CCES 1.0.3 Release Notes

(October 2013)

Introduction

This page describes the changes for CrossCore Embedded Studio 1.0.3. This release adds driver support for USB host mode, support for the ADSP-BF60x processor family revision 0.1, and general maintenance updates.

New Functionality

Support for ADSP-BF60x

Initial support for ADSP-BF60x family parts revision 0.1 was added in CCES 1.0.2. In CCES 1.0.3 the support has been updated. One change is that the tools workarounds for silicon anomaly 16000030 has been disabled by default when building projects for ADSP-BF60x family parts revision 0.1 using CCES 1.0.3.

Note that instruction parity errors are still disabled in the CRT startup code provided in CCES 1.0.3 for ADSP-BF60x parts to avoid silicon anomaly 16000005. As part of investigations into 16000005 a new related anomaly was discovered, 16000041. This new anomaly has been identified to occur when executing an IFLUSH instruction with instruction cache and parity enabled. The emulation software has a workaround for 16000041 beginning in the CCES 1.0.3 release. There is no documentation regarding 16000041 in the anomaly XML files in CCES 1.0.3 and there are no other workarounds other than for 16000041 supported by the CCES 1.0.3 emulator.

USB Host Mode Driver

With this release, the USB device mode driver is complemented by the addition of the USB host mode driver, enabling implementation of host mode USB Stacks such as the Micrium C/USB Host™ Stack for CrossCore® Embedded Studio 1.0.0, provided under separate license. The driver is included in the drivers' libraries for ADSP-BF526, ADSP-BF527, ADSP-BF548 and ADSP-BF609 platforms. Sources are located in the CCES 1.0.3 installation under

`Blackfin\lib\src\drivers\source\usb\controller\host.`

The library modules are built such that the faster multi-packet DMA (DMA Mode 1) is used for transferring data on the ADSP-BF526 (si-rev 0.2) and ADSP-BF609 platforms; single packet DMA (DMA Mode 0) is used for ADSP-BF527 and ADSP-BF548 platforms in line with known silicon anomalies (05000450, 0500456, 0500460, 0500465) for these parts. These defaults can be overridden by including the driver sources in your projects.

Hub support is available for ADSP-BF609 as this is the only Blackfin part with on-chip multipoint USB controller hardware. This allows multiple USB devices to be accessed concurrently by the processor. Currently, the driver has been successfully tested with 4 port hubs; failures with some 7 port hubs (ones with inbuilt secondary hub) have been identified.

Works with these hubs:

- AmazonBasics 4 port
- Belkin 4 port (CZB3372135 & CZ91166283)
- TrendNet TU 400E 3 port

Does not work with these hubs:

