

Release Notes for CrossCore Embedded Studio 2.9.3

Release Notes for CrossCore Embedded Studio 2.9.3 August 2020 © 2020 Analog Devices, Inc. http://www.analog.com

Table of Contents

1	Table of Contents2
2	Introduction3
3	New and Noteworthy4
3.1	CCES Runner now part of CrossCore Embedded Studio installation4
3.2	Linker Files Add-in
3.3	Simplified Memory drop-downs in debug-related views5
3.3.1	Memory Browser
3.3.2	Memory view \rightarrow Monitor Memory dialog8
3.3.3	Variable view \rightarrow Select Memory Space dialog
3.3.4	Plot View \rightarrow Edit Data Set dialog10
3.3.5	Image Viewer \rightarrow Image Configuration dialog11
3.3.6	Browse for Symbol dialog11
3.4	ROM support added to elfloader utility for SHARC+12
3.5	UART driver API to enable/disable parity added12
3.6	ADSP-2156x HADC/TMU support added12
3.7	Updated Device Drivers and System services Add-ins
3.8	Overhaul of Code Coverage Reports
4	Changes That Might Impact Backwards Compatibility14
4.1	ADSP-2156x initcode and preloads updated14
4.2	Output section sizes in linker map XML
4.3	ADSP-2156x parts adi_uart_StopDMA UART driver API changed14
4.4	ADSP-BF7xx cfft_fr16(), ifft_fr16() and rfft_fr16() fix14

2 Introduction

This document describes the changes for CrossCore Embedded Studio (CCES) 2.9.3. You can find the release notes for older releases in the docs sub-directory of your CCES installation as well as an Installation Guide which will help you install this release.

3 New and Noteworthy

3.1 CCES Runner now part of CrossCore Embedded Studio installation

CCES Runner utility which was available as a separate download and install via Help | Install New Software... and documented on EngineerZone (CCES Runner EngineerZone FAQ) is now available as part of the CCES installation.

CCES Runner documentation is also available in the Online Help.

3.2 Linker Files Add-in

A new Linker Files Add-in has been added to provide support for copying ARM linker files into your project. It is available for the ADSP-SC5xx ARM Cortex-A5 cores. The add-in is added to new Cortex-A5 CrossCore projects by default, but can also be added from the Overview page of the system.svc file after project creation.

The add-in:

- Adds initial apt.c and app.ld files to your project. The files are specific to the processor and target memory settings.
- Configures target memory for code and data in *Linker Files Configuration* page to get the right starting point for your project.
- Automatically updates the project linker settings to use the copied app.ld file.
- Allows you to edit the apt.c and app.ld files manually and make any changes needed for your project. Be sure to keep the files consistent with each other.
- Re-copies the apt.c and app.ld files to your project when changes to your project configuration are detected.
- Provides an option to configure whether to remove or retain the existing linker files when the add-in is removed from your project.



🏶 Debug 陷 Project Explor 🛛 🔍 🔍	🕼 my project Core0,c 🛛 👼 my project Core0/system.svc 💷 🖉 app.ld	- 0)
✓ ≌ my_project_Core0	Linker Files Configuration	()
> 🔊 Includes	Linker Files Options	
Y 🐸 src	This page allows you to configure the default linker LD script added to your project.	3
> Maintend Mainten		
> Marching My_project_Core0.h	Target Memory Options	
👻 🠸 system	Select default target memory for code and data: L2	
> beap_stack		
🕆 🗁 linker		
> 🖻 apt.c		
l❷ app.ld		
🖹 app.ld.backup		
app.ld.backup2		
app.ld.backup3		
apt.c.backup		
apt.c.backup2		
apt.c.backup3		
> 🗁 mcapi		
> 🗁 pinmux		
> 🗁 sru		
> Madi_initialize.c		
> Debug		
system.svc	Overview Heap and Stack Linker Files MCAPI Pin Multiplexing Signal Routing Unit	
	🕞 Console 🗈 Problems 💿 Executables 🗊 Memory Browser 🕞 Console 🖾	🗟 🚮 🚱 🗖
	Errors	
	Invalid target memory "L3" for new processor "ADSP-SC570" while changing processor in project. Using default setting "	'L2" instead.



