Abstract

Designed for the Texas Instruments (TI™) high performance TMS320C6000 digital signal processor (DSP) and power efficient TMS320C5000 DSP platforms, Code Composer Studio™ is a development environment that tightly integrates the capabilities of the following components within an extendible plug-in architecture:

- Integrated development environment (Code Composer) with editor, debugger, project manager, profiler, Probe Points, and more
- C Compiler, Assembly Optimizer and Linker (Code Generation Tools)
- Instruction Set Simulator
- Real-Time Foundational Software (DSP/BIOS™)
- Real-Time Data Exchange Between Host and Target (RTDX™)
- Real-Time Analysis and Data Visualization

Code Composer Studio integrates all host and target tools in a unified environment. It also simplifies DSP system configuration and application design to help designers get started faster than ever before. Code Composer Studio initially will be available for Windows 95 and NT4, with Windows 98 and Unix support to follow.

Contents

Introduction ................................................................................................................... .................................. 2
Code Composer Studio Meets the Needs of Today’s Complex DSP Applications................................. 3
  Open Plug-In Architecture ............................................................................................. 4
  Data Visualization ......................................................................................................... 6
  Real-Time DSP Environment ......................................................................................... 8
  Real-Time Data Exchange (RTDX) .................................................................................. 10
  Real-Time Analysis ....................................................................................................... 11
  Integrated Development Environment (IDE) ............................................................... 13
  Integrated Source Code Editor ..................................................................................... 13
  Visual Project Manager ................................................................................................. 13
  Integrated Debugger .................................................................................................... 14
  Interactive Profiler ........................................................................................................ 14
  Compile and Build Efficiently ....................................................................................... 15
  Inject and Extract Data Signals ..................................................................................... 17
  Multi-Target Debug ...................................................................................................... 18
  Customize and Automate Test ....................................................................................... 19
Summary ........................................................................................................................ 21
Introduction

With the emergence of new products that feature wireless digital communication, speech recognition, multimedia, and Internet telephony, designers are relying on DSPs to provide the computing horsepower for crucial real-time performance. Programmable DSPs provide today’s software engineer with the tools to reduce time to market while providing an optimized solution to the application challenge.

Effective software is required to fully harness the power of DSPs. In fact, software is arguably the most critical component of the product’s design. According to industry estimates, approximately 80% of the effort in the development of an embedded system and 80% of the system complexity lie in the software.

In spite of such findings, designers often overlook the availability of sophisticated and easy-to-use development tools during the selection of a processor for a real-time system. Such tools can have a large impact on meeting time-to-market requirements. A development system that offers its users an expandable array of tools working seamlessly together empowers the developer to focus on innovation, product differentiation, and time to market. Conversely, a development system that lacks cohesion, with a set of disparate tools that force the user to awkwardly deal with different and inconsistent user interfaces, can seriously impede a developer’s progress.

Historically, the DSP tools available were limited to either facilitating code generation (compilers, assemblers, and linkers) or debugging and analyzing sections of code (source code debuggers, and profilers). In most cases these tools are distinct with no automatic sharing of data, requiring the developer to constantly switch between different applications.
Debugging real-time applications is an equally frustrating experience. To understand their application's behavior under real-world conditions, developers want to be able to trace the flow of a program without halting execution. However, the typical debugging environment only allows for static debugging of code by single stepping between breakpoints. These debuggers are unable to provide information on the sequence or interaction of events that have occurred prior to the current point in the code (except by using printf statements, which changes the nature of the program and possibly obscures the real problem). Logic analyzers can also be used to debug code.

When multiple sections of real-time code are integrated, a new class of problems arises, caused by the complex interactions between sections of code. If developers do not have tools to assist them in determining whether the application design meets real-time constraints, problems can emerge, causing delays to the scheduled date of completion. If not caught, these problems can arise after a product has been shipped, requiring companies to fix the problem in the field at great cost.

It is clear, then, that as the popularity of DSPs continues to grow, the challenge before the industry is to provide tools that developers will want to use. The DSP development environment must provide the same level of sophistication and usefulness as mainstream microprocessor-centered tools. In addition, the toolset must go beyond code generation and basic debugging and profiling and offer tools specifically for DSP. Graphing windows that allow developers to view signal data in a more intuitive way, the ability to trace program flow without halting the processor, and real-time analysis to uncover subtle timing problems are as necessary to a DSP development system as an editor or compiler.