- Belkin 7 port
- iBall Lappie 4 port

CrossCore® Embedded Studio 1.0.2 Release Notes *(May 2013)*

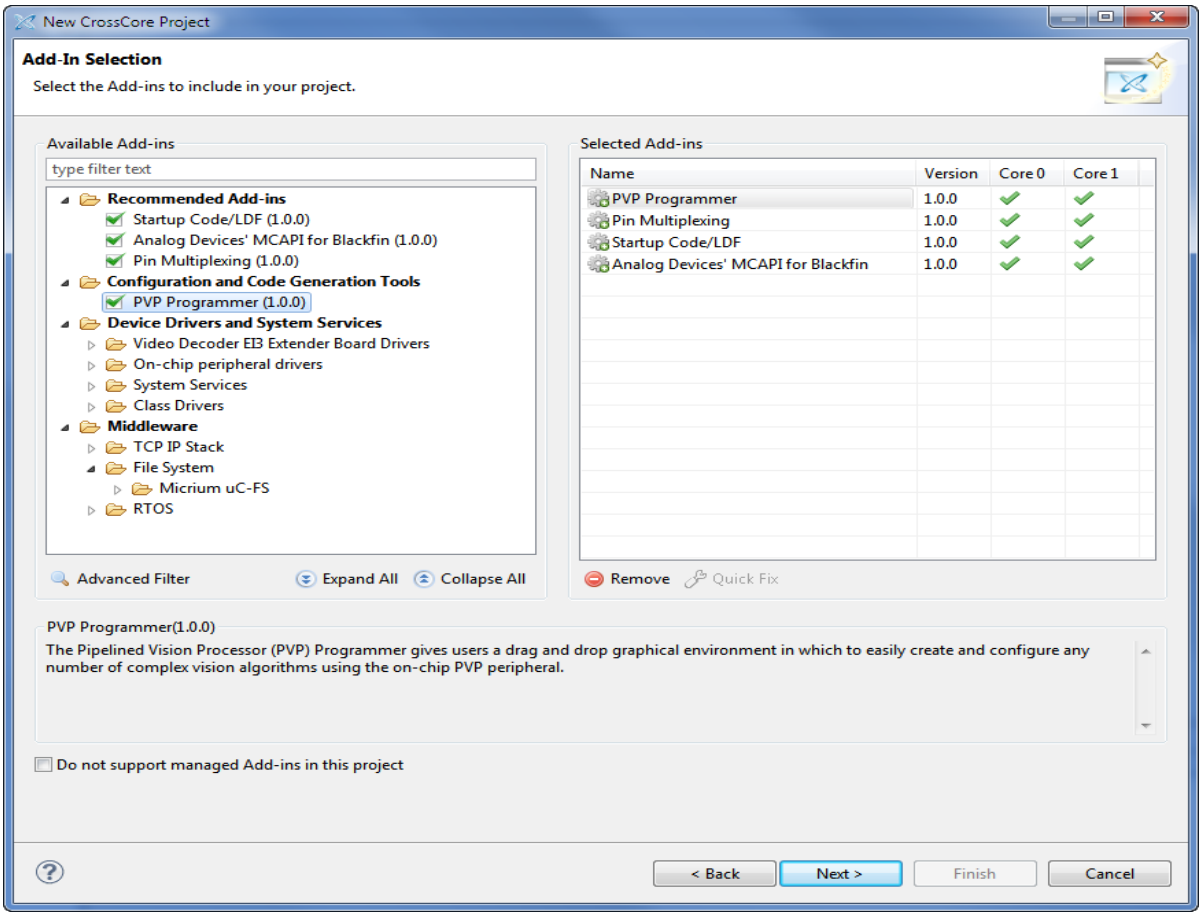
Introduction

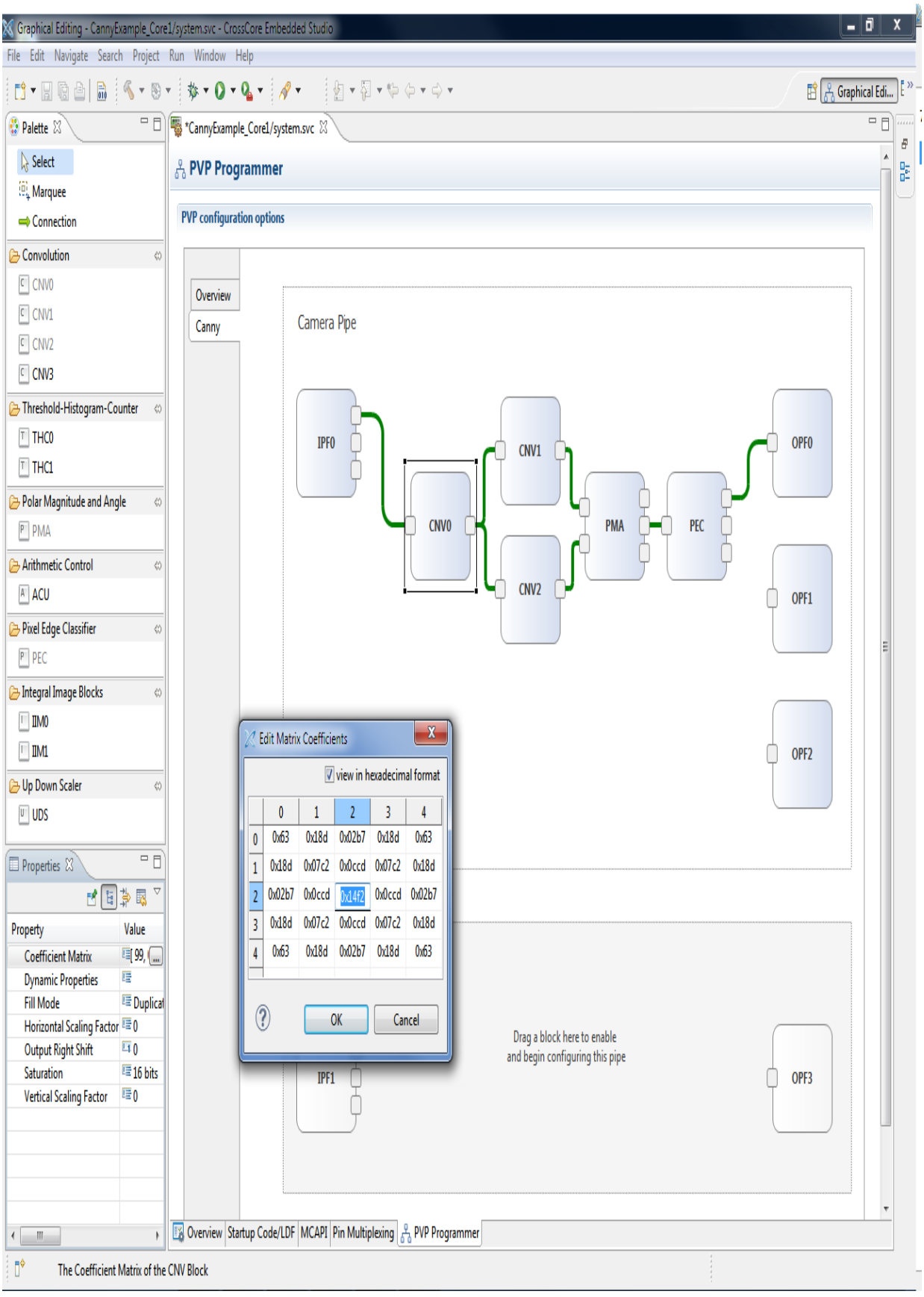
This page describes the changes for CrossCore Embedded Studio 1.0.2.

New Functionality

PVP Programmer

A new PVP Programmer Add-in has been added to this release to make configuration of the Pipelined Vision Processor found in some ADSP-BF60x parts much easier. The PVP Programmer can be added to a project targeted for supported parts by selecting PVP Programmer under Configuration and Code Generation Tools from the Add-in Selection dialog when creating a new project or from the Overview page of the system.svc file for existing projects.




Graphical Editing - CannyExample_Core1/system.svc - CrossCore Embedded Studio
File Edit Navigate Search Project Run Window Help
Palette
Select
Marquee
Connection
Convolution
CNV0
CNV1
CNV2
CNV3
Threshold-Histogram-Counter
THC0
THC1
Polar Magnitude and Angle
PMA
Arithmetic Control
ACU
Pixel Edge Classifier
PEC
Integral Image Blocks
IIM0
IIM1
Up Down Scaler
UDS
Properties
Property
Value
Coefficient Matrix
Dynamic Properties
Fill Mode
Horizontal Scaling Factor
Output Right Shift
Saturation
Vertical Scaling Factor
*CannyExample_Core1/system.svc
PVP Programmer
PVP configuration options
Overview
Canny
Camera Pipe
IPF0
CNV0
CNV1
CNV2
PMA
PEC
OPF0
OPF1
OPF2
OPF3
IPF1
Drag a block here to enable
and begin configuring this pipe
Edit Matrix Coefficients
view in hexadecimal format
0 1 2 3 4
0 0x63 0x18d 0x02b7 0x18d 0x63
1 0x18d 0x07c2 0x0ccd 0x07c2 0x18d
2 0x02b7 0x0ccd 0x14f2 0x0ccd 0x02b7
3 0x18d 0x07c2 0x0ccd 0x07c2 0x18d
4 0x63 0x18d 0x02b7 0x18d 0x63
OK
Cancel
Overview
Startup Code/LDF
MCAPI
Pin Multiplexing
PVP Programmer
The Coefficient Matrix of the CNV Block

New Compiler Switches

The SHARC compiler support the following new switches.

Switch	Platform	Description
-no-main-calls-exit	SHARC	Instructs the compiler not to plant a call to exit() at the end of main(). Normally, the compiler does this as it is more efficient, since it eliminates unused code. For dynamically-loadable modules, the return is necessary.

Pre-defined Compiler, Assembler and Linker Macro Changes

Macro	Description of change
__CCESVERSION__	is now defined to a value in the form 0xMMmmUUPP where PP is now a patch number. Previously these bits were documented as 00 and reserved for future use. Using the CCES 1.0.2.0 base kit tools the value of __CCESVERSION is defined to 0x01000200
__ADSPBF6xx__	Is defined to 1 when building for any of the ADSP-BF60x parts in C/C++, assembly and the LDF.
__ADSPBF5xx__	Is defined to 1 when building for any of the CCES 1.0.2.0 supported Blackfin parts other than ADSP-BF60x parts when it is not defined. When building for suitable parts it is defined in C/C++, assembly and the LDF.

Dynamically-loadable Modules

The dynamically-loadable module (DLM) support includes the following new functionality:

- dynreloc, a command-line utility for relocating DLMs on the host, rather than on the target.
- elf2dyn -a *sectname*=N, a switch which allows you to make the alignment constraints of DLM sections more strict than the default alignment emitted by the linker.
- elf2dyn -v, a new switch that reports the version of the elf2dyn utility.

New 214xx DDR2 macros

The following new bit position macros have been added to 21469.h:

Register	Macro	Description
DDR2PADCTL0	DATA_PWD	Data Pad Receiver Power Down
DDR2PADCTL0	DQS_PWD	DQS Pad Receiver Power Down
DDR2PADCTL0	DDR2CLK_PWD	Clock Pad Receiver Power Down
DDR2PADCTL1	ADDR_PWD	Address Pad Receiver Power Down
DDR2PADCTL1	CMD_PWD	Command Pad Receiver Power Down

New diagnostic checks in debug version of heap_install

The debug version of heap_install (linked when using the heap debugging libraries) now carries out the following additional checks:

Check	Error Type	Default Severity
Heap is insufficient size	_HEAP_ERROR_INVALID_INPUT	Error
Heap memory wraps around address space	_HEAP_ERROR_INVALID_INPUT	Error
Heap uses existing user ID	_HEAP_ERROR_INVALID_INPUT	Error
Heap uses existing start address	_HEAP_ERROR_INVALID_INPUT	Error

Support for ADSP-BF60x silicon revision 0.1

The CCES 1.0.2.0 toolchain provides support for silicon revision 0.1 of the ADSP-BF60x part family. This support includes a change to the startup code that is necessary to avoid parity error exceptions seen when running on revision 0.1 hardware. This change is incorporated automatically when the generated startup code is regenerated and is in the various pre-built default basiccrt crt.doj files. If you are using a custom startup source or have disabled regeneration of the generated files you will need to add the following BITSET instruction before accessing the ITEST_COMMAND or DTEST_COMMAND registers:

#include <sys/platform.h>
BITSET(R7, BITP_ITEST_COMMAND_PARCTL);

For example:

```
// Zero the ITEST_COMMAND and DTEST_COMMAND registers
// (in case they have unintialized values in them that
// cause a write somewhere when we enable cache).
BITSET(R7, BITP_ITEST_COMMAND_PARCTL);
I0.L = LO(ITEST_COMMAND);
I0.H = HI(ITEST_COMMAND);
I1.L = LO(DTEST_COMMAND);
I1.H = HI(DTEST_COMMAND);
[I0] = R7;
[I1] = R7;
CSYNC;
```

Note that any executable built with CCES 1.0.0.x or 1.0.1.x will need to be re-linked with an updated crt object before being used with 0.1 hardware.

New Features in the Loaders / Loader Collateral

ADSP-21371 Specific Loader Kernels

The ADSP-21371 is now supported by 371 specific kernels installed to:

- `SHARC/ldr/371_prom.dxe`
- `SHARC/ldr/371_spi.dxe`

with sources and projects available:

- `SHARC/ldr/371_prom`
- `SHARC/ldr/371_spi`

The Loader Build Artifact UI has been updated to present the 371 specific kernels as the default for the ADSP-21371.Likewise the loader has been updated to default the ADSP-21371 to a 371 kernel if it encounters a command-line that requires a default kernel.

