CrossCore® Development Tools by Analog Devices

Release Notes for CrossCore Embedded Studio 2.7.0

Contents

1 Introduction

This document describes the changes for CrossCore Embedded Studio (CCES) 2.7.0. 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

New in CCES 2.7.0: Use on Windows 7 requires Service Pack 1 or greater.

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

- Windows 7 Professional, Enterprise, or Ultimate (32 and 64-bit, SP1 or later)
- Windows 8.1 Pro or Enterprise (32 and 64-bit)
- Windows 10 Pro or Enterprise (32 and 64-bit)

Notes for Linux Users

New in CCES 2.7.0: CrossCore Embedded Studio now formally supports Ubuntu 16.04 in addition to 14.04.

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 and ADuCM4050 family of MCUs.

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

- Ubuntu 14.04 32-bit
- Ubuntu 16.04 32-bit

The following features are available and supported:

- Compilation using the GNU toolchain for the ARM Cortex-A core on ADSP-SC57x and ADSP-SC58x processors.
- Compilation using the GNU ARM toolchain for the ADuCM302x and ADuCM4x50 ARM Cortex-M cores.
- Debugging ADSP-SC58x, ADuCM302x and ADuCM4x50 via the IDE with GDB/OpenOCD.
- Development and debugging of Applications running under Linux on the ARM Cortex-A core on ADSP-SC57x and ADSP-SC58x processors.
- Development and debugging of bare-metal applications on the ADuCM302x and ADuCM4x50 ARM Cortex-M cores.

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 the ARM Cortex-A core on ADSP-SC57x and ADSP-SC58x processors)
- 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.

> **Caution (Windows 8.1 users)**
>
> *Prior to installation:* Ensure your machine is up-to-date with relevant Windows updates from Microsoft. CrossCore Embedded Studio relies upon the Microsoft Universal C Runtime from VisualStudio 2015, and this can silently fail to install if your machine is out of date. For more details, refer to Update for Universal C Runtime in Windows on Microsoft's web site.

To install CrossCore Embedded Studio, double-click `ADI_CrossCoreEmbeddedStudio-Rel2.7.0.exe`

To uninstall CrossCore Embedded Studio, open Control Panel / Programs and Features applet, and select to uninstall CrossCore Embedded Studio 2.7.0. You may need to delete the installation directory to clean up any leftover files.

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. The installation may not work properly when using Ubuntu Software and/or Ubuntu Software Center.

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

```
sudo dpkg -i adi-CrossCoreEmbeddedStudio-linux-x86-2.7.0.deb
```

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

```
sudo dpkg -r adi-cces-2.7.0
sudo dpkg -P adi-cces-2.7.0
sudo rm -rf /opt/analog/cces/2.7.0 (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/2.7.0/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 Ubuntu 16.04 32-bit

CrossCore Embedded Studio (CCES) can now be used on Ubuntu 16.04 32-bit

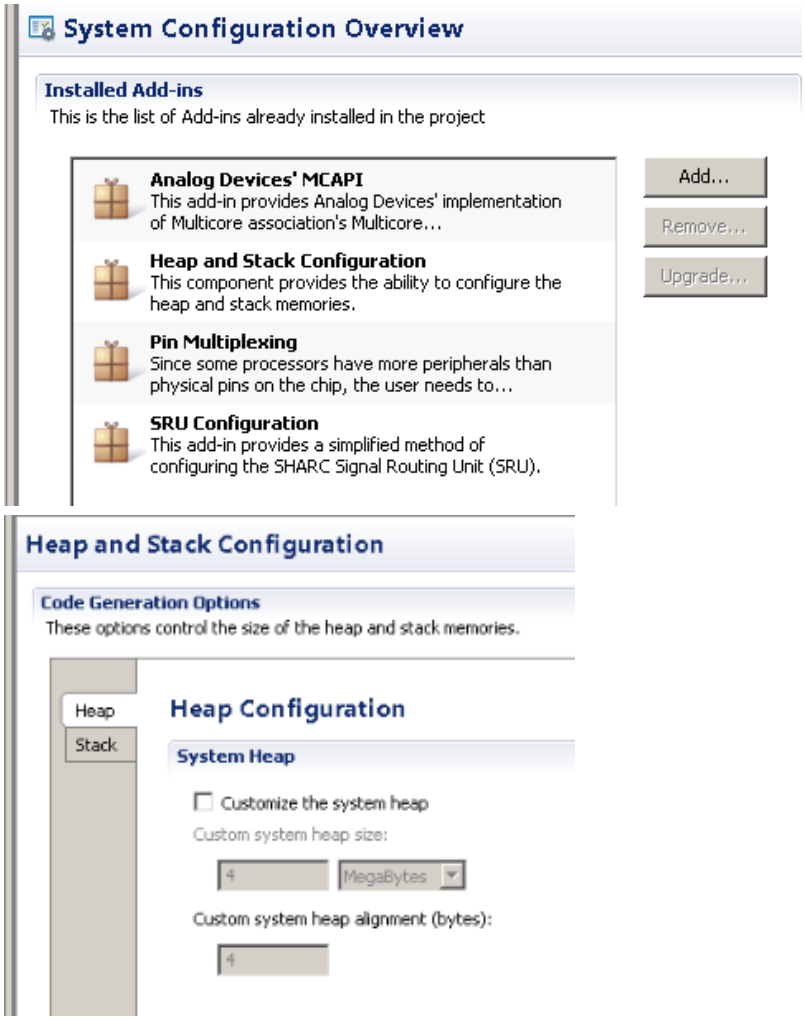
3.2 Cortex-M toolchain updated to GCC 6.x

The Cortex-M toolchain has been updated to release gcc-arm-none-eabi-6-2017-q2-update of the GNU ARM Embedded toolchain, which is based on the following components:

```
* gcc : ARM/embedded-6-branch revision 249437
  svn://gcc.gnu.org/svn/gcc/branches/ARM/embedded-6-branch/

* binutils : 2.28
  git://sourceware.org/git/binutils-gdb.git commit c0a558756bcf42dc25

* newlib and newlib-nano :
  git://sourceware.org/git/newlib-cygwin.git commit 0d79b021a4ec4e6b9

* gdb : 7.12 without target sim support
  git://sourceware.org/git/binutils-gdb.git commit 17265fcd6b8b640682
```

3.3 Heap and Stack Size configuration Add-in for ARM core on ADSP-SC5xx parts

When a new project is created for ADSP-SC5xx parts, a Heap and Stack Configuration Add-In is now installed by default in the System Configuration for the ARM Cortex-A core. This enables you to configure the sizes of your application's heap and stack memories. To change the settings, open the System Configuration by double clicking the system.svc file, and select the "Heap and Stack" tab. You may then customize the size of each heap or stack by checking the "Customize the ..." check box and selecting or entering the values you require.



3.4 The Register Browser now supports Cortex-A and Cortex-M with GDB

You can create multiple register tabs and select some core registers to monitor in different register tabs, also you can use the filter to find the target registers.

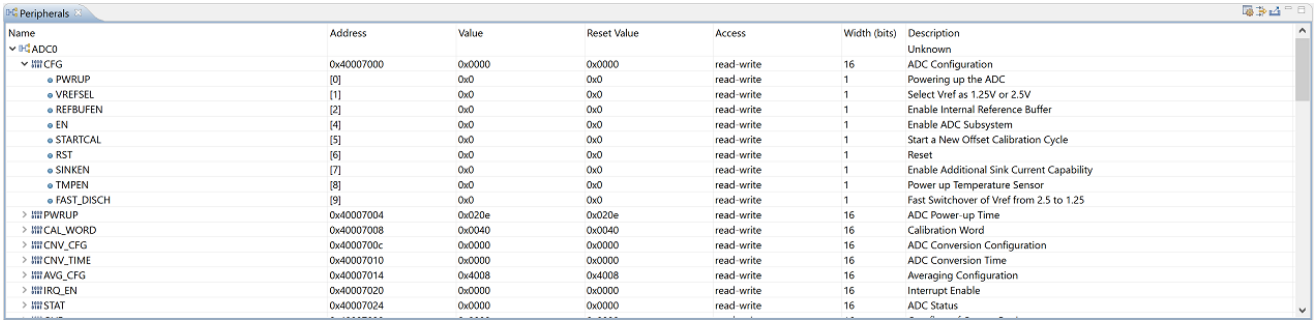
3.5 The Register Browser is now more responsive for large register groups

The response time for large register groups, such as SEC0, is faster now.

The IDE improves the implementation of register group data retrieval, and changes to use a paged tree to show the result. If the count of register items is large in the register browser view, for instance, 1000, CCES will show the first 500 items. It will continue to show the remaining items when you scroll down to the bottom of the tree.

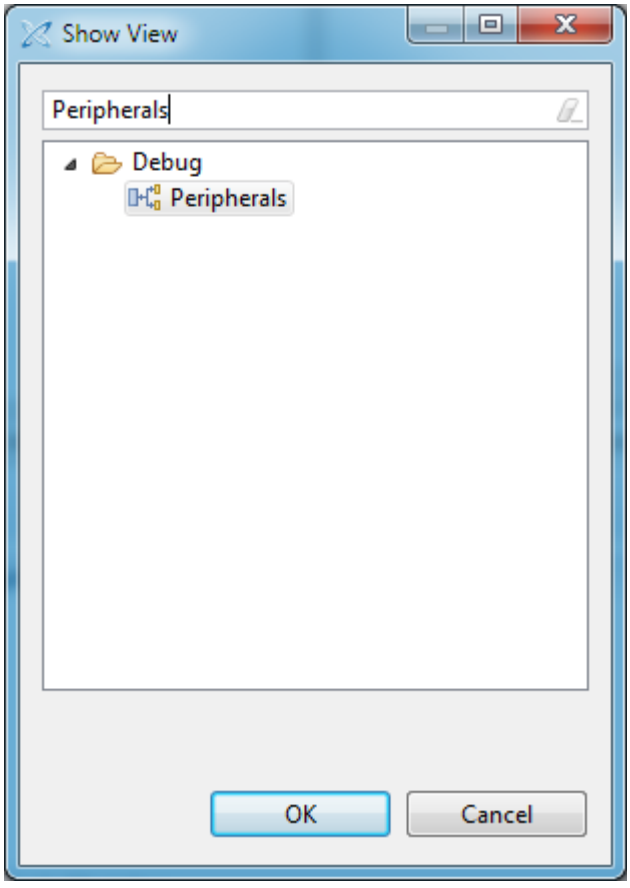
3.6 New Peripherals View which supports memory mapped registers for Cortex-M with GDB/OpenOCD

The Peripherals view displays each peripheral and its memory mapped registers and allows the user to view / edit the register values. It also allows the user to select certain registers / peripherals to show in the view, and the user can select to show / hide certain column(s). The Peripherals view also supports exporting registers' value into a local file.



