

What's New in Code Composer Studio Development Tools v 3.3

DSP SDS

ABSTRACT

This application report highlights new features and functionality in the Code Composer Studio™ (CCStudio) Integrated Development Environment (IDE) v. 3.3. Unlike 3.2 (which offers support for the DM644x and C64x+ device families), version 3.3 supports multiple device families. Several of the new debug options from the 3.2 version are now available for all platforms.

Topic	Page
1 Installing the 3.3 Version	2
2 Supported Devices and Simulators	2
3 IDE Property Manager	4
4 Status Bar	5
5 Register Window Changes	6
6 New Breakpoint Manager	7
7 New Cache Management Tools	9
8 Tool for Reporting Exceptions on C64x+ Devices	11
9 Adding Globals to the Watch Window	12
10 C6000 Functional Simulators Interrupt Latency Detection Feature	13
11 Displaying Software Pipeline Loops (SPLOOP) in the Disassembly Window	13
12 MMU Page Table Viewer	14
13 Memory Load/Save Utility	14
14 New Debug Menu Items	15
15 Troubleshooting Emulator Errors	15
16 IDE Changes	15

1 Installing the 3.3 Version

1.1 The 3.3 Package

Version 3.3 is the latest production release of the CCStudio Integrated Development Environment. The install is provided on a single CDROM and supports all TI catalog production DSP and SOC processors. The installation is a full installation and is not a patch or service pack to any previous installations. By default, the 3.3 version CD installs in a separate directory from the 3.2 version, so there are no dependencies between versions. For more information using the IDE and exploring the new features, see the Code Composer Studio Development Tools v3.3 Getting Started Guide ([SPRU509](#)).

1.2 System Requirements

Minimum Requirements:

- 500 MHz or higher Pentium compatible CPU
- 128 MB of RAM
- 600 MB of free hard disk space
- SVGA (800 X 600) display
- Internet Explorer (5.0 or later)

Recommended:

- 2 GHz or higher Pentium-compatible CPU
- 512 MB of RAM
- 16-bit color

Supported Operating Systems:

- Windows 2000 Service Pack 4
- Windows XP Home Service Pack 1 & 2
- Windows XP Pro Service Pack 1 & 2

2 Supported Devices and Simulators

CCStudio v3.3 supports the following platforms:

- ARM Platforms
 - ARM11- VPOM2420 Platform Simulator, VPOM2430 Platform Simulator
 - ARM7- VPOM2420 Platform Simulator, Big Endian, Little Endian, XDS510, XDS560 Emulators
 - ARM9 Simulator, Big Endian, Little Endian, XDS510, XDS560 Emulators
 - ARM925 Simulator, Little Endian
 - ARM926EJ-S Simulator, Little Endian
- C2000 Platforms
 - F240 XDS510, XDS560 Emulators
 - F2401 XDS510, XDS560 Emulators
 - F2402 XDS510, XDS560 Emulators
 - F2403 XDS510, XDS560 Emulators
 - F2406 XDS510, XDS560 Emulators
 - F2407 XDS510, XDS560 Emulators
 - F241 XDS510, XDS560 Emulators
 - F243 XDS510, XDS560 Emulators
 - C27xx Cycle Accurate Simulator, XDS510, XDS560 Emulators
 - F2810 Device Simulator, XDS510, XDS560 Emulators
 - F2812 Device Simulator, XDS510, XDS560 Emulators