Note that this is a change from CCES 1.0.1 which defaulted the ADSP-21371 to the 375 kernels. If you have an existing build that relied on the previous default, you may need to update your project.

The ADSP-21371 supports 32-bit external port whereas the ADSP-21375 supports 16-bit external interface. The ADSP-21375 boot kernel fails when used with a ADSP-21371 application that uses external memory.

ADSP-BF60x 0.1 Loader Collateral

There are additional pre-built DXE files / sources / projects in the Blackfin\ldr tree:

- `ADSP-BF60[6789] 0.1 and 0.0 rom_code`
- `ADSP-BF609 0.1 init_code`
- `Legacy ADSP-BFxxx projects and sources`

Byte Format for ADSP-214xx Non-Bootable Loader Files

An additional format is available in the SHARC loader when creating non-bootable loader files for the ADSP-214xx. Byte format is provided for use with the *-splitter sectname* switch. It can be set via Additional Options in the Loader Build Artifact.

See the Loader and Utilities Guide for additional information. The new switches are *-fBYTE* and *-u value*, documented in the ADSP-214xx Loader Command-Line Switches table. The Byte Format layout is documented in the Non-Bootable Loader Output Files section in the File Formats Appendix.

Services and Drivers

Two new APIs have been added

- An API to support the ADSP-BF609 Trigger Routing Unit (TRU)
- An API to support the ADSP-BF609 CRC DMA

Documentation for these APIs can be found the CrossCore® Embedded Studio 1.0.2 > System Run-Time Documentation > System Services and Device Drivers > ADSP-BF60x API Reference help section.

The USBD Device Driver now supports the ADSP-BF52x and ADSP-BF54x families in addition to the previously supported ADSP-BF60x family.

Changed Functionality

L2_sram_uncached being used for parity error handler

The default parity error handler for the ADSP-BF60x family parts is now defined in section L2_sram_uncached in order to follow the recommendation in the hardware reference that such a handler should be defined in uncached L2 memory.

If you are using a custom LDF that does not have an INPUT_SECTION for this section, you will need to add it. If you have modified your LDF to fill the memory section used by L2_sram_uncached, called MEM_L2_SRAM_UNCACHED, you may need to make 134 bytes available for the handler to avoid link failures not seen previously when using CCES 1.0.1.

Refer to the default non-generated ADSP-BF60x LDFs in <<CCES install location>>/Blackfin/ldf for an example of how this might be done. (ref. TAR-50291)

2146x / 2147x / 2148x Loader Kernels

The ADSP-2146x / 2147x/ 2148x loader kernels were updated to workaround PLL anomaly 15000020.

Symbol scrambling for ELF archives

Symbol scrambling for ELF archives using the *elfar* command's *-s* option now supports two-character scrambling keys, so as to extend the number of available keys. Additionally, the exclusion file for exempting symbols from scrambling can now contain comment lines starting with a # character.

C/C++ Library Documentation

The documentation for the third-party C/C++ libraries, provided by Dinkumware, is now included in the on-line help. In previous releases, this documentation was distributed within the Docs/cpl_lib within the Crosscore Embedded Studio installation. This documentation is provided without modification. Consequently, it may describe features that are not implemented on platforms supported by Crosscore Embedded Studio.

As part of the on-line help, the Dinkumware documentation may be returned as a result of help searches. Dinkumware documentation pages are identifiable by the copyright notice in the page footers, where copyright is assigned to Dinkumware Ltd., or to P.J. Plauger.

Removed Functionality

TBA

CrossCore® Embedded Studio 1.0.1.2 Release Notes

CrossCore® Embedded Studio version 1.0.1.2 is a patch release for CrossCore® Embedded Studio 1.0.1. It includes USB Device Controller driver fixes specific to the ADSP-BF609, ADSP-BF548, ADSP-BF527 and ADSP-BF526 family processors. The documentation for the controller driver has been updated and is included as part of this patch release.

This patch release is required by Analog Device's µC/USB Device™ Stack for CrossCore Embedded Studio version 1.0.1 product.

The CCES 1.0.1.2 patch release is a cumulative patch and includes the functionality included in the CCES 1.0.1.1 patch release.

Patch Utility Invocation

It is recommended that CrossCore® Embedded Studio is closed prior to applying the patch.

The patch utility will check for CrossCore® Embedded Studio v.1.0.1. If this version is not detected an error message will be generated and the patch utility will exit.

The patch can also be installed over CrossCore® Embedded Studio v.1.0.1.1.

The patch utility will overwrite existing files. A backup is not created by the patch utility. If a patched file has been modified and you wish to preserve the changes made you will need to create a backup of the file before applying the patch.

After the patch has been applied you will see "version 1.0.1.2" in the CrossCore® Embedded Studio splash screen each time it is started. You will also be able to see 1.0.1.2 in the CrossCore® Embedded Studio "About" page.

Uninstalling the patch will remove the patch from the installation database, but it will not remove the patched files.

There is no patch installation log generated by default. If you encounter patch installation issues, please contact technical support.

Support and Assistance

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:

processor.tools.support@analog.com

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

CrossCore® Embedded Studio 1.0.1.1 Release Notes

CrossCore® Embedded Studio version 1.0.1.1 is a patch release for CrossCore® Embedded Studio 1.0.1. It enhances the ADSP-BF609 EPPI driver such that DMA transfers between the EPPI controller and the EBIU can be configured to be 32, 64, 128 or 256 bits. In the CCES 1.0.1 release these transfers were fixed at 32 bits. This enhancement may be necessary for video applications operating at 720p.

Please note that this patch release is specific to CrossCore® Embedded Studio version 1.0.1 and will not work for any other version. This patch is intended for use with the Video Encoder EI3 Extender Board Support Package Version 1.0.1, the Video Decoder EI3 Board Support Package version 1.0.1, and the Camera EI3 Extender Board Support Package version 1.0.1.

Other Issues Addressed

Patch release 1.0.1.1 also addresses the following issues.

- The ADSP-BF609 SPI driver now allows both instances of the SPI controller to be opened. Previously only the first controller could be opened.
- The ADSP-BF609 SPORT controller driver now allows a SPORT to be opened in TX mode only . Previously opening the SPORT in TX mode only resulted in a SPORT STATUS interrupt being generated when there were no buffers to be processed.
- The ADSP-BF609 UART controller driver will no longer generate interrupts when opened in interrupt mode and there are no buffers to be processed.

Patch Utility Invocation

It is recommended that CrossCore® Embedded Studio is closed prior to applying the patch.

The patch utility will check for CrossCore® Embedded Studio v.1.0.1. If this version is not detected an error message will be generated and the patch utility will exit.

The patch utility will overwrite existing files. A backup is not created by the patch utility. If a patched file has been modified and you wish to preserve the changes made you will need to create a backup of the file before applying the patch.

After the patch has been applied you will see "version 1.0.1.1" in the CrossCore® Embedded Studio splash screen each time it is started. You will also be able to see 1.0.1.1 in the CrossCore® Embedded Studio "About" page.

Uninstalling the patch will remove the patch from the installation database, but it will not remove the patched files.

There is no patch installation log generated by default. If you encounter patch installation issues, please contact technical support.

Support and Assistance

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:

processor.tools.support@analog.com

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

CCES 1.0.1 Release Notes

Introduction

This document contains the release notes 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

Supported Processors

This release of CrossCore Embedded Studio adds support for the following processors:

- Blackfin Processors:
  - 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, ADSP-BF544, ADSP-BF544M, ADSP-BF547, ADSP-BF547M, ADSP-BF548, ADSP-BF548M, ADSP-BF549, ADSP-BF549M,
  - ADSP-BF561
  - ADSP-BF592-A
- SHARC Processors
  - 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

As with CrossCore Embedded Studio 1.0.0, this release also supports the following Blackfin processors:

- ADSP-BF606, ADSP-BF607, ADSP-BF608, ADSP-BF609

Tools Enhancements

Dynamically Loadable Modules

This release adds support for dynamically-loadable modules, through the *elf2dyn* command-line utility and the libdyn target library. For details, refer to the online help.

