Release Notes for CrossCore Embedded Studio 2.9.3

1 Table of Contents

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.

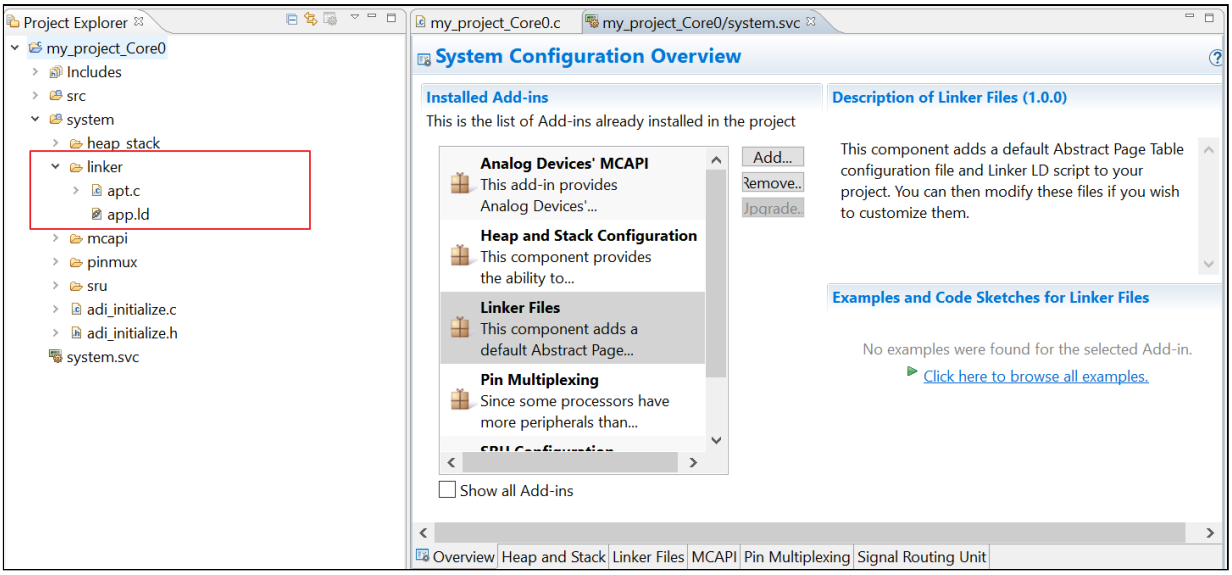
CCES Runner documentation is also available in the Online Help.

