP-21489

Micriμm software versions

µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio version 1.0.2 is based on Micriμm's µC/OS-III™ Real-Time Kernel version 3.03.01

There are several CrossCore Embedded Studio add-ins based on Micriμm's products which share common add-ins. To ensure that the same version of these add-ins is used by all the add-ins that require them, these add-ins are installed in a common location which is distinct from the µC/OS-III install folder. These common add-ins are

- µC/CPU which is installed in %COMMONPROGRAMFILES%\Analog Devices\uC-CPU v1.0.2. This installation includes µC/CPU version 1.29.01.
- µC/LIB which is installed in %COMMONPROGRAMFILES%\Analog Devices\uC-LIB v1.0.2. This installation includes µC/LIB version 1.37.00.

The documentation for all of these add-ins can be found in CrossCore® Embedded Studio Help under µC/OS-III™ 1.0.2

Version Compatibility

µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio version 1.0.2 requires CrossCore® Embedded Studio version 1.0.2 or later. This is due to the RTOS Status window changes which use a CCES provided framework for a status window for any RTOS.

Using µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio version 1.0.1 or earlier with CCES 1.0.2 is still possible but, if µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio version 1.0.2 is not installed the RTOS Status window provided with those products is not displayed.

Software requirements

In order to be able to add µC/OS-III Real-Time Kernel for CrossCore® Embedded Studio version 1.0.2 to a project you need to use CrossCore Embedded Studio version 1.0.2 or later. If you use an earlier version, none of the components related to µC/OS-III Real-Time Kernel for CrossCore® Embedded Studio 1.0.2 are available when you create a project.

µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio Software Anomalies

This section enumerates the most significant anomalies which relate to µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio. For a comprehensive list of all the public software anomalies visit http://www.analog.com/SoftwareAnomalies

Anomalies fixed in version 1.0.2

TAR-50123 Having both uCOS-III version 1.0.0 and a newer version installed always adds the newer uCOS-III sources

Due to an issue with µC/OS-III for CCES 1.0.0, if a machine has both µC/OS-III 1.0.0 and a more recent version of µC/OS-III installed, creating a new project and trying to add the µC/OS-III 1.0.0 add-in always results in the most recent µC/OS-III non platform-specific sources being added to the project. The platform-specific sources that are added are the ones in the 1.0.0 version.

The System Configuration Overview page will still display that µC/OS-III 1.0.0 is in the project even if most sources do not correspond to version 1.0.0.

TAR-50130 Deadlock waiting for user input

If you build an application for one silicon revision and you attempt to load it on a board that is a different silicon revision, then you get a prompt to confirm that you want to continue and load.

In some cases this can cause a deadlock which can only be solved by closing the IDDE.

µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio version 1.0.1 Release Notes

Introduction

This document contains the release notes for µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio version 1.0.1. It describes the release in detail and provides latest information that supplements the main documentation.

This release includes support for the processors listed in the next section, below. Users of previous releases should check the "Version Compatibility" section, below, for pertinent instructions on modifying existing applications for this new release.

For product support assistance, please contact our Processor Tools Support Team at <processor.tools.support@analog.com>.

Update Highlights

The focus of µC/OS-III™ Real-Time Kernel for CCES version 1.0.1 is improved reliability and usability rather than new features. These are the most notable changes:

- New static tasks display
  Developers of µC/OS-III applications set the priority and stack size of certain static tasks like the idle task in configuration pages. Tracking the priorities and stack sizes of all the static tasks can be difficult because the configuration happens across different windows. This gets more complicated when other products like µC/FS File System or µC/USB also create static tasks. For this reason, µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio version 1.0.1 now contains the "Application Tasks" window located in the µC/APP Tab which displays the information of all of the statically configured tasks in the same page. This page also displays in its tooltips the location where each task settings can be modified.

- More robust upgrade mechanism
  µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio 1.0.1 contains improvements to its add-in upgrade mechanism which help with the upgradeability of projects to future versions. This ensures that application-

specific files are not overwritten in the case of an add-in upgrade (or a remove followed by an add) and that the appropriate sources are always used.

- Full SHARC support
  Although µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio 1.0.0 supported SHARC processors, at the time of its release CrossCore® Embedded Studio did not. This version of the kernel provides the same level of functionality to SHARC processors as to all the Blackfin processors.

- Extra documentation
  Previous versions of the documentation included the Micriμm manual without any modifications. Due to the differences between µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio and other µC/OS-III offerings, the documentation now contains a section which specifies where the CCES add-in changes. Improvements to the documentation also include the tooltips in configuration windows which now display the kernel macros that are defined for each of the settings.
- Kernel metrics included
  Many customers are interested in kernel metrics. µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio now includes footprint and performance metrics for some of the processors supported. These metrics can be found in the kernel documentation within CCES.