Building Multi-Core Loader Files

This release provides an extension to the -NoFinalTag switch in the Blackfin loader for better control in combining multiple DXE files to a single ldr file. The -NoFinalTag can now be scoped to specific DXE files. For details, refer to the online help in the ADSP-BF60x Processor Loader Guide section in the Loader and Utilities manual.

Migrating VisualDSP++ files to CrossCore Embedded Studio

This release extends the functionality provided by the elf2elf migration utility, to include migration of .OVL files (for overlays) and .SM files (for VisualDSP++ projects that used SHARED_MEMORY). elf2elf now also includes a -merge switch that can combine such additional files into the main .DXE file. For further details, refer to the online help.

Image Viewer

A new debug view known as the Image Viewer has been added to this release. This view acts much like a memory window, however it allows you to view the contents of memory as an image in any number of configurable input formats. This is especially useful when debugging imaging applications where data is being retrieved from a camera and then processed.

Eclipse 3.7.2

The version of Eclipse upon which the IDE has been built has been upgraded to version 3.7.2. More information on the changes in this release of Eclipse can be found here:

- Eclipse 3.7.2 Release Notes

Bugs and enhancements addressed in this release can be found here:

- Issues addressed in Eclipse 3.7.2 and CDT 8.0.2

Application Loading Enhancements

In this release the Launch Configuration dialog has been enhanced to allow users to load multiple dxes at the same time with the option to merge symbols, perform resets or load symbols only on each application listed. This is useful for developing applications that are spread across multiple dxe files (say one for a ROM image and another for the standard application).

Add-in Wizard Enhancements

The overall look and feel of the Add-in page in the System Configuration Editor and the New Project Wizard has been enhanced to provide better usability as well as more helpful diagnostics when add-in conflicts are detected.

Linker Support for External Memory

The CCES 1.0.1 linker has support for external memory for the SHARC processors ADSP-2136[7-9], ADSP-2137[1,5], ADSP-2146[7,9], ADSP-2147[7-9], ADSP-2148[3,6-9] as follows:

(1) Synchronous external memory (e.g., DDR2) and asynchronous external memory (e.g., flash) are distinguished by specification of the keywords SYNCHRONOUS and ASYNCHRONOUS in the TYPE specification of the LDF MEMORY statement. If neither is specified for an external memory region the linker assumes the region is SYNCHRONOUS. Note that all default LDFs supplied with CCES 1.0.1 have been modified to specify these keywords.

(2) The linker xml files in $CCESDir\System\ArchDef defining the valid external memory ranges allow specification of the synchronicity (attributes synchronous and asynchronous) and memory bank (attribute bank). Here $CCESDir denotes the CCES 1.0.1 installation directory. e.g., C:\Analog Devices\CrossCore Embedded Studio 1.0.1

(3) The linker performs logical to physical address translation before comparing external memory regions in the LDF for overlap; omission of this translation could cause overlaps to be missed or non-overlapping regions to be reported as overlapping.
The linker translates the logical address specified in the LDF to the external memory physical address as follows:

External Memory Data

All accesses to external memory data must use normal word (i.e., 32 bit) addressing. The mapping of logical address (L) to physical address P in a bank beginning at logical address B and for memory width w is:

```
f = 32/w ;
P = (bank == 0) ? f * L  :  B + ( L - B) * f ;
```

External Memory Instructions

The mapping of logical address L to physical address P for instructions (bank 0 only) is dependent on the memory width w and whether the code is VISA or ISA:

For ISA code: P = (48/w) * L
For VISA code: P = (16/w) * L;

The impact of these items is that a user-written LDF imported from VisualDSP may cause the CCES 1.0.1 linker to generate errors where none appeared before. As a temporary work-around new external memory overlap errors may be inhibited by specifying the linker option -nomema; this turns off external memory address translation.

To eliminate such errors permanently will require editing of the LDF MEMORY statement for external memory regions: the keyword SYNCHRONOUS or ASYNCHRONOUS should be added and the logical address range amended, if necessary, to be compatible with those specified in the linker xml file for the processor.

Examples usage

The supported method to find and open examples with CrossCore Embedded Studio is via the Example Browser which is included in the release. Examples should also build and run correctly if they are opened in-place with Eclipse's "Import Project" menu. Importing examples with the "Copy To Workspace" tickbox selected may result in examples which do not build and/or run as expected.

Version Compatibility

This is to provide users with information for use in updating an existing application that was developed with the previous version of the product.

Macro Changes for ADSP-BF60x Headers

This release defines macros in the ADSP-BF60x processor headers that correspond to the ADSP-BF60x Blackfin (r) Processor Hardware Reference, Preliminary Revision 0.4, May 2012.

Your code may need changes if it relied on the defBF609.h / cdefBF609.h headers in the CCES 1.0.0 release. The following are the incompatibilities to be aware of:

- **DDR macros are now prefixed as DMC**

  CCES 1.0.0 names were ones such as REG_DDR0_CFG, BITM_DDR_CFG_EXTBANK, ENUM_DDR_CFG_EXTBANK1. With DMC as the prefix, these are now REG_DMC0_CFG, BITM_DMC_CFG_EXTBANK,

ENUM_DMC_CFG_EXTBANK1. To upgrade, make the following global edits:

REG_DDR0_ to REG_DMC0_
BITP_DDR_ to BITP_DMC_
BITM_DDR_ to BITM_DMC_
ENUM_DDR_ to ENUM_DMC_
PARAM_DDR_ to PARAM_DMC_
HAS_DDR to HAS_DMC

- **The SDRSIZE4G enumeration for SDRSIZE field in register DDR_CFG is invalid**

  b#-0110 is not a valid mask for the SDRSIZE field in REG_DDR0_CFG / REG_DMC0_CFG. Code using ENUM_DDR_CFG_SDRSIZE4G / ENUM_DMC_CFG_SDRSIZE4G will no longer build because the invalid macro was removed.

- **EMAC**

  The EMAC macros were substantially revised in CCES 1.0.1 from CCES 1.0.0. Consult the defBF60[6789].h headers in the installation and the Ethernet Media Access Controller (EMAC) chapter in the ADSP-BF60x Blackfin® Processor Hardware Reference on www.analog.com.

- **USB**

  The USB macros were substantially revised in CCES 1.0.1 from CCES 1.0.0. Consult the defBF60[6789].h headers in the installation and the Universal Serial Bus (USB) chapter in the ADSP-BF60x Blackfin® ProcessorHardware Reference on www.analog.com.

- **CAN_INT register in CAN module has bit field name changes**

CCES 1.0.1 (new names)	CCES 1.0.0
BITP_CAN_INT_CANRX	BITP_CAN_INT_CANTRX
BITM_CAN_INT_CANRX	BITM_CAN_INT_CANTRX
BITP_CAN_INT_MBRIRQ	BITP_CAN_INT_MBIRQ
BITM_CAN_INT_MBRIRQ	BITM_CAN_INT_MBIRQ

Other Known Issues

Nothing to report. For the latest anomalies, please consult our Software and Tools Anomaly page (http://www.analog.com/software-anomalies.html). This page will be available in October 2012.

CrossCore® Embedded Studio 1.0.0.1 Release Notes

CrossCore® Embedded Studio version 1.0.0.1 is a patch release for CrossCore® Embedded Studio 1.0.0. It addresses ADSP-BF60x silicon anomaly 16-00-0030 where if a core MMR register read instruction is interrupted and the core MMR register has read side effects, data, such as status flags or FIFO values, could be lost. Shutting off interrupts before these core MMR reads will prevent the anomaly from occuring. This release patches the services and drivers that perform these core MMR reads.

Please note that this patch release is specific to CrossCore® Embedded Studio version 1.0.0 and will not work for any other version.

Other Issues Addressed

Patch release 1.0.0.1 also addresses the following issues.

TAR-48756: Memory required by the SPORT and Linkport driver is insufficient in RTOS environment.

> Most device drivers require the application to pass a buffer to the driver when the driver is opened. The size of the buffer is specified in the driver header file. In the case of the SPORT and LinkPort drivers the specified size in the CCES 1.0.0 release is too small. This patch release includes a fix for this issue.