-
- F28xx Cycle Accurate Simulator, Simulator Tutorial
 - C5000 Platforms
 - C5401, C5402, C5403, C5404, C5406, C5407, C5409 Device Simulators
 - C541, C542, C543, C545, C545lp, C546, C548, C549 Device Simulators
 - C5410, C5416, C5420 Device Simulators
 - C5401 XDS510, XDS560 Emulators
 - C5402 XDS510, XDS560 Emulators
 - C5404 XDS510, XDS560 Emulators
 - C5407 XDS510, XDS560 Emulators
 - C55xx Functional Simulator
 - C55xx Rev 3.0 Functional Simulator, Cycle Accurate Simulator
 - C6000 Platforms
 - C6201 Device Simulator Little Endian, Big Endian
 - C6202 Device Simulator Little Endian, Big Endian
 - C6203 Device Simulator Little Endian, Big Endian
 - C6204 Device Simulator, Little Endian, Big Endian
 - C6205 Device Simulator, Little Endian, Big Endian
 - C620x XDS510, XDS560 Emulators
 - C6211 Device Cycle Accurate Simulator, Little Endian, Big Endian
 - C621x XDS510, XDS560 Emulators
 - C62xx CPU Cycle Accurate Simulator, Little Endian
 - C62xx CPU Cycle Accurate Simulator, Big Endian
 - C6216[Compiled] Device Functional Simulator, Little Endian, Big Endian
 - C64+ CPU Cycle Accurate Simulator, Little Endian, Big Endian
 - C64+ Cycle Accurate Simulator, Little Endian, Big Endian
 - C6455 Simulator (with VCP & TCP) Big Endian, Little Endian
 - C6455 XDS510, XDS560 Emulator with ICEPICK_C
 - DM6443 Cycle Accurate Simulator, Little Endian
 - DM6443 XDS510, XDS560 Emulator with ICEPICK_C
 - DM6446 Cycle Accurate Simulator, Little Endian
 - DM6446 XDS510, XDS560 Emulator with ICEPICK_C
 - TCI6482 Simulator (with RSA, VCP & TCP) Big Endian, Little Endian
 - TCI6482 XDS510, XDS560 Emulator with ICEPICK_C
 - C6411 Device Cycle Accurate Simulator, Little Endian, Big Endian
 - C6412 Device Cycle Accurate Simulator, Little Endian, Big Endian
 - C6414 Device Cycle Accurate Simulator, Little Endian, Big Endian
 - C6414, 15, 16 Rev1.0x XDS510, XDS560 Emulators
 - C6415 Device Cycle Accurate Simulator, Little Endian, Big Endian
 - C6416 Device Cycle Accurate Simulator, Little Endian, Big Endian
 - C64xx- VPOM2430 Platform Simulator
 - C64xx CPU Cycle Accurate Simulator, Little Endian, Big Endian
 - C64xx XDS510, XDS560 Emulators
 - C6416[Compiled] CPU Functional Simulator, Little Endian, Big Endian
 - DM642 Device Cycle Accurate Simulator, Little Endian, Big Endian
 - C6701 Device Simulator, Little Endian, Big Endian
 - C670x XDS510, XDS560 Emulators
 - C6711 Device Cycle Accurate Simulator, Little Endian, Big Endian
 - C6712 Device Cycle Accurate Simulator, Little Endian, Big Endian
 - C6713 Device Cycle Accurate Simulator, Little Endian, Big Endian
 - C671x XDS510, XDS560 Emulators
 - C672x CPU Simulator, Little Endian

- C672x XDS510, XDS560 Emulators
- C67xx CPU Cycle Accurate Simulator, Little Endian, Big Endian
- OMAP Platforms
 - OMAP- VPOM2420 Platform Simulator
 - OMAP- VPOM2430 Platform Simulator
 - OMAP1510 ES2 XDS510, XDS560 Emulators
 - OMAP1610 XDS510, XDS560 Emulators
 - OMAP1710 XDS510, XDS560 Emulators
 - OMAP2420 XDS510, XDS560 Emulators
 - OMAP310 XDS510, XDS560 Emulators
 - OMAPV1030 XDS560 Emulator

3 IDE Property Manager

Several debug tools now have enhanced interfaces on all platforms. Typically, to configure properties for a debug tool, you would right-click anywhere within a debug window and select Properties. Now, new and recently revised debug windows use the Property Manager for configuration, including the Memory Window, Breakpoint Manager and Cache Tag RAM Viewer. The Property Manager is displayed and remains on the right side of the main control window. The Property Manager lists available properties for a debug tool and lets you edit individual properties.

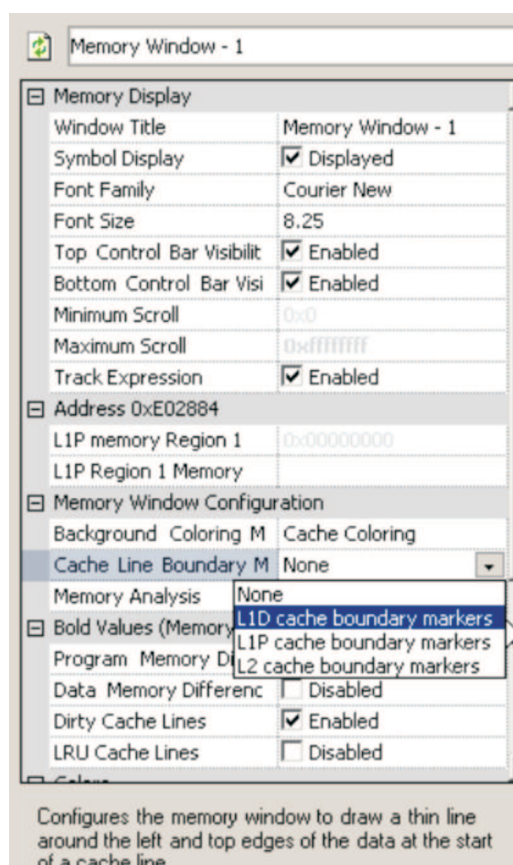


Figure 1. Property Manager for the Memory Window. Here, the drop-down menu indicates cache boundaries with lines that you can add.

New and recently revised debug windows use the Property Manager for configuration, including the Memory Window, Breakpoint Manager and Cache Tag RAM Viewer. Often the Property Manager may duplicate functionality already present in the debug window itself. However, the Property Manager allows you to see all the configured properties for a particular window (or element of a window) at a single glance. In the Breakpoint Manager Property Window, editing certain properties such as Action will cause additional options to be displayed for further editing.