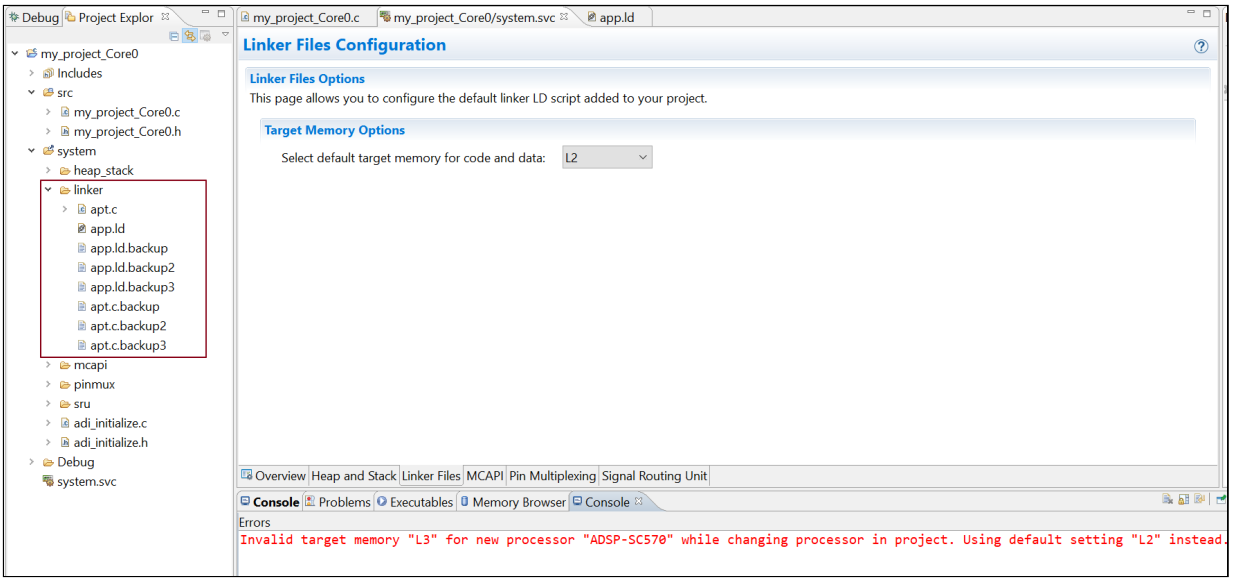
3.2 Linker Files Add-in

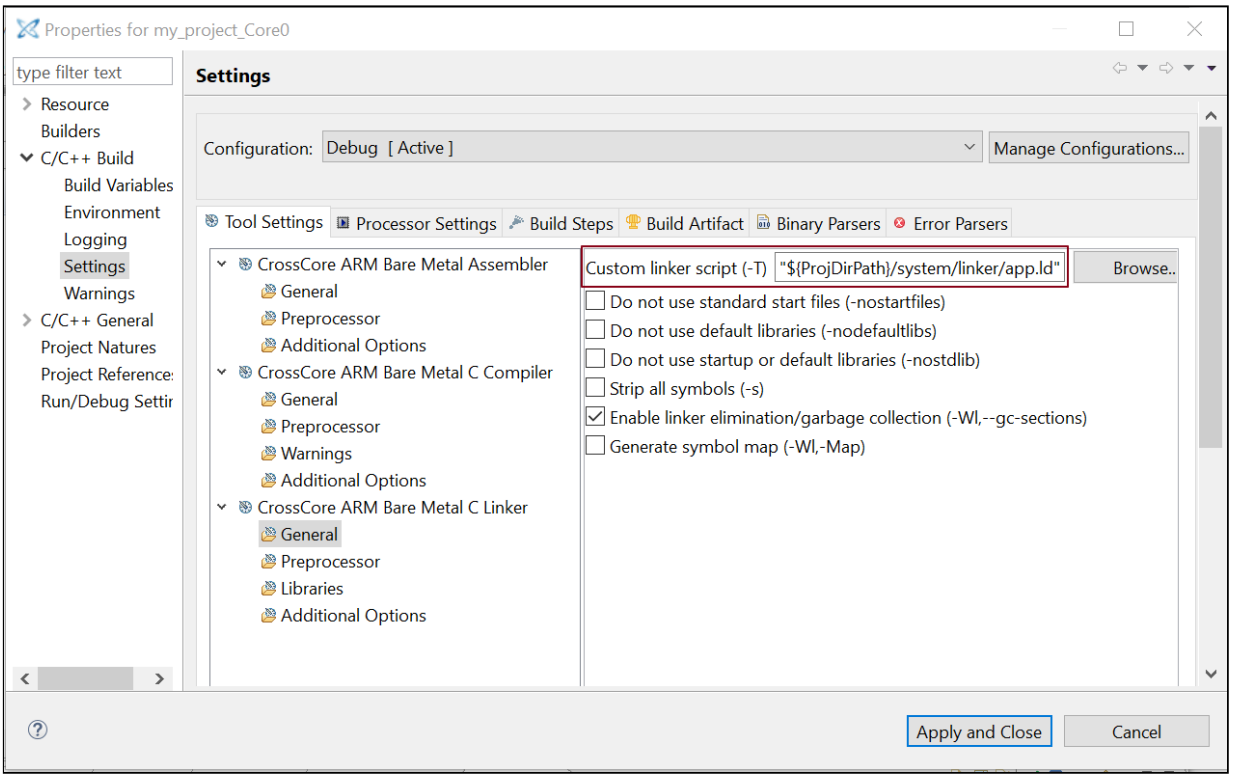
A new Linker Files Add-in has been added to provide support for copying ARM linker files into your project. It is available for the ADSP-SC5xx ARM Cortex-A5 cores. The add-in is added to new Cortex-A5 CrossCore projects by default, but can also be added from the `Overview` page of the `system.svc` file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







