Release Notes for CrossCore Embedded Studio 2.5.0

Release Notes for CrossCore Embedded Studio 2.5.0
December 2016

http://www.analog.com
processor.tools.support@analog.com

Contents

1 Introduction

This document describes the changes for CrossCore Embedded Studio (CCES) 2.5.0. You can find the release notes for older releases in the `docs` sub-directory of your CCES installation.

1.1 Supported Operating Systems

Notes for Windows Users

The following versions of Windows are supported for this release of CCES:

- Windows 7 Professional, Enterprise, or Ultimate (32 and 64-bit)
- Windows 8.1 Pro or Enterprise (32 and 64-bit)
- Windows 10 Pro or Enterprise (32 and 64-bit)

Notes for Linux Users

This release of CrossCore Embedded Studio for Linux has been provided to support the Linux Add-In for CrossCore Embedded Studio and support bare-metal development on Cortex-M processors such as the ADuCM302x family of MCUs.

The following Linux distributions are officially supported for this release of CCES:

- Ubuntu 14.04 32-bit

The following features are available and supported:

- Compilation using the GNU toolchain for the ADSP-SC58x ARM Cortex-A core.
- Compilation using the GNU ARM toolchain for the ADuCM302x ARM Cortex-M cores.
- Debugging ADSP-SC58x and ADuCM302x via the IDE with GDB/OpenOCD.
- Development and debugging of Applications running under Linux on the ADSP-SC58x ARM Cortex-A core.
- Development and debugging of bare-metal applications on the ADuCM302x ARM Cortex-M core.

The following features are only supported via the Windows version of CrossCore Embedded Studio:

- Development, simulation and debug of Blackfin processors
- Development, simulation and debug of SHARC processors (excluding ADSP-SC58x ARM core)
- Use of CrossCore Embedded Studio Add-Ins other than the Linux Add-In
- Debugging an Application using the CrossCore Debugger (TPSDK)

1.2 System Requirements

Verify that your PC has these minimum requirements for the CCES installation:

- 2 GHz single core processor; 3.3GHz dual core or better recommended
- 4 GB RAM; 8GB or more recommended
- 2 GB available disk space
- One open USB port

> **Note**
>
> A faster disk drive or SSD decreases the build time, especially for a large amount of source files. 8GB of RAM or more will substantially increase the performance of the IDE.

1.3 Obtaining Technical Support

You can reach Analog Devices software and tools technical support in the following ways:

- Post your questions in the software and development tools support community at EngineerZone®
- E-mail your questions about software and development tools directly from CrossCore Embedded Studio by choosing Help > Email Support or directly to processor.tools.support@analog.com
- E-mail your questions about processors and processor applications to processor.support@analog.com
- Submit your questions to technical support directly via http://www.analog.com/support
- Contact your Analog Devices sales office or authorized distributor

2 Installing CrossCore Embedded Studio

2.1 Installing CrossCore Embedded Studio on Windows

Note: Windows Only

> **Caution**
>
> Windows users may experience User Access Control (UAC) related errors if the software is installed into a protected location, such as `Program Files` or `Program Files (x86)`. We recommend installing the software in a non-UAC-protected location.

To install CrossCore Embedded Studio, double-click `ADI_CrossCoreEmbeddedStudio-Rel2.5.0.exe`

To uninstall CrossCore Embedded Studio, click the `Start Menu / Analog Devices / CrossCore Embedded Studio 2.5.0 / Uninstall CrossCore Embedded Studio 2.5.0` shortcut

2.2 Installing CrossCore Embedded Studio on Linux

Note: Linux Only

> **Caution**
>
> It is strongly recommended to use the command prompt to install CrossCore Embedded Studio. Post-install configuration may fail when installing via Ubuntu Software Center.

To install CrossCore Embedded Studio run the following command from the command prompt:

```
sudo dpkg -i adi-CrossCoreEmbeddedStudio-linux-x86-2.5.0.deb
```

To uninstall CrossCore Embedded Studio run the following commands from the command prompt:

```
sudo dpkg -r adi-cces-2.5.0
sudo dpkg -P adi-cces-2.5.0
sudo rm -rf /opt/analog/cces/2.5.0 (to clean up any leftover files)
```

2.2.1 Different users sharing the same CCES license on Linux

Many users can share a single valid `license.dat` file on a system by creating a symbol link to the valid license.dat in their own home directory (`~/.analog/cces`).
The user who installed license should ensure that the appropriate directory and file permissions are set-up to allow other users to access the valid license.dat.

