Release Notes for CrossCore Embedded Studio 2.4.0

Release Notes for CrossCore Embedded Studio 2.4.0
September 2016

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

Contents

1 Introduction

This document describes the changes for CrossCore Embedded Studio (CCES) 2.4.0. You can find the release notes for older releases in the `docs` sub-directory of your CCES installation.

1.1 Supported Operating Systems

This release of CCES is supported on the following operating systems:

- 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)

> **Note**
>
> 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.

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
- 1 GB RAM; 4GB 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. 4GB of RAM or more will substantially increase the performance of the IDE.
> - For proper viewing of documentation under Windows, Internet Explorer 9 or greater is recommended.

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 New and Noteworthy

2.1 ADuCM3027/9 Support

This release introduces support for the ADuCM302x family of processors. The processor family is fully supported in the CCES Eclipse IDE.

2.1.1 GCC ARM Embedded toolchain

ADuCM302x support is provided by the *GNU ARM Embedded* toolchain release 5-2016-q2. This is configured for Cortex-M processors, including libnano library configurations for low-memory-footprint applications.

2.1.2 CMSIS pack file pre-installed

Beyond the generic Cortex-M core support, processor support is provided by a CMSIS-PACK processor description package, which is pre-installed in the CCES 2.4.0 product. It can be found within the ARM/packs/AnalogDevices/ADuCM302x_DFP/1.0.0 directory of your installation.

2.1.3 New EZK license supported

This release includes a new EZK license for the ADuCM3029 EZ-KIT®. This license enables IDE support for ADuCM3029.

2.1.4 Limited Hardware breakpoints when running from ADuCM302x flash

By default, applications built for ADuCM302x processors are mapped into flash memory and execute from there. Applications running in flash have to be debugged using hardware breakpoints, rather than software breakpoints. The hardware configuration for the ADuCM302x processors only includes two hardware breakpoints. One of these is consumed by the automatic breakpoints placed on main() (when running to main), which leaves one free to use.

2.2 ADSP-BF70x Silicon Anomaly 19000058 Workaround

Silicon anomaly 19000058 where a system MMR read may cause the Blackfin+ core to hang, affecting ADSP-BF70x parts, has workarounds supported in CCES 2.4.0. The support comprises:

- ccblkfn compiler workaround for C/C++ source

- libraries - services, drivers and OSAL libraries for ADSP-BF70x parts have been built with the updated compiler. Assembly sources for other CCES libraries were reviewed and found to not require updates.

These workarounds are enabled by default when building for ADSP-BF70x parts using CCES 2.4.0. You can disable these workarounds by adding the `-no-workaround 19000058` switch to Compiler additional options and Linker driver additional options.

The compiler determines system MMRs reads by inspecting literal address such as are available when referencing one of the cdefBF707.h defined pREG macros. Volatile pointer accesses that the compiler cannot determine the address of are also considered to be possible system MMR accesses requiring the workaround by the compiler unless the `-no-assume-vols-are-mmrs` switch is used.

2.3 OpenOCD updates

2.3.1 OpenOCD has moved

OpenOCD can be used to debug programs built with the GNU ARM toolchain for Cortex-A and Cortex-M cores. OpenOCD can be used from the CCES IDE or on a command line. The location of OpenOCD has moved from ARM\arm-none-eabi\bin to ARM\openocd.

2.3.2 OpenOCD halts peripherals when debugging ADSP-SC5xx

OpenOCD now by default will halt the timers, watchdog timers, and counters when halting and stepping.

Note that when this feature is enabled, the target board will require a push-button reset or power cycle for the watchdog timer to work as expected, if switching from OpenOCD to CrossCore Debugger on any core, or after disabling this feature for OpenOCD.

To disable this feature, change the appropriate configuration file in directory: C:\Analog Devices\CrossCore Embedded Studio 2.4.0\ARM\openocd\share\openocd\scripts\target\adspsc*.cfg. At the top of the file, add the text:

```
set USE_CTI  0
```

2.4 Additional ADSP-SC5xx and ADSP-215xx Cycle-Accurate Simulator Support

2.4.1 New Part Support

In addition to supporting the ADSP-SC589, CCES 2.4.0 provides cycle-accurate simulator support for the SHARC+ cores of the following processors:

- ADSP-SC570
- ADSP-SC571
- ADSP-SC572
- ADSP-SC573
- ADSP-SC582
- ADSP-SC583
- ADSP-SC584
- ADSP-SC587
- ADSP-21571
- ADSP-21573
- ADSP-21583
- ADSP-21584
- ADSP-21587

Please note the functional simulator only supports the ADSP-SC589 and ADSP-21584 processors.