Furthermore, as designers begin to add more features to their products to keep step with or to leap ahead of their competitors, the complexity of their DSP applications will only increase. Efficient project management and an extensible architecture will ensure that development time isn’t needlessly spent manipulating source files and that the environment will grow to meet increasingly higher demands.

These issues represent the beginning of a paradigm shift away from the old days of DSP developers being forced to adapt to their development systems. Now developers will be able to customize their environment to the way they work and add tools as needed without having to tediously switch between standalone applications. Now tools will finally work together seamlessly to empower the developer.

**Code Composer Studio Meets the Needs of Today’s Complex DSP Applications**

Designed for the Texas Instruments high-performance 'C6000 and power efficient 'C5000 platforms, Code Composer Studio is a development environment that tightly integrates the capabilities of the following components, all within an extendible plug-in architecture:

- Integrated Development Environment (Code Composer) With Editor, Debugger, Project Manager, Profiler, Probe Points, and More
- C Compiler, Assembly Optimizer and Linker (Code Generation Tools)
- Instruction Set Simulator
- Real-Time Foundational Software (DSP/BIOS)
- Real-Time Data Exchange Between Host and Target (RTDX)
Real-Time Analysis and Data Visualization

Code Composer Studio integrates all host and target tools in a unified environment. It also simplifies DSP system configuration and application design to help designers get started faster than ever before. Code Composer Studio will initially be available for Windows 95 and NT4, with Windows 98 and Unix support to follow.

Figure 1. Code Composer Studio Saves Time at Each Phase of the Code Development Cycle

Code Composer Studio provides the unique features to cover every stage of the DSP code development cycle.

Open Plug-In Architecture

Today’s highly complex system designs often require hardware and software tools with more capabilities than those previously available. Code Composer Studio’s open plug-in architecture allows developers to easily extend the IDE to match their growing needs. By leveraging DSP’s most extensive third-party network, designers can select third-party tools to plug into the Code Composer Studio development environment. Not only is this a better alternative to buying separate applications to meet new requirements, It also helps reduce the risk to developers. By allocating fewer resources to developing individual utilities, developers can spend more time testing the end product, resulting in increasing an application’s robustness.
Some examples of possible plug-ins include block diagram code generation systems, hardware configuration and diagnostics tools, algorithm libraries, and filter design packages. Data is shared between all tools, increasing reliability and saving time by requiring fewer manual operations between tools. The common look and feel of plug-ins provides a familiar environment to developers and shortens the learning curve. Therefore, developers can focus their innovation on creating new applications and adding value for their customers.
Data Visualization

The ability to view data and signals in their native format greatly facilitates interpretation, analysis, and testing. This requires a flexible visualization tool with many DSP applications display types, such as eye diagrams, constellation plots, and FFTs (Fast Fourier Transforms). Developers need a way of changing variables to immediately see the effect on their signals. The visualization tool should be able to view these signals while the application is running.
Figure 4. Speed Up Analysis and Testing by Viewing Signals in Their Native Format With Built-In Application-Specific Graphs

Code Composer Studio’s tightly integrated data visualization allows data and signals to be viewed in their native format for easy interpretation and analysis with many display types. As the program runs, a snapshot of the signal is connected to a data visualization window allowing the signal to be seen as it progresses through an algorithm during debugging. Variables can be changed on the fly and their effects seen almost immediately, without requiring any additional code. This method is more efficient than those employed by other development environments where developers have to modify their code, extract the data, open up a new visualization application, view the data, change a variable, recompile, run, extract the data, and then run the visualization application again. Code Composer Studio’s integrated data visualization is a tremendous time saver; it is more efficient and more robust.

Code Composer Studio provides many built-in graphing windows, including those frequently used in communications, imaging, and general DSP systems. Eye diagrams allow the developer to see variations in the signal caused by noise and time jitter. Constellation plots allow a programmer to evaluate the algorithm’s ability to extract meaning from a signal. An FFT waterfall consists of a series of spectral snapshots that show how the spectral content of a signal changes over time. This is very useful for measuring noise characteristics over time for data modems or developing speech recognition systems.
Real-Time DSP Environment

Foundational Software

Today’s new systems require a minimum set of services at the target level that enable scheduling, communications, resource control, and analysis. These services are critical to developers and they will either purchase a real-time operating system (RTOS), kernel library, or create their own. Custom kernels, though small, are usually expensive to build and difficult to maintain or port to other targets.

