ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *Molex / Innovative Integration SBC6711* at our website: *Click HERE*

SBC6711 Development Package Manual

The SBC6711 Development Package Manual was prepared by the technical staff of Innovative Integration, December 23, 2003.

For further assistance contact:

Innovative Integration
2655 Park Center Drive
Simi Valley, California 93065

PH: (805) 520-3300
FAX: (805) 579-1730

email:techsprt@innovative-dsp.com
Website:www.innovative-dsp.com

This document is copyright 2001 by Innovative Integration. All rights are reserved.

VSS\Distributions\SBC6711\Documentation\SBC6711.book

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

List of Tables. 7
List of Figures. 9

CHAPTER 1 *Introduction* 11

A Note about this Manual. 12

CHAPTER 2 *Installation* 13

Host Hardware Requirements. 13
Software Installation. 13
Tools Registration. 16
Hardware Installation 18
JTAG Emulator Hardware Installation (for DSP boards Only, the JTAG is NOT needed with ChicoPlus) **18**
Baseboard Installation (for DSP boards AND ChicoPlus). **18**
Code Composer Studio Setup. 20
Borland Builder Setup and Use 23

CHAPTER 3 *Integrated Development Environment* 27

The Texas Instruments C Compiler Toolset 27
C Compiler Toolset Usage **28**
Code Composer Studio. 28
Editor. **28**
Debugger. **28**

CHAPTER 4 *Support Applets*. 29

The Terminal Emulator “UniTerminal” 29
The COFF File UniDownloader. 38
Using the Application **38**
Command Line Switches. **38**
Manual Operation. **39**
The COFF File Dump Utility 40

CHAPTER 5 *Creating a Binary File* 42

The Flash Burn Utility 43
Target Page **44**
Talker Page **45**
Flash Page **46**
Info Page. **48**
Example Burn Sequence **48**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Common Problems when Embedding Code 49

CHAPTER 6 *Developing Target Code* 51

Introduction 51
Components of Target Code (.c, .asm, .cmd) 52
Edit-Compile-Test Cycle using Code Composer Studio 52
A Simple Code Composer Studio Project 52
Build Options (M62, Q62, SBC62 Boards) 54
Build Options (M67, Q67, SBC67 Boards) 56
Build Options (SBC6711 Boards) 58
Build Options (M44 Boards) 60
Build Options (SBC32 Boards) 62
Automatic projectfile creation 65
Rebuilding a Project 65
Running the Target Executable 65
Anatomy of a Target Program 66
Use of Library Code 67
Compiling/Assembling/Linking Outside Code Composer Studio 67
The Next Step: Developing Custom Code 68

CHAPTER 7 *Developing Host Code* 69

Dynamic Link Library 69
Sample Host Programs 70
DspComponent 72
Host Example Programs for the SBC671x Baseboard 73
Overview 73
MessageTest 74
The Target Application 76
VcMessage Test 77

CHAPTER 8 *Creating Target Software* 81

C Code Development 81
C Compiler 81
C Library Reference 82
Zuma Toolset Libraries 82
Texas Instruments C Libraries 85
Target Hardware Interaction 86
Digital Input/Output 87
C Language Digital I/O Functions 88
Timers 88
STDIO Communication 89
Using Interrupts 90
The SBC6711 Message Interface 91

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Overview of the Message Interface 91
Target API for Messaging 92
Host USB Support 92
Host API for Bulk Transport 93
Example Programs for the Bulk Transport Interface 93
The SBC6711 RS-232 Interface 93
Example Program for the RS232 Interface 94
Building Flash Programmable Applications 94
Example Target Programs for the SBC6711 95
HELLO 95
TEST 96

CHAPTER 9 *DOS Environment Requirements* 97

CHAPTER 10 *SBC6711 Hardware* 99

SBC6711 Hardware Functions 99
Memory Map 100
SBC6711 Hardware Initialization Requirements 102
External Memory 102
SBC6711 OMNIBUS 103
SBC6711 OMNIBUS Memory Mapping 104
OMNIBUS Power 104
FIFOPort I/O Expansion 105
Transmitting and Receiving FIFOPort Data 106
Monitoring FIFO Status 107
FIFOPort Reset 108
Controlling the FIFOPort Programmable Level Flag 108
FIFOPort Flags 108
Designing External Hardware for use with the FIFOPort 109
‘C6711 Serial Ports 110
RS232 Serial Port 110
RS232 Interrupts 111
USB Port 111
Resetting the SBC6711 over USB port 112
USB Interrupts 113
DMA Support and the USB Interface 113
USB Software Development 113
Timers 114
On-chip Timers 114
AD9851 Direct Digital Synthesizer 115
External Timer/Counters 116
Timer Output Control 117
Digital I/O 118
Digital I/O Timing 119
Digital IO Electrical Characteristics 120

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Interrupts **120**
Conditioning for Interrupt Input Signals ***121***
Shared/Dedicated Interrupts ***122***
External User Interrupt Usage **123**
Watchdog Timer **123**
Multi-Card Timing Synchronization **123**
JTAG Test Bus **125**
Miscellaneous Control **125**
Power Requirements **126**
Updating the SBC6711 logic **126**

CHAPTER 11 *Appendices* ***127***

Connector pinouts **127**
JP4, JP8, JP5 - OMNIBUS I/O Connectors ***127***
JP6, 7, 9, 10 – OMNIBUS Bus Connectors ***130***
JP3 – Digital I/O Connector ***132***
JP2 – FIFOPort Connector ***133***
JP13 - SyncLink/ClkLink ***134***
JP18, JP19 – Processor Serial Port Connectors ***135***
JP17 – JTAG Debugger Connector ***136***
JP16 – Power Input Connector ***137***
JP22 – Asynchronous Serial Port Connector ***138***
Board Layout **138**
Mechanical Drawing (Board Revision A & B) ***139***
Connector Location (Board Revision A & B) ***140***
Mechanical Drawing (Board Revision C & D) ***141***
Connector Location (Board Revision C & D) ***142***

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

List of Tables

TABLE 1. Common Problems when Embedding Code in Flash ROM 50
TABLE 2. Zuma Toolset Source Directories 82
TABLE 3. Zuma Toolset Support Subdirectories 83
TABLE 4. Texas Instruments Standard Library Functions 85
TABLE 5. SBC6711 External Peripheral Memory Map 86
TABLE 6. Digital I/O Access Memory Location 87
TABLE 7. Digital I/O Direction Configuration 87
TABLE 8. Digital I/O Library Functions 88
TABLE 9. C Language Timer Function 88
TABLE 10. STDIO Driver Functions 90
TABLE 11. Messaging System Target Functions 92
TABLE 12. Messaging System Host Functions 93
TABLE 13. RS232 Communication Functions 94
TABLE 14. Required Disk Directory Structure for II Development Tools. 98
TABLE 15. SBC6711 External Memory Map 101
TABLE 16. SBC6711 Bus Control Register Initialization Values 102
TABLE 17. SBC6711 I/O Bus Memory Mapping 103
TABLE 18. I/O Bus Power Ratings 105
TABLE 19. FIFOPort Level Status Register 107
TABLE 20. FIFO Port Timing Parameters 110
TABLE 21. UART Control Registers 111
TABLE 22. USB Control Registers 112
TABLE 23. AD9851 Control Registers 115
TABLE 24. External Timer/Counter Configuration Register Addresses 116
TABLE 25. External Timer/Counter Input Clock Control Registers Bit Definitions 116
TABLE 26. External Timer/Counter Period Formulas 117
TABLE 27. External Timer/Counter Data Registers Addresses 117
TABLE 28. Timer Pins Output Sources 117
TABLE 29. DDS Pin Output Sources 118
TABLE 30. Digital I/O Control Registers 118
TABLE 31. Digital IO Port Control Register Bit Definitions 119
TABLE 32. Digital I/O Port Timing Parameters 120
TABLE 33. External Interrupt Input Control Register Addresses 120
TABLE 34. External Interrupt Control Register Bit Definitions 121
TABLE 35. External Interrupt Status and Acknowledge Register Addresses 122
TABLE 36. SyncLink 0 Signal Selection Register Bit Definitions 124
TABLE 37. SyncLink 1 Signal Selection Register Bit definitions 124
TABLE 38. SyncLink 2 Signal Selection Register Bit definitions 124
TABLE 39. ClockLink Signal Selection Register Bit Definitions 125
TABLE 40. Miscellaneous Control Register Definition 126

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TABLE 41. OMNIBUS Bus Connectors 130
TABLE 42. I/O Module Bus Connectors 131
TABLE 43. Digital I/O Connector 132
TABLE 44. FIFOPort Connector 133
TABLE 45. SyncLink Connector 134
TABLE 46. Processor Serial Port Connector 135
TABLE 47. JTAG Debugger Connector 136
TABLE 48. Power Input Connector 137
TABLE 49. Asynchronous Serial Port Connector 138