3.3 Simplified Memory drop-downs in debug-related views

In CrossCore Embedded Studio, there are multiple dialogs in the debugging related views that have a memory drop-down for SHARC/SHARC+ processors, which allows user to specify the memory type for the given address or symbol. In releases prior to CCES 2.9.3, this drop-down was a list of all memory types of the related processor, and you needed to determine and select the relevant memory type for the symbol. As of CCES 2.9.3, the drop-down is simplified to only show the applicable memories according to the address or symbol you have specified. If there is only one applicable memory for the address or symbol, the drop-down will be disabled and the correct memory type will be selected automatically. For the cases that no memory is

applicable for the given address or symbol, the drop-down will be empty and the OK button in the dialog will be disabled.

The new dialogs with the updated memory drop-down are as below:

3.3.1 Memory Browser

The memory drop-down in Memory Browser view is hidden for Blackfin as before since the processors in this family have only one memory type.

For other processors, the memory drop-down will be disabled if there is only one applicable memory for the given address or symbol.

💷 Console 🧟	Tasks [Problen	ns 💽 Exe	ecutables	🚺 Memory Browser 🛛	⑧ ≧ ≝ ☜ ☜ [1] 14 ▽ □ □ ◯						
main	main V Go New Tab											
0xffa0010c - m	0xffa0010c - main <hexadecimal> 🕄</hexadecimal>											
Address	0	1	2	3		^						
FFA0010C	00	E8	03	00	Blackfin							
FFA00110	F9	BØ	B8	BØ								
FFA00114	00	E3	06	00								
FFA00118	00	60	01	E8								
FFA0011C	00	00	10	00								
FFA00120	00	E8	03	00								
FFA00124	00	60	B8	B0		~						

📮 Console 🧔	Tasks 📳	Probler	ns 🚺 Ex	ecutable	s 🚺 Memory Browser 🛛 👘 🖉 🔍 🖶 👔						
SHARC L1 BW	SHARC L1 BW 00 V Zdata V Go New Tab										
0x2413d0 - zd	0x2413d0 - zdata <hexadecimal> 🕴</hexadecimal>										
Address	0	1	2	3	^						
002413D0	36	16	2E	C1							
002413D4	<mark>C0</mark>	66	46	BF							
002413D8	42	53	B1	40							
002413DC	FE	BF	30	40							
002413E0	72	DD	C1	C0							
002413E4	18	F3	44	C1							
002413F8	F0	F7	1B	C1	×						

📮 Console 🧔	🗟 Console 🧟 Tasks 🔝 Problems 🕡 Executables 🚺 Memory Browser 🖾 🔹 🔹 😂 🖄 🖄 🗂 🖆 🤝 🗖									
SHARC L1 NW	00	~ 0x0009	0000					`	Go New Tab	
SHARC L1 NW	00	mal> ¤								
SHARC LT EP C	0		2	3					^	
00090000	00000000	00000000	00000000	6	0000000					
00090004	46888888	00020010	00000000	-	C700000					
00090008	063E001C	00000000	00000000	6	00000000					
0009000C	00000000	00000B3E	0B3E0000	6	0000000					4
00090010	00000B3E	0B3E0000	00000000	4	6B60000					
00090014	063E001C	00000000	00000000	6	0000000					
00090018	00000000	469A0000	063E001C	6	0000000					
0009001C	00000000	00000000	041C0574	6	002063E					
00090020	0F760500	00000000	00000000	6	0000000					
00090024	041C0574	0003063E	0F760600	6	00000000					
00090028	00000000	00000000	041C0574	6	004063E					
00000000	0F760700	00000000	00000000	P	00000000				~	