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

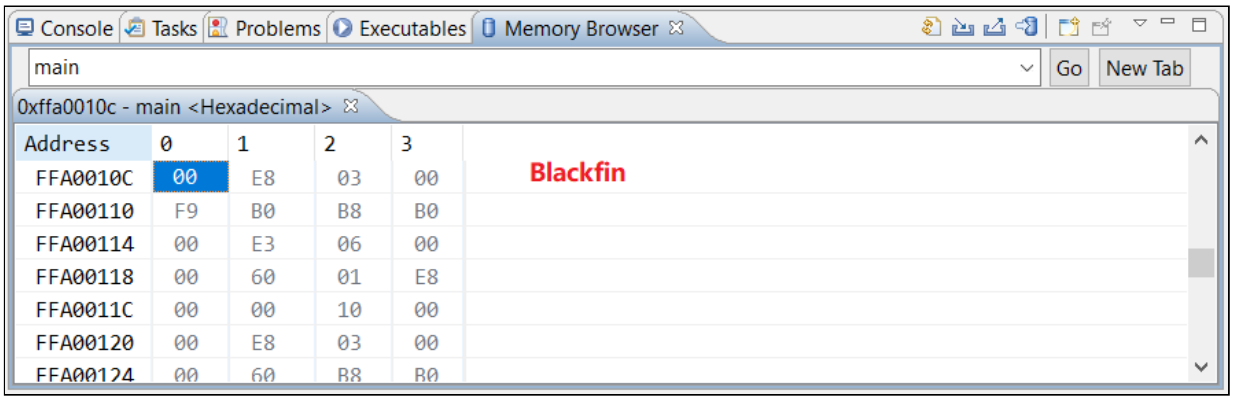
applicable for the given address or symbol, the drop-down will be empty and the OK button in the dialog will be disabled.

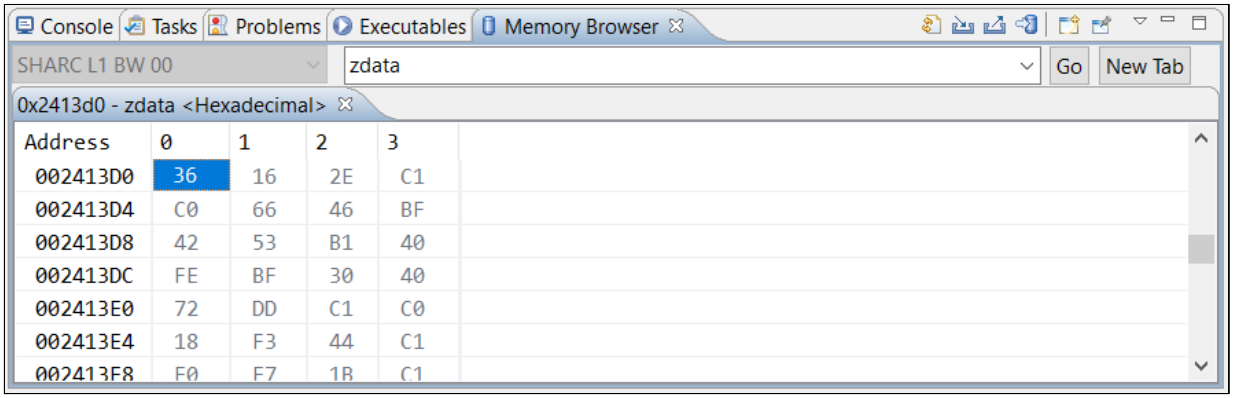
The new dialogs with the updated memory drop-down are as below:

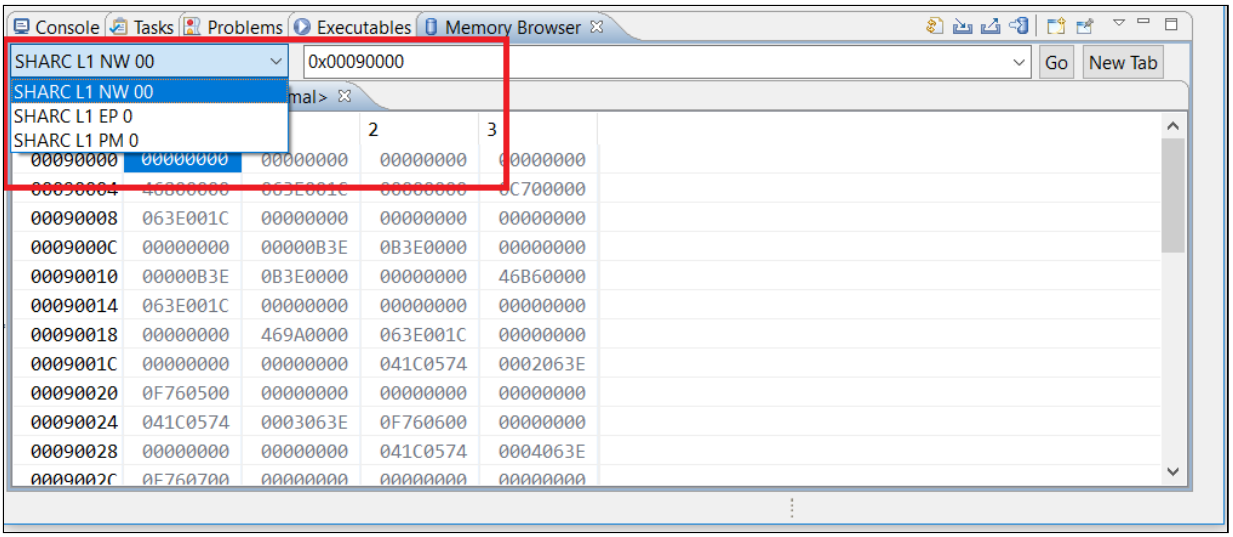
3.3.1 Memory Browser

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.

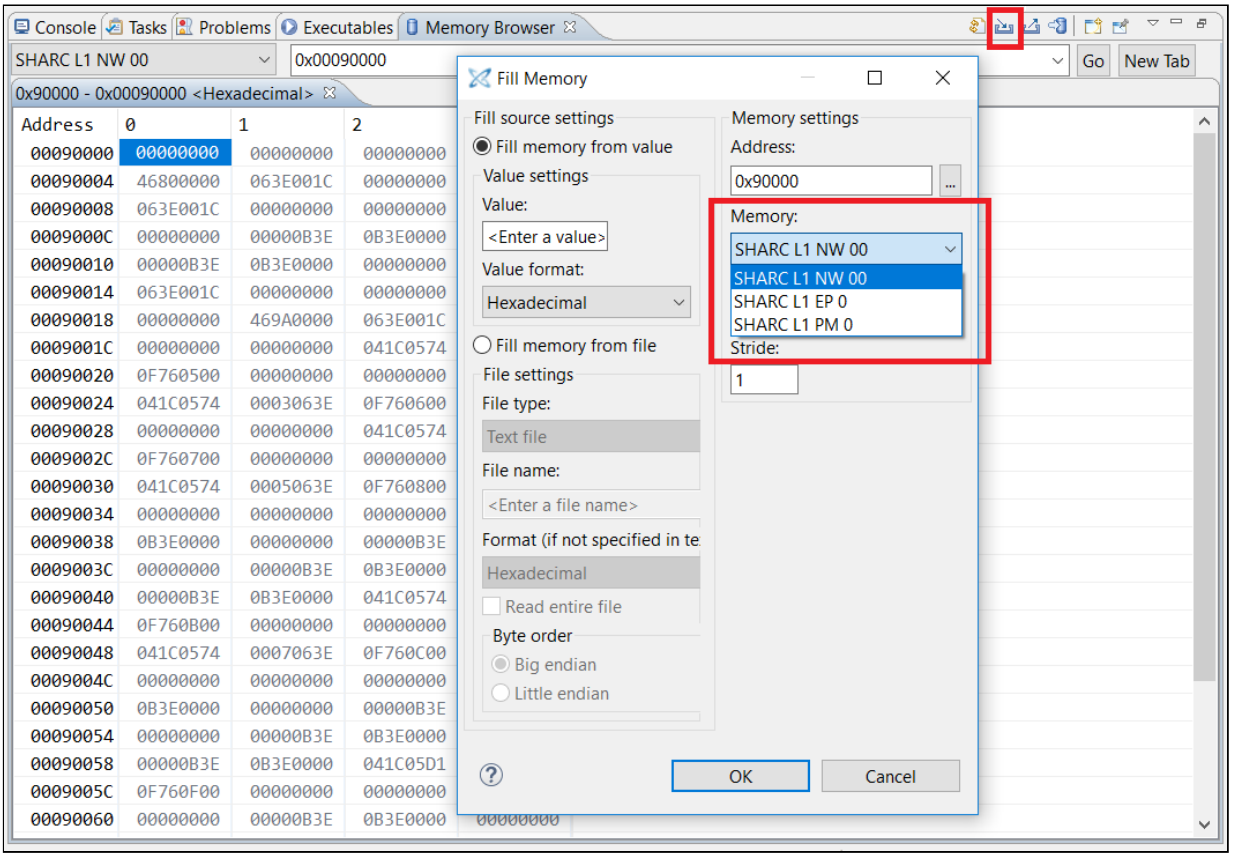
For other processors, the memory drop-down will be disabled if there is only one applicable memory for the given address or symbol.