Supported Processors

µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio 1.0.1 supports the same processors as version 1.0.0. These supported processors are:

Blackfin

- ADSP-BF504, ADSP-BF504F, ADSP-BF506F
- ADSP-BF512, ADSP-BF514, ADSP-BF516, ADSP-BF518
- ADSP-BF522, ADSP-BF524, ADSP-BF526, ADSP-BF523, ADSP-BF525, ADSP-BF527
- ADSP-BF531, ADSP-BF532, ADSP-BF533, ADSP-BF534, ADSP-BF536, ADSP-BF537, ADSP-BF538, ADSP-BF539
- ADSP-BF542, ADSP-BF542M, ADSPBF547, ADSP-BF547M, ADSP-BF548, ADSP-BF548M, ADSP-BF549, ADSP-BF549M
- ADSP-BF544, ADSP-BF544M
- ADSP-BF561
- ADSP-BF592-A
- ADSP-BF606, ADSP-BF607, ADSP-BF608, ADSP-BF609

SHARC

- ADSP-21160, ADSP-21161
- ADSP-21261, ADSP-21262, ADSP-21266
- ADSP-21362, ADSP-21363, ADSP-21364, ADSP-21365, ADSP-21366, ADSP-21367, ADSP-21368, ADSP-21369
- ADSP-21371, ADSP-21375
- ADSP-21467, ADSP-21469
- ADSP-21477, ADSP-21478, ADSP-21479
- ADSP-21483, ADSP-21486, ADSP-21487, ADSP-21488, ADSP-21489

Micriμm software versions

μC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio version 1.0.1 is based on Micriμm's μC/OS-III™ Real-Time Kernel version 3.03.01

There are several CrossCore Embedded Studio add-ins based on Micriμm's products which share common add-ins. To ensure that the same version of these add-ins is used by all the add-ins that require them, these add-ins are installed in a common location which is distinct from the μC/OS-III install folder. These common add-ins are

- μC/CPU which is installed in %COMMONPROGRAMFILES%\Analog Devices\uC-CPU v1.0.1. This installation includes μC/CPU version 1.29.01.
- μC/LIB which is installed in %COMMONPROGRAMFILES%\Analog Devices\uC-LIB v1.0.1. This installation includes μC/LIB version 1.37.00.
- μC/CLK which is installed in %COMMONPROGRAMFILES%\Analog Devices\uC-CLK v1.0.1. This installation includes μC/CLK version 3.09.03

The documentation for all of these add-ins can be found in CrossCore® Embedded Studio Help under μC/OS-III™ 1.0.1

Version Compatibility

Both the add-In framework provided with CCES 1.0.0 and μC/OS-III 1.0.0 have several undocumented issues related to add-In upgrades. Because of this it is recommended that you do not use the CCES add-in framework to upgrade existing projects that use μC/OS-III 1.0.0 add-ins to use μC/OS-III 1.0.1. Instead, it is recommended that entirely new projects are created

Software requirements

μC/OS-III Real-Time Kernel for CrossCore Embedded Studio requires CrossCore Embedded Studio version 1.0.1 or later to be installed to build projects.

Adding µC/OS-III to a project

When a µC/OS-III add-inis added to a CrossCore Embedded Studio project, the CCES add-in framework executes the following actions

- Adds links to the kernel files in the installation folder
- Adds generated files based on the configuration window settings
- Sets up the required toolchain options.
- Copies os_app_hooks.c to your project. This file is a copy of a micrium template which contains hook functions which are used depending on configuration options. This file can be edited by the application developer and will not be removed from the project when the file system is removed or overwritten if it is already present in the destination folder.
- Copies app_ucos3_cfg.h to your project. This file is empty by default but can contain application-specfic macros and definitions. This file can be edited by the application developer and will not be removed from the project when the file system is removed or overwritten if it is already present in the destination folder.

MISRA-C Compliance Conformance Checking

The Motor Industry Software Reliability Association (MISRA) published a set of guidelines for the C programming language to promote best practice in developing safety related electronic systems in road vehicles and other embedded systems. The CrossCore® Embedded Studio compiler fully supports the MISRA-C 2004 Guidelines, and can detect violations of the rules at compile-time, link-time, and run-time.

The µC/OS-III Real-Time Kernel for CrossCore Embedded Studio complies with MISRA by documenting known violations of the MISRA Compliance standard. The violations mean that µC/OS-III sources will not compile with MISRA conformance checking enabled, and any file that includes µC/OS-III headers will also not compile.

µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio Software Anomalies

This section enumerates the most significant anomalies which relate to µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio. For a comprehensive list of all the public software anomalies visit http://www.analog.com/software-anomalies.html

Anomalies fixed in version 1.0.1

TAR-48258 Application-specific files are unexpectedly overwritten after re-adding the RTOS

Due to a mistake in uC/OS-III for CrossCore Embedded Studio, removing and re-adding the RTOS to a project can result on application-specific files being unexpectedly overwritten. These files can be modified by applications to suit their needs and should not have been overwritten. It is recommended that if an application requires changes to these files, a safe copy of them is made to a separate location before removing and re-adding uC/OS-III.