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

List of Figures

FIGURE 1. Terminal Emulator “UniTerminal” Applet 30
FIGURE 2. UniTerminal File Menu 31
FIGURE 3. UniTerminal DSP Menu 32
FIGURE 4. UniTerminal Option Menu 33
FIGURE 5. UniTerminal Display Options Tab 33
FIGURE 6. UniTerminal DSP Options Tab 34
FIGURE 7. UniTerminal Sound Options Tab 35
FIGURE 8. UniTerminal Support Options Tab 36
FIGURE 9. UniTerminal Help Menu 36
FIGURE 10. The COFF Dump Utility 40
FIGURE 11. COFF Dump Utility Output 40
FIGURE 12. Creating a New Project in Code Composer Studio 53
FIGURE 13. Adding Files to a Code Composer Studio Project 53
FIGURE 14. Code Composer Studio Project Window 54
FIGURE 15. Code Composer Studio Compiler Build Options 55
FIGURE 16. Code Composer Studio Linker Build Options 56
FIGURE 17. Code Composer Studio Build Results Window 56
FIGURE 18. Code Composer Studio Compiler Build Options 57
FIGURE 19. Code Composer Studio Build Results Window 58
FIGURE 20. Code Composer Studio Compiler Build Options 59
FIGURE 21. Code Composer Studio Linker Build Options for Non-Bios Project 60
FIGURE 22. Code Composer Studio Build Results Window 60
FIGURE 23. Code Composer Studio Compiler Build Options 61
FIGURE 24. Code Composer Studio Assembler Build Options 61
FIGURE 25. Code Composer Studio Linker Build Options 62
FIGURE 26. Code Composer Studio Build Results Window 62
FIGURE 27. Code Composer Studio Compiler Build Options 63
FIGURE 28. Code Composer Studio Assembler Build Options 63
FIGURE 29. Code Composer Studio Linker Build Options 64
FIGURE 30. Code Composer Studio Build Results Window 64
FIGURE 31. The MessageTest Example 75
FIGURE 32. The VclMessage Example 78
FIGURE 33. SBC6711 Block Diagram 100
FIGURE 34. FIFOPort Block Diagram 106
FIGURE 35. FIFO Port Timing 109
FIGURE 36. DSP On-Chip Timer 0 Clock Source Control Register 114
FIGURE 37. DSP On-Chip Timer 1 Clock Source Control Register 114
FIGURE 38. Digital I/O Port Timing 119
FIGURE 39. SyncLink Connector Pin Orientation 134

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

FIGURE 40. Power Connector Pin Positions (side view, from front of connector, showing connector keying and locking tab along with printed circuit board position) 137

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CHAPTER 1

Introduction

This document describes the Zuma software development environment for Innovative Integration (I.I.) digital signal processor (DSP) cards. The environment comes complete with ANSI compliant C/C++ code Compiler, Assembler, Linker, Debugger, and Windows interface software and represents the most complete package available for DSP code creation for Texas Instruments DSP processors.

Each Developer's Package consists of four major features:

- TMS320-based DSP board.
- Texas Instruments Floating Point C Compiler/Assembler toolset.
- Code Composer JTAG-based hardware-assisted debugger.
- Zuma software toolset including:

  DSP Peripheral Library - supporting on-board peripherals and DSP functions, with full source code.

  Custom 32-bit Windows 9x/NT/2000 compatible dynamic link library (DLL) - which utilizes a custom, 32-bit, Ring 0/Kernel-mode device driver for host PC software application development.

  Host Support Applets - for automatic program download, terminal emulation, COFF file dumping and on-board flash programming.

  Sample Applications - showing Host PC as well as target DSP coding techniques.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

This manual discusses installation issues and includes full documentation on all Innovative Integration software tools (please see the accompanying manuals for specific information on the T.I. toolset or Code Composer Studio software packages). Installation is discussed first, followed by brief introductions to each of the software packages and instructions on their use. General software development issues are presented, and a tutorial on DSP software development, particularly as it relates to the integrated use of the software packages included in this kit, are also discussed. References are given for the peripheral libraries and host DLL packages in the Appendices.

A Note about this Manual

Certain typography conventions are used in this manual to indicate user operations, file types, etc., as follows:

- Windows application menu commands are identified and presented as pipe-delimited strings indicating the menu entries which are being discussed. For example, the Load Program menu item under the File menu in the Code Composer package would be named by the following string:

  File | Load Program

  Computer readable files and keyboard input/output are represented in Courier font, with user input in bold. For example, a program file will be referred to by name as

  ```
  C:\SBC6711\TALKER\TALKER.OUT
  ```

  while user input and commands look like

  ```
  ROM MYPROG.OUT
  ```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CHAPTER 2

Installation

Thank you for purchasing from Innovative Integration! We appreciate your business. This chapter describes the software and hardware installation procedure for Windows 2K or XP. Development under other versions of Windows are not supported at this time, though drivers are provided to allow runtime operation under Win9x, and WinME.

Do *NOT* install the hardware card into your system at this time. This will follow the software installation.

Host Hardware Requirements

The software development tools require an IBM or 100% compatible Pentium IV-class or higher machine for proper operation. An Intel-brand processor CPU is *strongly recommended*, since AMD and other "clone" processors are not compatible with the Intel MMX and SIMD instruction-set extensions which the Armada Host library utilizes extensively to improve processing performance within a number of its components. The host system must have at least 128 Mbytes of memory (256MB recommended), 100 Mbytes available hard disk space, and a CDROM drive. Windows 2000 or XP (referred to herein simply as *Windows*) is required to run the developer's package software, and are the target operating systems for which host software development is supported.

Software Installation

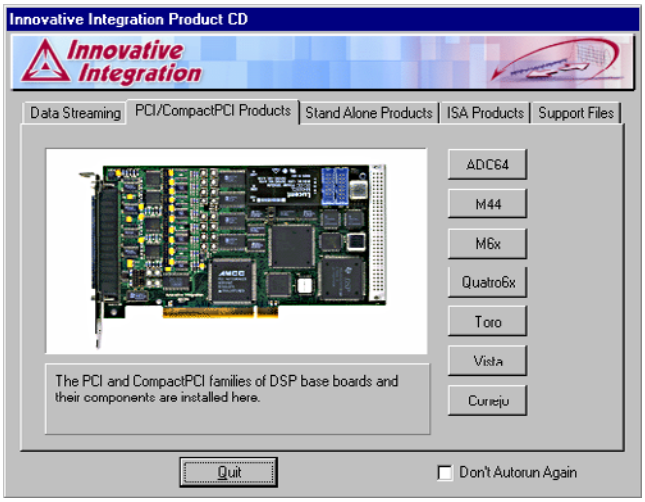
The development package has an installation program that will guide you through the installation.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Note: Before installing the host development libraries (VCL components or MFC classes), you must have Microsoft MSVC and/or Borland C++ Builder installed on your machine, depending on which of these IDEs you plan to use for Host development. If you are planning on using one of these environments, it is imperative that these environments be tested and known-operational before proceeding with the library installation. Additionally, you must install Code Composer Studio prior to installation of the development package software if you have a DSP board. (This will not be necessary if you have purchased a streaming board such as ChicoPlus.) If these items are not yet installed, then the installation program will not permit installation of the associated development libraries. However, drivers and DLLs-only may be installed, to facilitate field deployment.

Win2K/XP Users Only. You must have Administrator Privileges to install and run the software/hardware onto your system, refer to the Windows documentation for details on how to get these privileges.