TAR-49037: Line buffering needs to be enabled when redirecting STDIN via the STDIO service

> The STDIO service has been patched such that the STDIN is now placed into line buffering mode when STDIN is redirected to USB.

The SPI driver in CCES 1.0.0 incorrectly loads the Tx and Rx Word Count registers when configured for 16/32 bit transfers. When loading these two registers with 8 bit transfers, the SPI functions correctly. This patch fixes the issue when 16/32bit transfers are used.

Patch Utility Invocation

It is recommended that CrossCore® Embedded Studio is closed prior to applying the patch.

The patch utility will check for CrossCore® Embedded Studio v.1.0.0. If this version is not detected an error message will be generated and the patch utility will exit.

The patch utility will overwrite existing files. A backup is not created by the patch utility. If a patched file has been modified and you wish to preserve the changes made you will need to create a backup of the file before applying the patch.

After the patch has been applied you will see "version 1.0.0.1" in the CrossCore® Embedded Studio splash screen each time it is started. You will also be able to see 1.0.0.1 in the CrossCore® Embedded Studio "About" page.

Uninstalling the patch will remove the patch from the installation database, but it will not remove the patched files.

There is no patch installation log generated by default. If you encounter patch installation issues, please contact technical support.

List of patched files

The following file is new and will be added

%CCES_INSTALL%\blackfin\include\sys\16000030.h

The following files will be modified

%CCES_INSTALL%\blackfin\include\sys\anomaly_macros_rtl.h

%CCES_INSTALL%\blackfin\include\drivers\linkport\adi_linkport_bf6xx.h
%CCES_INSTALL%\blackfin\include\drivers\sport\adi_sport_bf6xx.h

%CCES_INSTALL%\blackfin\src\drivers\uart\adi_uart_bf60x.c
%CCES_INSTALL%\blackfin\src\drivers\twi\adi_twi.c
%CCES_INSTALL%\blackfin\src\drivers\linkport\adi_linkport_bf6xx.c
%CCES_INSTALL%\blackfin\src\drivers\sport\adi_sport_bf6xx.c
%CCES_INSTALL%\blackfin\src\drivers\spi\adi_spi_bf6xx.c
%CCES_INSTALL%\blackfin\src\drivers\spi\adi_spi_data_bf6xx.c

%CCES_INSTALL%\blackfin\src\services\stdio\adi_stdio.c

%CCES_INSTALL%\blackfin\include\drivers\usb\controller\device\musbmhdrc\adi_usbd_dev_musbmhdrc.h
%CCES_INSTALL%\blackfin\src\drivers\usb\controller\device\musbmhdrc\adi_usbd_dev_musbmhdrc.c
%CCES_INSTALL%\blackfin\src\drivers\usb\controller\device\musbmhdrc\adi_usbd_dev_musbmhdrc_intrpt.c
%CCES_INSTALL%\blackfin\src\drivers\usb\controller\device\musbmhdrc\adi_usbd_dev_musbmhdrc_local.h
%CCES_INSTALL%\blackfin\src\drivers\usb\controller\device\musbmhdrc\adi_usbd_dev_musbmhdrc_state.c

%CCES_INSTALL%\blackfin\lib\bf609_rev_any\debug\libdrv.dlb
%CCES_INSTALL%\blackfin\lib\bf609_rev_any\debug\libssl.dlb
%CCES_INSTALL%\blackfin\lib\bf609_rev_any\libdrv.dlb
%CCES_INSTALL%\blackfin\lib\bf609_rev_any\libssl.dlb
%CCES_INSTALL%\blackfin\lib\bf609_rev_none\debug\libdrv.dlb
%CCES_INSTALL%\blackfin\lib\bf609_rev_none\debug\libssl.dlb
%CCES_INSTALL%\blackfin\lib\bf609_rev_none\libdrv.dlb
%CCES_INSTALL%\blackfin\lib\bf609_rev_none\libssl.dlb

Support and Assistance

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:

processor.tools.support@analog.com

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

CrossCore® Embedded Studio Release Notes, March 29, 2012 (1.0.0)

Preface

Thank you for purchasing CrossCore® Embedded Studio for Analog Devices Processors.

The first release of our latest development environment, CrossCore Embedded Studio 1.0.0 is designed to make developing software applications for Analog Devices processors even easier.

CrossCore Embedded Studio represents a major step forward in embedded system development. It combines an industry-leading integrated development environment (IDE) with Analog Devices' advanced optimizing compiler technology. It also supports standards such as ISO/IEC C and C++, MISRA-C, Embedded C and MCAPI™, plus additional features to get your products to market even sooner.

Purpose of This Document

This document briefly introduces features of CrossCore Embedded Studio 1.0.0, which supports the ADSP-BF60x family of Analog Devices Blackfin processors.

Details of this product and other products in the CrossCore family are found in the "Related Documents" section of these release notes or in the online Help that is accessible from within CrossCore Embedded Studio.

Intended Audience

This publication is primarily intended for programmers looking for a short overview of CrossCore Embedded Studio. For additional introductory information, you can view the video tutorials that are available through the "Welcome Page" that appears when you first start using CrossCore Embedded Studio, or at:

http://videos.analog.com/category/products/processors-dsp/ .

Technical or Customer Support

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. To do this: 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 processor 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: http://ez.analog.com/community/dsp

Platform and Processor Support

This release of CrossCore Embedded Studio supports the ADSP-BF60x family, which are dual-core Blackfin processors. The following processors are supported:

ADSP-BF606

ADSP-BF607

ADSP-BF608

ADSP-BF609

Support for SHARC processors, and other processors in the Blackfin family will be available in a future update to CrossCore Embedded Studio.

Related Products and Documentation

CrossCore Embedded Studio can work in concert with other software add-ins and hardware as part of a comprehensive software development solution. All of these products install easily into the CrossCore Embedded Studio environment and are available through the Analog Devices web site for CrossCore Embedded Studio, www.analog.com/cces. Associated documentation can also be accessed there. This section identifies these products.

Software Add-Ins:

1. Real Time Kernel

2. USB Stack (Device)

3. File System

4. lwIP Lightweight TCP/IP Stack

Development Hardware:

1. ADSP-BF609 Evaluation Hardware for the ADSP-BF60x Blackfin Family of Processors

2. WVGA/LCD EI3 Extender Board

3. Video Decoder & Video Encoder EI3 Extender Boards

4. Audio EI3 Extender Board

5. Camera EI3 Extender Board

6. HPUSB and USB Emulators

7. ICE-100B Emulator

For target processor information, refer to your processor's hardware reference manual, programming reference, or data sheet. All documentation is available online and also directly from within the IDE without needing an external URL.

Online Technical Documentation

Full documentation for CCES and any optionally installed add-ins is available from your Windows start menu by clicking Analog Devices -> CrossCore Embedded Studio 1.0.0 -> CrossCore Embedded Studio Help or by clicking the Help -> Help Contents menu directly from within the IDE. This documentation includes the following manuals:

- Graphical Development Environment
- System Runtime Documentation
- Licensing Guide
- Assembler and Preprocessor Manual
- C/C++ Compiler and Library Manual for Blackfin® Processors
- Linker and Utilities Manual
- Loader and Utilities Manual
- Development Hardware Documentation
- ADSP-BF6xx Blackfin Processor Hardware Reference
- ADSP-BF5xx/BF60x Blackfin Processor Programming Reference

Additional documentation for Analog Devices processor hardware can be downloaded directly from the IDE by clicking Help -> Install New Software… and selecting “CrossCore Embedded Studio Software and Documentation” in the “work with” field of the Install dialog.

Introduction

This chapter describes CrossCore Embedded Studio, the requirements for running version 1.0.0 and some of the benefits provided by this release.

Product Release Description

CrossCore Embedded Studio integrates an Eclipse-based IDE with the latest versions of our mature code generation and debugging tools, enabling programmers to move easily between editing, debugging, and deployment of final products.

Release 1.0.0 includes the code generation tool chain comprised of the processor-specific software necessary for completing a project: assembler, C/C++ compiler and libraries, linker, loader, splitter, and utilities.