2.2.2 OpenOCD needs to be run as sudo on Linux

In order to debug an Application with GDB and OpenOCD (Emulator) on Linux, OpenOCD needs to have permissions to access your USB device. You can set-up the necessary permissions when installing CCES on Linux by selecting 'Configure OpenOCD permissions' option on the installation dialog or afterwards by running `sudo sh /opt/analog/cces/2.5.0/Setup/setup_openocd_permissions.sh`.

If you debug an Application with GDB and OpenOCD (Emulator) using the IDE and OpenOCD fails, because it cannot access your USB device, a dialog will appear with a message telling you that you can run the `setup_openocd_permissions.sh` script.

If you start CCES with sudo permission, then there should be no problems with OpenOCD accessing your USB device.

3 New and Noteworthy

3.1 Freestanding support for C99 native complex types on SHARC

The CCES compiler now supports the `_Complex` keyword when building in C99 mode, to allow you to write complex arithmetic in accordance with Annex G of ISO/IEC 9899:1999. Although this is a freestanding implementation (meaning that the full set of library functions defined in Annex G is not available) a small set of functions and macros are also available in the `complex.h` header file. Including `complex.h` also enables you to use the more natural spelling `complex.`

3.2 Workarounds for SHARC+ silicon anomaly 20000083

CCES 2.5.0 contains changes to help work around SHARC+ silicon anomaly 20000083 (In presence of the pipe flush, the compute to DAG data forwarding can go wrong for pre modify reads).

- New cc21k `-workaround 20000083` switch has been added. When used the compiler will ensure that the compiler generated assembly doesn't include the instructions sequences that can trigger the anomaly.
- New easm21k `-anomaly-detect 20000083` switch added. When used the assembler will check assembly source and issue a new warning, `ea2569`, when the instruction sequence that can trigger anomaly 20000083 is found.
- Updated runtime library functions : C source rebuild using the compiler workaround and assembly source updated where required as indicated by easm21k

The `-workaround 20000083` and `-anomaly-detect 20000083` switches are enabled by default when building for affected SHARC+ parts. The updated runtime libraries are linked by default.

3.3 Additional ADSP-215xx/ADSP-SC5xx Functional Simulator Support

3.3.1 New Part Support

In addition to supporting ADSP-SC589 and ADSP-21584, CCES 2.5.0 provides functional simulator support for the SHARC+ cores of the following processors:

- ADSP-21573

- ADSP-SC573
- ADSP-SC584

The functional simulator is described in the CrossCore Embedded Studio Online Help, under the following topics:

- 'Simulator User's Guide'
- 'Integrated Development Environment > Debugging Targets > Debugging ADSP-SC5xx SHARC Targets > Using CrossCore Functional Simulator for SHARC+ Cores of an ADSP-SC5xx Application (SHARC+ Debug)'
- 'Integrated Development Environment > Debugging Targets > Debugging ADSP-215xx SHARC Targets > Using CrossCore Functional Simulator for SHARC+ Cores of an ADSP-215xx Application (SHARC+ Debug)'

3.4 Halt peripherals on suspend' is available for ADSP-SC5xx/ADSP-215xx processors

3.4.1 General information

This feature helps debug code that uses timers, watchdog timers, and counters. When the 'Halt peripherals on suspend' option is enabled for a core, the peripherals halt whenever the core halts. This will prevent timers/WDTs from expiring while no application is being executed and also from counters continuing while the core is halted.

Using the CCES IDDE, this option is disabled by default for all cores. If the option is enabled for more than one core, the peripherals will halt whenever any of the selected cores halt.

Note that for multi-core processors such as ADSP-SC5xx, these peripherals are shared by all cores. When multiple processors are in a single JTAG scan chain, the peripherals are halted only for the processor for which the option is enabled.

The peripherals will continue ticking for a short period after the core halts before stopping. In other words, peripherals will expire differently when running through the code versus when debugging the code.

CCES Debugger

To enable the feature with CCES Debugger, in the Debug Configurations window, navigate to the 'Target Options' tab, select the desired core(s), and enable the option 'Halt peripherals upon suspend'. If you have already launched and are connected to the target board, a relaunch is necessary for the option to take effect.