To begin the installation, start the host operating system and insert the installation CD. The installation should autostart after you put it into the CD. If the CD does not auto start, click on the **Start** button, then **Run**. Enter the path to the SETUP.EXE program located at the root of your CD-ROM drive, i.e. E:\SETUP.EXE. The setup program will run. Select the appropriate tab. From there, select the appro-



priate baseboard button, depending on the type of hardware being installed. The baseboard-specific install screen will automatically come up. You should see a screen similar to the following. **Note:** If you



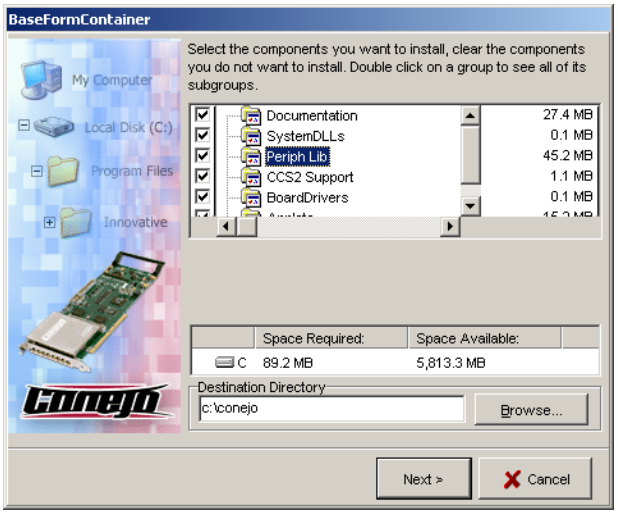
are attempting a full installation with development libraries for a DSP board and have not already installed Code Composer and your Host IDE (Borland Builder or MSVC), you should do so before pro-

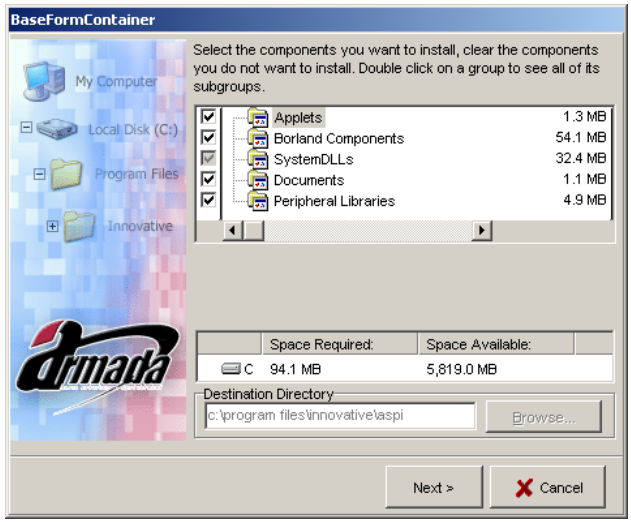
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ceeding with the installation and NOT click Install at the first selection window. In this case, click the Exit button to terminate. If you are deploying a completed application and need driver and DLL support files only to be installed, proceed. There will be a chance to check drivers-only and DLLs-only in each of the sub-installs.

In the example above, each of the check boxes have sub-installs associated with them, and will open a sub-install screen window with another set of check boxes allowing you to select files to be installed. For example, the first sub-install for “Conejo - (including Applets, Examples and Drivers” is shown below.

The second image below, is for “Armada - (including C++Builder or MS Visual C++ support)”. It follows a similar pattern and presents another set of choices as illustrated.





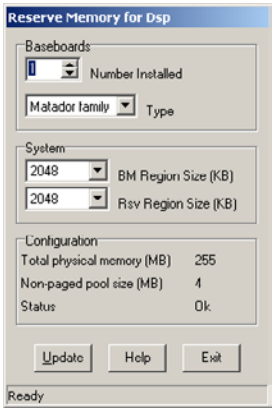
You may customize your installation, including or omitting components shown in these sub-install menus. At this point in each sub-install, check the desired components to be installed and click the ‘Next>’ button.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The host examples included in this installation are written in C++ using Borland Builder or Microsoft MSVC, and C on the target side using Code Composer Studio (C++ on some targets). They are provided to illustrate the various features of the board and how these features are utilized.

Bus Master Memory Reservation Applet. Near the conclusion of the installation process, **ReserveMem.exe** will run. This will allow you to set the memory size needed for the busmastering to occur properly. This applet may be run from the start menu later if you need to change the parameters.

For optimum preformace each Matador Family Baseboard requires 2 MB of memory to be reserved for its use. To reserve this memory, the registry must be updated using the ReserveMem applet. Simply select the **Number of Baseboards** you have on your system, click **Update** and the applet will update the registry for you. If at any time you change the number of boards in your system, then you must invoke this applet found in **Start | Programs | <target board> | ReserveMemDsp**.

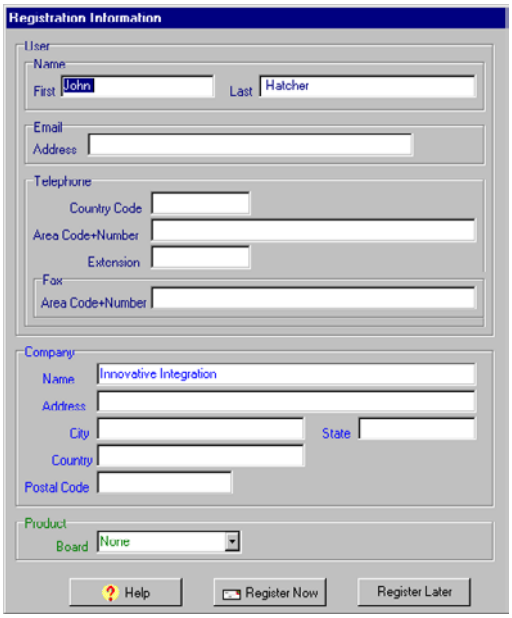


After updating the system exit the applet by clicking the **exit** button to resume the installation process.

Tools Registration

Before beginning DSP and Host software development, you must register your installtion with Innovative Integration. Technical support will not be provided until registration is successfully completed. Additionally, some of the development applets will not operate until unlocked with a passcode provided during the registration process. To Register, click Start | Program Files | Vista | New User to start the *NewUser.exe* registration application. The registration form below will be displayed:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

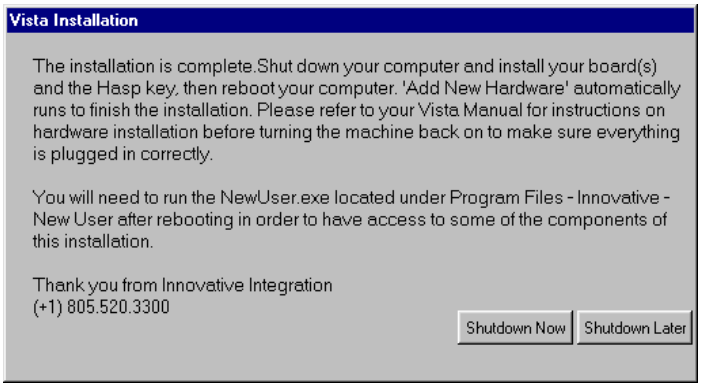


You should fill it out completely and return it to Innovative, either via email or fax. Upon receipt, Innovative will provide access codes to enable downloading of update software from our website, telephone hotline support and unrestricted applet access.

At this point you have finished the installation process and you are ready to begin development. Exit with the **OK** button.

Finally, the following screen appears. At this point, you should shut down your computer. This will also allow you to physically place the board(s) into the machine in addition to allowing newly installed components to be initialized

.



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Hardware Installation

The software components of the Development Package have been installed. To proceed with the Development Package Kit installation, it will be necessary to configure and install your hardware.

JTAG Emulator Hardware Installation (for DSP boards Only, the JTAG is NOT needed with ChicoPlus)

First, the emulator hardware must be configured and installed into your PC. The emulator hardware is described in the table below:

Type	Features
Pod-based	Uses a special ribbon cable with integrated line drivers to connect the target DSP emulation signals to the JTAG debugger card. Usable on 3.3 volt or 5 volt designs. (Including ‘C54x and ‘C6x)

PCI Pod-Based Emulator Installation

To install the PCI pod based emulator, follow the instructions below:

1. Shut down Windows and power-off the host system.
2. Perform the board installation in an “ESD” or static safe workstation.
3. Power off the host system and **touch the chassis** of the host computer system to dissipate any static charge.
4. Remove the card from its protective static-safe shipping container, being careful to handle the card only by the edges.
5. Touch the chassis of the PC to dissipate any built up static charge.
6. Securely install the JTAG board in an available PCI slot in the host computer.
7. Connect the JTAG pod to the host-pod cable. Connect the host-pod cable to the connector located on the end bracket of the JTAG PCI plug-in board.

Baseboard Installation (for DSP boards AND ChicoPlus)

To install the baseboard:

1. Shut down Windows and power-off the host system.
2. Perform the board installation in an “ESD” or static safe workstation.
3. Power off the host system and **touch the chassis** of the host computer system to dissipate any static charge.
4. Remove the card from its protective static-safe shipping container, being careful to handle the card only by the edges.
5. Touch the chassis of the PC to dissipate any built up static charge.
6. Connect the 14-pin connector on the JTAG PCI pod to the DSP board JTAG connector **(ChicoPlus users skip this step.).**
7. Securely install the baseboard into an available PCI slot in the host computer.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A Few Considerations BEFORE Power-up. Double-check everything before applying power. Are the JTAG (if needed) and baseboard cards seated correctly in the slot?

It can't be overemphasized: double check your cabling BEFORE connection to the baseboard. Also, don't cut corners and hot plug the cables. This can cause latch-up of components on the card and damage them irreparably. Be aware that the cables to analog inputs are an important part of keeping the signals clean and noise-free. Shielded cables and differential inputs (where applicable) help to control and reduce the noise.