The files are located in the application folder and are called
system\uCOS-III\AppSources\os_app_hooks.c
system\uCOS-III\Cfg\os_app_hooks.h
system\uCOS-III\Cfg\app_ucos3_cfg.h

TAR-48440 The file app_ucos3_cfg.h is read-only when it shouldn't

The file system/uCOS-III/AppSources/app_ucos3_cfg.h is designed to be edited by uC/OS-III applications instead of app_cfg.h when they need to add any application-specific details. uC/OS-III Real-Time Kernel for CrossCore Embedded Studio made the file read-only so when a user tries to overwrite it, the IDE reports that it cannot be overwritten. At this point the IDE offers an option to remove the read-only attribute. Selecting that option will allow users to write to the file.

TAR-49137 The SHARC RTOS examples contain app_heaptab.s instead of app_heaptab.c

The SHARC uCOS-III examples in the 1.0.0 release contain a file called app_heaptab.s to define the heaptable, This is incorrect and the heap table definition should be done in a C file. To fix the problem follow the following steps:

- Double-click on the system.svc file, and remove and re-add the Startup/LDF component.This should add app_heaptab.c to the project.
- Remove app_heaptab.s from the folder system/startup_ldf

TAR-49323 Context switch hits anomaly 09000022/15000004 and can cause an incorrect context restore

A section of the SHARC context switch for uC/OS-III uses a global variable as part of its internals to restore some registers. If the variable is placed in external memory then the restore sequence would be vulnerable to anomaly 09000022/15000004.

The impact of this anomaly on the context switch is that ASTATy and USTAT2 may not be restored correctly for a task that is being switched-in, or a task that is being restored after an interrupt has occurred.

TAR-50050 Statistics task fails without debug enabled

In uCOS-III for CCES selecting to have the statistics task without debug variables results in a compilation error of the form
C:\Analog Devices\uCOS-III-Rel1.0.0\uCOS-III\common\uCOS-III\Source\os_stat.c", line 79: cc0020: error: identifier "cpu_sr" is undefined
    CPU_CRITICAL_ENTER();
The problem happens with the following combination in the uCOS-III tab > uC/OS Debug/Statistic page

- "Enable statistics task" ticked
- "Include debug variables" unticked

Changing either of those options avoids the compilation errors.

TAR-50227 Nested interrupts can cause task corruption with uCOS-III projects

The sequence used to save a task context on SHARC processors has a bug where if a nested interrupt occurs, the pointer to the context record for the task is changed to point to the context record for the interrupted ISR. If there are no task switches initiated from either ISR, then the bug does not show and the system continues without problem. If there is a task switch, then the outgoing task is essentially corrupted and so will not be restored correctly if scheduled to run.

TAR-50260 SHARC interrupt vector table is not interrupt-safe and causes uCOS-III to fail

The first instruction for non-IRQ/non-timer interrupts in the SHARC interrupt vector table are not interrupt-safe so a nested interrupt can occur. If a nested interrupt occurs at that instruction then the task-level context is not saved and restored correctly, and execution will not return correctly from interrupt level.

TAR-50264 uCOS3 context switch does not disable LIRPTL interrupts

uCOS-III requires that all interrupts are disabled during context saves and restores. In one point in the context switch, the SHARC port needs to clear the mask registers. To do this, uCOS-III for CCES only clears the bits in IMASK leaving the interrupt mask bits in LIRPTL untouched. This means that certain interrupts could still occur corrupting the save/restore sequence potentially corrupting the context save/restore sequence.

Outstanding anomalies

These are the currently known anomalies which affect µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio.

TAR-50123 Having both uCOS-III version 1.0.0 and a newer version installed always adds the newer uCOS-III sources

Due to an issue with µC/OS-III for CCES 1.0.0, if a machine has both µC/OS-III 1.0.0 and a more recent version of µC/OS-III installed, creating a new project and trying to add the µC/OS-III 1.0.0 add-in always results on the most recent µC/OS-III non platform-specific sources being added to the project. The platform-specific sources added are the ones in the 1.0.0 version.

The System Configuration Overview page will still display that µC/OS-III 1.0.0 is in the project even if most sources are not.

µCOS/III™ Real-Time Kernel for CrossCore® Embedded Studio version 1.0.0 Release Notes

What is µC/OS-III Real-Time Kernel for CrossCore Embedded Studio

µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio is the result of a partnership between Analog Devices and Micriµm to provide user-friendly programming environment for µC/OS-III applications running on Analog Devices' Blackfin and SHARC processors. µC/OS-III Real-Time Kernel for CrossCore Embedded Studio provides an integrated environment with CrossCore Embedded Studio which offers the advantage of an industry-standard IDE combined with Analog Devices' advanced optimizing compiler technology.

The µC/OS-III is Micriµm's newest RTOS, designed to save time on embedded system projects. In addition to the features inherent in µC/OS-II, µC/OS-III also manages a number of application tasks only constrained by the processor's access to memory. µC/OS-III supports a large amount of priority levels which is restricted in Blackfin and SHARC processors to 65535. Typically, configuring µC/OS-III for between 32 and 256 different priority levels is adequate for most embedded applications.

µC/OS-III does not allocate memory for the kernel objects. Instead, applications pass the memory for tasks, semaphores, mutexes, event flags, message queues, timers and memory partitions which means that the RTOS never fails allocating memory at runtime.

