Application Report
SPRAA07 - May 2004

What's New in Code Composer Studio Development Tools v 3.0

Steve White and Robert Nagle — *Code Composer Studio, Applications Engineering*

ABSTRACT

This application report highlights new features and functionality in Code Composer Studio™ Integrated Development Environment (IDE) v3.0. It describes enhancements, bug fixes and changes in device support from the previous version. Version 3.0 supports the C6000™ platform only; future versions will support the C2000™, C5000™ and OMAP™ platforms.

Requirements:

- Windows 2000 or Windows XP, with Internet Explorer 5.5 or later
- 600 Mhz processor, 128MB of RAM, 500MB of hard drive space
- PC display must have a minimum of 16 bit color.

Contents

List of Figures

1 Introduction

Code Composer Studio Integrated Development Environment (IDE) v3.0 contains a new feature, Application Code Tuning (ACT), for optimizing applications and algorithms. It also includes: a more powerful Parallel Debug Manager, enhancements to the IDE, the ability to dynamically connect and disconnect from targets (*Connect/Disconnect*), additional configuration options for the Memory Window, and changes in the default symbolic debug format from COFF/STABS to COFF/DWARF. In this release, the only devices supported will be C6000. However, succeeding versions will support C2000, C5000 and OMAP.

2 Installation Changes

The default install directory for the IDE is **C:\CCStudio**. It will no longer be necessary to install a separate version of the IDE for each family of devices.

3 Changes in the User Interface

Here is a short listing of changes to the GUI:

- Right-clicking from the source code window or the disassembly window allows the user to set or disable various kinds of breakpoints or probe points (i.e., hardware, software, conditional). The Selection Margin now uses icons of different shapes and colors to indicate the type of breakpoint/probe point and whether they are currently enabled.
- The Memory Windows Options dialog contains additional options and is accessible as a right-click option from several new places (see section 7).
- The addition of Application Code Tuning (ACT) offers a choice of two layouts (Standard and Tuning) and new menu options under the main Profile menu. Some options formerly available under the old Profiler menu are now listed under this new Profile menu.
- Options have been changed or added under the Customize » Options dialog tabs. Several of them have to do with default startup behavior for the IDE (i.e., for target connection or file reloading). For example, by default, the IDE's default behavior *no longer connects automatically* to the target upon startup. Other changes to the Options tab reflect changes to IDE components mentioned elsewhere in this document.
- The way that GEL files are listed in the Project View has changed. The 3.0 version of the IDE displays GEL files from CCS Setup with a yellow icon and normal GEL files with a white icon. If a GEL file fails to load, it will not be listed at all in the GEL folder.
- The Symbol Browser is now listed as a menu item under the View menu and includes several changes (see section 8).

4 Application Code Tuning (ACT)

4.1 What It Does

The Code Composer Studio IDE now offers a series of tools (called Tuning Tools) to help developers to optimize (or tune) applications. These tools allow users to set goals and track progress towards goals by collecting different sets of data. A new Advice Window provides on-the-fly recommendations about how to make code meet user goals.

The IDE has two Layouts, (*tuning layout* and *standard layout*), which can be toggled from the main toolbar. Standard layout is the default, while tuning layout (represented by a tuning fork icon on the toolbar) opens advice windows on the left side. The tuning layout focuses attention on the optimization needs of the user's program. The advice window walks the user through the optimization/tuning process specific to the DSP device and a selected goal.

Changing the layout does *not* require having to close or save your work. When you switch layouts, the Control Window will momentarily disappear and reappear in the new layout. Layouts mainly affect the way windows are displayed in the Code Composer Studio IDE.

4.2 Tuning Dashboard

The Tuning Dashboard panel comes up on the left when the user switches to Tuning Layout (by selecting the Tuning Fork icon on the toolbar). This panel provides advice and allows the user to launch tuning tools (i.e., Advice Window, Goals Window, Profile Setup, and Profile Viewer). Tuning Tools can also be launched from the Profile menu.

4.3 Compiler Consultant

After analyzing source code and compiler build options, the Compiler Consultant recommends ways to improve code performance. The tool displays two types of information: Compile Time Loop Information (created by the Compiler) and Run Time Loop Information (gathered by profiling the application). Each time the application is compiled, Compiler Consultant will analyze the code and create suggestions for different optimization techniques to improve code efficiency. These suggestions include compiler optimization switches and pragmas to insert into the code that give the compiler more application information.