Highlighting individual elements within a debug window lets you configure specific instances or elements. For example, in the Breakpoint Manager ([Section 6](#)), each row corresponds to a different breakpoint. Highlighting one row in the Breakpoint Window reveals properties for that specific breakpoint in the Property Manager. Highlighting a different row reveals those for a different breakpoint. To view the property window, right-click anywhere in the Breakpoint Manager, Memory Window, or Cache Tag RAM Viewer windows and select Properties from the context menu, or select View→Property Window.

4 Status Bar

New options were added to the status bar that is displayed at the bottom of the CCStudio application window. In addition to the normal target connect messages, there is also a basic status indicator. The left side of the status indicator shows if the target is running or not. The right side of the status indicator turns yellow to indicate that the target was recently halted. This occurs when you manually halt the process, or when CCStudio temporarily halts the target to carry out another internal process. The indicator will turn grey after a few seconds.

The status bar will also show messages about the current options used by CCStudio:

- Process Mode (ARM processors only): The status bar displays the name of the current mode used by the executed process. The options are:
 - ARM – Indicates that the process is in the ARM mode.
 - THUMB – Indicates that the process is in the Thumb mode.
- Endianness: The status bar denotes the Endianness sequencing method being used with either LE for Little Endian or BE for Big Endian.
- Jazelle Indicator: The word JAVA is displayed in the status bar when Jazelle is enabled.
- MMU Indicator (ARM processors only): The status bar displays either MMU Off or MMU On to indicate the status of the Memory Management Unit (MMU) mode. Note: This feature is available for ARM 9 and ARM 11 targets.
- Privileges (ARM processors only): The status bar indicates the privilege mode for the application by displaying either USER mode or SUPERVISOR mode.
- Task Level Debugging Indicator: The status bar indicates the status of Task Level Debugging (TLD) by displaying TLD when TLD has been enabled on the device. Note: TLD support is not available for all operating systems.
- Descriptions: The center of the status bar displays text which describes the actions of individual menu commands and toolbar items as you hold the mouse cursor over them, and the path of the active source file window. The right area of the status bar shows the line and column position of the cursor when viewing a source file.
- Profile Clock: The Profile Clock is displayed on the right side of the status bar, if it has been enabled. See the Application Code Tuning Online Help for more information.

5 Register Window Changes

5.1 Creating Custom Register Groups

The Register Window now uses a two-panel window display that appears at the bottom of the main control window when you select one of the registers from the View menu. The left panel consists of logical groups of registers; the right panel consists of individual registers that belong to this logical group. Clicking on a register type/group on the left panel shows registers of that particular type. Clicking the mouse on the register value on the right lets you edit (or even copy) its value.

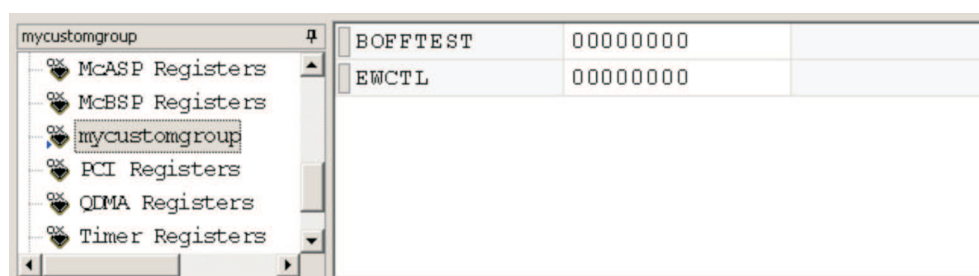


Figure 2. Register Window. This two panel display puts registers into logical categories and custom groups. Highlighting the category on the left lets you view (and edit) individual values on the right.

In addition, you can create custom groups of registers to view only the registers most likely to be of interest for a project. To do this, right-click anywhere within the Register Window, select the Customize Register Group item from the context menu, and create a custom group. After you create a group with registers, the new custom group will appear in the Register Window's left panel. Clicking on the group's name displays the associated registers.

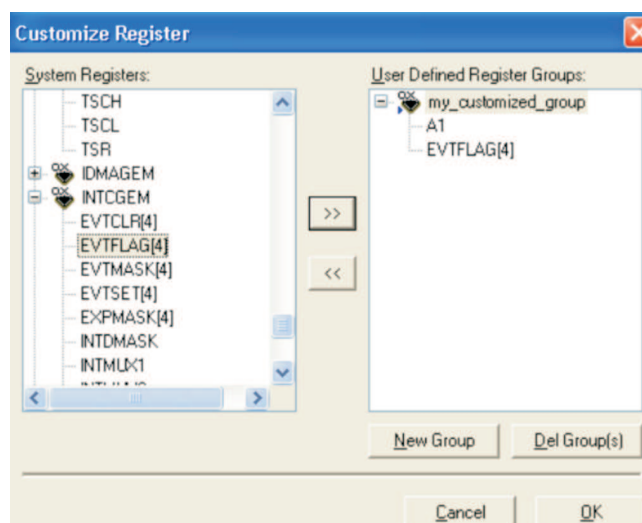


Figure 3. Customizing Register Groups. Grouping individual registers in a custom group provides easy access to the important registers in the Register Window.