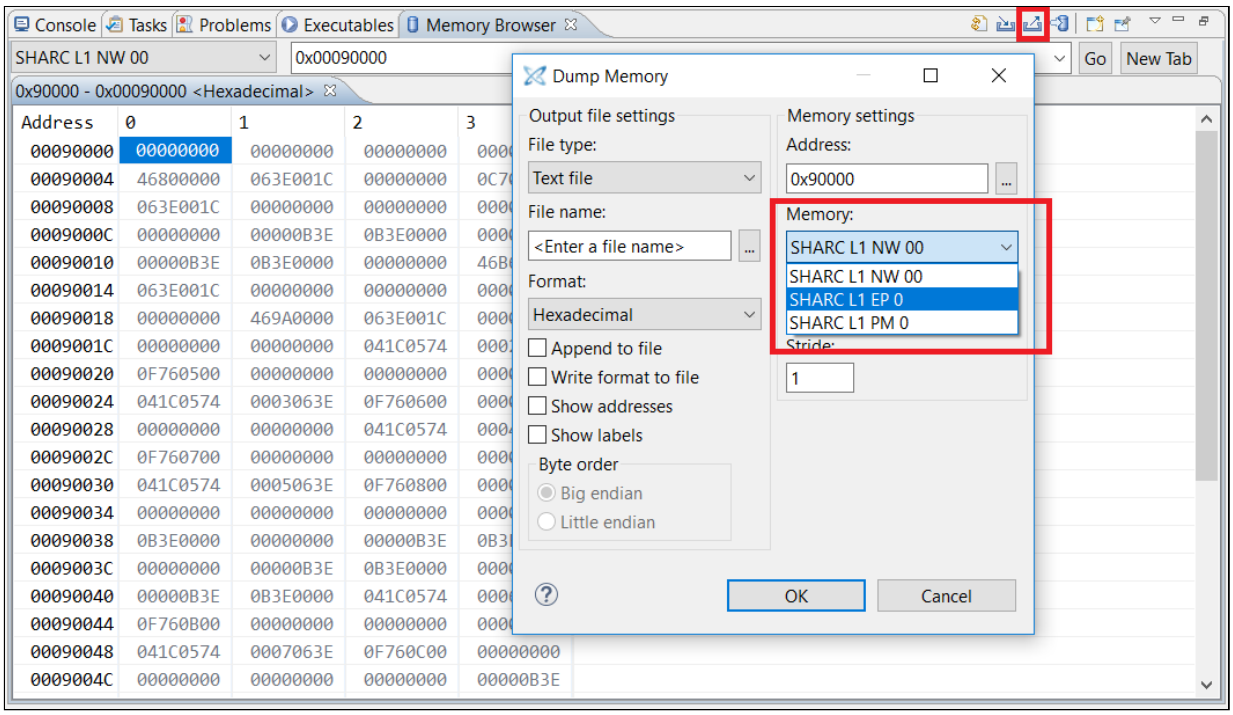




Fill Memory dialog

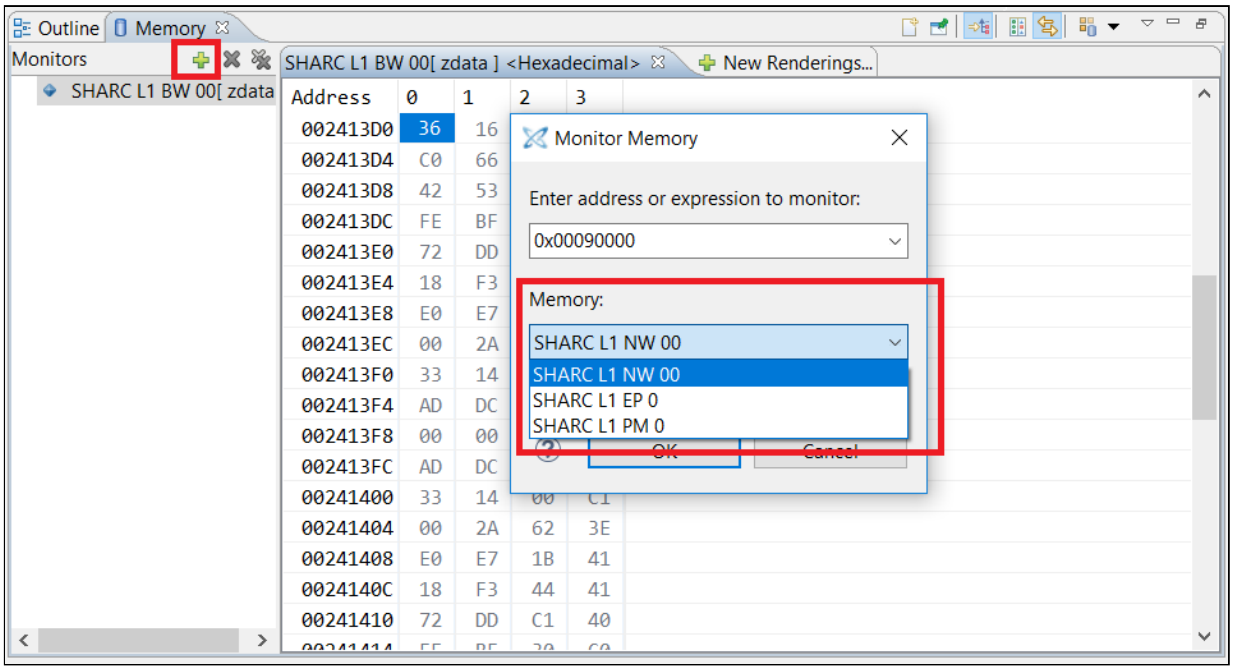


Dump Memory dialog