Release 1.0.0 System Requirements

To install and run Release 1.0.0, your computer must provide the following software, configuration, and system resources:

- 2 GHz single core Intel Pentium 32-bit processor (or x86 compatible); A 3.3 GHz or faster dual core machine is recommended.
- Windows XP Professional SP3 (32-bit only), Windows Vista Business Enterprise/Ultimate SP2 (32-bit only), Windows 7 Professional/Enterprise/Ultimate (32 and 64-bit)
- At least 1 GB of internal memory; 4 GB or more is recommended
- At least 2 GB of available disk space

Windows Vista and Windows 7 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. The default installation location is “C:/Analog Devices/CrossCore Embedded Studio 1.0.0”.

Getting Started with CrossCore Embedded Studio

The first time you start CrossCore Embedded Studio you will see the Welcome Screen which has links to learning resources to help you use this product effectively. You can return to the Welcome Screen anytime by selecting Welcome from the Help menu. The Welcome Screen has links to the following areas:

Getting Started Video

The Getting Started video will introduce you to the basics of this product.
It’s recommended that new users begin by watching this informative presentation.

Working with Projects

The *Working with Projects* section gives you three easy ways to begin working with

CrossCore Embedded Studio projects.

- Selecting “Create a new project” will open a dialog that guides users through the new project creation process, along with options for creating a customized Linker Description File and startup code.

- Selecting “Import an existing CCES project” will import an existing CrossCore Embedded Studio project into your workspace.

- Selecting the Example Browser will enable you to search all of the examples that have been installed on your system. You can search by any number of criteria including processor, processor family, platform, language, and keyword. Many of the related products in the CrossCore family also come with examples. When these products are installed, their examples will also be available through the Example Browser.

Viewing Documentation

Under the *Viewing Documentation* heading you will find a link to open the online help facility and a link to find and download additional processor documentation.

Getting Help

Finally, the Welcome Page includes links to get additional help on CrossCore Embedded Studio.

Processors and DSP Community on EngineerZone: http://ez.analog.com/community/dsp. Here you can search the FAQs, blogs, and forums for more information. You can also use the forum to ask questions to the community.

Private Technical Support: http://www.analog.com/support. Submit a private support request to our technical support team.

Online Training Modules

One of the best ways to become familiar with CrossCore Embedded Studio is to visit the online training site at:

http://videos.analog.com/category/products/processors-dsp/.

Here are some of the available training modules:

- CrossCore Embedded Studio Introduction and Overview
- Navigating Through the Eclipse Based IDE
- Creating, Configuring, and Building Projects
- Debugging on a Hardware Target
- Creating and Debugging a Boot Image
- System Services and Device Drivers in CrossCore Embedded Studio
- Transitioning from VDK to a new RTOS
- An introduction to the ADSP-BF609 Blackfin Processor

License Activation

The New License Wizard starts automatically the first time you run CrossCore Embedded Studio or thereafter if no valid licenses are detected. When the wizard appears, click Yes to start the New License Wizard. If you don't use the wizard when you first begin CrossCore Embedded Studio you can start the wizard later by choosing Help->Manage Licenses->New and the New License Wizard will appear.

On the first page of the wizard you will be asked whether you would like a free 90 day evaluation of the product or if you have a serial number that you would like to register. If you choose to enter a serial number then you will be asked to enter it at this time. Enter the number exactly as it appears, including dashes.

The next page will ask you to select a license activation method. If you have internet access the "one-step" activation is recommended. The one-step activation will install, register, and validate your license and you will be ready to begin using CrossCore Embedded Studio.

The alternative method of activation will install a temporary license so you can continue to use CrossCore Embedded Studio. You will need to register and activate your permanent license through the Analog Devices website.

Full information on CrossCore Embedded Studio licensing can be found by choosing

Help->Help Contents->CrossCore Embedded Studio 1.0.0->Licensing Guide.

Notable Features

IDE

The CrossCore Embedded Studio IDE is based on the industry standard Eclipse environment. Eclipse features a best-in-class smart editing environment and a language-aware editor. The editor provides code completion of language constructs and source browsing of names.

The project environment features different perspectives, making it easy for users to switch back and forth between views pertinent to specific development tasks. One perspective can be used for editing and building your project while another can provide the visibility necessary for debugging your program.

In addition to the Eclipse features, Analog Devices has enhanced the environment to provide project customization. The build options can be easily changed to take best advantage of the code generation tools including the Optimizing C/C++ Compiler.

User Interfaces for Source Generation

The Startup Code/LDF add-in makes it easy to create and configure a DSP Project. It helps you create a new project with startup code that sets up peripherals like cache, DMA, and I/O, etc.

In additional to startup code, a linker description file (LDF) can be generated and added to the project to link in only necessary libraries, and specify SDRAM partitioning, etc. This takes away much of the complexity of configuring your application's memory layout.

The Startup Code/LDF add-in automatically adds the generated files to your project, and updates them to reflect changes as you configure your project's settings, while preserving your custom modifications to the generated files.

Analysis and Diagnostic Support

CrossCore Embedded Studio provides tools to more quickly identify common code errors at runtime and provide you with the information necessary to efficiently optimize your program.

Reports for Instrumented Profiling, Heap Debugging, and Code Coverage

CrossCore Embedded Studio provides support for generating various profiling and tracing reports in HTML that can be launched from within the IDE.

The compiler has several profiling and tracing facilities. These include:

- Heap debugging (.hpl files): To identify memory leaks, corruption and other problems relating to malloc and free.
- Instrumented profiling (.prf files): To identify the functions that consume the most cycles.
- Code coverage (.pgo files): To identify which parts of your application have been exercised by your testing.

Each of these facilities generates data files during application execution, containing profile or trace data. These files can be converted into HTML-formatted reports through the IDE.

To generate an HTML report, open the File menu, and select the New...>Code Analysis Report option. There is a choice for each of the report types.

For further details go to Help->Help Contents->CrossCore Embedded Studio 1.0.0->C/C++ Compiler and Library Manual for Blackfin Processors, Chapter 2, “Analyzing Your Application” section.

Stack Overflow Detection

The Blackfin compiler supports stack overflow detection in CrossCore Embedded Studio

1.0.0. This facility is enabled through the -rtcheck-stack compiler switch, and instruments the generated code to check the stack pointer against the end of the stack. If an overflow is detected, the application jumps to adi_stack_overflowed, where a breakpoint is placed automatically by the IDE. For more information go to Help->Help Contents->CrossCore Embedded Studio 1.0.0->C/C++ Compiler and Library Manual for Blackfin Processors, Chapter 2, “Stack Overflow Detection” section.

Fatal Error Diagnostic Information

In CrossCore Embedded Studio 1.0.0, the run-time libraries use a common, extensible API for reporting fatal application errors. When the API is invoked, the application transfers control to the adi_fatal_error function. The IDE automatically places a breakpoint on this label when your application is loaded, so the application halts. When this occurs, the IDE will retrieve diagnostic information passed to the API by the run- time library, and will display it in the console window.

Inter-core Communication

CrossCore Embedded Studio 1.0.0 includes support for the Multicore Communications

API (MCAPI™), version 2.0, defined by The Multicore Association.

MCAPI allows communication between cores using

- unconnected messages
- connected packet streams
- connected scalar streams.

Communication within a core is also supported.

MCAPI is available on all Blackfin processors, but is most useful with dual-core processors such as ADSP-BF609. MCAPI is enabled by default when creating new projects for ADSP-BF609. MCAPI configuration can be added to your project via the System Configuration utility.

To see the MCAPI 2.0 Specification document, open CrossCore Embedded Studio Help: Help -> Help Contents->CrossCore Embedded Studio 1.0.0->System Runtime Documentation->Multicore Communication API (MCAPI) Specification.

The MCAPI examples can be found here: Help -> Browse Examples and search on "MCAPI".

Peripheral Controller Driver Model

The driver model deployed in CrossCore Embedded Studio was designed with the following guidelines in mind:

- Ease of use
- Minimal footprint
- Minimal latency

For further information go to Help->Help Contents->CrossCore Embedded Studio 1.0.0->System Runtime->System Services and device drivers

For examples and/or "Code Sketches" using the driver model go to Help->Browse Examples and search on the appropriate keywords (e.g. "audio", "video", "twi", etc.) (Code Sketches are interactive code examples where input values can be modified to show their effect on the code in real time.)