5.2 Bitfield View

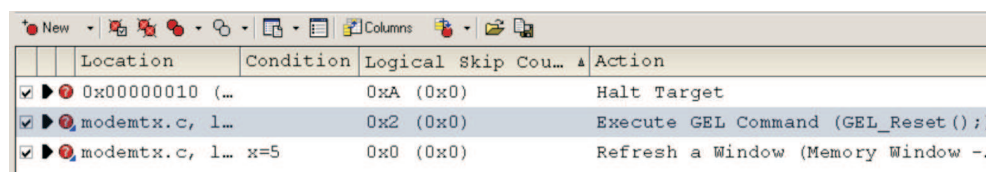
Some registers can also be expanded to show the individual bits for that register. This option is only available for some devices. To enter bitfield mode, right-click on the register window and select Tree View from the context menu. Any registers that support a bitfield will have a small + sign next to them. These can be expanded to view the bitfield. The value for each bit can be edited by double-clicking on the value. Any edited values will appear in red.

6 New Breakpoint Manager

6.1 Different User Interface

The interface for creating and configuring breakpoints has changed substantially. In previous IDE versions, to configure a breakpoint, you needed to open a separate dialog and follow a complex series of steps. The new Breakpoint Manager interface lets you do all the steps within a single window that can be left open throughout the debugging process. To open the Breakpoint Manager, select Breakpoints from the Debug menu.

The Breakpoint Manager lists breakpoints as rows on a table and lets you configure them by clicking the mouse on a column in that row. (You can also configure individual breakpoints by opening up the Property Manager.) You can also customize your view, save/load breakpoint configurations, and sort columns. The Breakpoint Manager uses colors and icons to denote the status and type of a particular breakpoint.



	Location	Condition	Logical Skip Count	Action
✓ [icon]	0x00000010 (...)	0xA (0x0)		Halt Target
✓ [icon]	modemtx.c, l...	0x2 (0x0)		Execute GEL Command (GEL_Reset();)
✓ [icon]	modemtx.c, l... x=5	0x0 (0x0)		Refresh a Window (Memory Window -...)

Figure 4. Breakpoint Manager. Note the icons on each breakpoint row: question mark (conditional breakpoints), check boxes (enable/disable), blue triangle underneath the circle (non-halt action).

6.2 Associating Actions with Breakpoints

By using the drop-down menu on the Action column, you can associate a breakpoint with a particular action: Halt Target, Update All Windows, Execute GEL Command, Refresh a Window, Enable/Disable a Group and Perform File I/O.

6.3 Merging Probe Point Functionality into the Breakpoint Manager

Previous versions of CCStudio IDE used the term *probe point* when referring to a point you could designate to update a debug window or perform read/write actions on a file. This functionality still exists; however, probe points in this version will be treated by the Breakpoint Manager as another type of breakpoint with different actions. If you click on the appropriate cell under the Action column, a series of drop-down options appear.

You can still associate a data input/output file with the breakpoint as you did before (by clicking File → File I/O). With the new Breakpoint Manager, you can also do this by clicking on the Action column for a breakpoint and selecting the *Perform File I/O* drop-down option.

6.4 Breakpoint Groups and Configurations

You can create custom filters by creating breakpoint groups. For example, suppose that some breakpoints are triggered only under certain circumstances (i.e., they are conditional breakpoints). You could put all the conditional breakpoints into a group, so only one type of breakpoint would be visible.

To create a custom breakpoint group, click on the Create/Filter Breakpoint Groups icon at the top of the Breakpoint Manager column. After you click it, a new dialog will let you add new breakpoints. You can then assign an individual breakpoint to a group by clicking on the appropriate column and selecting the desired group from the drop-down list.

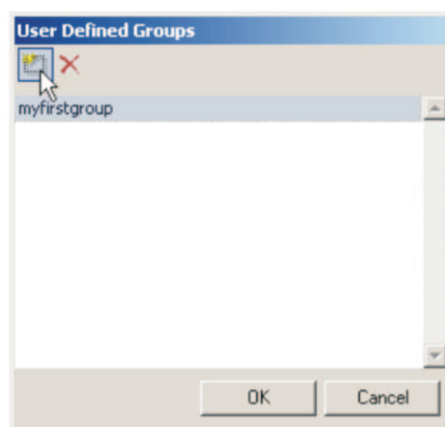


Figure 5. Creating a Breakpoint Group. Create a custom group and then add breakpoints to it.