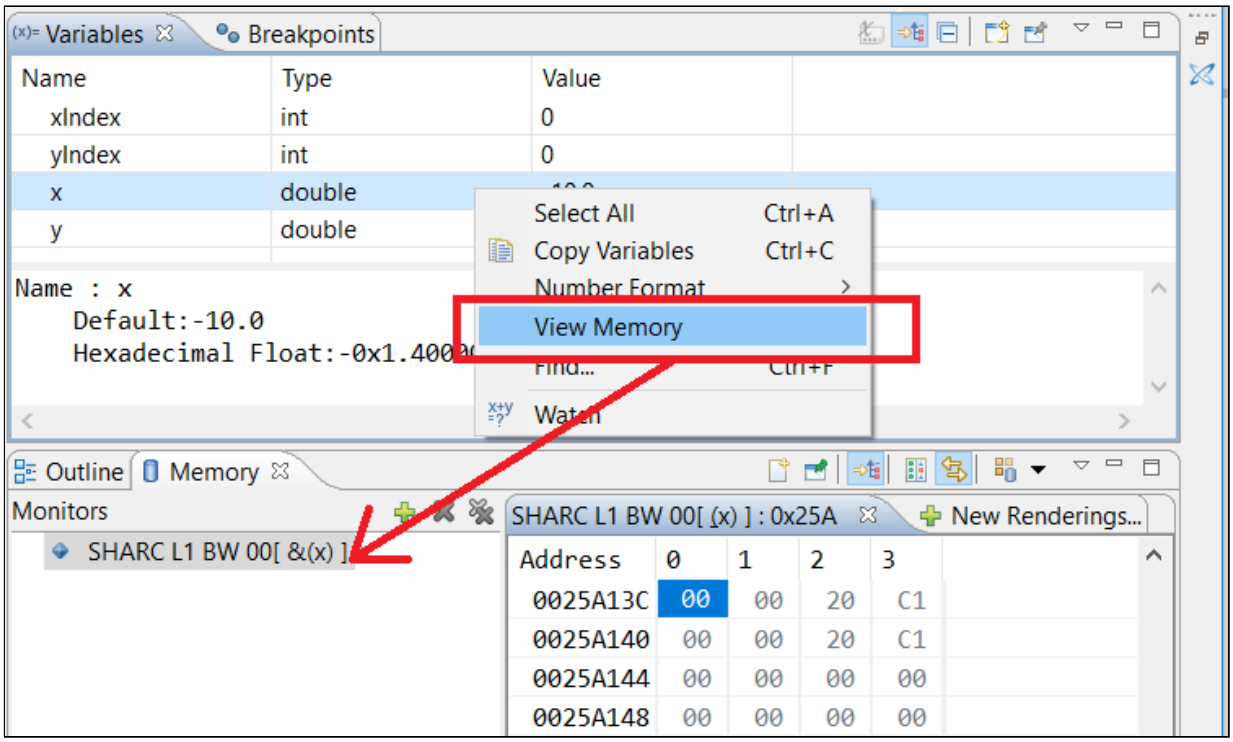
3.3.2 Memory view → Monitor Memory dialog

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.