OpenOCD using the IDE

Only the ARM core can be controlled using OpenOCD. To enable the feature, in the Debug Configurations window's 'Target' tab, tick the 'Halt peripherals upon suspend' checkbox. If you have already launched and are connected to the target board, a relaunch is necessary for the option to take effect.

OpenOCD using command line

This feature is enabled by default when using OpenOCD from the command line.
To disable this feature, change the appropriate configuration file in directory: `${CCES}\ARM\openocd\share\openocd\scripts\target\adspsc*.cfg`.

At the top of the file, add the text: `set USE_CTI 0`

3.4.2 Other Processors

Except for the ADSP-BF70x, other processors do not have the ability to enable or disable this feature. The checkbox will appear checked or unchecked depending on whether the feature is always enabled or not available.

3.4.3 Restrictions

- An executable must be loaded to the core for the feature to work properly. (CCES-16261)
- The core must go from running state to suspended state for the peripherals to halt. Therefore if the CCES Debugger Debug Configurations option to 'Run immediately after load' is not selected, the peripherals will not halt even though the core is halted since no run has taken place. (CCES-16255)
- If OpenOCD is used to enable the 'Halt Peripherals on Suspend' option then the board must be power cycled when CCES Debugger is next used, or after disabling the feature in OpenOCD. (CCES-13506)

3.5 'Halt peripherals on suspend' is now disabled by default for ADSP-BF70x processors, can be enabled

The `Halt peripherals on suspend` feature which was added in CCES 2.4.0 and enabled by default, will now be disabled by default, and can be enabled if desired.

Having this featured disabled will help when debugging code that modifies TRU0_GTCL or TRU0_SSR69. When stepping through code that modifies these registers with the 'Halt peripherals on suspend' feature enabled, the registers will be corrupted.

Note that in CCES 2.4.0, these registers were corrupted even when running through code. This issue is now resolved in CCES 2.5.0.

To control the feature with CCES Debugger, in the Debug Configurations window, navigate to the 'Target Options' tab and enable or disable the option 'Halt peripherals upon suspend'. If you have already launched and are connected to the target board, a relaunch is necessary for the option to take effect.

3.6 Improvements to core timer interrupts for ADSP-SC5xx/ADSP-215xx Functional and Cycle-Accurate Simulation

Core timer interrupts were not working properly in CCES 2.4.0 and earlier when using the Functional and Cycle-Accurate simulators on ADSP-SC5xx/ADSP-215xx SHARC+ cores. These issues are now resolved.

3.7 New SHARC+ cache APIs for ranges - can be used to set an non-cacheable range for OTP access

Two new functions have been added for SHARC+ cache configuration. These are:

```
adiCacheStatus adi_cache_set_range(void * _start, void * _end, adiRangeReg _rr,

adiCacheStatus adi_cache_set_disable_range(adiRangeReg _rr);
```

The startup code support has been modified to use `adi_cache_set_range()` and `adi_cache_set_disable_range()` functions.

It is intended that users can use the new `adi_cache_set_range()` API to disable caching for OTP accesses - a documented requirement in the HRM. For an example of how to do this please see below:

```
#include <sys/cache.h>  /* contains prototype and types for
adi_cache_set_range call */
#include <sharc_rom.h>  /* defined ROM_OTP_BASE_ADDR and
ROM_OTP_END_ADDR macros */
/* ... snip...*/
   adiCacheStatus r = adi_cache_set_range((void *)ROM_OTP_BASE_ADDR,
                                          (void *)ROM_OTP_END_ADDR,
                                          rr4,

adi_cache_noncacheable_range);
```

For more information on these new functions, please see their references pages in the CCES help.

3.8 CCES IDE updated to use Java SE Runtime Environment 8

The Java run-time environment that the CrossCore Embedded Studio Eclipse-based IDE uses has been upgraded to Java SE Runtime Environment (build 1.8.0_102-b14).

Please visit http://www.oracle.com/technetwork/java/javase/overview/index.html for more information.

3.9 CMSIS-Pack file support

CrossCore Embedded Studio (CCES) now supports CMSIS-Pack files.

CMSIS-Pack files are installed by default into **`C:\Analog Devices\CrossCore Embedded Studio 2.5.0\ARM\packs`**. Their location can be configured using the preference **`Window | Preferences | CMSIS Packs`**.