After Power-up. After completing the installation, boot your PC into Windows.

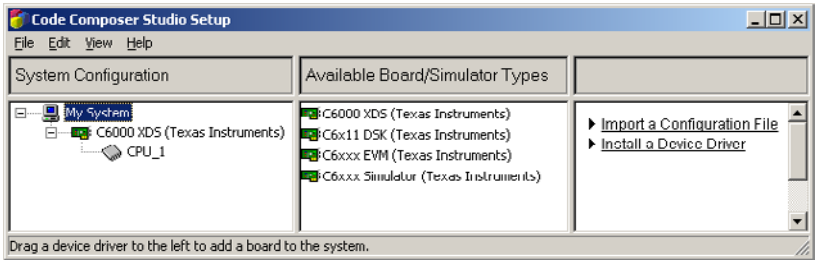
The target cards are plug and play compliant, which Windows should detect and auto-configure at start-up. Under rare circumstances, Windows will fail to auto-install the device-drivers for the JTAG (if needed) and baseboards. If this happens, please refer to the "TroubleShooting" section.

The next section is NOT used with the Chico Family products. If your board is a Chico Family product, you are DONE.

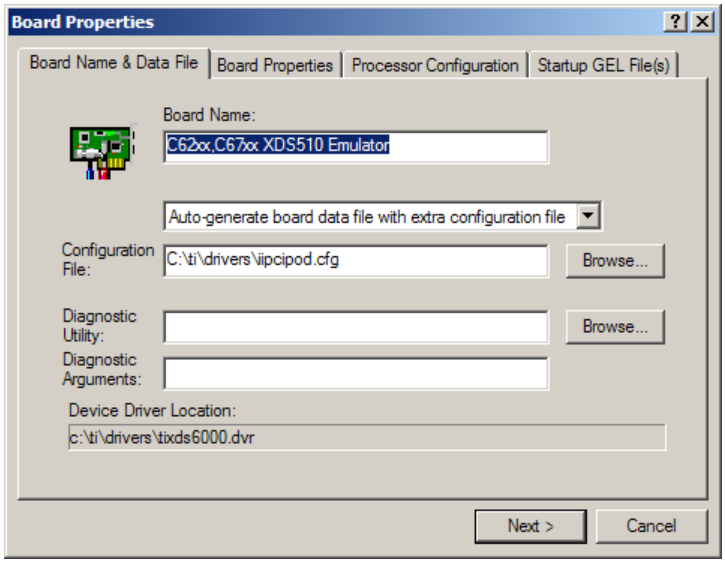
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Code Composer Studio Setup

To setup Code Composer Studio and activate the Innovative-supplied Code Hammer JTAG board driver, the Code Composer Setup Utility must be run. Since the Code Hammer debugger is XDS510-compatible, Code Composer Studio setup must be configured to use the XDS510 driver for the C6000 after removing the default simulator driver from the System Configuration. This driver is named C6000 XDS within the Code Composer setup utility. In the figure below, it is the first item listed in the Available Board/Simulator Types column of the Setup program.



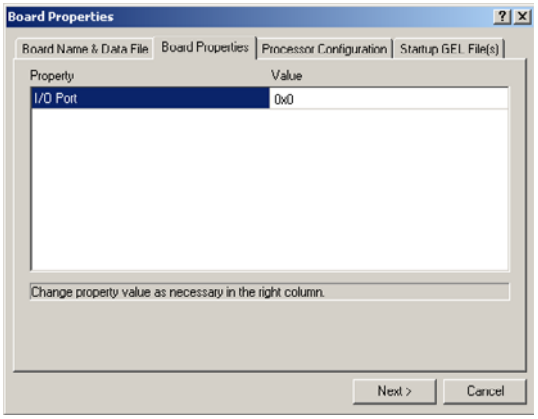
Click this driver from the “Available Board/Simulator Types” control within the setup utility and drag it into the “System Configuration” control. Then, right-click on this C6000 XDS object to invoke the Properties Dialog for the driver.



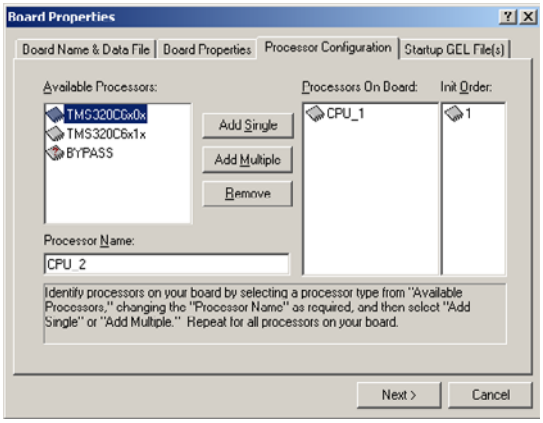
Under the “Board Name & Data File” tab, the board name edit box should list "C6000 XDS (Texas Instruments)". The “Configuration File” combo box should be changed to "Auto-generate board data file with extra configuration file". The “Configuration File” edit box should be changed to "<drive>:\ti\Drivers\IIPciPod.cfg", where <drive> is the letter for the drive onto which CCS 2.0 was installed.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Under the “Board Properties” tab, the I/O port value for the driver should be set to virtual device address “0”.



Under the “Processor Configuration” tab, one or more processors of the appropriate type (C6x0x or C6x1x) should be added to the “Processors on the Board” list box. Note that for multi-board targets

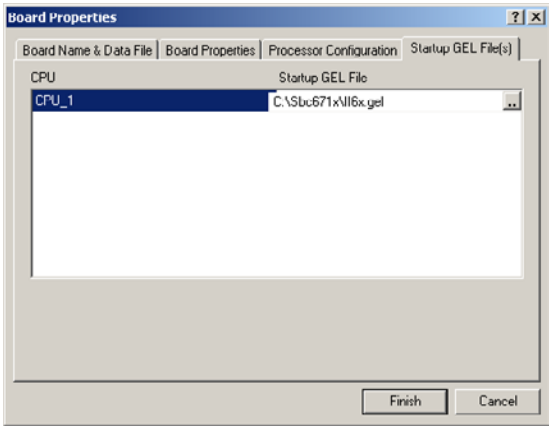


such as the Quadia, one processor should be added for each device in the JTAG scan path.

Under the Startup Gel Files tab, the Startup Gel File combo box should be the board-specific initialization GEL script for the target board. Innovative supplies an appropriate, board-specific GEL initializa-

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

tion file for each product located in the root Zuma toolset directory. For example, for the Vista, this should be set to c:\Vista\II6x.gel.



Finally, the setup configuration should be saved. After saving the configuration and shutting down the setup tool, Code Composer Studio should launch successfully. If you encounter difficulty launching CCS 2.0, Run the JtagDiag.exe utility provided in your Zuma toolset to reset the debugger interface. Then, run the MatadorDownload.exe utility and press Boot, to boot a default target application. Then restart Code Composer Studio.

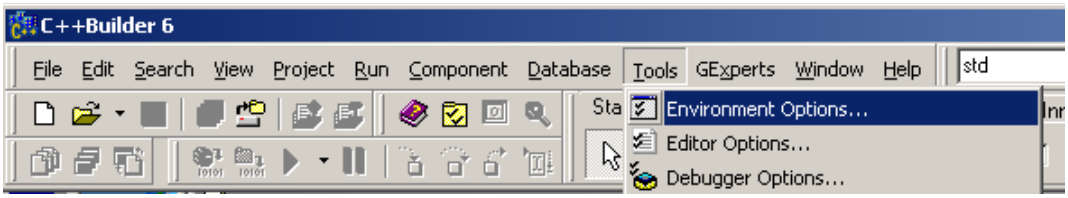
As a consequence of these steps, a new ccBrd0.dat file will be created. This file has been customized for the particular target being used.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

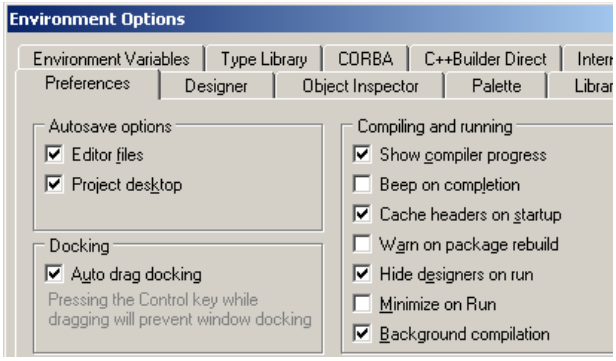
Borland Builder Setup and Use

Following the normal installation of the Innovative Integration toolset components, numerous VCL components and C++ classes are automatically added to the BCB IDE. Additionally, Innovative recommends that the following IDE and project options be manually changed in order to insure simplified use and proper operation:

Automatic saving of project files and forms during debugging. Invoke the Tools | Environment Options dialog from the main BCB toolbar.