6.5 C64x Simulator Watchpoints

Watchpoints halt simulation when an access is made to a targeted memory location. The access could be from the CPU or from the DMA (if supported by the particular device configuration). Debug reads and writes to memory locations do not cause watchpoints to trigger. Watchpoints are supported by the following configurations:

- C64xx CPU Cycle Accurate Simulator
- C6416 Device Functional Simulator

To use this option, you must load a project and open the Breakpoint Manager by selecting Debug → Breakpoints. In the Breakpoint Manager, select New Watchpoint from the dropdown menu next to the New Item icon in the toolbar. The New Watchpoint dialog will open. Enter a location for the watchpoint and an event to break on. Location can be specified either as an absolute memory address, or a symbol name. For example, to create a watchpoint on a global variable defined in a .c file as:

```
int IntegerA;
```

You would specify the watchpoint location as:

```
&IntegerA
```

You can choose from the following events:

- Memory Read
- Memory Write
- All Access Types

Choose an event and click OK. The Breakpoint Manager will show your new watchpoint. See the online help for Watchpoints for more information on modifying existing watchpoint properties or halting the target on these breakpoints.

6.6 C55x Hardware Breakpoints

For C55x emulators, you have multiple additional options under the New Item menu in the Breakpoint Manager. For instance, you can create a new hardware breakpoint, watchpoint, or counter. These options are supported on all C55x emulation devices included in the CCStudio IDE. To use the options, you must load a project and open the Breakpoint Manager by selecting Debug → Breakpoints. In the Breakpoint Manager, select the dropdown menu next to the New Item icon in the toolbar. There will be four options visible: Software Breakpoint, Hardware Breakpoint, Hardware Watchpoint, or Hardware Counter. Choosing one of the hardware items opens a new parameters dialog for you to input basic information for the selected item. For hardware breakpoints, you can choose the location from the parameter dialog. For hardware watchpoints, you can choose the location and the access type (read, write, etc.). For counters, you can choose whether you would like to count cycles or events. If you choose event, the menu will show multiple event types under the dropdown menu for Event. See your hardware documentation for more information on these events.

7 New Cache Management Tools

When using a hierarchical memory architecture, handling cache coherency issues can be a challenge. Generally if multiple devices, such as the CPU or peripherals, share the same cacheable memory region, cache and memory can become incoherent. In the past it has been hard to pinpoint cache coherency problems, but this IDE version contains two tools designed specifically to make it easier to spot and solve cache coherency problems.

7.1 Enhanced Memory Window

7.1.1 New Functionality

The new memory window contains several new functions. Select View→Memory to display the memory window.

Drop-down boxes make the memory window easier to configure. For example, you can use a drop-down menu to set the memory formatting style or appearance.

A new combo box on top left lets you search for specific memory addresses, or you can even track a specific expression if an expression is entered. By using the right arrow buttons, you can also set a scrolling range for the window and have better control over how finely you can scroll down in the window.

The Property Manager associated with the Memory Window lets you control color, font, and symbol displays.

You can now edit/change specific memory values by clicking on a cell's contents. A message bar (on the bottom left) informs you of any potential error messages.

The Memory Window also lets you switch between views of multiple targets/cores on the fly. A drop-down menu on the top right of the Memory Window lets you switch target views quickly. If you have loaded more than one device through Code Composer Setup, the drop-down box will let you view memories of other cores without needing to open up another Control Window. This option is helpful if you are using peripherals that share memory with another core.

7.1.2 Managing Cache with the New Memory Window

The Memory Window can now help manage cache problems by bypassing cache for devices that use cache. This feature is available on some simulators and selected hardware targets for C64+ devices. Checkboxes on the bottom let you bypass caches to inspect memory as seen by external entities. If all boxes are checked, the memory window will show memory values from the highest memory level containing the address. You can also assign different colors for different kinds of cache, which allows you to quickly view the active cache at any given memory address.



Figure 6. Memory Window. Different colors identify different kinds of cache. The L-shaped lines are cache line boundaries.

7.2 Cache Tag RAM Viewer

Cache Tag RAM Viewer is another tool for solving cache coherency problems for C64x devices. This feature is available on some simulators and selected hardware targets for C64+ devices. A cache tag RAM stores various data related to cache states:

- **Tag address:** Address bits that map the line to the physical memory
- **Valid:** Whether the line is in cache
- **Dirty:** Whether the line was written by the CPU
- **LRU:** Least recently used state

To launch the viewer, choose Tools→CacheTagRAMViewer Control→Cache Tag RAM Viewer. The Cache Tag RAM viewer displays cache tag RAMs as a table with sortable columns. The table provides a static view into cache, which makes it easier for developers to optimize cache usage, memory layouts, and access patterns. You can sort columns to group dirty lines together or analyze the layout of a particular cache.

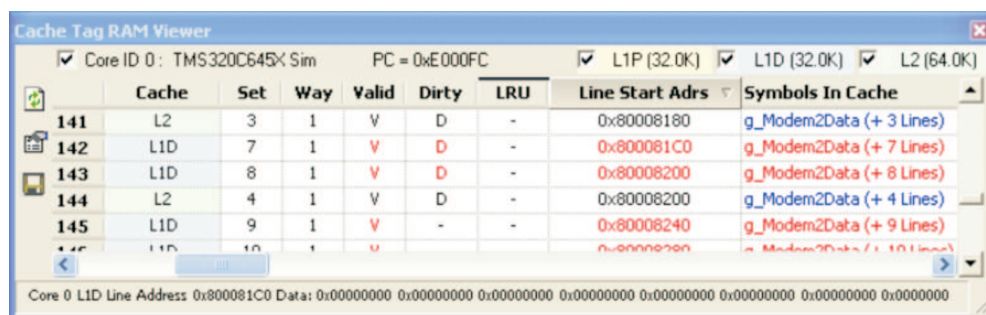


Figure 7. Cache Tag RAM Viewer. You can highlight individual columns to group or arrange Cache Tag RAM by a certain criteria.