3.9.1 Pre-installed CMSIS-Pack files

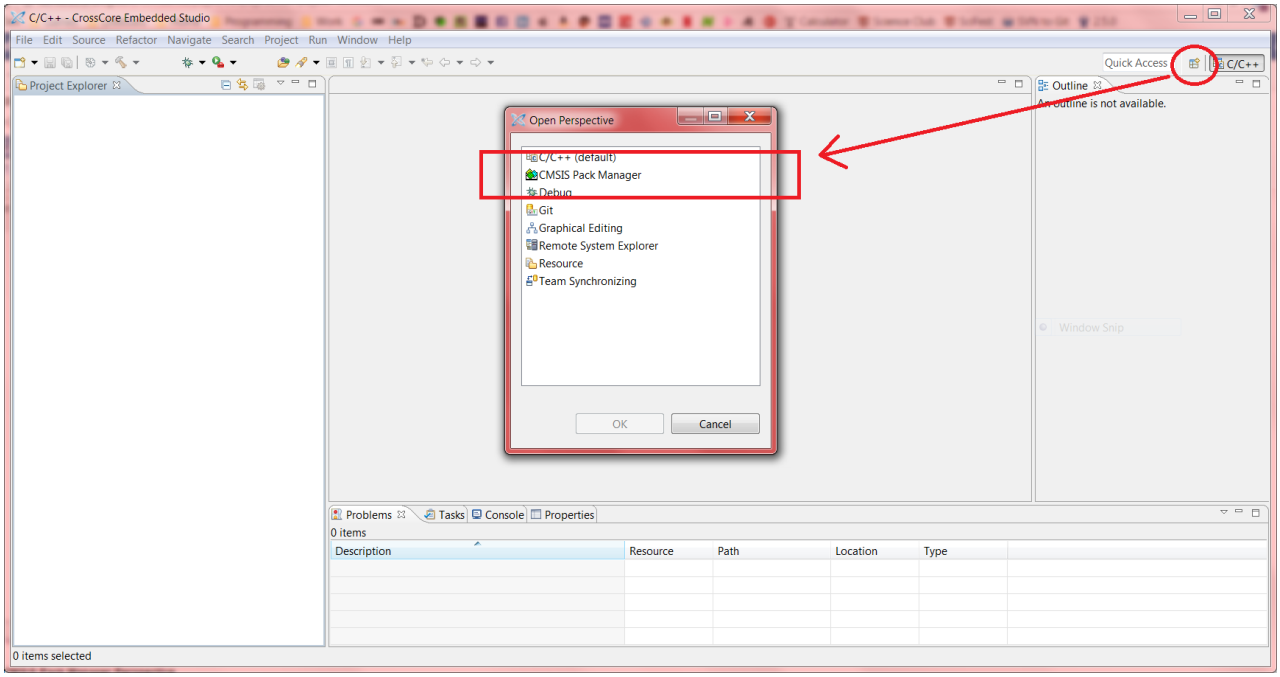
Analog Devices' ADuCM302x Device Family Pack (DFP) is pre-installed into **`C:\Analog Devices\CrossCore Embedded Studio 2.5.0 \ARM\packs\AnalogDevices\ADuCM302x_DFP\1.0.3.`**

3.9.2 CMSIS-Pack files can be managed using the IDE's CMSIS Pack Manager Perspective

CMSIS-Pack files can be installed and removed using the CMSIS Pack Manager.