4.4 Cache Tune

By graphically representing cache memory accesses, CacheTune makes it easier to identify non-optimal cache usage. This display of cache hit/miss data as a two-dimensional graph can be utilized to improve performance in cache-based memory devices. All memory accesses are color-coded by type. Various filters, panning, and zoom features facilitate quick drill-down to view specific areas. This visual/temporal view of cache accesses enables quick identification of problem areas, such as areas related to conflict, capacity, or compulsory misses. All of these features help the user to improve an application's overall cache efficiency.

4.5 Code Size Tune (CST)

CodeSizeTune (CST) replaces the Profile-based Compilation (PBC) tool from version 2.2 of the IDE. CST analyzes tradeoffs between code size and cycle count more quickly and easily for an application. Using a variety of profiling configurations, CodeSizeTune profiles the user's application, collects data on individual functions, and determines the best combinations of compiler options. CST graphs function-specific options sets (see Figure 1), permitting a choice of the configuration best suited for the user's needs. (For more information, see section 10).

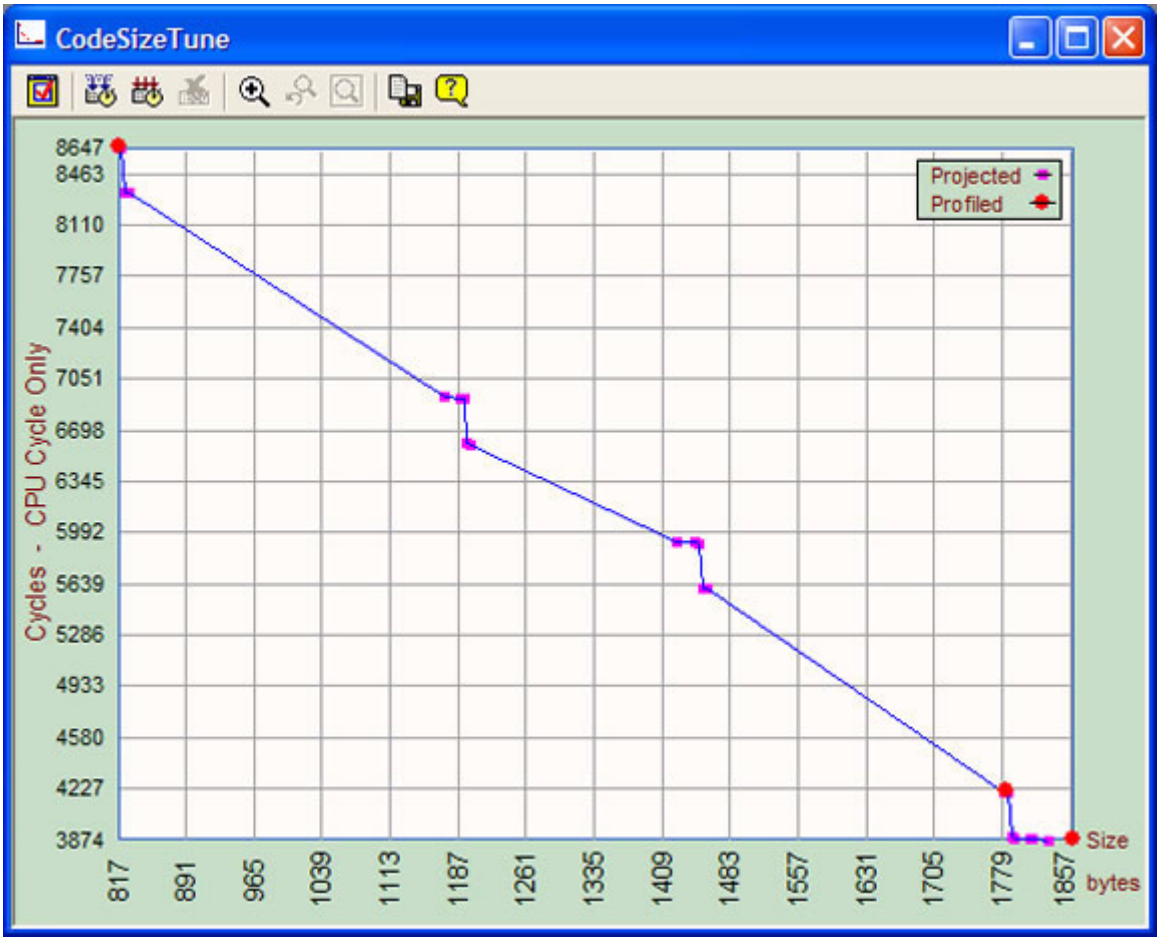

Figure 1. Using CodeSizeTune to Profile Data

5 Parallel Debug Manager (PDM)

The Parallel Debug Manager (PDM) has several changes from earlier versions.