7.2.1 Tracking and Viewing Cache Changes

By setting properties in the Property Manager window, you can observe changes in your cache tags. If you enable certain properties, the viewer will show only dirty or changed cache lines or memory differences. Another option lets you log changes to the Output Window.

The Cache Tag RAM viewer allows you to ensure that key functions and data structures are resident in cache and not being bumped from cache (either entirely or partially). Conflicts are indicated when address ranges or symbols associated with these functions are only partly displayed or when the cache set holding them contains different data. When you hover your mouse over a cache line, the viewer indicates whether a particular cache line contains the same symbol.

Line Start Adrs	Symbols In Cache
0x8000C340	g_Modem3Data (+ 13 Lines)
0x80004340	g_ModemData (+ 13 Lines)
0x8000C380	g_Modem3Data (+ 14 Lines)
0x80004380	g_ModemData (+ 14 Lines)
0x8000C3C0	g_Modem3Data (+ 15 Lines)
0x800043C0	g_ModemData (+ 15 Lines) Symbols were previously: g_Modem2Data (+ 14 Lines)
0x8000C400	g_Modem3Data (+ 16 Lines)
0x80004400	g_ModemData (+ 16 Lines)
0x8000C000	g_Modem3Data (0x8000C000)
0x8000C080	g_Modem3Data (+ 1 Line)

Figure 8. Cache Conflict Indicators. The tooltip indicates that the symbol on address 0x80004380 has changed, indicating a cache conflict.

8 Tool for Reporting Exceptions on C64x+ Devices

Exceptions are used to signal programs. For example, exceptions resulting from memory protection or security violations inform the developer that there are problems with memory maps or bad pointers (CPU or DMA).

If your device supports exceptions, you can configure exception reporting by selecting Debug→Exceptions. An option dialog will open, check the feature marked Enable Exception Handling on the bottom left to view and choose exception types. The Exception Reporting tool provides preconfigured reporting options (such as *Report Internal Exceptions*, *Report dropped CPU interrupts* and *Report Memory Protection Faults*). You can also customize which exceptions to report.

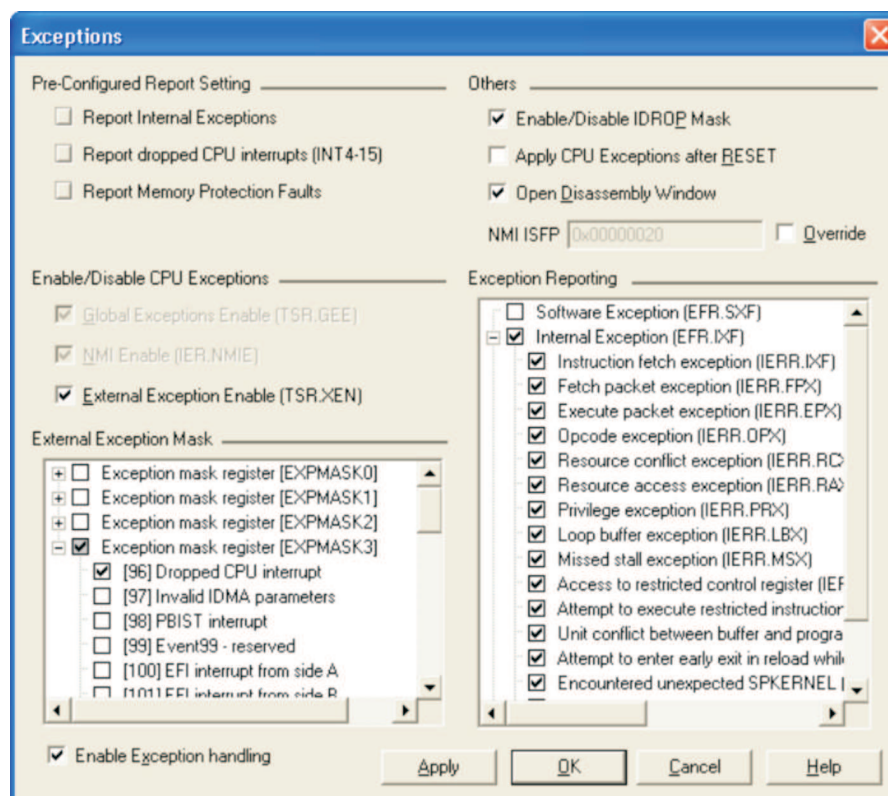


Figure 9. Exception Reporting Tool. This tool lets you select exceptions you wish to be reported to the Output Window. It comes with preconfigured reports.