The Cycle Accurate Simulator is described in the CrossCore Embedded Studio Online Help, under 'Simulator User's Guide > Usage' and under 'Integrated Development Environment > Debugging Targets > Debugging ADSP-SC5xx SHARC Targets > Using CrossCore Cycle-Accurate Simulator for SHARC+ Cores of an ADSP-SC5xx Application (SHARC+ Debug)'

2.4.2 Instrumented profiling support

Instrumented profiling can now be used with the cycle-accurate simulator.

2.5 ADSP-SC5xx/ADSP-215xx read side effect registers can be overridden

Certain registers are known to cause side effects when read through the CCES debugger using the Register Browser and Registers windows. Since there is no way for users to stop these registers from being read, we have compiled a list of registers that the debugger will no longer read from the processor when the register window in which they reside is open. The value shown for these registers instead of reading them from the processor is all E's. If there are any children registers for these read side effect registers, their contents are considered NA and should be ignored. The list of registers compiled thus far is as follows:

- `GICCPU1_INT_ACK`
- `EMAC0_TM_STMPSTAT, EMAC0_LPI_CTLSTAT, EMAC0_MMC_RXINT, EMAC0_MMC_TXINT, EMAC0_DMA0_MISS_FRM, EMAC0_DMA2_MISS_FRM`
- `EMAC1_MMC_RXINT, EMAC1_MMC_TXINT, EMAC1_DMA0_MISS_FRM`
- `USB0_INTRTX, USB0_INTRRX, USB0_IRQ, USB0_DMA_IRQ, USB0_LPM_IRQ`
- `USB1_INTRTX, USB1_INTRRX, USB1_IRQ, USB1_DMA_IRQ, USB1_LPM_IRQ`
- `FIR0_MACSTAT`
- `IIR0_DMASTAT`
- `DAI0_IRPTL_H, DAI0_IRPTL_L`
- `DAI1_IRPTL_H, DAI1_IRPTL_L`
- `TWI0_RXDATA8, TWI0_RXDATA16`
- `TWI1_RXDATA8, TWI1_RXDATA16`
- `TWI2_RXDATA8, TWI2_RXDATA16`
- `UART0_RBR`
- `UART1_RBR`
- `UART2_RBR`
- `DMC0_CPHY_CTL`
- `DMC1_CPHY_CTL`

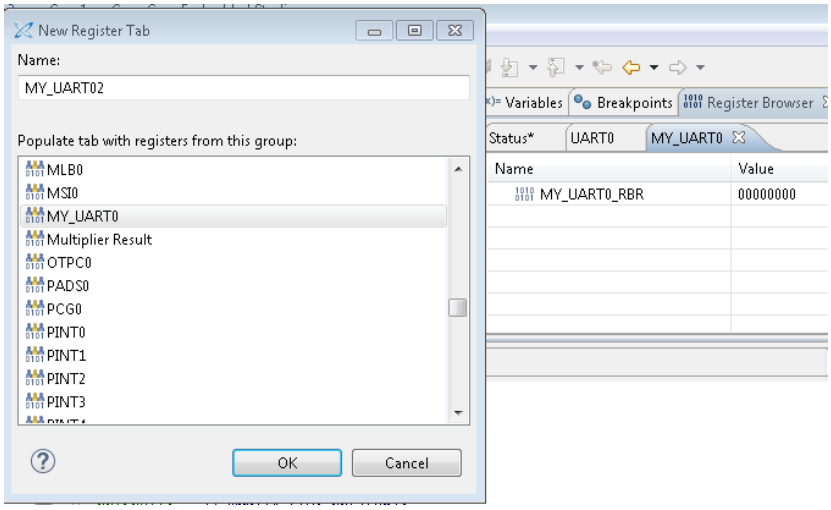
Using a custom xml file, these registers can now be read if needed. An example custom xml file is shown below that will allow the UART0_RBR register to be read even though the standard register window will not allow the read to occur. The register name used needs to be different from the register name in the list above in order to read the register correctly. The correct read-address, write-address, and mask width is essential to allowing the register access to be successful.