- Users can connect or disconnect from targets “on the fly” by right-clicking the processor on the right pane of PDM (see Figure 2).
- The interface allows an expanded view of processors, with several drop-down filters to reveal a list by group, CPU, or board.
- Red highlighting on the processor icon (on the left pane) indicates that the processor is not connected to the system or that it has updated status information.
- The User can now put processors into **loosely-coupled groups**, (i.e., where the processors are not all on the same physical scan chain). Choosing Group View from the second drop-down menu and System on PDM’s left pane shows which groups are synchronous and which are not.

Global breakpoints work *only* when processors in a group belong to the same physical scan chain.

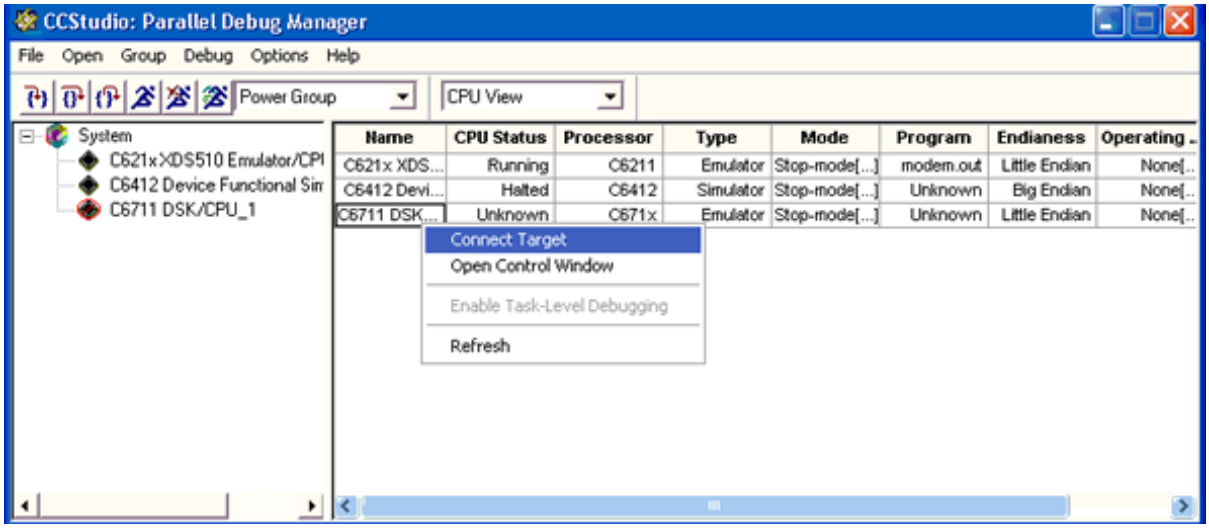

Figure 2. New Interface for Parallel Debug Manager (PDM)

6 Connect/Disconnect

The Code Composer Studio IDE now makes it easier to connect and disconnect with the target dynamically. New functionality (Connect/Disconnect) allows you to disconnect from your hardware target and even to restore the previous debug state when connecting again.

- The Status Bar will briefly flash a help bubble to indicate changes in the target's status. When the target is disconnected, the status bar will indicate this fact, as well as the last known execution state of the target (i.e., halted, running, free running or error condition). When connected, the Status Bar will also indicate if the target is stepping (into, over or out), and the type of breakpoint that caused the halt (software, hardware).
- After a connection to the target (except for the first connection), a menu option entitled "Restore Debug State" will be available under the Debug Menu. Selecting this option will enable every breakpoint that was disabled at disconnect. You can also reset them normally (by pressing F9 or selecting Toggle Breakpoints from the right-click menu). Breakpoints from cTools jobs and emu analysis will not be enabled.
- If you have the PDM open, you can connect to a target by right-clicking on the cell corresponding to the target device underneath the Name column.

7 Memory Window Options

The Memory Window Options dialog contains several new features. To solve cache coherency problems, the user can bypass any cached memory value by selecting Bypass Cache. Another option, Highlight Cache Differences, highlights the memory value on any memory portion that has been previously cached. This option is only supported on hardware devices.

A new option, Track Expression, allows you to view expressions and their evaluated values. (Previous versions automatically replaced an expression with its evaluated value after the user's mouse left the field). You can also edit the original expression. If the evaluated value changes, the expression will be removed, and you will see only the evaluated address value.

This dialog is now accessible as a right-click menu option for several of the windows (including the Watch Window, the Register Windows and View»Memory).

8 Symbol Browser Changes

The Symbol Browser menu item now is listed under the View menu. By right-clicking from any tab, the user can use the Find Symbol function to locate matching strings (highlighted in red). Every tab (except the type tab) now contains an Address column. Double-clicking on the function name will open up the source code containing the function. Double-clicking on the address (in either the Globals or Labels tab) will open up the disassembly window at the given address.