Standardized Interrupt Management

To make the programming of Analog Devices processors consistent, the same APIs and methodology will be available for Blackfin and SHARC processors and supported operating systems in future updates. These interrupt management APIs begin with the adi_int prefix.

The types of interrupts that can be managed with these APIs are:

SHARC processors:

- All core interrupts

Blackfin processors:

- All core interrupt levels
- All system interrupts
- Exceptions
- Non-maskable interrupt (NMI)

The same interrupt APIs and methodology are also available for the following operating environments:

- Bare-metal (no operating system). Support for this configuration is shipped with CrossCore Embedded Studio
- Real time operating system. Support for this configuration is shipped in the real time kernel add-in available for CrossCore Embedded Studio.

This interrupt management mechanism is not only available for use within applications, but it is also the mechanism used by CrossCore Embedded Studio's system services and device drivers.

For further information go to Help->Help Contents->CrossCore Embedded Studio 1.0.0-

>System Runtime Documentation->Interrupt Support.

Compiler Language Standards Support

Embedded C Support

The Blackfin compiler in CrossCore Embedded Studio 1.0.0 provides support for the native fixed-point types *fract* and *accum*, defined in Chapter 4 of the "Extensions to support embedded processors" ISO/IEC draft technical report TR 18037. These native fixed-point types allow you to write your applications in a more natural manner, without sacrificing performance.

For instance, the following function is an example of a dot product implemented using *fract* and *accum*, with natural fractional, saturating multiplication, addition, and assignment operators instead of built-in functions.

```
#include <stdfix.h>
accum dot_product(fract *a, fract *b, int n)
{
accum sum = 0.0k;
int i;
for (i = 0; i < n; i++)
sum += a[i] * b[i];
return sum;
}
```

For more information go to Help->Help Contents->CrossCore Embedded Studio 1.0.0->C/C++ Compiler and Library Manual for Blackfin Processors, Chapter 1 "Using Native Fixed-Point Types".

C99 and C++2003 Support

The compiler conforms to the ISO/IEC language standards:

- C99 is the default C language accepted by the compiler. This is a freestanding implementation of the ISO/IEC 9899:1999 C language standard.
- C89 mode is available through the -c89 compiler switch. In this mode, the compiler supports a hosted implementation of the ISO/IEC 9899:1990 C

language standard.

- C++2003 is the default C++ language accepted by the compiler. This is a hosted implementation of the ISO/IEC 14882:2003 C++ language standard.

In all these modes, the compiler accepts Analog Devices' language extensions. Some language features are standard features in some modes, and extensions in others; for further information go to Help-> Help Contents->CrossCore Embedded Studio 1.0.0-

>C/C++ Compiler and Library Manual for Blackfin Processors, and refer to the feature descriptions for details on standards conformance.

Full C++ Standard Library

The Blackfin compiler supports both the abridged and full C++ standard libraries. For more details go to Help->Help Contents->CrossCore Embedded Studio 1.0.0->C/C++ Compiler and Library Manual for Blackfin Processors for documentation on the -full- cpplib switch.

MISRA-C:2004 Support

The compiler provides comprehensive support for MISRA-C: 2004, a set of guidelines published by the Motor Industry Software Reliability Association (MISRA). The compiler detects violations at compile-time, at link-time and at run-time.

For details go to Help->Help Contents->CrossCore Embedded Studio 1.0.0->C/C++ Compiler and Library Manual for Blackfin Processors and find the section "MISRA-C Compiler "in Chapter 1 "Compiler".

Device Programmer

The Device Programmer is a utility for programming device memory, such as parallel or serial flash devices, on a target board. The Device Programmer is invoked in a command window as cldp, which is installed in the root folder of CrossCore Embedded Studio. The device programmer can also be added as a post build step in CrossCore

Embedded Studio to automatically program device memory after creating a Loader File artifact.

The Device Programmer interfaces with the target board using a device programmer interface application (dpia) that is installed separately from CrossCore Embedded Studio as part of the EZ-Board/EZ-Kit board support installation. If not interfacing to an Analog Devices, Inc. EZ-Board/EZ-Kit, then users must create their own dpia.

For more information on how to use the Device Programmer or creating a custom dpia please go Help->Help Contents->CrossCore Embedded Studio 1.0.0->Graphical Development Environment >Device Programmer.

Known Limitations

Supported Processors

This release of CrossCore Embedded Studio supports the ADSP-BF60x family of Blackfin processors. Support for SHARC processors, and other processors in the Blackfin family will be available in a future update to CrossCore Embedded Studio.

Note that the documentation for CrossCore Embedded Studio already reflects our plans to include these other processors.

Note that many components of the toolchain for other processors are included in this release; however, they should not be considered production-quality.

Simulator

There is no simulator for the ADSP-BF60x family of processors. Consequently, to run programs for this family you will need the ADSP-BF609 EZ Kit Evaluation Hardware and its associated board support package or your own board with a BF60x processor and board support software.

Image Viewer

Support for the Image Viewer will be available in a future update.

Pipeline Viewer

There is no support for the Pipeline Viewer in this release.

ADSP-BF60x Loader

The CrossCore Embedded Studio loader does not provide a switch for creating forward blocks for the ADSP-BF609 where an entire boot stream can be contained in the payload of a block, and that entire stream can be forwarded to a peripheral.

Documentation

The following documentation is not up-to-date in this release. Instead the included documentation represents the toolchain and libraries at the time of the CrossCore Embedded Studio Beta Release.

- C/C++ Compiler for SHARC Processors.
- C/C++ Library Manual for SHARC Processors.

Anomalies

The following table is a list of known anomalies in CrossCore Embedded Studio 1.0.0.

Tar	Summary	Release Note
TAR-48052	part function names in call stack for C++ functions	When debugging an application, the displayed name for C++ functions in the call stack will not give all available information about the functions. Information about namespace scope, templates, etc. will be missing. For example, the function "my_test::array<short, long>::array(int)" will be displayed as "array(int)". This issue may make it difficult to navigate sources that use these standard C++ features
TAR-47733	Symbol manager should reset breakpoints when memory initialization is complete	If code that has breakpoints set in it is initialized using the runtime initialization support, the instruction at the breakpoints will be corrupted and the program will fail to execute properly. To avoid the problem, do one of the following: 1) Disable run-time initialization support when you need to rely on the use of breakpoints (for example, when debugging) 2) Ensure that code containing breakpoints is not initialized at runtime. This can be done in one of two ways: - by adding the following line to the the appropriate source file: #pragma file_attr("requiredForROMBoot") - By adding the function name (with a prefixed underscore) to the list of functions defined in the LDF as "OBJS_LIBS_WITH_BREAKPOINTS"
TAR-48245	cldp crashes with commands file that does not have a new line at the end	When using a command file(-@ filename), the command line device programmer will not work if the file does not have a new line at the end.
TAR-48048	Duplicate entries for HPUSB emulator seen during Found New Hardware Wizard	The USB device driver for the Analog Devices, Inc. line of emulators inadvertently did not receive an update of its version. Therefore, there may be times when installing the driver where the user may be asked to pick between device drivers with the same exact version. It is safe to choose the oem*.inf file where * is a higher number.
TAR-48373	Memory Browser loses memory tabs on relaunch	If you create a new memory tab in the Memory Browser view and then re-launch your debug configuration using the "Relaunch" menu, it loses the memory tab that you just created. To reproduce: 1. Launch any debug configuration 2. Open the Memory Browser view