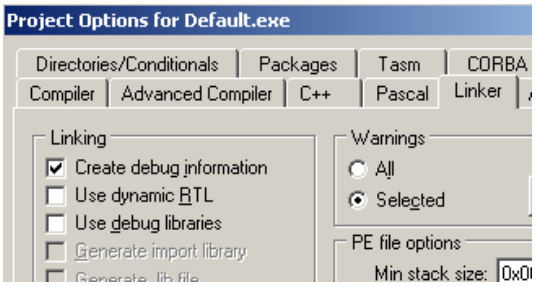


This will invoke the Environment Options dialog:



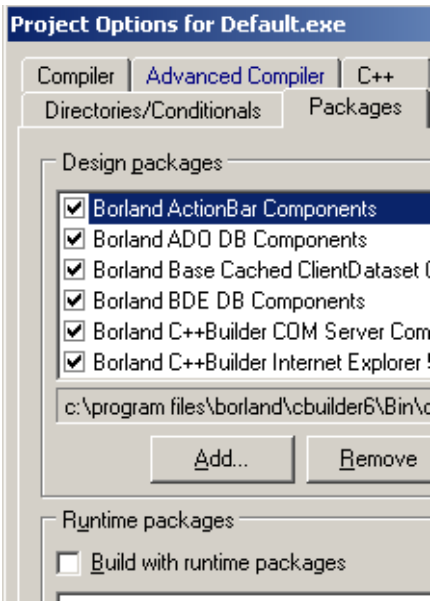
Enable autosaving of Editor Files and the Project Desktop, so that project files are automatically saved each time a project is rebuilt and debugged.

Static-binding of built executables. Click on Project | Options on the main BCB toolbar to invoke the Project Options dialog. Then click on the Linker tab. Uncheck Use Dynamic RTL checkbox.



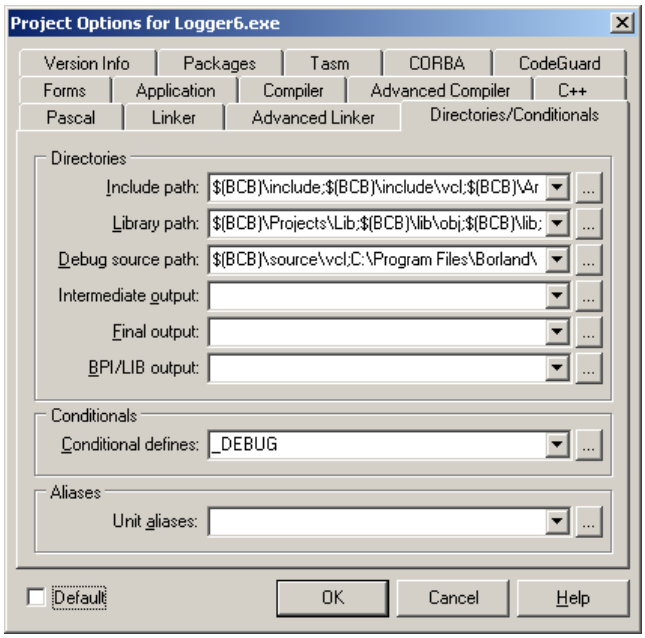
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Next, click on the Packages tab and uncheck the Build with runtime packages checkbox.



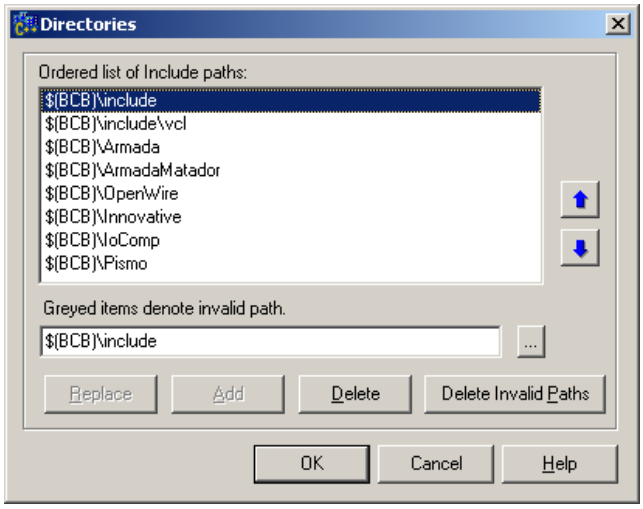
These options insure that projects are built with minimal dependencies on external DLLs. See the FAQ "**What DLLs do I have to deploy with my newly created executable"** in the Troubleshooting chapter for details on which DLLs must be deployed with user-written executables.

Appropriate library and include paths. Click on Project | Options on the main BCB toolbar to invoke the Project Options dialog. Then click on the Directories | Conditionals tab to edit the default Include and Library paths which should be used when constructing an Armada-based application.

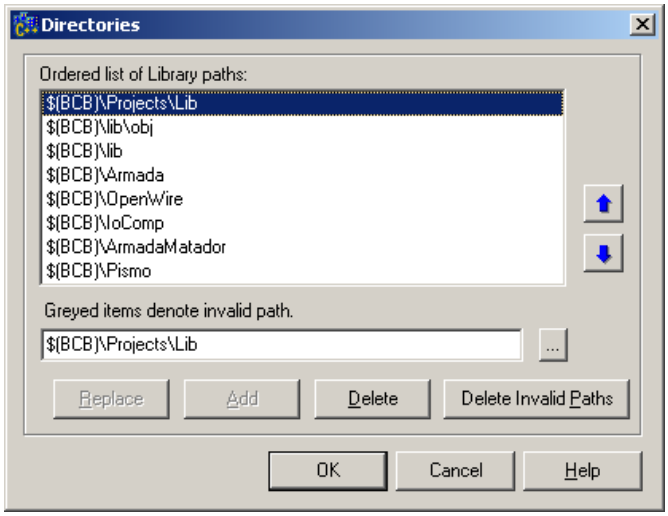


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Next, click on the ellipses next to the Include Path edit box to invoke the Include Path editor dialog. Add entries for Armada, ArmadaMatador, OpenWire, IoComp and Pismo, as shown below, then click Ok to accept these edits.



Next, click on the ellipses next to the Library Path edit box to invoke the Library Path editor dialog. Add entries for Armada, ArmadaMatador, OpenWire, IoComp and Pismo, as shown below, then click Ok to accept these edits.



These changes insure that the standard Armada headers and object files are available to projects during compilation. Note that these paths may either be added to the default BCB project, by editing these options without first opening a specific project, or to specific projects after opening them. The advantage of the former is that the settings are automatically present on allsubsequently-created projects.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CHAPTER 3

Integrated Development Environment

The C Developer's Package consists of several software tools, integrated to work together to provide a complete DSP design environment for Innovative Integration DSP boards. This section discusses the tools included in the development package and gives descriptions of each applets features and use. A brief introduction is given regarding the software programs provided and their use within the Developer's Package. The user is referred to the individual manuals accompanying these software products for complete documentation.

The Texas Instruments C Compiler Toolset

The compiler supplied with the Developer's Package is the Texas Instruments (T.I.) Floating Point C/C++ Compiler toolset for the DSP target board. The compiler runs under Windows as a cross compiler, generating executable applications for the DSP processor which are then downloaded and executed using the other tools in the Developer's Package. The compiler is ANSI C compatible and supports nearly all standard C functions. Additional libraries provided with the Developer's System include C standard I/O and peripheral drivers for the A/D, D/A, bit-I/O and timers. Assembly language may also be mixed with C code for higher performance where required.

Typical application programs will consist of one or more C (`.C`), header (`.H`), and Assembly language (`.ASM`) source files, as needed. Additionally, target program generation requires use of a linker command file (`.CMD`) which specifies the memory map for the target and optionally includes commands defining the libraries to be linked into the final application.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Users of the Code Composer Studio editor/debugger will also employ make (`.MAK`), workspace (`.WSP`) and special Code Composer-specific script files (`.GEL`). The example programs included in the Developer's Package illustrate the use of these files also and give example files to use as a basis for custom DSP applications.

C Compiler Toolset Usage

The C compiler may be run directly from a DOS Prompt window under Windows as described in the TI toolset documentation. Also included in the installation directory are make files useful for manually rebuilding applications programs within the DOS environment. To rebuild an example or library from the command line, execute **`gmake -f <filename.mak>`**.