9 Code Generation Tool Changes

Please check the codegen release notes for details on new features and fixed bugs.

The default symbolic debug format is now **COFF/DWARF format** instead of COFF/STABS format. There are now three options for debug information:

- The **-g** option generates DWARF information.
- The **-gn** option disables the generation of all symbolic debugging information.
- The **-gt** option generates the legacy COFF/STABS format.

Performance improvements for control code. The *long long* native data type is now supported. New atomic intrinsics have been added for enabling and disabling interrupts.

New command line utilities. The *object file display utility* prints the contents of object files (.obj), executable files (.out), and/or archive libraries (.lib) in both human-readable and XML formats. The *strip utility* removes symbol table and debugging information from object and executable files. The *name utility* prints the list of names defined and referenced in a COFF object (.obj) or an executable file (.out). The *hex conversion utility* can create boot tables.

New Linker Capabilities. The linker now creates copy tables and supports section splitting. The linker is now capable of outputting information in XML format. The ***-priority*** linker option now provides an alternate search mechanism for libraries. Specifying this option causes each unresolved reference to be satisfied by the first library that contains a definition for that symbol.

The linker ***--args=size*** option instructs the linker to allocate memory on the target so that the loader can use that memory to store all the contents on the argv array and the argc variable.

The assembler now provides enhanced symbolic debugging information through the use of .asmfunc/.endfunc directives.

Usability Improvements for Linear Assembly. Support has been added for circular addressing, user-controlled register allocation, volatile loads and stores, and partitioning information associated with registers rather than instructions.

10 Functionality That has been Removed or Replaced

Visual Linker and Code Maestro have been removed from the v.3.0 release of Code Composer Studio IDE.

Profile-Based Configuration (PBC) has undergone substantial changes. Its functionality has been incorporated into CodeSizeTune (CST), which is one of the tuning tools of Application Code Tuning (see section 4). The old Assistant Window is now in the Advice Window and contains more step-by-step information about how to use the component. Functions formerly available in this old Assistant Window are now launched from toolbar icons in the ACT Advice Window. For example, you can now configure profile setup using ACT's Profile Setup tool. CST includes more informative messages about errors and status; it also allows you to save CST projected points not only on the graph, but also to different configurations. Because compilers and simulators have improved considerably, profile results of code size and cycle counts may differ from results produced by older versions of Code Composer Studio IDE.

11 Simulation Changes

Naming conventions have changed for simulators.

11.1 Changes in Device Support

- C6412 and DM642 device cycle accurate simulators and device functional simulators are added to this release.
- C64xx Cache Simulator configuration is *not* supported for this release.
- Simulator configuration name changes. The simulator configuration names available in the Import Configurations of Code Composer IDE Setup are renamed to better reflect the hardware modeled and the cycle accuracy aspect. The simulator names now explicitly indicate whether it is is a CPU only simulator or a device simulator. For example, C6412 Functional Simulator now becomes C6412 **Device** Functional Simulator. Also, the string *Cycle Accurate* is added to specific configuration names to indicate the cycle accuracy of such configurations. For example, C6412 Device Simulator now becomes C6412 Device **Cycle Accurate** Simulator.

11.2 Resource Conflict Detection Support

- The C6000 CPUs have four functional units on each of the two sides, A and B. These units offer enhanced parallelism and allow for execution of up to eight parallel instructions. These units, along with the Data Access Paths and CPU registers, primarily constitute the resources of a C6000 CPU. These resources have various constraints on their simultaneous use by different instructions. If the constraints are violated by the executing code, application behavior is not guaranteed.
- The C6000 simulators now detect and report such conflicts by default. Resource conflict detection in the simulators is crucial because many of these violations manifest only at runtime, making it impossible to detect them at compile time.

11.3 New Event Names

Names for analysis events are changed in this release. You can view a complete list of these events by selecting Debugging » Analysis Tools for Debugging » Simulator Analysis from the online help's Table of Contents.

12 Bug Fixes

You can view debug fixes and known problems by viewing the Release Notes. To do this, select Help » Contents from the main menu and choose the link titled Release Notes.

13 Documentation Changes

Topics in the Main Help have been reorganized to match the software developer's development cycle more closely. There are new Instruction Set Technical Overviews for all platforms, tutorial updates and several new multimedia tutorials, including ones for the Application Code Tuning (ACT) Components. The online help now includes a link to the online DSP knowledgebase.

Also, the Application Code Tuning component uses "advice windows" to give real-time adaptive advice and information about the user's program or project.