		3. Type 0 for the address and click Go 4. In the Debug view, right-click on the top level node and choose "Relaunch" Workaround: Terminate your debug configuration first and then re-launch it.
TAR-48160	Warnings and errors when the project name contain "-"	Projects with a dash (-) in their name may fail to build To reproduce: 1. Install CCES, and create a project named TAR-48041, other things set as default 2. Build the new created projects, there are warnings 3. Open the project Properties setting page, Enable MISRA-C 4. Rebuild the project, there is a error indicate the name should not include "-" Workaround: Do not use a dash in the name of your project.
TAR-47906	Project paths that contain an ampersand (&) will not build	If a project path has the ampersand character in it then it fails to build. To reproduce: * Create a new project where the path to the project has a ampersand in it. * Build the project to see the error. Workaround: Do not use ampersands in the path or project name.
TAR-48189	String index out of range in Debug As, Debug Configuration menu	When there is more than one project open and you select a dxe to run or debug, you may see a pop- up about 'String index is out of range' and cannot load the dxe. Workaround: To load the dxe, select to run or debug from the project level instead of selecting the individual dxe.
TAR-47759	"Invalid format: STRING.Format" errors when debugging	When debugging an executable you may occasionally see a number of "Invalid format: STRING.Format" errors in the output console and no application output. Workaround: None. This error can be safely ignored.
TAR-48176	Not resolved errors in system/uCLIB/source lib files	When using the wizard to create a project that has uC/OS-III and then clicking on the system/uCLIB/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 . To reproduce: * File, New, CrossCore Project. Select 609 and silicon rev any. Next. * Deselect MCAPI, Startup/LDF and pin muxing. Select RTOS. * Everything else default. * In Project Explorer, click on any of the files in system/uC-LIB/Source. Workaround: None
TAR-48254	Boot fails when Start Address (-p) with Initialization File (- init) due to incorrect NEXT PTR argument in initialization FIRST block	Do not use "Start address (-p <alternate- address>)" when building a ldr file with "Boot format Intel HEX (-f hex)" and "Initialization file (-init <init.dxe>)" at this release. Either: 1) Build the ldr file with "Boot format ASCII (-f ASCII)" with the default start address and specify the offset when programming the memory using the Device Programmer: cldp -offset <alternate-address> ... or 2) Build the ldr file with "Boot format HEX (-f hex)" and add "-kp <alternate-address>" in Additional Options instead of "Start address (-p <alternate-address>)".
TAR-44454	String comparison functions fail on signed values	String comparison functions (strcmp, strncmp and memcp) can return the wrong result if the parameter strings contain non-ASCII characters.
TAR-48374	BF60x def headers have incorrect casts for some registers in the EMAC module	In the "cdef" header files for BF60x processors (which contain C register and bitfield definitions), a number of macros for memory-mapped registers in the EMAC module are incorrect.The macros should use (volatile uint32_t *) instead of (void * volatile *). The list of incorrect macros is given below, followed by correct definitions: pREG_EMAC0_DMA_RXDSC_ADDR pREG_EMAC0_DMA_TXDSC_ADDR pREG_EMAC0_DMA_TXDSC_CUR pREG_EMAC0_DMA_RXDSC_CUR pREG_EMAC0_DMA_TXBUF_CUR pREG_EMAC0_DMA_RXBUF_CUR pREG_EMAC1_DMA_RXDSC_ADDR pREG_EMAC1_DMA_TXDSC_ADDR pREG_EMAC1_DMA_TXDSC_CUR pREG_EMAC1_DMA_RXDSC_CUR pREG_EMAC1_DMA_TXBUF_CUR pREG_EMAC1_DMA_RXBUF_CUR

		Correct definitions: #define pREG_EMAC0_DMA_RXDSC_ADDR ((volatile uint32_t*)REG_EMAC0_DMA_RXDSC_ADDR) /* EMAC0RX Descriptor List Address */ #define pREG_EMAC0_DMA_TXDSC_ADDR ((volatile uint32_t*)REG_EMAC0_DMA_TXDSC_ADDR) /* EMAC0TX Descriptor List Address */ #define pREG_EMAC0_DMA_TXDSC_CUR ((volatile uint32_t*)REG_EMAC0_DMA_TXDSC_CUR) /* EMAC0TX current descriptor register */ #define pREG_EMAC0_DMA_RXDSC_CUR ((volatile uint32_t*)REG_EMAC0_DMA_RXDSC_CUR) /* EMAC0RX current descriptor register */ #define pREG_EMAC0_DMA_TXBUF_CUR ((volatile uint32_t*)REG_EMAC0_DMA_TXBUF_CUR) /* EMAC0TX current buffer pointer register */ #define pREG_EMAC0_DMA_RXBUF_CUR ((volatile uint32_t*)REG_EMAC0_DMA_RXBUF_CUR) /* EMAC0RX current buffer pointer register */ #define pREG_EMAC1_DMA_RXDSC_ADDR ((volatile uint32_t*)REG_EMAC1_DMA_RXDSC_ADDR) /* EMAC1RX Descriptor List Address */ #define pREG_EMAC1_DMA_TXDSC_ADDR ((volatile uint32_t*)REG_EMAC1_DMA_TXDSC_ADDR) / * EMAC1TX Descriptor List Address */ #define pREG_EMAC1_DMA_TXDSC_CUR ((volatile uint32_t*)REG_EMAC1_DMA_TXDSC_CUR) /* EMAC1TX current descriptor register */ #define pREG_EMAC1_DMA_RXDSC_CUR ((volatile uint32_t*)REG_EMAC1_DMA_RXDSC_CUR)

		/* EMAC1RX current descriptor register */ #define pREG_EMAC1_DMA_TXBUF_CUR ((volatile uint32_t*)REG_EMAC1_DMA_TXBUF_CUR) /* EMAC1TX current buffer pointer register */ #define pREG_EMAC1_DMA_RXBUF_CUR ((volatile uint32_t*)REG_EMAC1_DMA_RXBUF_CUR) /* EMAC1RX current buffer pointer register */
TAR-48372	___cplb_ctrl is used as part of the instruction/parity workaround but may not be initialized before use	The ___cplb_ctrl variable is checked during the startup code sequence to see if instruction caching is enabled; if it is, instruction parity is disabled to avoid anomaly 16000005 ("Using L1 Instruction Cache with Parity Enabled is Unreliable"). If runtime initialization support is enabled (i.e. the "- mem" switch is used or "Runtime initialization" is selected in the linker project options), __ _cplb_ctrl will be checked before it is initialized by the runtime initialization sequence, and will contain a random value. This can mean that parity and instruction caching get enabled together and the anomaly will be hit. To avoid this issue, do one of the following: - do not use runtime initialization. - disable instruction caching.. - disable parity support by defining __ parity_ctrl to zero.
TAR-48443	USB Controller Driver header file MISRA errors	The USB controller driver is prebuilt and is included in the driver library file libdrv.dlb. If you are rebuilding this driver library from source and you are building with the -misra-strict compiler option you will encounter some MISRA-C warnings and errors.
TAR-48549	Watchdog services are not included in libssl	The ADSP-BF609 Watch Dog Timer (WDT) service sources were not built into the System Services library Blackfin\lib\bf609_rev_any\libssl.dlb Blackfin\lib\bf609_rev_none\libssl.dlb To use the WDT you will need to include its source file into your project. Blackfin\lib\src\services\source\wd\adi_wd.c The header file to include to include in your project is #include <services\wd\adi_wd.h> Please note that the documentation for the WDT is not available in

		help system. However, Blackfin\lib\src\services\source\wd\adi_wd.c contains the documenation for the WDT APIs. As of this release you will need to browse Blackfin\lib\src\services\source\wd\adi_wd.c for API documentation.
TAR-48092	The Power Service is unable to put the processor into deep sleep	The Power Service has an adi_pwr_SetPowerMode() API which is used to set the processor dynamic power management operating mode. This API is currently capable of setting the operating mode to Full On, Active, Active PLL Disabled or Sleep. The Deep Sleep and Hibernate modes are not currently supported by this API. The adi_pwr_SetPowerMode() API doesn't return an error code and the mode is not changed if attempting to set the mode to Deep Sleep or Hibernate.
TAR-48392	Disabling PPIRx broadcast in the Video Subsystem driver clobbers VSS connection register contents	If the Video Subsystem API adi_vss_EnablePPIRxBcast is called to disabled the EPPI receive broadcast, the VSS_CONN register contents will be cleared. This is due to a bug in the implementation of the API. By default the EPPI Rx broadcast is disabled. So it is not required to call the adi_vss_EnablePPIRxBcast to disable the broadcast. This will be fixed in the upcoming update of the CCES.