TI’s DSP/BIOS run-time foundation software, included with Code Composer Studio, provides optimized basic run-time services, such as low-latency threading and I/O, to programs executing on the DSP target hardware. It also provides Real-Time Analysis facilities, described below, that supply unprecedented visibility into the target DSP application behavior without using breakpoints. The DSP/BIOS run-time system occupies less than two k-words of target memory and consumes less than 1 MIPS (million instructions per second) of processing power with all instrumentation enabled. The DSP/BIOS is packaged as a library of relocateable, re-entrant object modules (subroutines). Only those DSP/BIOS modules that are actually used in the application are linked into the application. The DSP/BIOS run-time system’s small size makes it practical to embed directly into a DSP’s on-chip boot ROM.
The DSP/BIOS scheduling paradigm is a static, pre-emptive, priority-based, subroutine model. A single stack is used for all threads and registers, and memory resources are kept to a minimum. This static scheduling and resource control allows extensive pre-compile configuration of system objects. Static configuration allows low overhead intrinsic diagnostics within the target application.

Code Composer Studio provides a graphical host-based static configuration tool. The graphical tool makes it easy for developers to control a wide range of parameters accessed by the DSP/BIOS run-time system. All DSP/BIOS objects are statically configured and initialized using this tool. Because all DSP/BIOS services are initialized at compile time, the target application is not required to perform run-time creation of DSP/BIOS services. This shrinks the run-time memory footprint significantly because no dynamic service creation code is required on the target DSP.
Real-Time Data Exchange (RTDX)

Data visualization tools, such as those described above that are included with Code Composer Studio, are excellent for proving the correctness of DSP algorithms. Once algorithms are integrated into applications, the real-time behavior of the system must be observed. Real-Time Data Exchange, RTDX for short, provides significant benefits over alternative methods of system debugging. Until recently, developers were forced to stop their application to exchange data with the host computer. This intrusive method of debugging may yield misleading information because the isolated snapshot of a halted high-speed application does not present an accurate view of the system’s real-world operation.

RTDX gives designers the industry’s first DSP system that provides real-time, continuous visibility into the way target applications operate in the real world. RTDX allows developers to transfer data between the host computer and DSP devices without stopping their target application. This important breakthrough shortens development time by giving developers a much more realistic representation of the way their systems operate. Just as modern medical diagnostic equipment provides a real-time, ongoing analysis of the way a patient’s body is functioning, RTDX allows designers to continually monitor their systems and gain real-time insight into their running applications.
Figure 6. Connect Host and Target Applications Together and Exchange Data in Real Time to Gain Insight Into How the Application is Behaving

With RTDX hard-disk drive, designers can test their applications without halting the drive with improper signals to the servomotor, and engine control designers can analyze changing environmental conditions such as temperature while the application is running.

Real-Time Analysis

To analyze the execution of real-time event-driven DSP applications, developers must be able to view both the data output and the timing relationships between tasks in the system. RTDX provides the pathway to transfer the data to provide host-side visual display of real-time execution information. DSP/BIOS provides a library of high-level analysis routines and host-side graphical visualization to ease the use of this unique debugging resource. The DSP/BIOS run-time itself is implicitly instrumented using this library to give deep visibility into the task-level behavior of an application.
The foundations of DSP/BIOS’s real-time analysis library are the Log and Statistics modules. The Log module provides a “real-time printf” capability. The text strings are stored and formatted on the host, so the target need only transfer data to the host. This requires tens of cycles, as compared to hundreds or more for target-side formatting of printf strings. The Statistics module records the maximum, minimum, and average of a target-side value. All calculations are done on the host, so only a small amount of information must be recorded and transferred from the target. The Statistics module is ideal for tracking the performance parameters of a DSP system.

Building on the Log module, DSP/BIOS integrates real-time instrumentation directly into the run-time scheduler. Log messages from the scheduler are displayed in a graphical strip-chart, much like a software logic-analyzer. The strip chart shows the execution sequence of software interrupts and other threads. This execution view immediately allows you to pinpoint program errors that occur only through a sequence of real-time events in your system. Using breakpoints to debug such problems significantly changes the timing of your system, masking timing errors.

DSP/BIOS uses RTDX to transfer this information in real time, letting the real-time behavior of the system be seen. This instrumentation is implicit and user selected—all you do is select it in the graphical configuration tool as a compile time option. Less than one MIPS of processing power is used with all DSP/BIOS instrumentation enabled.