Code Composer Studio

Code Composer Studio is a flexible, high-performance, integrated code generation environment developed by Texas Instruments and bundled into the Innovative Integration Zuma toolset. For complete documentation to the features of Code Composer Studio package, refer to the accompanying *Code Composer Studio Manual* provided. If the user wishes to compile outside of Code Composer Studio (or has not purchased the package), these make files may be used from the DOS command line to rebuild individual project files or the entire target file set.

Editor

Code Composer Studio supports code editing and emulates the most popular editing packages (CUA, etc.). Code Composer Studio is also a Window editor. Custom DSP code development can take place entirely within the Code Composer Studio environment using its project management tools to place source files, libraries and linker command files into projects (`.MAK`) in order to build executables. The example programs included in the Developer's Package each have a Code Composer project file (`.MAK`) associated with them which may be used to re-compile the example.

Debugger

Code Composer Studio is a software program for high-level TI C and Assembly Language debugging which supports high-performance, JTAG or MPSD-based hardware assisted debugging directly on the target DSP. This allows the user access to the internal register set, peripherals, and bus of the target board in order to load, run, and debug applications. Also integrated into the Code Composer Studio software package is a code management subsystem for editing files as well as creating and compiling DSP projects.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CHAPTER 4

Support Applets

The Developer's Package includes a number of support applications supporting general DSP development: the terminal emulator (`UNITERMINAL.EXE`), the COFF file downloader (`UNIDOWNLOAD.EXE`), the COFF file display utility (`COFFDUMP.EXE`), the PromImage utility (PROMIMAGE.EXE) and the FLASH prom programming facility (`BURN.EXE`). This section describes the functionality of each of the applications and their use within the development system.

The functions provided by each of the applications may be configured through menu selections available within each of the applets themselves. See the discussion below for applet-specific parameters.

The Terminal Emulator "UniTerminal"

The terminal emulator "UniTerminal" provides a C language-compatible, standard I/O terminal emulation facility for interacting with the `stdio` library running on an Innovative Integration target DSP processor. Display I/O calls such as `printf()`, `scanf()`, and `getchar()` are routed between the DSP target and the Host UniTerminal applet where ASCII output data is presented to the user via a terminal emulation window and host keyboard input data is transmitted back to the DSP. The terminal emulator works almost identically to console-mode terminals common in DOS and Unix systems, and provides an excellent means of accessing target program data or providing a simple user interface to control target application operation.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

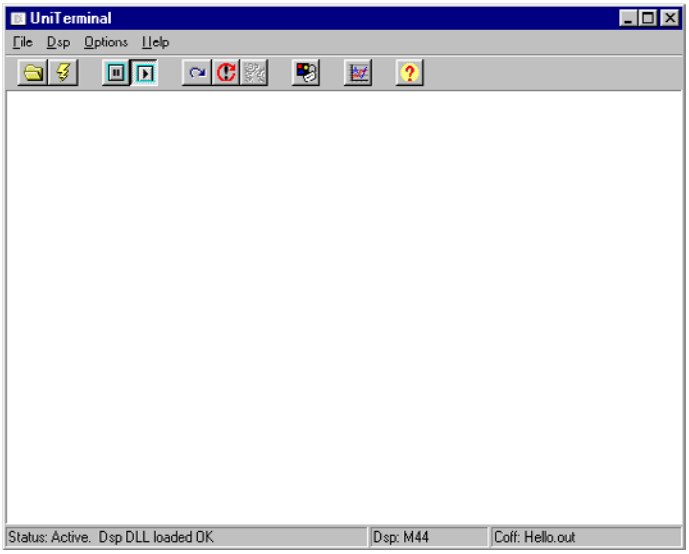


FIGURE 1. Terminal Emulator "UniTerminal" Applet

The terminal emulator is straightforward to use. The terminal emulator will respond to stdio calls automatically from the target DSP card and should be running before the DSP application is executed in order for the program run to proceed normally. The DSP program execution will be halted automatically at the first stdio library call if the terminal emulator is not executing when the DSP application is run, since standard I/O uses hardware handshaking, except on RS232 bus-based targets. The stdio output is automatically printed to the current cursor location (with wraparound and scrolling), and console keyboard input will also be displayed as it is echoed back from the target.

The terminal emulator also supports Windows file I/O using the library routines `fopen()`, `fclose()`, `fread()`, `fwrite()`, `fseek()`, and `fflush()`. Refer to the `Zuma.hlp` file for the types and usage of these library functions as their usage is not 100% ANSI compliant.

Important Note: Before using UniTerminal, you must register your Zuma Toolset. Until you do so, usage will be restricted to a 20-day trial period for UniTerminal and other applets contained in the Zuma Toolset. To register, fill out the contents of the Registration Form, then click on the Register Now button. This will print a Registration report which, must be faxed to Innovative Integration. Innovative Integration will E-mail you an Access Code, which must be typed into the Registration Form for all the Zuma features to be enabled.

Terminal Emulator Menu Commands. The terminal emulator provides several menus of commands for controlling and customizing its functionality. These functions are available on the menu bar, located at the top of the UniTerminal main window. Speed button equivalents for each of the menu options are also available on the button bar located immediately beneath the menu bar. The following is a description of each menu entry available in the terminal emulator, and its effects.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

File Menu:

.

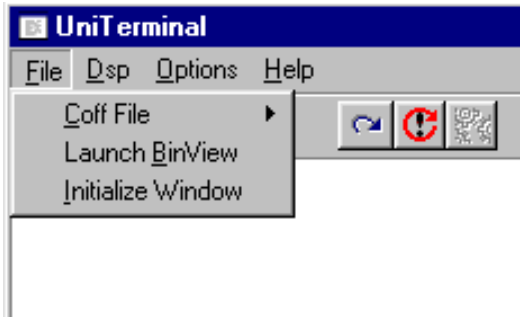


FIGURE 2. UniTerminal File Menu

- File | COFF File | Download - provides for COFF (Common Object File Format) program downloads from within the terminal emulator. When selected, a file requester dialog box is opened and the full pathname to the COFF filename to be downloaded is selected by the user. Clicking “Open” in the file requester once a filename has been selected will cause the requester to close and the file to be downloaded to the target and executed. Clicking “Cancel” will abort the file selection and close the requester with no download taking place.

Quatro6x Users: Terminal supports downloading of either Coff files (`.OUT`) or multi-processor out (`.MPO`) files. The (`.MPO`) files provide a means of downloading separate (`.OUT`) files to multiple processors simultaneously, which greatly simplifies the task of synchronizing execution in a multi-processor environment.

- File | COFF File | Reload - Reloads and executes the COFF file last downloaded to the target. It provides a fast means to re-execute the application program most recently loaded into the target board.

NOTE: File | COFF File | Download and File | COFF File | Reload functions physically resets the target DSP (in order to initiate the target Talker program) prior to the download. When using the terminal emulator in conjunction with the Code Composer debugger, use Code Composers File | Load Program facility to download executable code to the target rather than the terminal emulator’s download facility. Since the Code Composer mechanism does not physically reset the target during the download, it is not reliant on the target Talker to perform the download.

If you attempt to download using the COFF Download menu within the terminal emulator while using Code Composer, you may receive the diagnostic dialog box, which indicates that Code Composer has *halted* the target processor via the JTAG hardware link. While in this halted state, the terminal emulator cannot invoke the Talker program on the target DSP in order to perform the software download. To correct this problem, execute the Debug | Run Free menu command from within Code Composer to release the DSP from JTAG control. Afterwards, clear the terminal emulator error message dialogs and retry the terminal emulator COFF Download.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- File | Launch Binview – launches Binview, so that you may use it to examine the contents of a binary windows data file.

- File | Initialize Window - Resets the application window. Use this feature to reset the application if emulation ceases and will not resume, as an alternative to restarting the application. Emulation can be disrupted by outside programs, such as the JTAG-based Code Composer debugger.

DSP Menu:

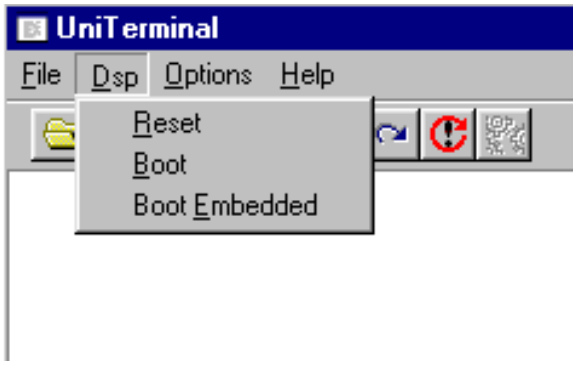


FIGURE 3. UniTerminal DSP Menu

- Dsp | Reset - causes the terminal emulator to assert, then de-assert the target's physical reset pin, bringing the target board into a cold-start, uninitialized condition. This function is also available via the speed button toolbar. This is a special utility function which is not needed in normal operation.