3.9.3 CMSIS Pack Manager Perspective

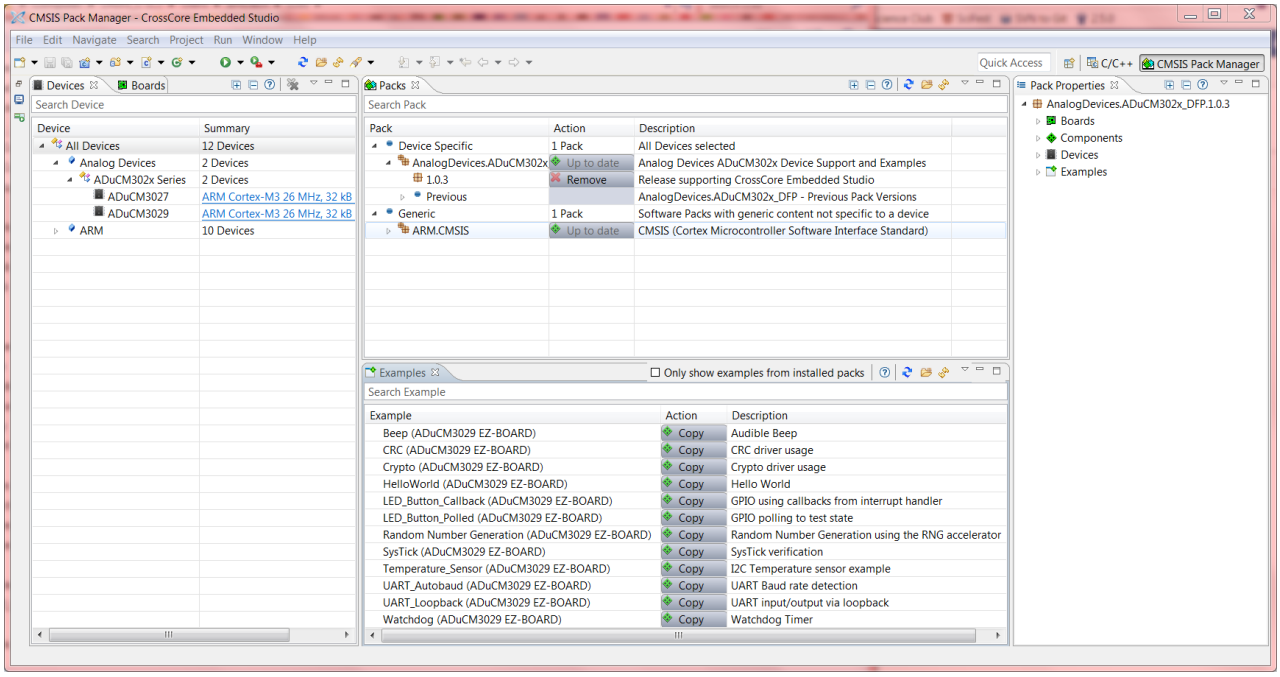
To open the CMSIS Pack Manager Perspective, click "Open Perspective" icon on tool bar and select "CMSIS Pack Manager" in the Open Perspective window.



3.9.4 CMSIS Pack Manager Perspective

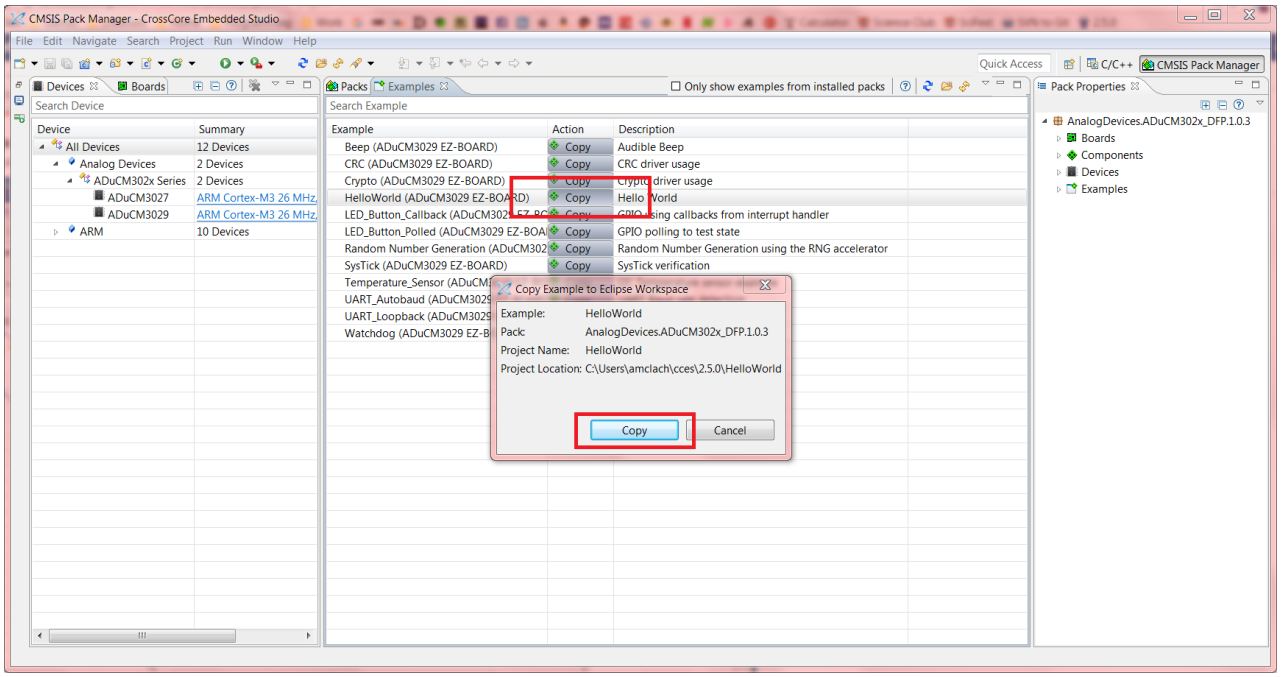
The CMSIS Pack Manager Perspective opens Views to examine the supported Devices, Boards, Packs and available Examples.