The run-time memory requirement for the DSP/BIOS Real-Time Analysis features is user controlled during configuration. This means that it can be left enabled in deployed applications. Field service technicians with modest equipment can then bring detailed failure analysis back to software maintenance engineers. It can also be used in the test lab for quality assurance with only minimal impact on overall resource usage.

**Figure 7. Quickly Find and Fix Subtle Real-Time Problems by Visualizing System Performance and Timing**

![Diagram showing real-time analysis and instrumentation](image)

**Integrated Development Environment (IDE)**

Code Composer Studio is a fully integrated development environment that provides access to all its features and toolsets from a single easy-to-use interface.
Figure 8. Save Time and Increase Productivity by Using a Single Control Panel to Access All Development Tools: Editor, Debugger, Project Manager, Real-Time Analysis, Compiler, Graphical Signal Analysis, and More

Integrated Source Code Editor

A full-fledged source code editor tuned for writing C and DSP assembly code offers developers a multitude of ways to be productive. Color syntax highlighting for C, assembly, and scripts allows users to instantly spot errors in keywords or missing comment delimiters. Floating toolbars support advanced operations, such as finding the next matching brace and indenting text. The search and replace function makes it easy to change variable names. To provide users with the most customizable working environment possible, edit windows support either docking to or floating outside the parent window in any configuration. Fully integrated with other facilities in Code Composer Studio, such as the debugger, the editor allows developers to easily edit code and see both source and disassembly at the same time.

Visual Project Manager

Code Composer Studio’s visual project management system provides a fast way of visualizing, accessing, and manipulating all project files from the same window. Files are organized into functional categories, such as source files, include files, libraries, configuration files, and script files. Files can be added to the DSP project simply by dragging and dropping files from the file manager. All of the features of the project manager are designed with the express purpose of saving time and making development easier.
Integrated Debugger

Code Composer Studio’s integrated debugger has DSP-specific capabilities and advanced breakpoints to simplify development. Conditional and hardware breakpoints can be based on C-expressions, local variables, or CPU register symbols. A script file can be executed when a particular breakpoint hits. Script files are easily created using the GEL scripting language. GEL uses a ‘C-like’ syntax, making it easy to learn, and provides powerful access to Code Composer Studio features, target variables, and memory. Global breakpoints are also available for multiprocessor systems. Because the address is determined at build time, setting breakpoints by source line number saves time. Breakpoints can be configured using dialog boxes, making it possible to activate groups of breakpoints with a click of a button.

Developers can launch multiple data windows, which are capable of displaying symbolic information. Code Composer Studio allows viewing interleaved source and disassembly. Programs can be debugged quickly by selecting the option to step into, over, or out of C-functions and assembly sub-routines.

The advanced watch window makes it easy to add variables directly from the edit window. Developers can easily “drill down” through complex structures and arrays by expanding and collapsing them, and even edit variables. GEL script functions can be added to the watch window, allowing execution at every breakpoint. Highlighting enables changes in variables to be easily and immediately seen. The quick watch window displays the value of a variable in a C or assembly source window instantaneously. A variable can easily be added to the watch window using the mouse. Four watch windows are available to help organize data.

Interactive Profiler

Developers must be able to evaluate code execution times to develop an efficient, high performance system. Code Composer Studio’s interactive profiler makes it easy to quickly measure code performance and ensure the efficient use of the DSP target’s resources during a debugging and development session.

Profile points accumulate hits and collect statistics on the number of instruction cycles executed or other events that have elapsed since the previous profile point was hit. This allows high-usage areas of code to be targeted during optimization, helping developers produce finely tuned code. Profile information can be gathered in one section of code while another is being debugged.

Profiling is available for blocks of assembly or C code in any combination and for other processor events, such as the number of branches, sub-routine calls, or interrupts taken. To increase productivity, all profiling facilities are available throughout the development cycle.
Compile and Build Efficiently

Although writing assembly code is very common for DSPs, writing software in C helps to reduce development time, especially if developers are unfamiliar with the device architecture. Therefore, it is especially important to use a compiler optimized for DSP rather than general-purpose processors. This minimizes the code size, thereby reducing the memory required and the system cost. Unlike other compilers that rely on public domain (GNU) derived technology, Code Composer Studio's compiler takes advantage of limited registers and makes tight DSP-specific loops. With ten years experience and several patents, the robust and reliable compiler and optimizer technologies allow developers to spend less time hand coding and more time concentrating on delivering new applications.

Texas Instruments is committed to continually improving the compiler. For example, the version 3 compiler for the 'C6000 produces code that is up to 20% smaller and 20% faster than code produced by version 2.