µC/OS-III allows multiple tasks to run at the same priority level. When equal priority tasks are ready to run, µC/OS-III runs each for a user-specified time. Each task can define its own time quanta and give up its time slice if it does not require the full time quanta.

µC/OS-III has a number of internal data structures and variables that it needs to access atomically. It protects these critical regions by disabling interrupts for very small numbers of cycles. Interrupt response with µC/OS-III is deterministic.

µC/OS-III contains a comprehensive set of error-checking functionality being able to detect a varied set of user errors. Each API function provides an error code regarding the outcome of the function call.

µC/OS-III's footprint can be scaled to only contain the features required for a specific application. The execution time for most services provided by µC/OS-III is both constant and deterministic; execution times do not depend on the number of tasks running in the application.

µC/OS-III Real-Time Kernel for CrossCore Embedded Studio version 1.0.0 is released with Micriμm's µC/OS-III version 3.03.00.

Getting Started with µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio

Installation

CrossCore® Embedded Studio v.1.0.0 or newer must be installed prior to installing µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio.

Please make sure to close CrossCore Embedded Studio before proceeding with the installation. If CrossCore Embedded Studio is left open during the installation, it will have to be restarted after installing µC/OS-III Real-Time Kernel.

µC/OS-III Real-Time Kernel for CrossCore Embedded Studio installs the following:

- µC/OS-III Real-Time Kernel. Its default installation directory is C:\Analog Devices\uCOS-III-Rel1.0.0.
- µC/LIB . This software is always installed into Common Program Files directory. This location is determined by the %CommonProgramFiles(x86)% environment variable in 64-bit operating systems or by %CommonProgramFiles% in 32-bit operating systems.
- µC/CPU. This software is always installed into Common Program Files directory. This location is determined by the %CommonProgramFiles(x86)% environment variable in 64-bit operating systems or by %CommonProgramFiles% in 32-bit operating systems.

Analog Devices strongly recommends installing µC/OS-III Real-Time Kernel outside of the Program Files directory to prevent possible permission issues related to UAC (User Access Control). If you have already installed the product under Program Files then we recommend that you uninstall it and re-install it in a different location.

Note: Multiple versions of the µC/OS-III Real-Time Kernel can be installed on the same system. Only a single instance of a specific version of the product can be installed on a system.

License Checking

The installation process checks for a valid license for µC/OS-III™ Real-Time Kernel. If a valid license is not detected, the installer will start the Manage Licenses utility for entering and activating a license. The installer will fail in a non-interactive mode when valid license is not present.

Installation Logging

The installer does not create a log file by default. If you encounter installation issues, you can generate an installation log file by running the installer from the command prompt.

Change to the directory containing downloaded installer executable and run the following from the command prompt:

```
ADI_uCOS-III-Rel1.0.0.exe /v"/l*v c:\temp\installer.log"
```

License

The installation process checks for a valid license for µC/OS-III™ Real-Time Kernel. Refer to the Licensing Guide in your CCES installation which can also be found in http://www.analog.com/CrossCoreLicensingGuide.

Support and Assistance

There are several options for contacting support:

- Submit your questions online at:

  http://www.analog.com/support

- E-mail your Processor and DSP software and development tools questions from within CrossCore Embedded Studio .

  Go to “Help->E-mail Support…”. This will create a new e-mail addressed to processor.tools.support@analog.com, and will automatically attach your CrossCore Embedded Studio version information (ProductInfo.html).

- E-mail your Processors and DSP applications and processor questions to:
  - processor.support@analog.com OR
  - processor.china@analog.com (Greater China support)

- Post your questions in the Processors and DSP online technical support community in Engineer Zone at

http://ez.analog.com/community/dsp

Supported processors

µC/OS-III™ Real-Time Kernel for CrossCore®Embedded Studio includes support for existing Blackfin and SHARC processors, The support will only be visible, however**,** if the CrossCore Embedded Studio toolchain supports those processors. For example, if you use CrossCore Embedded Studio version 1.0.0 then you will only be able to use µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio for ADSP-BF60x processors. If you use CCES 1.0.1, then you can use µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio for all the Blackfin and SHARC processors that CCES supports.

Software requirements

µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio requires CrossCore Embedded Studio version 1.0.0 or later to be installed to build projects.

Getting started with a project that uses the RTOS

Adding the RTOS to a project

Creating a project which includes the RTOS

In order to create a project you should follow the instructions provided in the CrossCore Embedded Studio help. As part of the project creation, the page Add-in selection contains a list of all the available add-ins for the project that you are creating based on the installed products and the project's chosen processor and type. You can see the µC/OS-III add-in in the Middleware section under the RTOS category.

The µC/OS-III add-in generates the `OSInit()` call for initializing the RTOS, and this code should be called prior to initializing other services/devices. To ensure timely initialization, the IDE adds any required code to a global C function named adi_initComponents(). If you choose to add a default C source to your project then the resulting project contains a call to adi_initComponents(). If the template C code is not selected at project creation then it is the application developer's responsibility to add initialization code for the components required including the RTOS.

There are no calls added to create start-up tasks or to `OSStart()` to start the OS. These API calls remain the application's responsibility.