Fill Memory dialog

🖳 Console 🖉 Tasks 🔝 Problems 🜔 Executables 🕕 Memory Browser 🖾 👘 🖉 🖄 🖄 🖄 🖄 🖄 🖄 🖄 🖄 🖄											
SHARC L1 NV	V 00	~ 0x000	90000	M. market		Go New Tab					
0x90000 - 0x0	00090000 <he< th=""><th>kadecimal > 🛛</th><th></th><th>K Fill Memory</th><th>- L X</th><th></th></he<>	kadecimal > 🛛		K Fill Memory	- L X						
Address	0	1	2	Fill source settings	Memory settings	^					
00090000	00000000	00000000	00000000	Fill memory from value	Address:						
00090004	46800000	063E001C	00000000	Value settings	0x90000						
00090008	063E001C	00000000	00000000	Value:	Memory:	1					
0009000C	00000000	00000B3E	0B3E0000	<enter a="" value=""></enter>							
00090010	00000B3E	0B3E0000	00000000	Value format:							
00090014	063E001C	00000000	00000000	Hexadecimal ~	SHARC LT NW 00						
00090018	00000000	469A0000	063E001C	Ticxudeeimar	SHARC L1 PM 0						
0009001C	00000000	00000000	041C0574	 Fill memory from file 	Stride:						
00090020	0F760500	00000000	00000000	File settings	1						
00090024	041C0574	0003063E	0F760600	File type:							
00090028	00000000	00000000	041C0574	Text file							
0009002C	0F760700	00000000	00000000	File name:							
00090030	041C0574	0005063E	0F760800	. Cater a file manual							
00090034	00000000	00000000	00000000	<enter a="" me="" name=""></enter>							
00090038	0B3E0000	00000000	00000B3E	Format (if not specified in te							
0009003C	00000000	00000B3E	0B3E0000	Hexadecimal							
00090040	00000B3E	0B3E0000	041C0574	Read entire file							
00090044	0F760B00	00000000	00000000	Byte order							
00090048	041C0574	0007063E	0F760C00	Big endian							
0009004C	00000000	00000000	00000000	O Little endian							
00090050	0B3E0000	00000000	00000B3E								
00090054	00000000	00000B3E	0B3E0000								
00090058	00000B3E	0B3E0000	041C05D1	?	OK Cancel						
0009005C	0F760F00	00000000	00000000								
00090060	00000000	00000B3E	0B3E0000	0000000		×					

Dump Memory dialog

🖳 Console 🖉 Tasks 🔝 Problems 🕡 Executables 🕕 Memory Browser 🔅 🔹 🖄 🖏 👘 🖆									
SHARC L1 NV	V 00	~ 0x000	90000		Ma			~	V Go New Tab
0x90000 - 0x0	00090000 <hex< th=""><th>(adecimal > 🕱</th><th></th><th></th><th>X Dump Memory</th><th></th><th>— U</th><th>×</th><th></th></hex<>	(adecimal > 🕱			X Dump Memory		— U	×	
Address	0	1	2	3	Output file settings		Memory settings		^
00090000	00000000	00000000	00000000	000	File type:		Address:		
00090004	46800000	063E001C	00000000	0C70	Text file	\sim	0x90000		
00090008	063E001C	00000000	00000000	000	File name:		Memory:		
0009000C	00000000	00000B3E	0B3E0000	000	<enter a="" file="" name=""></enter>		SHARC L1 NW 00	~	
00090010	00000B3E	0B3E0000	00000000	46B	Senter a nic namez		SHARC L1 NW 00		
00090014	063E001C	00000000	00000000	000	Format:		SHARC L1 FP 0		
00090018	00000000	469A0000	063E001C	000	Hexadecimal	\sim	SHARC L1 PM 0		
0009001C	00000000	00000000	041C0574	000:	Append to file		Stride		
00090020	0F760500	00000000	00000000	000	Write format to file		1		
00090024	041C0574	0003063E	0F760600	000	Show addresses				
00090028	00000000	00000000	041C0574	0004	Show labels				
0009002C	0F760700	00000000	00000000	000	Byte order				
00090030	041C0574	0005063E	0F760800	000	Big endian				
00090034	00000000	00000000	00000000	000	O Little endian				
00090038	0B3E0000	00000000	00000B3E	0B31					
0009003C	00000000	00000B3E	0B3E0000	000					
00090040	00000B3E	0B3E0000	041C0574	000	?		OK Cano	:el	
00090044	0F760B00	00000000	00000000	000					
00090048	041C0574	0007063E	0F760C00	0000	0000				
0009004C	00000000	00000000	00000000	0000	00B3E				~

3.3.2 Memory view → Monitor Memory dialog

As with Memory Browser, the memory drop-down in the Memory view "Monitor Memory" dialog is hidden for Blackfin processors and shown for SHARC/SHARC+ processors.