For more information on the Views in this Perspective, please see the *CMSIS C/C++ Development User's Guide* in the CCES On-line Help.



3.9.5 Examples provided

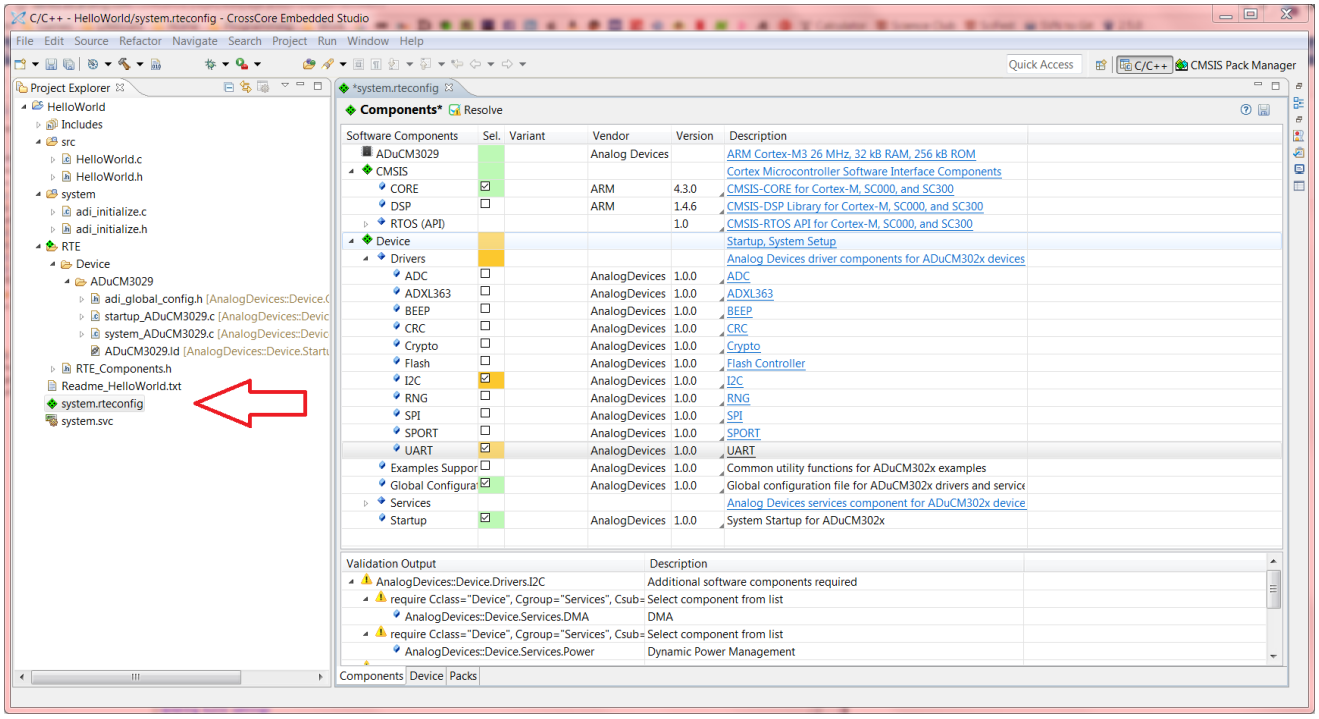
The CMSIS Pack Manager Perspective opens an Examples View that can be used to open example projects supplied within a CMSIS-Pack file.



3.9.6 Adding CMSIS components to a project for a Cortex-M based processor (e.g. ADuCM3029)

CrossCore Projects created for Analog Devices' Cortex-M based processors, such as ADuCM3029 include a **`system.rteconfig`** file. Opening this file within the IDE will open the Run-Time Environment (RTE) Configuration editor.

Components from the CMSIS-Pack, such as drivers and services, can be added to a project by selecting them in the editor and clicking Save.