Adding the RTOS to an existing project

Every CrossCore Embedded Studio project contains a System Configuration file called `system.svc` which is located in the root of the project. The file is the IDE's interface for

managing the various pre-written software components used in the "system" implemented by a project. Double-clicking any `system.svc` file in a navigation view opens that file in the System Configuration Utility which allows you to see the add-ins that you currently have in your project. Clicking on Add and selecting µC/OS-III add-in from the Middleware section under the RTOS category adds µCOS-III to your project.

The µC/OS-III add-in generates code for initializing the RTOS, and this code should be called prior to initializing other services/devices. To ensure timely initialization, when system components are configured the IDE adds any required code to a global C function named `adi_initComponents()`. You should call this function early in the `main()` routine.

Notes:

- If the IDE detects that `adi_initComponents()` is not yet present in `main()`, it prompts you to add it and offers to insert it for you.
- There are no calls added to create start-up tasks or to `OSStart()` to start the OS. These API calls remain the application responsibility.
- Adding the RTOS to a project means that thread-safe versions of libraries are used, including the Runtime library and the System Services/Device Drivers library. It is the application's responsibility to ensure that any functions that should only be called from task level are not called from main but from a task function.

Configuration

µC/OS-III application developers traditionally configure applications by creating header files which contain a long list of macro definitions. In order to have a more intuitive configuration mechanism, µC/OS-III Real-Time Kernel for CrossCore Embedded Studio contains a configuration tab which can be accessed by double-clicking the `system.svc` file and selecting the µC/OS-III tab. Selecting all the desired fields in the configuration tab generates the appropriate files which are part of the project.

Note that the defaults may not necessarily be appropriate for your application and you should set them to suit your needs. For example, the default Task Rate of the Statistics Task (if you choose to add it) is 0 which means that this task would run continuously without any other tasks being able to run. The configuration window tool tips should give you an indication of how you should set your values or look in the documentation for more details.

µC/OS-III project structure

When adding µC/OS-III to a CrossCore Embedded Studio project all the µC/OS-III specific files get placed in the `system` folder. Please do not change this organization. In the system folder the following structure gets created

- A `uCOS-III` folder. This folder contains any sources and header files which are part of the µC/OS-III RTOS. It contains the folders:
    - Source, TLS and Ports. These folders contain the actual RTOS code which should not be modified.
    - AppSources and Cfg. This folder contains a copy of the templates of headers and sources that are provided by Micriµm and that applications can customize to suit their needs.
    - GeneratedSources. This folder contains headers and sources generated by the product based on the GUI configurations. These files should not be modified.
- A `uC-CPU` folder. This folder contains any sources and header files which are required by Micriµm µC/CPU software. µC/CPU provides a processor-independent interface to the supported processors and toolchains that is used in all Micriµm products.
- A `uC-LIB` folder. This folder contains any sources and header files which are required by Micriµm µC/LIB software. µC/LIB provides a clean and organized implementation of some of the most common standard library functions, macros and constants. µC/LIB is required by many Micriµm products including µC/OS-III.
- A `uC-Common` folder. This folder contains sources and headers which are common to several Micriµm products but that are not part of any Micriµm product themselves. These include `app_cfg.h` which is needed by all Micriµm applications.

Interrupts

CrossCore Embedded Studio for Analog Devices processors provides a coherent interrupt management mechanism which allows for the same interface to be used in RTOS and non-RTOS applications. This means that interrupt service routines in µC/OS-III applications must be written in C and use the adi_int interface, regardless of whether they use µC/OS-III APIs. Any thread-safety requirements or interactions with tasks are handled by the adi_int interface. For more information on the adi_int API, in CrossCore Embedded Studio go to Help > Search and enter adi_int.

Examples of the usage of this interrupt management mechanism are the System Services and Device Drivers provided with Crosscore Embedded Studio. By using the adi_int interface, the same services and drivers can be used in all applications regardless of whether an operating system is used.

BSP module

In some of the examples described in the µC/OS-III manual provided by Micriµm for all platforms, there is a mention to some BSP-specific functions. µC/OS-III Real-Time Kernel for CrossCore Embedded Studio does not provide any BSP-specific functionality and the RTOS is ready to be used from main without any BSP calls. In order to access hardware-specific functionality the System Services and Device drivers functionality provided with CrossCore Embedded Studio

Examples and sketches

Sketches

CrossCore Embedded Studio provides a mechanism by which small code fragments, called sketches, can be generated with parameterized input provided by the user. The resulting code can then be copied and pasted to a project. µC/OS-III Real-Time Kernel for CrossCore Embedded Studio provides a few sketches which demonstrate the API calls that are required for

- Creating a task
- Pending on signals (semaphores, mutexes, etc)
- Posting signals (semaphores, mutexes, etc)

The CCES sketch mechanism is designed for small code fragments. For this reason some of the µC/OS-III sketches hard-code certain elements like task name or signals to pend in OSPendMulti. The intention of these sketches is to quickly show the intended code sequence and they should be modified to suit the application's needs.