🗄 Outline 🚺 Memory 🖄						🖹 🛃 🤿	1 🕄 🖘 👫 👻	~ - 8
Monitors 🚽 🕷 🗞 SHAF	RC L1 BW	00[zo	data] ·	<hexac< th=""><th>decimal> 🛛 🕂 🕂 New Renderings</th><th>.]</th><th></th><th></th></hexac<>	decimal> 🛛 🕂 🕂 New Renderings	.]		
 SHARC L1 BW 00[zdata Add 	ress	0	1	2	3			^
00	2413D0	36	16	Жм	Ionitor Memory	×		
00	2413D4	<mark>C0</mark>	66		,			
00	2413D8	42	53	Ente	r address or expression to monitor:			
00	2413DC	FE	BF	0.0				
00	2413E0	72	DD	UXU	0090000			
00	2413E4	18	F3	Merr	2017			
00	2413E8	EØ	E7	wien	югу.			
00	2413EC	00	2A	SHA	ARC L1 NW 00	\sim		
00	2413F0	33	14	SHA	RC L1 NW 00			
00	2413F4	AD	DC	SHA	RC L1 EP 0			
00	2413F8	00	00					
00	2413FC	AD	DC		Cancer			
00	241400	33	14	00	CI			
00	241404	00	2A	62	3E			
00	241408	E0	E7	1B	41			
00	24140C	18	F3	44	41			
00	241410	72	DD	C1	40			~
<u></u>	241414	ГГ	DE	מר	<u>^</u>			

3.3.3 Variable view → Select Memory Space dialog

When "View Memory" in the Variable view, the "Select Memory Space" dialog will only show when there are more than one applicable memory for the selected variable. As with other dialogs, the drop-down in this dialog will only list applicable memories for the selected variable or address.



🗐 Console 🖉 Tasks 🔝 Problems 💽 Executables 🔗 F	ot 🛛 🛞	ଇ୍ର୍ଜ୍ 📑 🖻 🔻 🗝 🖻
	Untitled	^
	🔀 Plot Configuration	- 🗆 X
	General	
	Type: Surface plot	✓ Settings
	Data sets	
20	🖄 Data Set 0 [Address='zdata', Row Count=21, Column C	Count=21] New
10	Edit Data Set —	
sg 5	ame: Data Set 0	·
× -5	Z-Axis	
-10	ddress: zdata	
-20	femory: SHARC L1 BW 00	~ -
0	ows: 21 Columns: 21	
5	Data type: Signed Integer	~
10		
Y-Axis		
	? ОК	Cancel
<		>

3.3.4 Plot View \rightarrow Edit Data Set dialog

🖳 Console 🧟 Tasks 🔝 Problems 💽 Executables 層 Image Viewer 🛛 8 Q. Q. 2 1 🗹 🔀 Image Configuration × Memory Info Image Source Source location: DSP Memory Start address: zdata \sim Source format: Raw pixel data SHARC L1 BW 00 Memory: C:\Analog Devices\ADS File name: Stride: Image Info Options Update on halt Write to memory Pixel format: Grayscale (8-bit) Pack data Reverse pack Width (pixels): 512 Height (pixels): 512 ? Import. Export... OK Cancel < > Zoom: 167% Address: Byte values ∨ 0x93 Col: 482 Row: 96 Rotation: 0° Flip: None Gamma: 1.00

3.3.5 Image Viewer → Image Configuration dialog

3.3.6 Browse for Symbol dialog

The "Browse for Symbol" dialog can be accessed by clicking the button next to the address field above the memory drop-down in the dialogs mentioned above. The old "Show symbols in all memory types" checkbox in this dialog has been removed. Now this dialog will always show a list of all symbols for the running program.