The Peripherals View can be opened from the main menu from within the Debug perspective:

`Window -> Show View -> Other... under Debug`



For more documentation on how to use the Peripherals view, see the CrossCore Embedded Studio Online Help under **Integrated Development Environment / Eclipse Documentation / CMSIS C/C++ Development User's Guide / Peripherals View**.

3.7 Strided data for large-regime FFTs using the DMA descriptor chain model on ADSP-SC58x FFTA accelerator

New functions have been added to support strided (non-contiguous) input and output data when using the FFTA to perform large-regime (num points >= 4096) FFTs using the DMA descriptor chain model.

The new functions are:

- `accel_cfft_large_set_tcbs_strided`
- `accel_cfft_large_windowed_set_tcbs_strided`
- `accel_cfft_large_mag_sq_set_tcbs_strided`
- `accel_cfft_large_windowed_mag_sq_set_tcbs_strided`
- `accel_ifft_large_set_tcbs_strided`
- `accel_ifft_large_windowed_set_tcbs_strided`
- `accel_rfft_large_set_tcbs_strided`
- `accel_rfft_large_windowed_set_tcbs_strided`
- `accel_rfft_large_mag_sq_set_tcbs_strided`
- `accel_rfft_large_windowed_mag_sq_set_tcbs_strided`

Please see the SHARC compiler and library manual for more information.

3.8 Improved support for long command-line strings

Long toolchain command lines, such as a compiler command with many include directories, can cause the build to fail unexpectedly if the command-line length exceeds the length supported by the Windows operating system. To reduce the size of the command line and avoid this problem, CCES now generates "includes-XXX.txt" files during the build, where 'XXX' is a unique identifier for the file contents. These files contain the '-I' command for each include directory in the same order as listed in the Tool Settings. The "includes-XXX.txt" files are added to the command line via "@includes-XXX.txt" and can be found in the project's Debug / Release build directory.

This change has been applied to ARM, Blackfin and SHARC compilers for all processors.

3.9 C++ project creation support in the headless builder

The CCES Headless Builder has been updated to allow users to create C++ projects via the command line using the "`-language C++`" command line switch and via the json input file (using "`-input-file`"). The language defaults to 'C' if the option is omitted.