- Dsp | Boot - performs the Dsp | Reset function, then boots the Talker program on the target. On plug-in cards, the Talker program is downloaded into target program memory, the target is reset (launching the Talker program), then UniTerminal polls for the Talker sign-on command via the stdio mailbox. On stand-alone, single-board targets, the target is reset which launches Talker from onboard Flash memory, then UniTerminal polls for the Talker sign-on command via the stdio mailbox. This function is also available via the speed button toolbar. This is a special utility function which is not needed in normal operation.

- Dsp | Boot Embedded - performs the Dsp | Reset function, simultaneously driving the CTS line inactive which causes single board targets to launch the application code embedded into sector one of their onboard Flash memory. This function is also available via the speed button toolbar. This is a special utility function which is only useful for single-board DSP users when testing embedded application programs.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Option Menu:

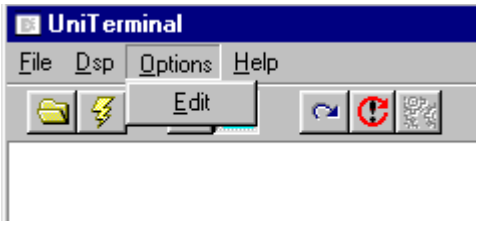


FIGURE 4. UniTerminal Option Menu

- Options | Edit - invokes the user options dialog. The four, tabbed pages on the dialog box allow customization of display settings, dsp type, use of sound and specification of the location of support applets, such as BinView. This function is also available via the speed button toolbar.

Display Options Tab:

The display options tab on the Options dialog (seen below) contains controls to allow user-customization of the appearance of UniTerminal.

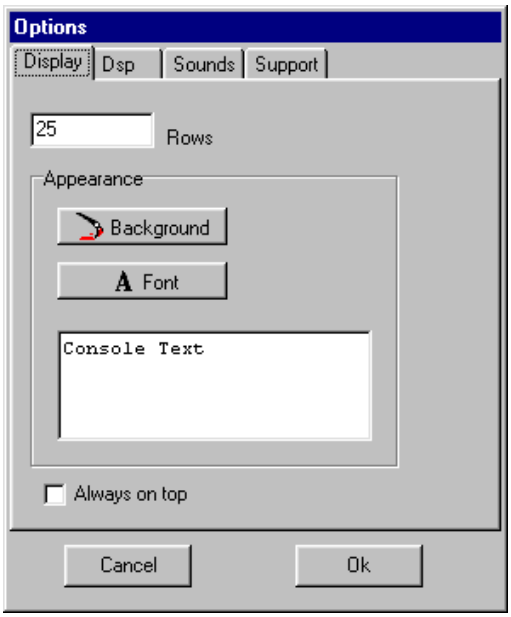


FIGURE 5. UniTerminal Display Options Tab

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The Rows edit box controls the number of rows of text available within the terminal emulator window.

The Appearance group box contains separate buttons which are used to adjust the background color and font of the text within the terminal emulation window.

The Always On Top check box controls whether UniTerminal is forced to remain a foreground application, even when it loses keyboard focus. This is useful when running stdio-based code from within the Code Composer environment, when it's preferable to make terminal visible at all times. The terminal will remain atop other windows when this entry is checked. Select the entry again to uncheck and allow the terminal emulator window to be obscured by other windows.

Display Options Tab:

The Dsp tab on the Options dialog (seen below) contains controls to allow selection of the type and number of the target board being serviced by this instance of UniTerminal.

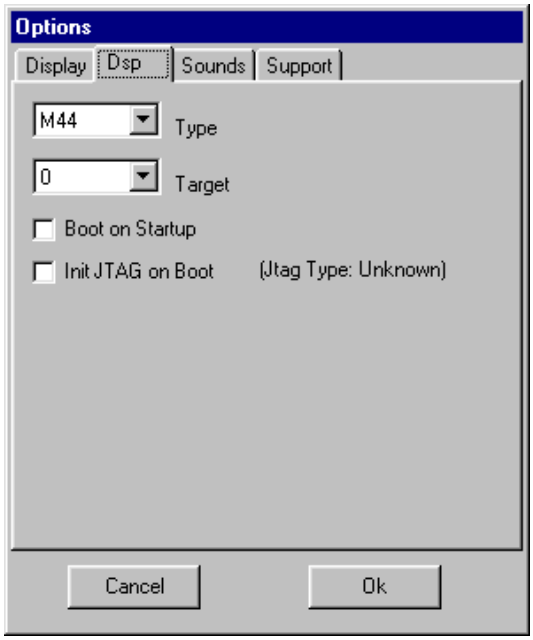


FIGURE 6. UniTerminal DSP Options Tab

The Type combo box specifies the type of target board being serviced by this instance of UniTerminal.

The Target combo box specifies the number of the target board being serviced by this instance of UniTerminal.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The Boot On Startup check box controls whether the target processor is booted when UniTerminal is initially invoked. Disable this check box when using UniTerminal in conjunction with Code Composer, to avoid resetting the target and corrupting the JTAG state upon UniTerminal invokation.

The Init JTAG on Boot check box controls whether the JTAG is reset when the DSP board is booted. If this box is checked, the JTAG reset will disconnect any prior logical connections upon running UniTerminal. Disable this check box when using any debugger other then Innovation Intergration’s. This is it to prevent any communication problem between the target board and debugger.

Sound Options Tab:

The Sounds tab on the Options dialog contains controls to specify whether UniTerminal provides audible indicators during its operation.



FIGURE 7. UniTerminal Sound Options Tab

The Error Indications check box controls whether sounds are played upon receipt of errors during emulation.

The Alert Indications check box controls whether sounds are played upon receipt of errors during emulation.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Support Options Tab:

The Support tab on the Options dialog contains controls to allow specification of the location of companion support applets used by UniTerminal.



FIGURE 8. UniTerminal Support Options Tab

The Binview Button is used to browse to the location of the BinView binary data viewer applet. UniTerminal communicates directly (using TCP/IP, which must be installed) with the BinView applet when target applications attempt to display graphical data files.

Help Menu:

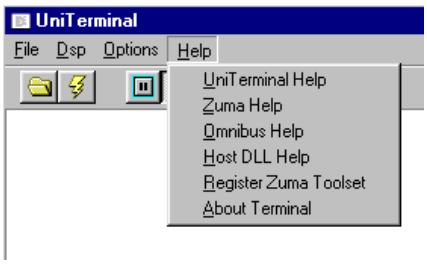


FIGURE 9. UniTerminal Help Menu

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The Help menu provides convenient access to the technical resources/Help Files provided in the Zuma Toolset.

- Help | Register Zuma Toolset - invokes a dialog box to be filled out and sent to Innovative Integration

- Help | About Terminal - invokes a dialog displaying copyright and version information about UniTerminal, plus information pertaining to the use of Host resources by the target DSP board.Use this information when contacting Innovative Integration for technical support regarding UniTerminal.

Terminal Emulator Command Line Switches. The terminal emulator also provides the following command line switches to further modify program behavior. The switches must be supplied via the command line or within Windows shortcut properties (see the Installation section for more information), and will override the default behavior of the applet.

Multiple instances of UniTerminal may be invoked simultaneously in order to support installations utilizing multiple target boards. Instances of UniTerminal, after the first loaded instance must be configured via command line switches in order to properly communicate with their associated target.

- **-b***boardtype* - Use the -b switch to force an instance of UniTerminal to communicate with a specific type of target board, boardtype. Supported boardtypes are listed below:

Board Type	Supported Product(s)	Communication Path
None	No DSP selected	N/A
ADC64	ADC64 DSP Board	PCI Bus
M44	M44 DSP Board	PCI Bus
PC31	PC31 DSP Board	ISA Bus
PC32	PC32 DSP Board	ISA Bus
PC44	PC44 DSP Board, up to two installed TIM processors	PCI Bus
PCI32	PCI32 DSP Board	PCI Bus
PCI44	PCI44 DSP Board, up to three installed TIM processors	PCI Bus
M6x	M62/M67 DSP Board	PCI Bus
Q6x	Q62/Q67 DSP Board with four C6x processors	PCI Bus
SBC31	SBC31 stand-alone DSP Board via serial port	COM1: or COM2:
SBC32	SBC32 stand-alone DSP Board via serial port	COM1: or COM2:
SBC54	SBC54 stand-alone DSP Board via serial port	COM1: or COM2:
SBC6x	SBC62/SBC67 stand-alone DSP Board via serial port	COM1: or COM2:
SBC6711	SBC6711 stand-alone DSP Board	USB

- **-t***X* - Use the -t to allow the user to force the UniTerminal to interact with a specified target number. This switch is particularly useful in multi-board installations to create instances of the emulator for targets other than target 0. See the Installation section for more information on multi-board installations. The *X* parameter specifies the logical target number with which to communicate. **NOTE:** For single-board targets, specify target 0 for boards connected via COM1 and target 1 for boards connected via COM2.