3.9.7 Enhanced documentation regarding SHARC+ MODE2 Status Register and Emulation Restrictions

A new section has been added to the Online Help to guide users when setting the SHARC+ MODE2 status register and using the CrossCore Debugger for emulation. The new section is listed under 'Integration and Development Environment > Debugging Targets' and contains the following information:

The emulation software modifies the SLOWLOOP bit (bit 7) in the MODE2 status register depending on whether a user wants slow loop operation for user mode debug operations or not. It is recommended that applications do not modify this bit. Instead, users should read the MODE2 status register, change only the bits needed, and then write the new value back.

See `About Hardware Counter-Based Loops for SHARC+ Cores` and `Target Options Dialog Box for SHARC Processors` in the Online Help for more information.

4 Changes That Might Impact Backwards Compatibility

4.1 Silicon revisions no longer supported

Support for the following silicon revisions has been removed from CCES 2.5.0:

- SHARC
  - ADSP-SC58[23479] 0.0
  - ADSP-2158[347] 0.0
  - ADSP-2136[789] 0.1
- Blackfin
  - ADSP-BF70[0-7] 0.0
  - ADSP-BF60[6789] 0.0
  - ADSP-BF52[246] 0.1
  - ADSP-BF51[2468] 0.1

4.2 New ADuCM3027/9 projects should use the Analog Devices' ADuCM302x Device Family Pack (DFP)

CrossCore Projects for Analog Devices' Cortex-M based processors, such as ADuCM3029, which were created with CCES 2.4.0 will use Add-in components to add start-up code, device drivers and services to the project. In CCES 2.5.0, this has changed so that new projects use components from the CMSIS-Pack instead.

Unfortunately there is no easy way to migrate a CrossCore Project for Analog Devices' Cortex-M based processors created with CCES 2.4.0 over to using CMSIS-Pack components. CCES 2.5.0 does not offer a migration option.

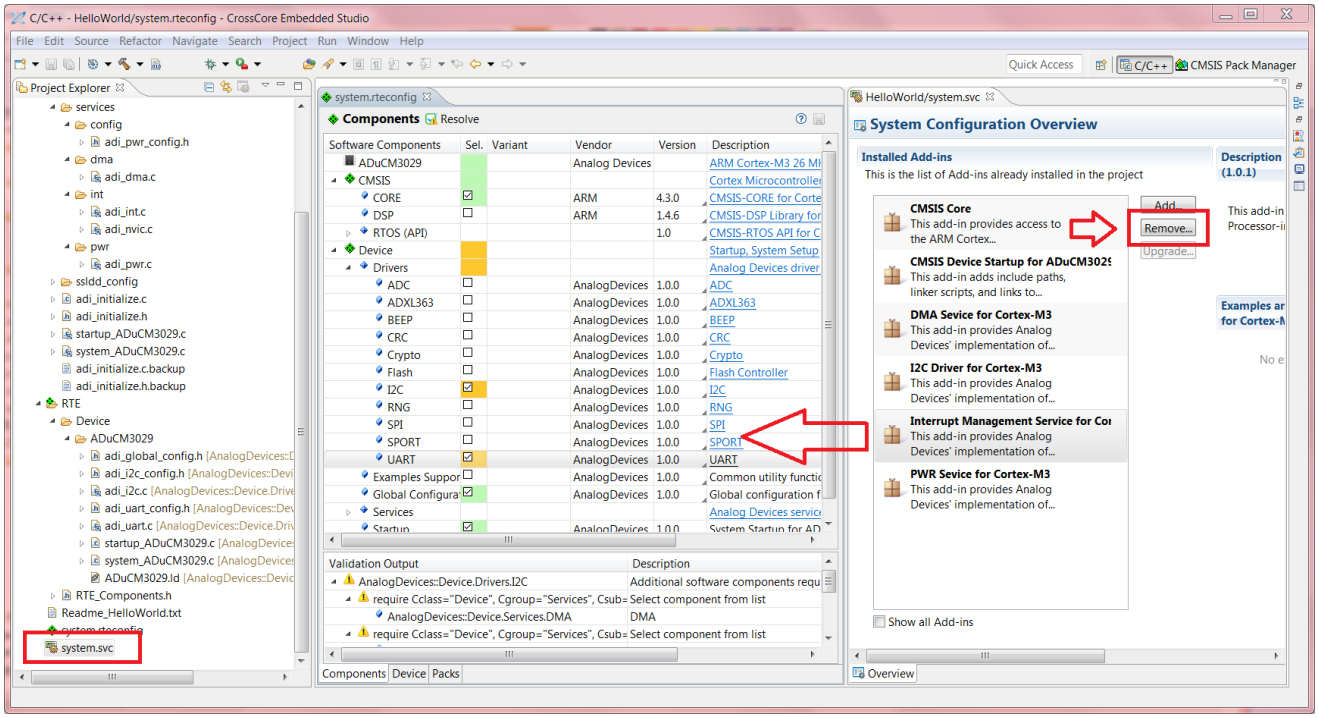
Please contact processor.tools.support if you would like assistance moving an existing project over to using CMSIS-Pack components.