Custom Window XML

```
<?xml version="1.0" standalone="yes"?>
<custom-cces-proc-xml
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="\Analog Devices\CrossCore
Embedded Studio 2.4.0\System\ArchDef\ADSP-custom-board.xsd"
    processor-family="Sharc"
    file="sc589_custom_window.xml">
<custom-register-extended-definitions>
<register name="MY_UART0_RBR" read-address="0x31003020" write-address
="0x31003020" bit-size="32" type="IO" mask="FFFFFFFF" group="MY_UART0
" description="MY UART0 RBR" />
</custom-register-extended-definitions>

</custom-cces-proc-xml>
```

To open the new custom register window in a Register Browser, use the ***group*** name found in the custom xml file. In this example, group="MY_UART0". The screenshot below shows how to access the register.



For more information on how to use a custom board support file, please search the CrossCore Embedded Studio online help for *custom board support*.

Note: These registers should not be accessed using the Memory Browser.

2.6 New tool for validating a secure boot stream

A new tool, `securebootsim.exe`, is now provided to enable you to validate a boot stream that has been signed using the `signtool` utility. More details can be found in the "Validating A Signed Boot Stream" section of the online help.

2.7 Improved debugging with ARM infinite loops (CCES-13753)

Using the CrossCore Debugger, it is now possible to halt, step and relaunch when the ARM core is executing a single-instruction loop such as generated with the instruction "while (1);".

2.8 Improved debugging with system reset, reset MP group (CCES-15619)

When a problem occurs during a system reset, the CrossCore Debugger now emits a 'Problem Occurred' pop-up, with 'TpsdkServer] Failed to reset MP group: Error code 8004806e' in the Details pane. When this is seen, reset board and try again. In previous releases, the same problem would cause CCES to hang.

3 Changes That Might Impact Backwards Compatibility

3.1 MP Resume for ADSP-215xx/ADSP-SC5xx (CCES-15487)

In previous releases, MP Resume would start running the cores such that the booting core(core 0) starts first and then cores 1 and 2 in the case of ADSP-SC5xx processors. Since adi_core_enable() will not work properly on a core that is not running, this order has been updated. Now the booting core(core 0) is run last and the other 2 cores are run in order. So MP resume will now run cores 1 and 2, and then run core 0 last.

Note that when using the "Run" and "Suspend" commands on individual cores, this order should be observed in order to properly perform a core reset on the SHARC+ cores when adi_core_enable() is executed.

In Release 2.3.0, this issue could cause the SHARC+ cores not to be reset properly. This may cause differences in behavior between 2.4.0 and 2.3.0 when using the CrossCore Debugger.

Online and FAQ documentation for 2.3.0 and earlier included steps to manually set the RCU0_MSG_SET bit to release cores, or use of a custom xml to set this register when loading. These methods should not be used when debugging. The adi_core_enable() command executes a core reset in addition to releasing the cores,which simply setting the RCU0_MSG_SET register does not.

3.2 Fix for CCES-15729 - System reset is not restoring some registers to their reset value for ADSP-SC5xx/ADSP-215xx

In CCES 2.3.0, the CCES Debugger was not properly restoring some registers to their reset value after a system reset. Some affected registers are MODE1, MODE2, and FLAG registers.

3.3 Fix for CCES-15522 - 2 core resets in a row corrupts core MMR registers for ADSP-SC5xx/ADSP-215xx

From CCES 2.0.0 onwards, re-starting or relaunching repeatedly with core resets, or clicking on the IDDE's Reset button repeatedly, could cause core memory mapped registers to become corrupted and show a value of 0x3be.

Starting in CCES 2.4.0, during core resets, interrupts and the system interface of the core being reset are now disabled, and the core is in IDLE mode. These changes may cause differences in behavior between 2.4.0 and previous releases of the CrossCore Debugger.

3.4 Fix for CCES-15345 - Restart ignores the 'Reset core before load' setting

Before CCES 2.4.0, when Restarting a session, reset always occurred regardless of the "Reset core before load” option in the Debug Session. Starting in CCES 2.4.0, reset will not occur during a restart if the option is not set.