If the tool reports an exception, the CPU will halt and display a message in the output window. At that point, you can view this information and decide whether to terminate the application (and reset the CPU) or let an exception handler deal with it and continue. Having access to these exception reports increases flexibility and potentially saves development time.

9 Adding Globals to the Watch Window

The Watch Window now allows you to watch file static and global variables. To use this option, you must load a program and open the Watch Window. Right click on the window and select the Add Globals to Watch option from the context menu. The Add Globals to Watch dialog has a list of all the file static or global variables currently defined for the loaded program. This dialog also shows the scope for each variable. Choose a variable by clicking in the checkbox and selecting the Add to Watch button. Select and add as many expressions as needed, and then click Close to return to the Watch window. The selected expressions will appear in the Watch tab. Scope information is shown as part of the variable name (example syntax: 'lowlev.c':init). If the loaded symbols change and a static expression added to the watch window is no longer defined, the Watch window will display an error message.

10 C6000 Functional Simulators Interrupt Latency Detection Feature

This tool allows you to measure the worst-case interrupt latency of the code, including programming interrupt constraints such as disabling GIE/NMIE, and architectural behavior such as non-serviceability of interrupts in branch delay slots.

Note: The feature is available on the C6000 functional simulators.

While programming, you may find that the algorithm has a larger interrupt latency than quoted. If so, you may not discover it until late in the development life cycle, and it may require a delay to correct. Characterization of the interrupt latency of the code is needed so that you can determine the real time latencies in the application. The application consists of multiple components that cannot be individually designed. The Interrupt Latency Detection (ILD) feature on the C6000 simulator provides you with a deterministic measure of the worst-case interrupt latency of the code. As a result, you can quote the interrupt latency of the code for a given set of test vectors.

To use the ILD feature, you must first load the ILD GEL file:

1. Launch CCStudio.
2. From the File menu, choose Load GEL.
3. Browse to the GEL file *InterruptLatencyDetector.gel*, in the directory C:\CCStudio_v3.3\cc\gel.
4. Click OK to load the GEL file.

This should add the following options to your GEL menu under GEL Interrupt Latency Detector: StartProfile, SetThreshold, and StopProfile. To begin profiling, you need to build and load your application, and create a Start Profile Point and End Profile Point for the .c file for your application. Then set the threshold latency by selecting Interrupt Latency Detector→SetThreshold from the GEL menu. Run the application until the Start Profile Point, then start profiling by selecting GEL→Interrupt Latency Detection→StartProfile. Run the application until the End Profile Point. Stop profiling by selecting GEL→Interrupt Latency Detection→StopProfile. Exit CCStudio, as the profile log is complete only upon exiting the simulator.

After running, the tool outputs an Interrupt profile log as an XML file with the following entries, corresponding to the intervals where the interrupts are disabled by the architecture:

- Start and end profile point
- Start and end CPU cycle

The XML file only captures the first 500 entries. It is generated in the same path and same name as the loaded COFF file for the application. For example, if the application to be profiled is c:\Apps\test\debug\app.out, the interrupt profile log is available in c:\Apps\test\debug\app.xml after profiling the application for interrupt latency. For a multi-core simulator, multiple XML files are generated, one per core. The file name has the core index to distinguish the files.

11 Displaying Software Pipeline Loops (SPLOOP) in the Disassembly Window

More recent targets like C64x+ support Software Pipelined Loop (SPLOOP), an algorithm for processing software loops more efficiently. The SPLOOP facility stores a single iteration of loop in a specialized hardware buffer and selectively overlays copies of the single iteration in a software pipeline to construct an optimized loop execution.

To view the SPLOOPS on an appropriate target, choose Project→Build Options, and then select the following build options: Opt Level: (-2 or higher) and Target Version: mv64+. SPLOOP is supported on C64x+ based devices. This IDE version lets you view SPLOOP in the disassembly window during execution. If you step through disassembly code using SPLOOPS, more than one cursor arrow may appear in the selection margin of the disassembly code ([Figure 10](#)). These arrows appear to go back and forth every time you step through your code. The multiple cursor arrows are a virtual representation of the loops executing in parallel. So within the scope of the SPLOOP, you may see 3 cursor arrows (if there are 3 loops) or 4 cursor arrows (if there are 4), etc.