Some general optimizations that are incorporated into the compiler are:

- Branch optimization/control-flow simplification
- Alias disambiguation
- Copy propagation
- Common sub-expression elimination
- Redundant assignment elimination
- Loop induction variable optimization/strength reduction
- Loop rotation
- Loop invariant code motion
- Inline expansion of function calls
- File level optimizations
- Data flow optimizations
- Expression simplification
- Register variables
- Register tracking/targeting
- Cost-based register allocation

The 'C6000 compiler is designed in conjunction with the 'C6000 VelociT™ Very Long Instruction Word (VLIW) architecture, giving developers the most sophisticated tools available. Often, time-critical applications can be coded entirely in portable, standard C!

Unique to the DSP market, Code Composer Studio includes the only compiler that optimizes code at the program level, which is often critical to performance. Code Composer Studio’s compiler contains many 'C6000-specific optimizations, such as: software pipelining, conversion/predicate execution, memory address cloning, and memory address dependence elimination.
In addition to using C, 'C6000 developers can organize assembly instructions to execute in the least number of cycles with the world’s first assembly optimizer. Code optimization and parallelization is critical to achieving peak performance on highly parallel architectures such as the 'C6000. Code Composer Studio’s assembly optimizer gives developers more performance per cycle. As shown in Figure 9, instructions that would normally take 12 cycles to complete can be executed in only 3.

With Code Composer Studio’s extensive context-sensitive on-line help, it is no longer necessary to spend valuable development time searching through manuals. Selecting the keyword of interest in the edit window and pressing F1 takes you directly to the help topic. The help system contains comprehensive information on the DSP target, including register definition and instruction summaries.

All build options can be set with easy-to-use dialogs for the compiler, assembler, and linker, thus eliminating the need to remember cryptic commands. The build options are saved with the project for easy retrieval. All compiler and assembly tools can be launched quickly from the toolbar. Developers can quickly resolve compile errors by double-clicking on the error. This saves time by opening the source file containing the error at the appropriate location, allowing the line in question to be fixed immediately. Batch builds can be accomplished using GEL’s built-in functions to build multiple projects faster and automatically.
**Inject and Extract Data Signals**

Developers have traditionally had to embed test code in an algorithm to store data in a file. They would then launch a graphing application and view the data within that standalone application. For required changes, they returned to their debugging session, made the appropriate algorithm edits, recompiled and ran the program, and repeated the process of saving data for later viewing and analysis. Code Composer Studio seamlessly integrates such graphing facilities within the IDE, removing the need to embed any test code in the algorithm.

The Probe Point, a unique and powerful feature in Code Composer Studio, is a sophisticated form of breakpoint. It allows developers to define a point in the algorithm where they can perform an oscilloscope-type function. Unlike a breakpoint, program execution resumes after hitting a Probe Point. Many activities can be performed when a Probe Point is reached, such as injecting or extracting signal data, updating a graphical plot, or executing a GEL script. Probe Points can be easily set, removed, or moved to other areas in the algorithm simply by clicking on the icon in the toolbar. Recompiling or writing additional code is not necessary.

*Figure 10. Use Probe Points to Inject and Extract Data Signals, Execute Scripts, and View Data in Different Formats, All Without Modifying Source Code*

Code Composer Studio makes testing easy with the many options available for file I/O. Numerous files can be connected at once, making it easy to simulate conditions.
Multi-Target Debug

Code Composer Studio now supports the development of complex systems with multiple boards. Code Composer Studio’s Parallel Debug Manager (PDM) provides synchronized control over multiple processors that reside on a single board. It can be used to launch individual parent windows to control each processor.

Figure 11. Debug and Control Multiple Targets in Complex Systems

Save debug time by defining up to 32 different groups of CPUs for specific functions via intuitive dialog boxes. The Parallel Debug Manager can be used to broadcast commands to different groups of CPUs in the JTAG scan path. A global breakpoint command on one processor can halt other processors when this breakpoint is encountered. The Parallel Debug Manager lets developers open up separate debug windows for any CPU on any board in the system.
Customize and Automate Test

The GEL scripting language lets users customize their development environment by adding menu items and creating a graphical user interface (GUI) to control the target DSP application. Because GEL is based on C, it is easy to learn and implement. GEL is a powerful scripting language with many built-in functions to access target DSP resources (such as DSP memory locations and COFF variables) as well as host resources (such as adding new menu items, dialog boxes, and sliders). Many GEL functions are built-in and can be called from anywhere in Code Composer Studio.