🔀 Browse for Symbol		— 🗆	×									
Select a symbol:												
type filter text	type filter text											
Symbols:												
Symbol	Address	Memory Space	^									
interrupt_table.e	0x90080	SHARC L1 PM 0										
lib_prog_term.e	0x1c0d49	SHARC L1 SW 3										
clear_imask	0x1c0f56	SHARC L1 SW 3										
clear_imask0	0x1c0f56	SHARC L1 SW 3										
clear_imask1	0x1c0f5b	SHARC L1 SW 3										
set_imask	0x1c0e82	SHARC L1 SW 3										
set_imask0	0x1c0e82	SHARC L1 SW 3	~									
<			>									
(V)	OK	Can	cel									

3.4 ROM support added to elfloader utility for SHARC+

The -romsplitter switch directs the loader utility to produce an in-place image representing the ROM sections in the input executables and to ignore any RAM sections. ROM section content is placed directly at its target address in the image, without creating loader stream blocks.

The -combine-rom switch, on the other hand, directs the loader utility to integrate ROM sections into a boot stream. ROM section content is placed directly at its target address in the image, whereas RAM sections are transformed into loader blocks that are fitted around the ROM section content.

With both options, word-addressed ROM sections are transformed to byte-addressed content in the image, and their addresses translated accordingly. By default, addresses in the produced image are relative to the start address of the SPI range, which is 0x60000000. This can be changed with option -rom-base. In linker description files, the RAM or ROM type of a section is determined by the definition of the memory segment that the section is mapped into.

3.5 UART driver API to enable/disable parity added

UART driver now provides a new adi_uart_EnableParity() function, to enable/disable the UART parity configuration dynamically.

3.6 ADSP-2156x HADC/TMU support added

The Housekeeping ADC (HADC) and Thermal Monitoring Unit (TMU) features are now fully supported by CCES for all ADSP-2156x family processors.

3.7 Updated Device Drivers and System services Add-ins

If you uninstall a device driver module component add-in provided through system.svc, only driver source files will be removed from the project; the static configuration file (adi_xxx_2156x_config.h) will not be deleted from the project.

If you then re-install the same device driver module component add-in, this will overwrite the existing static configuration file with the default one which is provided in the CCES installation. You should take a back-up/ copy of the existing static configuration file prior to re-installing the driver add-in, if required.

3.8 Overhaul of Code Coverage Reports

Code Coverage reports have been updated as follows

- More accurate results.
- Statistical summary added.
- Style improvements.

Search for "Generating Code Analysis Reports" in the CCES help for details about code coverage reports.

4 Changes That Might Impact Backwards Compatibility

4.1 ADSP-2156x initcode and preloads updated

The initcode source projects and prebuilt initcode and preload executables for EZ-KIT support now set DCLK to 667 MHz as required for silicon revision 0.2. In previous CCES releases DCLK was set to 500 MHz as required for silicon revision 0.0.

4.2 Output section sizes in linker map XML

Following the fix for issue CCES-22172, the size reported for each output section in linker map XML output is the sum of the input sections and fragments placed into it. Previously, it was reported as the difference between the start address of the first input section/fragment and the end address of the last input section/ fragment within it. This meant that if an output section was broken up into multiple parts, for example because a part of it fitted into an alignment gap in another output section, too large a size was reported for it.

The change does also mean that alignment gaps within contiguous output sections are no longer counted towards their reported size. This should provide more reliable reporting of code and data sizes, as alignment gap sizes can change somewhat unpredictably when sizes of input sections change.

4.3 ADSP-2156x parts adi_uart_StopDMA UART driver API changed

The adi_uart_StopDMA() API has been updated to add a new parameter, ADI_UART_STOP_DMA_CHANNEL Channel, which will allow you to select whether the function stops the DMA channel for TX, RX, or both TX and RX. Previously, this API would always stop both TX and RX DMA channels.

4.4 ADSP-BF7xx cfft_fr16(), ifft_fr16() and rfft_fr16() fix

There has been a fix to correct the dynamic scaling done by the Blackfin+ cfft_fr16, ifft_fr16 and rfft_fr16 functions. The fixed versions of these functions are now always defined by libdsp.dlb and the versions in the ADSP-BF70X parts Utility ROM will no longer be used. This will mean a small increase in the code footprint of applications calling these functions, since they will now be occupying RAM. On the positive side, these functions may me linked to L1 memory for improved performance, if needed.