Examples

In addition to the code sketches, µC/OS-III Real-Time Kernel for CrossCore Embedded Studio provides a few simple examples which show how to use the RTOS. Each example is shipped for three platforms: ADSP-BF609, ADSP-BF533 and ADSP-21469 and can be used in both Release and Debug configurations. The examples can be run on an EZ-KIT for all platforms or on a simulator for the platforms that have both (ADSP-BF533 and ADSP-21469).

Notes

- The examples use features that can be disabled in the configuration windows. Disabling any of the used features results in expected link errors.
- Double-clicking on an example from the example browser or the system overview page opens the project in the installation folder without copying it to your workspace. If you want to modify the example in any way, it is recommended that it gets copied to your workspace. If you would like to copy the project to your workspace note that you may have to copy the sources separately. See the Known Issues section for more details.

Location

In order to locate µC/OS-III examples and sketches, you can use the following:

- Open CrossCore Embedded Studio's Example Browser which can be found in CrossCore Embedded Studio under Help. Select in the Product section "Micrium uCOS-III v 3.03.00 [1.0.0]" for a full list of examples and sketches.
- Open CrossCore Embedded Studio's System Overview page which can be found by double-clicking on the system.svc file. Select the µC/OS-III add-in for a list of the available examples and sketches.
- Import projects located in your µC/OS-III installation folder under `uCOS-III/Examples`.

Documentation

µC/OS-III Real-Time Kernel for CrossCore Embedded Studio contains the processor-agnostic section of the µC/OS-III Real-Time Kernel manual as provided by Micriμm as well as the manuals for Micriμm's µC/CPU and µC/LIB. To find these manuals go to CrossCore Embedded Studio Help > Help Contents where you can see a whole section on the Micriμm products released with this add-in.

Integration with CrossCore® Embedded Studio

System view

CrossCore Embedded Studio provides the System view which is used by µC/OS-III Real-Time Kernel for CrossCore Embedded Studio. Use the System Configuration Overview tab to add the µC/OS-III system add-in to a CrossCore Embedded Studio project.

To access the System Configuration Overview tab, do one of the following:

- In a navigation view, double-click the `system.svc` file of a project. The System Configuration utility appears with the overview tab selected.
- If the utility is already open, select the Overview tab.

As well as being able to add, remove and upgrade add-ins from this window, it also provides a list of examples and sketches associated with the selected add-in.

For more information about the System Configuration utility, see the CrossCore Embedded Studio help.

Configuration tabs

When the µC/OS-III Real-Time Kernel for CrossCore Embedded Studio gets added to a project, several configuration tabs get added to the System view. These include tabs for

Micriμm components which are required by several products like μC-LIB and a configuration tab for μC/OS-III. These configuration tabs provide an easy mechanism to generate any macro definitions required by the Micriμm products.

If the μC/OS-III add-in is used by a CrossCore Embedded Studio project, its configuration tab includes the following pages:

- General Options
- Debug and Statistics Options
- Signals Options
- Task Options
- Application - General Options
- Application - Timer and Tick Options

For more information about each of the configuration options see the section μC/OS-III tab in the μC/OS-III node in CrossCore Embedded Studio's help.

RTOS Status window

To help with the debugging of μC/OS-III applications, μC/OS-III Real-Time Kernel for CrossCore Embedded Studio integrates a window in CrossCore Embedded Studio's debug perspective which displays the current state of the application. When you halt execution of a μC/OS-III program, CrossCore Embedded Studio reads data for tasks, semaphores, mutexes, message queues, event flags groups and memory partitions, and displays the state and status data in this window.

When one of the above μC/OS-III objects is created, it is added to the display. An entity is removed from the display when it is deleted.

The following requirements must be met in order to display the contents of the RTOS Status window:

- μC/OS-III add-in must have added to the executable
- The "Include debug variables" tickbox must be selected in the μC/OS-III configuration under μC/OS-III Debug/Statistics (enabled by default)
- The application must have been built with debug information (enabled by default in Debug configuration)

Initially, information displays in a collapsed state, showing only the name of the item. Clicking the triangle next to the item expands the view to display the existing instances and their properties including for example for tasks their state, priority, stack range etc.

You can find this window in the Debug perspective if you select Window > Show View > RTOS Status

Options not supported

µC/OS-III Real-Time Kernel for CrossCore Embedded Studio version 1.0.0 does not allow the setting of some of the CrossCore Embedded Studio options and will raise an error if these options are set. The options supported may change in future releases. This version does not support:

- Stack overflow detection (-rtcheck-stack). CrossCore Embedded Studio can track the usage of the system stack and will report errors if the stack has overflowed. In order to provide a similar support in the RTOS case, the µC/OS-III Real-Time Kernel needs to provide information about each task stack to CrossCore Embedded Studio so it could monitor the use of each of the tasks stacks. This information is not provided in this version of µC/OS-III Real-Time Kernel.
- Instrumented profiling (-p). CrossCore Embedded Studio can provide a summary of cycle counts for functions within an application. In order to provide the correct information in the RTOS case, the µC/OS-III Real-Time Kernel needs to provide information about each task. This information is not provided in this version of µC/OS-III Real-Time Kernel so instrumented profiling cannot be set.