3.5 Fix for CCES-15483 - Hidden SLOWLOOP window may cause CCES to appear hung when stepping, launching, or setting breakpoints

There are improvements to the window that allows users to modify the MODE2 SLOWLOOP settings for debugging F1 active loops on ADSP-SC5xx/ADSP-215xx processors.

However, when using workspaces created using CCES 2.3.0, the Target Options settings for the 'Allow fast loops when breakpoints are set' checkbox are lost and will be disabled by default. The settings need to be set properly using the 'Target', 'Settings...', 'Target Options' menu. Note the 'Target' menu can be viewed only when connected to the target.

3.6 Fix for CCES-14828 - No such file or directory linking error due to overly long command

To avoid exceeding the command-line length limit of the Windows operating system, the IDE may pass command-line objects to the linker via a file. This can change the order in which objects to be linked appear on the command-line to the linker, and may lead to differences in how the resulting executable is linked.

3.7 SHARC cfftf and ifftf functions no longer preserve values of scratch registers

The `cfftf` and `ifftf` fast N-point complex radix-2 Fast Fourier Transform SIMD functions in the SHARC libdsp library previously saved the contents of scratch registers on entry and restored these registers at the end. This was unnecessary as the C/C++ compiler does not expect a scratch register to be preserved over calls to either of these functions. These unnecessary save and restore instructions have been deleted in CCES 2.4.0.

If the `cfftf` or `ifftf` functions are being called from assembly, using symbols `_cfftf_simd` or `_ifftf_simd,` please ensure that there are no uses of data in scratch registers expected to remain after the calls.

For details regarding which registers are scratch, please see the following CCES help topic: SHARC® Development Tools Documentation > C/C++ Compiler Manual for SHARC® Processors > Compiler > C/C++ Run-Time Model and Environment > Registers.

3.8 New cache warning from ADSP-215xx/ADSP-SC5xx SHARC+ LDFs

The SHARC+ non-generated CCES default LDFs, used when *not* using the Startup code/LDF addin or a custom LDF, have been enhanced to output a warning if the caches are configured in an unsupported mode. Below is an example of the warning:

```
[Warning pp0018] "C:\analog\cces2.4.0\SHARC/ldf/ADSP-SC589.LDF":241 'Ca
Preprocessor finished with 0 error(s) 1 warning(s)
```

Avoid the warning by modifying the caches enabled to match one of the documented supported modes.

3.8.1 New cc3832 cc21k compiler error for reserved uses of seg_dmda_nw

The cc21k compiler had been enhanced to issue new error cc3832 when building SHARC+ byte-addressed source containing `#pragma section` or `#pragma default_section` uses of reserved section `seg_dmda_nw`.

3.8.2 New cc3199 cc21k and ccblkfn warning when more than one LDF is supplied on a command-line

The cc21k and ccblkfn compilers have been enhanced to issue a warning when more than one LDF is seen on a command-line. The new warning id is cc3199.

3.8.3 Changes to A5 Abstract Page Tables for SPI Flash memory (CCES-15989)

The TLB entry for SPI Flash memory (0x60000000-0x7FFFFFFF) has an updated descriptor, ADI_MMU_R0_DEVICE, to prevent issues with speculative accesses when the memory has not been configured. No change is required when using the default APT files, but a corresponding change will be required for any customized files.

3.9 Default silicon revision for ADSP-BF70x parts updated

The command-line tools for ADSP-BF70x parts when `-si-revision` switch is not used now default to build for revision 1.1 rather than 1.0.

3.10 New SPI driver size error checks

The debug versions of the SPI driver APIs have been enhanced to detect when buffer sizes are overly large (>65535 bytes) and would cause runtime failures if used. The following APIs are affected:

- `adi_spi_ReadWrite`
- `adi_spi_SetTxWordCount`
- `adi_spi_SetRxWordCount`

4 Known Issues

4.1 No Examples for ADuCM302x

The CMSIS-PACK installed in this release contains the "personality" files for the ADuCM302x family (header files, linker scripts, etc.) and source files for system services and device drivers, but does not yet contain examples for ADuCM302x processors. These examples and appropriate projects will be provided in a future release of the CMSIS-PACK.

4.2 Run-time library functions for ADuCM302x are not thread-safe

The GCC ARM Embedded toolchain's run-time library functions for ADuCM302x, like malloc are not thread-safe. If you want to use the run-time library functions from multiple threads, you should do your own locking.

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