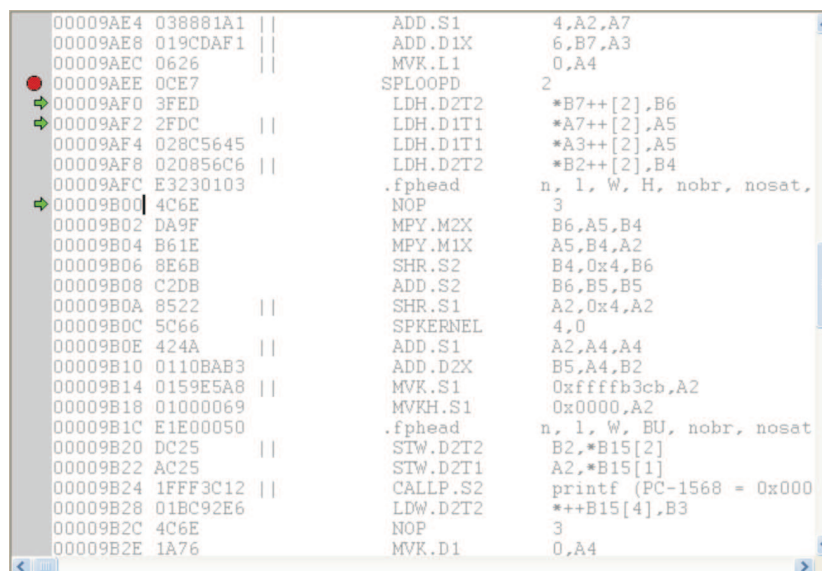


Figure 10. Software Pipelining Loop (SPLOOP) in the Disassembly Window. The SPLOOP begins at the breakpoint (depicted as a red circle). When loops are executing in parallel, multiple cursor arrows will appear to move.

12 MMU Page Table Viewer

The MMU Page Table Viewer tool is a new option for CCStudio on ARM 9 and ARM 11 (v5 MMU configuration only) devices that allows you to visualize how the Memory Management Unit (MMU) translates virtual memory to physical memory (see the Virtual and Physical Memory topic in the online help for more information on memory and MMUs). The MMU Page Table Viewer allows you to accurately visualize the address translation tables and configuration of the MMU. It decodes the two level page table hierarchy of the MMU, displaying the logical to physical memory mapping and the page frame attributes. The Page Table Viewer has a flexible XML driven display, which supports ARM 9 and ARM 11 MMU page tables. The Viewer display can be set to either automatically or manually update, and when sorted by logical address, only the visible page table entries are read from the target. However, the Viewer is currently only supported for emulation, and there is no visualization of cached Translation Lookaside Buffer (TLB) page table entries. Note: The MMU Page Table Viewer currently provides visibility into the ARM v5 MMU. It does not support the ARM v6 MMU.

13 Memory Load/Save Utility

Previous versions of CCStudio supported data file formats in either a COFF output file (.out) or a CCStudio data file (.dat). This tool allows you to support raw binary data transfers, and also allows for faster download speeds. This new option is available for ARM and C5000 targets. It has been optimized to take advantage of the newer emulation capabilities of the XDS560 to allow improvements of data download speeds (host to target), as well as additional improvements in speed regardless of emulator. There is also an option to perform byte swapping for every 16-bit word during the data transfer. You can use the Write file to memory tab to load a file, and the Save memory to file tab to save a file.

To load a file on the target, type the full path of the file, or use the Browse button to select it. Select an address and a page and click Write. The writing process can be canceled at any time by clicking Cancel. The plug-in also displays the progress of the operation. The destination address can be a physical address or the address of a symbol. The page combo is disabled for targets that have only one memory page. To save data from the target to a file type the full path of the file, or use the Browse button to select it. Select an address, page and size and click Save memory. The writing process can be canceled at any time by clicking Cancel. The plug-in also displays the progress of the operation.

14 New Debug Menu Items

The Debug menu has three new items:

14.1 Low Power Run

This is the same as a normal run, except that the target is not prevented from going into low power mode (sleep mode). If power is lost, the lower left corner of the CCStudio window will display a message. Losing power may cause breakpoints to be lost. After a halt, all breakpoints will be verified, and disabled if they are no longer present.

14.2 Advanced Resets

In addition to the default reset option (Reset CPU), you may have a choice of alternate resets depending on your target. These are defined differently on each target; your target documentation will describe any new resets that are available through this menu item.

14.3 Halt on Reset

When selected, the target will halt after a reset (inside the reset vector). Otherwise, the target will run. Some user-initiated resets will not necessarily run the target immediately, this is more for an external reset or when running and executing a reset. This value is undetermined until the target is connected. If you have hardwired this state in the hardware after a power up, the value will not be correct until it is connected. You can force a change on the target even when it is not connected.

15 Troubleshooting Emulator Errors

We have added a new section to the online help under Customer Support called Troubleshooting Emulator Errors. This section describes some of the emulation related error messages that can be generated by CCStudio, as well as some possible options to avoid or resolve the issue. The list is ordered by the error number and contains errors generated at the GTI, OTI, and PTI levels. Scan level errors are not included. Clicking on an error will take you to the description and possible resolution for that error.

16 IDE Changes

Note that support for CodeSizeTune, Peripheral Context Display (PCD), and the CSL graphic user interface has been removed in this version of CCStudio. For more information about CSL or to find out if CSL is supported by your device, check your device product folder on www.ti.com, or contact the Product Information Center (<http://www-k.ext.ti.com/sc/technical-support/product-information-centers.htm>).

Also, the OMAP System DMA and Interrupt Controller registers have been moved from the Tools menu to the View→Registers menu option for OMAP processors.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		Applications	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
Low Power Wireless	www.ti.com/lpw	Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265

Copyright © 2006, Texas Instruments Incorporated