```
ccesc.exe
    -nosplash
    -application com.analog.crosscore.headlesstools
    -command projectcreate
    -data c:/workspace
    -project ./test
    -project-name foo
    -processor ADuCM4050
    -type Executable
    -language C++
```

JSON input file

```
...
"basicInfo" : {
    ...
    "language" : "C++",
    ...
}
...
```

3.10 Allow the user to specify FPU for Cortex-M parts via headless tools

The CCES headless builder has been updated to allow users to disable the FPU for Cortex-M parts using the `-nofpu` command line switch and via the json input file (using `-input-file`). If the option is omitted, CCES will use the processor default for the FPU.

```
ccesc.exe
    -nosplash
    -application com.analog.crosscore.headlesstools
    -command projectcreate
    -data c:/workspace
    -project ./test
    -project-name foo
    -processor ADuCM4050
    -type Executable
    -nofpu
```

JSON input file

```
...
"basicInfo" : {
    ...
    "fpu" : "NO_FPU",
    ...
}
...
```

3.11 DDR configuration register settings for ADSP-SC5xx and ADSP-215xx processors have been modified

The values of the certain DDR configuration registers in the preload and initcode files have been updated to resolve JEDEC compliance issues with DDR memory on ADSP-SC5xx EZ-Kits. For ADSP-SC58x/2158x, the PADCTL2 register has been updated, and for the ADSP-SC57x/2157x, the PADCTL2 and EMR1 registers have been updated.

3.12 Device Driver for CAN controller

CAN Driver support for ADSP-SC589 has been added to CCES 2.7.0 release.

The CAN driver is divided into two modules:

- A CAN device driver provides API’s to configure and use CAN controller
- A CAN stack driver which is a wrapper for the CAN device driver to meet specific requirements like queuing of messages in case mailboxes are not free, discarding of old TX messages, etc...

The CAN driver supports both CAN controllers. It is not mandatory to use the CAN Stack driver. The CAN driver can be used stand alone and user can directly call CAN Device driver APIs.

4 Changes That Might Impact Backwards Compatibility

4.1 RR4 non-cacheable range in start-up code for ADSP-SC57x and ADSP-SC58x

The start-up code for the SHARC+ cores of ADSP-SC57x and ADSP-SC58x family processors now set up range register pair RR4 to configure the ARM core's portion of SDRAM as uncached. This is done to allow the ARM core to use its SDRAM memory to allocate buffers that are accessed by applications running on the SHARC cores without the possibility of the SHARC caches holding stale values or not writing back values required by the ARM core. This change is necessary for applications using MCAPI.

The use of RR4 in this way is not done when any of the following is true

- build target is ADSP-2157x, ADSP-2158x or any other ADSP-21xxx processors as they do not have an ARM core
- build target is ADSP-SC570 or ADSP-SC571 processors as these processors don't support SDRAM
- when the SHARC+ caches are disabled in Startup Code/LDF addin setting.
- when "Use External Memory (SDRAM)" is not enabled
- for projects that are using a custom start-up or LDF

There are two possible methods to restore the previous behaviour should you not want to use RR4 as described above:

- Unset the Startup/LDF addin "***Non-cacheable range for ARM core SDRAM***" check-box found on the Cache Configuration using the system.svc configuration editor.
- Call the `adi_cache_set_disable_range()` function at the start of `main()` to disable the range again, for example:

```
#include <sys/cache.h>

int main(int argc, char argv[]) {
   /* disable range RR4 that was setup in the start-up */
   adiCacheStatus r = adi_cache_set_disable_range(adi_cache_rr4);

   /* ... rest of application ... */
}
```

To add support for this new functionality to custom start-up and LDF files

1. Add labels `___sdram_arm_start` and `___sdram_arm_end` to your ADSP-SC5xx part LDF. Refer to `<CCES 2.7.0 install folder>/SHARC/ldf/ADSP-SC589.ldf` for an example of how this is done.
2. Insert a call to `__lib_set_noncacheable_arm_cache_range` in your custom start-up code. See `<CCES 2.7.0 install folder>/SHARC/lib/src/crt_src/SC5XX_hdr.asm` for an example of how this is done.

4.2 Emulator runs from current PC on terminate (ADSP-BF70x only)

Previous releases of CCES would jump to a stall loop and run in that loop continuously on terminate. Now the default behavior has been changed to have the processor continue running from the current location when terminating the connection.

5 Known Issues

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