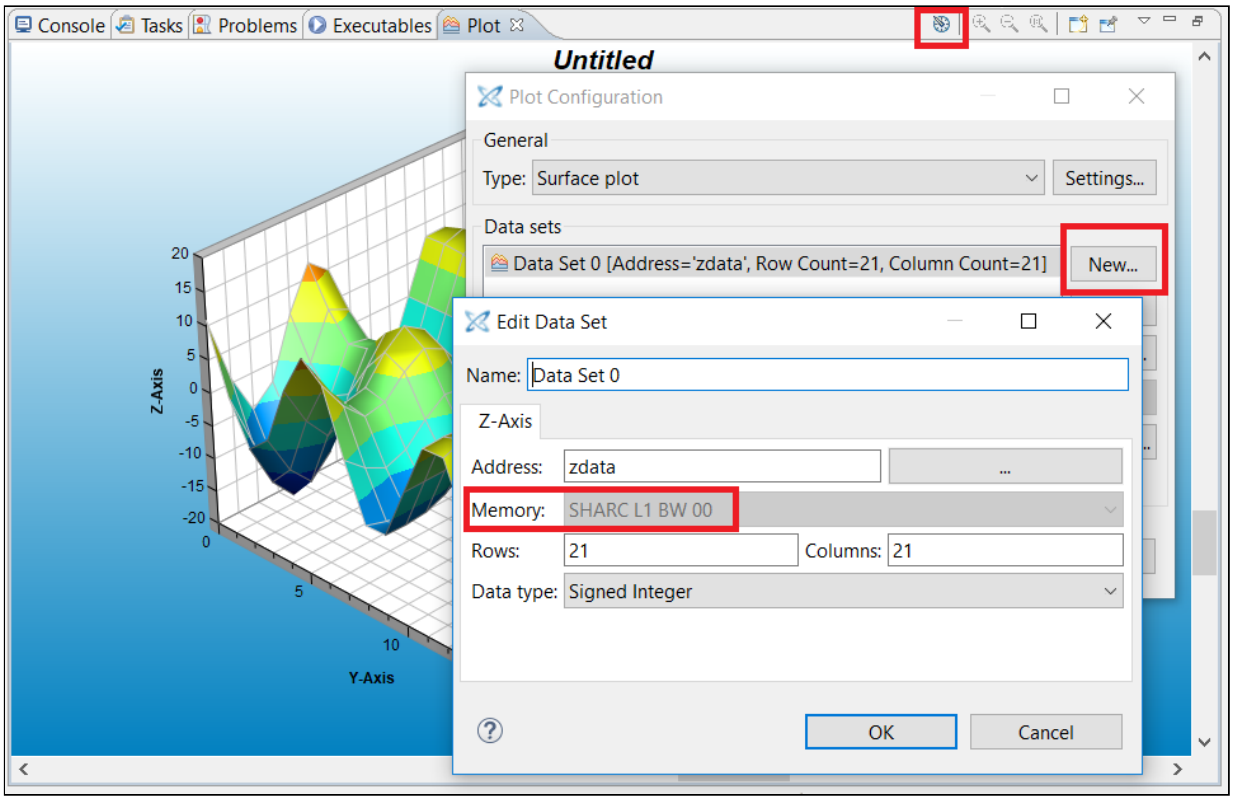


3.3.3 Variable view → Select Memory Space dialog

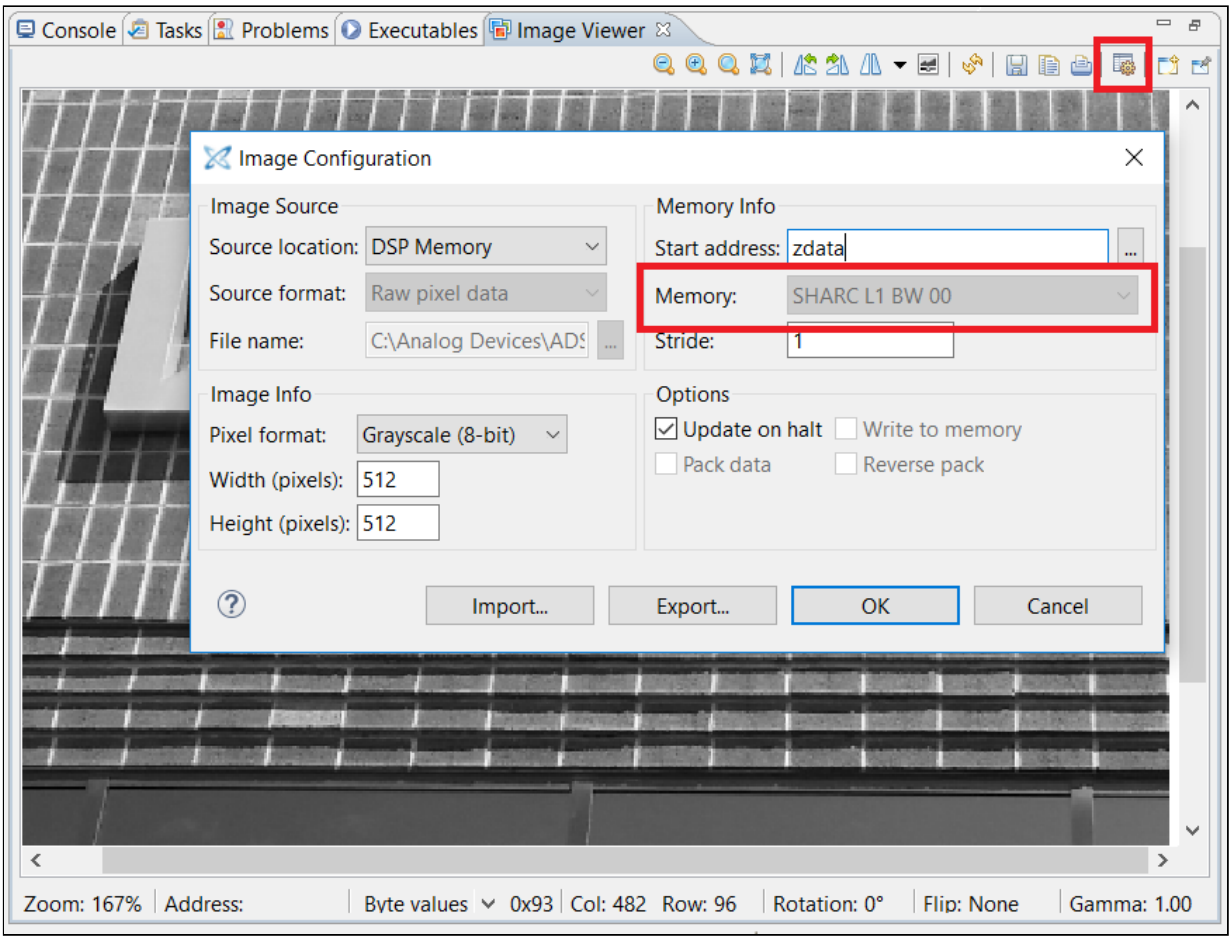
When "View Memory" in the Variable view, the "Select Memory Space" dialog will only show when there are more than one applicable memory for the selected variable. As with other dialogs, the drop-down