4.2.1 Recommended

We recommend that all new CrossCore Projects for Analog Devices' Cortex-M based processors use CMSIS-Pack components, added using the Run-Time Environment (RTE) Configuration editor.

System Configuration editor is still available to CrossCore Projects that are set-up to use CMSIS-Pack components

It is still possible to add start-up code, device drivers and services to a CrossCore Project that is set-up to use CMSIS-Pack components via the System Configuration editor (rather than the RTE Configuration editor) as shown below, but doing so is not recommended.



4.2.2 Backwards compatibility

Add-in components used in by CrossCore Projects for Analog Devices' Cortex-M based processors created with CCES 2.4.0 will be upgraded when they are opened with CCES 2.5.0.

Project options, such as include search paths and project resource links, such as start-up code, device drivers and services will be updated, so that they point to the new files in the pre-installed Analog Devices' ADuCM302x Device Family Pack (DFP).

4.3 Interrupt handler names in the startup code for ADuCM3027/9 do not match what is expected by the device drivers

The `startup_ADuCM302*.c` files in the ADuCM302x CMSIS pack (`ARM/packs/AnalogDevices/ADuCM302x_DFP/1.0.3/Source/GCC`) for CCES fill the interrupt vector table with dummy handlers that are intended to be overridden by applications.

Unfortunately in the CCES 2.4.0 release, the startup files use interrupt handler names based on the default CMSIS startup code, which do not match the handler names in the startup code for IAR and Keil. The device drivers and services use the latter names, which means that their interrupt handlers do not override the default ones and will not be invoked when built in CCES 2.4.0. This has been corrected in this release of CCES.

Fix for Custom LDF project setting is ignored on the IDDE linker command-line (CCES-14516)

The IDDE no longer ignores the -T switch for custom LDF project. This may result in backward compatibility issues as the linker will no longer use the default `app.ldf` provided in the project when the -T switch is used.

4.4 Linked objects on the command line should be in alphabetical order (CCES-15752)

CCES links an application by placing the object file names into an input file. The object file (s) are written to that file in no particular order. For example, they are not sorted alphabetically.
You may have a project that happens to work because of the order in which the objects are placed in the input data file, and you may find that your project does not build successfully in later releases of CCES. In such a case, you should review your project set-up to see if there was a problem that already existed but did not show up until the order of the objects on the linker command line was changed.

4.5 Taking the address of the cabs , cexp and conj functions

In the order to support C99 native types, the `cabs`, `cexp` and `conj` functions have been declared in `complex.h` as macros. If you need to take the address of these functions, please define the `ADI_COMPLEX_STRUCT_FORM` or `ADI_COMPLEX_C99` macros before including the `complex.h` header, according to which definition of complex types you wish you use.

5 Known Issues

5.1 X11 forwarding is required to create or build projects from a command line with CCES on Linux remotely by SSH

Note: Linux Only

If you want to use the CrossCore Embedded Studio headless tools application to create or build projects from a command line on Linux remotely by SSH, then you will need to setup:

1. X11 forwarding needs to be enabled on both the client side and the server side.
2. Have X Server (e.g. Xming) setup on your client side.

To make sure your server side has enabled X11 forwarding, check if its `/etc/ssh /sshd_config` contains:

```
X11Forwarding yes
X11DisplayOffset 10
```

If you use Putty as client, enable the X11 forwarding option by checking `Configuration > Connection > SSH > X11 > Enable X11 forwarding` and adding an X display location (e.g. localhost:0.0).

If you use Cygwin as client, enable the X11 forwarding by the following commands:

```
export DISPLAY=localhost:0.0
ssh -XY username@remote_server_ip
```

5.2 For More Information

For the latest anomalies please consult our Software and Tools Anomalies Search page.