MISRA-C Support

MISRA C is a software development standard for the C programming language developed by the Motor Industry Software Reliability Association (MISRA). Its aims are to facilitate code safety, portability, and reliability in the context of embedded systems, specifically those systems programmed in ANSI C. The compiler detects violations of the MISRA rules at compile-time, link-time, and run-time.

Although the µCOS-III source is MISRA compliant because Micriμm provide a list of the rules that it breaks, in order to be able to build applications with CrossCore Embedded Studio MISRA options, the compiler checks have been suppressed from all the sources that are part of the kernel itself, including all files in the `system/uCOS-III/Source`, `system/uCOS-III/Ports` and `system/uCOS-III/TLS` folders.

µC/OS-III follows most of the MISRA-C:2004. The list of rules that the platform-agnostic part of µC/OS-III does not adhere to can be found in the CrossCore Embedded Studio Help under Micriμm µC/OS-III™ 1.0.0 > Micriμm µC/OS III Manual > MISRA-C:2004 and µC/OS-III.

Although there is no public list for any other pieces of software that come with µC/OS-III Real-Time Kernel for CrossCore Embedded Studio, there are further rules that we have disabled in order to be able to build all our sotware with CrossCore Embedded Studio -misra-strict option.

Rule 2.1, Rule 2.4, Rule 5.1, Rule 5.2, Rule 5.7, Rule 6.3, Rule 8.12, Rule 8.5, Rule 9.2, Rule 10.1.a , Rule 10.5, Rule 11.3 , Rule 11.4 , Rule 11.5, Rule 12.13, Rule 12.1, Rule

12.6, Rule 12.8, Rule 13.7 , Rule 14.7, Rule 15.2, Rule 16.2 , Rule 16.3, Rule 16.7 , Rule 17.1 , Rule 17.4, Rule 18.1, Rule 19.10, Rule 19.13, Rule 19.4, Rule 19.6, Rule 19.7, Rule 20.4, Rule 20.5, Rule 20.6, Rule 20.8.

For description of each of these rules search in the CrossCore Embedded Studio Help for MISRA Rule <number>.When enabling strictest MISRA option in a CrossCore Embedded Studio project that contains the µC/OS-III add-in, you will get some linkage-related errors from the code located in `system/uCOS-III`. The errors relate to the file `os_app_hooks.c` which is application-specific and can be modified by users to remove those errors by either changing the code to suit their needs or by disabling MISRA-C checking accross source files in the Project Options.

Interaction with other CrossCore® Embedded Studio software

Other add-ins for CrossCore Embedded Studio may make use of features of µC/OS-III Real-Time Kernel for CrossCore Embedded Studio which means that disabling those features in the µC/OS-III configuration window might stop projects which include those add-ins from linking.

Thread-safe runtime libraries

Adding µC/OS-III Real-Time Kernel for CrossCore Embedded Studio to a project switches automatically to use the thread-safe runtime libraries. These libraries use mutexes to protect certain I/O structures and a TLS slot for storage of information like errno or variables related to `rand()`, `strtok()` etc.

Removing µC/OS-III mutex or TLS support from an application could cause link errors if these are used by the thread-safe runtime library.

System Services and Device Driver libraries

Although the System Services and Device Drivers functionality does not require an RTOS, if µC/OS-III is in the application then it uses some objects to ensure its thread-safety. Certain services or device drivers use semaphores or mutexes.

Removing µC/OS-III mutex or semaphore support from an application could cause link errors if these are used by the System Services/Device Drivers library.

MCAPI

Although the Multi-core API (MCAPI) functionality does not require an RTOS, if µC/OS-III is in the application then it uses some objects to co-ordinate between tasks. Analog Devices' implementation of MCAPI uses Event flags.

Removing µC/OS-III flags support from an application could cause link errors if these are used by MCAPI.

µC/FS™ File System for CrossCore® Embedded Studio

Although µC/FS File System for CrossCore Embedded Studio does not require an RTOS, if µC/OS-III is in the application then it uses some objects to ensure its behavior in a multi-threaded environment. In its first release µC/FS requires semaphores, and 2 task-specific registers if working-directory functionality is enabled. Under certain conditions the API `OSTimeDlyHMSM()` can also be required.

Removing µC/OS-III semaphores, task-specific registers or OSTimeDlyHMSM support from an application could cause link errors if these are used by µC/FS.

µC/USB Device™ Stack for CrossCore® Embedded Studio

µC/USB Device Stack for CrossCore Embedded Studio requires the presence of an RTOS, although not necessarily µC/OS-III Real-Time Kernel for CrossCore Embedded Studio. When running in a µC/OS-III application, µC/USB Device Stack requires multiple µC/OS-III objects like semaphores and Task-specific registers slots.

Removing any of the µC/OS-III functionality that is required by a µC/USB Device application could cause link errors.

Note that adding µC/USB Device to a project which already has µC/OS-III may change some RTOS settings. The full list of specific changes is displayed by the µC/USB Device addition process.

lwIP Lightweight TCP/IP Stack for CrossCore® Embedded Studio

lwIP Lightweight TCP/IP Stack for CrossCore Embedded Studio requires the presence of an RTOS, although not necessarily µC/OS-III Real-Time Kernel for CrossCore Embedded Studio. When running in a µC/OS-III application, lwip requires semaphores and TLS slots.