in this dialog will only list applicable memories for the selected variable or address.



3.3.4 Plot View → Edit Data Set dialog

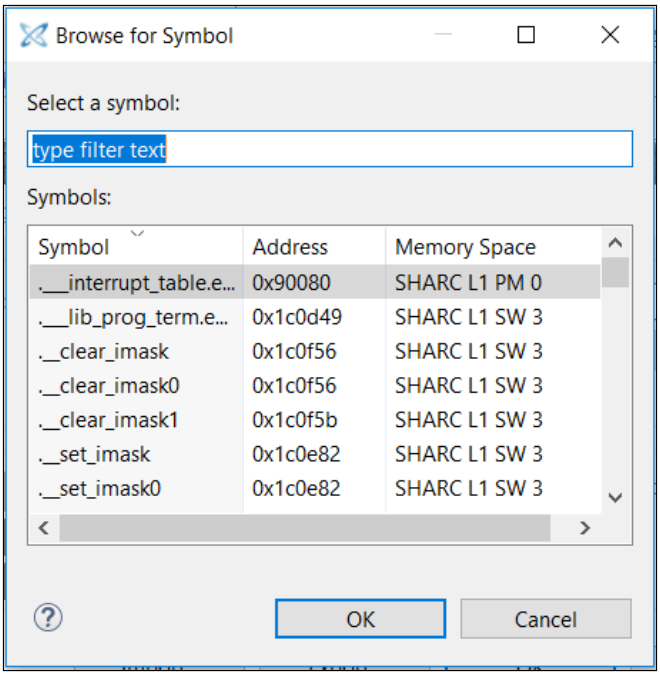


3.3.5 Image Viewer → Image Configuration dialog



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.



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.