Without purchasing any other tools, GEL can be used to create a DSP application GUI prototyper. Examples of GEL routines that can be created include:

- User interface to DSP applications
- Hardware diagnostics
- Board setup
- Frequently accessed functions
- Application-specific commands
- Connecting a slider to a variable
- Regression test control
It is easy to add menu items to the tools drop-down menu via a simple text file. A GEL dialog box can be created to accept user parameters that can be passed to GEL functions and then executed. When the execute button is pressed, the dialog’s parameters are passed to the menu item’s C-function (contained in the text file) and interpretively executed.

Results can be printed to any number of GEL output windows to highlight the state of the testing. GEL test functions can perform several test cases or load DSP programs onto the target, run them, and collect test results. Test cases can examine memory locations, verify registers, target variables, and much more.
Summary

As DSPs make their way into an ever-growing number of applications, the software that drives these products will play an increasingly crucial role in the product’s success in the marketplace. As a result, designers must scrutinize more than raw processor performance when selecting a DSP to build their products around. The sophistication of the tools that will be used to create the DSP software must also be considered. Basic code generation, debugging, and profiling capabilities of existing development tools is no longer sufficient to meet the demands of developing complex real-time systems.

Code Composer Studio from Texas Instruments empowers DSP developers with an advanced, integrated set of tools for every phase of the code development cycle: a visual project manager, source code editor, compiler and build tools, debugger, profiler, and data visualization tools. Every tool is accessible from the same control window, removing the need to switch between several standalone applications, a tedious practice that has beleaguered other development systems.

Code Composer Studio does more than match the flexibility and power of microprocessor development environments—it takes users to an entirely new level of DSP development. The ability to debug real-time applications is finally in the hands of developers. Code Composer Studio’s real-time analysis capability helps developers quickly find and fix subtle problems that only occur when an application is running at full speed. And Code Composer Studio’s extendible plug-in architecture allows users to add their own application-specific third-party tools as part of their projects, fully integrated with the standard toolset. Add plug-ins supporting block diagram code generation, board diagnostics, new data visualizations, and more.

Today’s complex real-time systems demand both high-performance processors and a truly visual development environment. Code Composer Studio, together with the industry-leading TMS320 family of DSPs, meets this demand and offers developers an unbeatable combination of the finest DSP technology for creating new and exciting DSP products.
## TI Contact Numbers

**INTERNET**

*TI Semiconductor Home Page*

www.ti.com/sc

*TI Distributors*

www.ti.com/sc/docs/distmenu.htm

**PRODUCT INFORMATION CENTERS**

**Americas**

Phone  +1(972) 644-5580  
Fax  +1(972) 480-7800  
Email  sc-infomaster@ti.com

**Europe, Middle East, and Africa**

Phone  
Deutsch  +49-(0) 8161 80 3311  
English  +44-(0) 1604 66 3399  
Español  +34-(0) 90 23 54 0 28  
Francais  +33-(0) 1-30 70 11 64  
Italiano  +33-(0) 1-30 70 11 67  
Fax  +44-(0) 1604 66 33 34  
Email  epic@ti.com

**Japan**

Phone  
International  +81-3-3457-0972  
Domestic  0120-81-0026  
Fax  
International  +81-3-3457-1259  
Domestic  0120-81-0036  
Email  pic-japan@ti.com

**Asia**

Phone  
International  +886-2-23786800  
Domestic  
Australia  1-800-881-011  
TI Number  -800-800-1450  
China  10810  
TI Number  -800-800-1450  
Hong Kong  800-96-1111  
TI Number  -800-800-1450  
India  000-117  
TI Number  -800-800-1450  
Indonesia  001-801-10  
TI Number  -800-800-1450  
Korea  080-551-2804  
Malaysia  1-800-800-011  
TI Number  -800-800-1450  
New Zealand  000-911  
TI Number  -800-800-1450  
Philippines  105-11  
TI Number  -800-800-1450  
Singapore  800-0111-111  
TI Number  -800-800-1450  
Taiwan  080-006800  
Thailand  0019-991-1111  
TI Number  -800-800-1450  
Fax  886-2-2378-6808  
Email  tiasia@ti.com

TI, Code Composer Studio, DSP/BIOS, RTDX, and VelociTI are trademarks of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.
IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

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

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE (“CRITICAL APPLICATIONS”). TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty, or endorsement thereof.

Copyright © 1999 Texas Instruments Incorporated