Removing µC/OS-III mutex or TLS support from an application could cause link errors if these are used by lwIP.

Common Micriμm Components

There are several CrossCore® Embedded Studio add-ins based on Micriμm's products which share common components. To ensure that the same version of these components is used by all the add-ins that require them, these components are installed in a common location which is distinct from the add-in install folders. These common components are

- μC/CPU which is installed in `%COMMONPROGRAMFILES%\Analog Devices\uC-CPU v1.0.0`. This installation includes μC/CPU version 1.29.01.
- μC/LIB which is installed in `%COMMONPROGRAMFILES%\Analog Devices\uC-LIB v1.0.0`. This installation includes μC/LIB version 1.36.02.

The documentation for these components can be found in CrossCore® Embedded Studio Help under μC/OS-III™ 1.0.0 > Components Shared by Add-ins.

Known issues with μC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio

These are the currently known problems which affect μC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio.

TAR-48108 Status window cannot detect RTOS if -g is not on

The RTOS status window relies on debugging information in the .dxe file to determine if a RTOS has been implemented in the application. In order for this information to be generated the -g compiler switch must be used when compiling the source code which means the RTOS status view works only with the Debug configuration. At this time there is no workaround.

TAR-48180 Some projects included in the installation cannot be built by users

There are some projects located within the μCOS-III release which do not build because the files related to μC-CPU and μC-LIB cannot be located. These projects build the OSAL libraries which are not designed to be built or used directly by applications. In the unlikely case that there is a requirement to rebuild these libraries then the include paths from the projects can be modified to have the folders where all the μC-CPU and μC-LIB header files are located.

TAR-48258 Application-specific files are unexpectedly overwritten after re-adding the RTOS

Due to a mistake in µC/OS-III for CrossCore Embedded Studio, removing and re-adding the RTOS to a project can result on application-specific files being unexpectedly overwritten. These files can be modified by applications to suit their needs and should not have been overwritten. It is recommended that if an application requires changes to these files, a safe copy of them is made to a separate location before removing and re-adding µC/OS-III.

The files are located in the application folder and are called:

```
system\uCOS-III\AppSources\os_app_hooks.c
system\uCOS-III\Cfg\os_app_hooks.h
system\uCOS-III\Cfg\app_ucos3_cfg.h
```

TAR-48440 The file app_ucos3_cfg.h is read-only when it shouldn't

The file `system/uCOS-III/AppSources/app_ucos3_cfg.h` is designed to be edited by µC/OS-III applications instead of `app_cfg.h` when they need to add any application-specific details. µC/OS-III Real-Time Kernel for CrossCore Embedded Studio made the file read-only so when a user tries to overwrite it, the IDE reports that it cannot be overwritten. At this point the IDE offers an option to remove the read-only attribute. Selecting that option will allow users to write to the file.

TAR-48505 RTOS examples are not portable to other workspaces

The µCOS-III examples released with the µCOS-III product reference their source folder (`src`) and their readme in terms of a relative path from the project location. The reason for this was so the same sources and readme could be used for all processors that the example can be run on.

This method of linking files means that if a user chooses to import the project with the "Copy projects into workspace" box selected then the src and readme files are not copied and the example does not work.

If you want to import one of the examples to your workspace then you should follow these instructions:

- Import the example copying it to your workspace. This step creates a new project in the workspace which contains two invalid links, to the readme and to the `src` folder.
- Remove the existing link to the src folder and to the readme by selecting them on the new project and either press the "Delete" key or right-click and choose Delete.
- Close the project.
- Copy the readme and `src` folder to the location of the project in the new workspace. This will automatically add those files to the project. The location of

the µCOS-III examples in the file system is: `%RTOS_INSTALL_FOLDER%\uCOS-III\Examples`.

- Reopen the project.
- If the project was previously built in the installation folder then run "Clean" before rebuilding.

The example should build as work as expected.

TAR-48176 Not resolved errors in system/uC-LIB/Source lib files

This is an issue with CrossCore Embedded Studio but it is included here because it may seem like an issue with µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio.

When using the wizard to create a project that has µC/OS-III and then clicking on the `system/uC-LIB/Source` files `lib_ascii, lib_mem.c, lib_math.c and lib_str.c` files, you may see a number of errors in the Problems window. These are spurious errors detected by the editor and will not impact the ability to build or debug the project.

In order to remove these errors right-click on your project and select Index > Rebuild.

TAR-48520 Strange control characters cause the generation of code coverage to fail

This is an issue with CrossCore Embedded Studio but it is included here because it may seem like an issue with µC/OS-III™ Real-Time Kernel for CrossCore® Embedded Studio.

If a source file in your application contains strange control characters, such as form feeds (FF), then generating a code coverage report will fail. This is the case for example of some of the Micriµm µCOS-III sources.

In the general case, to workaround the issue, perform a search and replace of strange control characters in the source files used by your application, and re-run the code analysis report wizard. We don't recommend editing the RTOS sources.