- *"filespec."* - Use the *"filespec."* field, which includes both the path and file name as a unit, to allow the user to force the terminal emulator to download the specified file to the target DSP board, as soon as the terminal emulator is loaded. This field is particularly useful in situations where the UniTerminal is "shelled to" from within an other Host applications to facilitate the automatic execution of target applications employing standard I/O.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The COFF File UniDownloader

The UniDownload application provides a means of downloading debugged programs to a target DSP. The utility is useful in situations where a host application program capable of performing the DSP application download operation is unavailable or unnecessary.



Using the Application

The application may be operated through the use of buttons on the toolbar located at the top of the applet window, or through the use of command line switches. The latter technique is more common, since this allows the applet to be invoked within the Windows startup folder to cause DSP application execution to commence when Windows boots.

Command Line Switches

The following command line switches are supported. Each switch consumes an in line argument, which must be present for proper operation.

Switch Argument

-b name

-f "filename"

-t number

The **-b** switch is used to specify the name of the target DSP to which code is to be downloaded. The following DSP board names are acceptable: None, ADC64, M44, PC31, PC32, PC44, PCI32, PCI44, SBC31, SBC32, SBC54, M6x, Q6x, SBC6x, SBC6711 Case is significant and the name must be spelled exactly as shown. **Note:** The command line must be formatted such that a space character is present between the switch and its argument

The **-f** switch is used to specify the name of the COFF executable image to be downloaded to the target. The filename parameter should be a full unambiguous pathspec to be image file.

The **-t** switch is used to specify the target instance number, 0..15. In a single target system, zero should be supplied.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Manual Operation

The applet may be operated manually using the buttons located at the top of the applet window. The operation of each of the buttons is described below.



The file open button is used to browse to and subsequently download a COFF image file located somewhere on the local machine.



The reload button is used to re-download the last COFF executable image which was successfully downloaded.



The help button is used to invoke the help documentation associated to this application.



The board type combo box is used to change the type of DSP target to which code is to be downloaded. The following DSP types are supported: None, ADC64, M44, PC31, PC32, PC44, PCI32, PCI44, SBC31, SBC32, SBC54, M6x, Q6x, SBC6x, SBC6711.



The target number combo box is used to change he instance of the DSP target to which code is to be downloaded.



The event log is used to display the history of the status of manual operations which have been performed.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The COFF File Dump Utility

The COFF downloader utility provides users with the capability to generate a report detailing the memory usage of target DSP programs generated using the TI tool set.

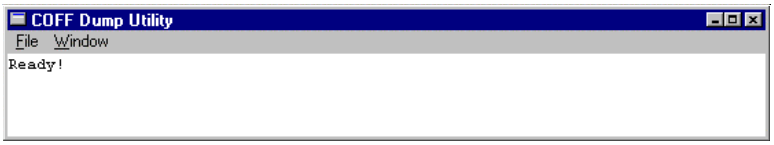


FIGURE 10. The COFF Dump Utility

`COFFDUMP.EXE` parses through COFF files stored in files on the hard disk and ascertains the complete memory consumption by the DSP program. Memory usage for each of the sections defined in the applications command file are tabularized and the results are written to the Windows NotePad scratch buffer. If desired, Notepad can then be used to write the data to disk or to a printer.

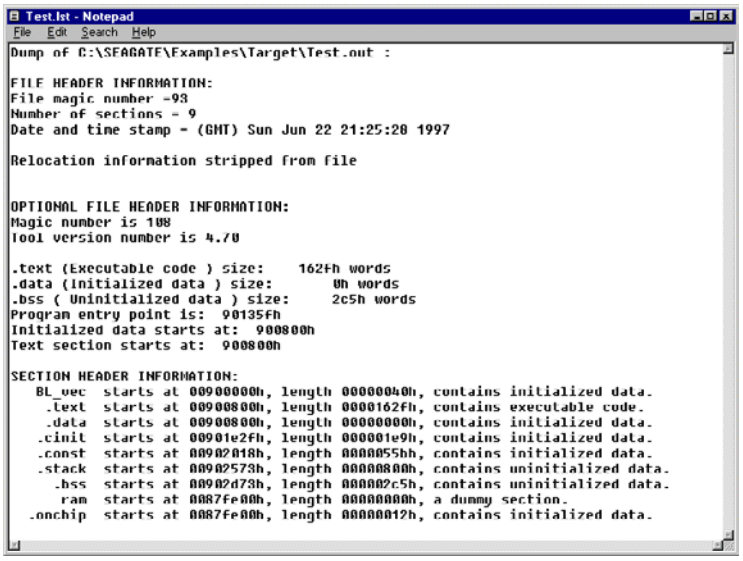


FIGURE 11. COFF Dump Utility Output

COFF Dump Utility Menu Commands. The following is a brief description of commands available from the COFF Downloader menus:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

File Menu

- File | Dump – Involves the standard Windows file selector window for COFF output files (`.OUT`). Parses through selected file and writes diagnostic dump of contents of executable image to NotePad scratch buffer.

- File | Exit - exits the dump utility program.

Window Menu

- Window | About - presents program copyright and version information.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CHAPTER 5

Creating a Binary File

When using single-board computers, such as the SBC32 or SBC6711, it is possible to embed application code into Flash ROM so that the application begins executing immediately after power-up. In order to accomplish this, you must first create a Code Composer project, build it, and then debug/test it by downloading it with the terminal emulator and Code Composer.

The standard Code Composer build process will generate a .out file, which is suitable for use within the terminal emulator or Code Composer, but is unsuitable for placement in Flash ROM. Once you have a viable .out file, it must be converted into a binary file before it can be burned into the target DSP card's Flash ROM. The process of converting an .out file into a binary file (`.bin`) is target specific.

SBC31, SBC32 and SBC54 targets

To briefly describe this conversion process, the `.out` file from the Code Composer project will be converted to a `.a0` file using a hex converter and a `.cmd` file. The `.a0` file is a readable ASCII version of the `.out` file. Then this intermediate `.a0` file is converted to a `.bin` file using a hex to bin converter (hex2bin.exe) provided in your release. To accomplish this conversion quickly and easily, Innovative Integration has provided a template (`application.mk`) that can be modified to convert your file. This template is located in the target DSP card's root directory.

To convert your `.out` file into a `.bin` file to be burned to the Flash ROM, proceed as follows:

1. To start off, you must have a viable `.out` file (ie. yourfile.out).

2. Next you will have to create a `.mk` file for the out to bin conversion, using the template provided `application.mk`.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- Copy the template, `application.mk` file (in `C:\<I.I. Target Board>`) to the directory where `yourfile.out` resides and rename it to `yourfile.mk`.

- Open this new file with a text editing program like Notepad.

3. Edit the new .mk file as follows:

   - Change the OUTPUT and OUTPUT_BASE to the desired output name for the bin file. For example, to convert yourfile.out to yourfile.bin, edit this template file in the following manner:

     OUTPUT = yourfile.out

     OUTPUT_BASE = yourfile.

   - You must also be sure that the application.cmd and the hex2bin files are found by the new convertyourfile. To do this:

     a. Change the HEX_ARGS to point the where the application.cmd file is (ie if you have copied this file to the ..examples\target directory, then you must change HEX_ARGS = ..\..\application.cmd.

     b. Similarly, you must point to the location of the hex2bin file in the HEX2BIN variable.

   - Then save the changes and close the editing program.

Note: It is always a good idea to remove or rename any old `.bin` file of the same name to ensure a new .bin file is created.

4. Open a DOS prompt and CD to the directory where the yourfile.out and the Convertyourfile.mk files reside. Type: **nmake -f yourfile.mk** and press `<Enter>`. This will create the yourfile.bin in the same directory.

SBC67 and SBC6711 Targets.

Run the supplied applet PromImage.exe. Browse to your application .out file then click the Make button to generate the image file. The auto-generated image file will have the same name as the input .out file, but will be named with the .bin extension. This file is suitable for embedding into sector one of the flash ROM on the target board.

Note: SBC67x must be equipped with version 1.2 or later of Talker in order to use these image files.

The Flash Burn Utility

The Flash ROM programming utility `(BURN.EXE)` provides the capability to embed debugged application code within the flash ROM onboard Target DSPs which feature FLASH ROM. Currently, this includes each of the Innovative Integration single board computers (the SBC31, SBC32, SBC54,

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

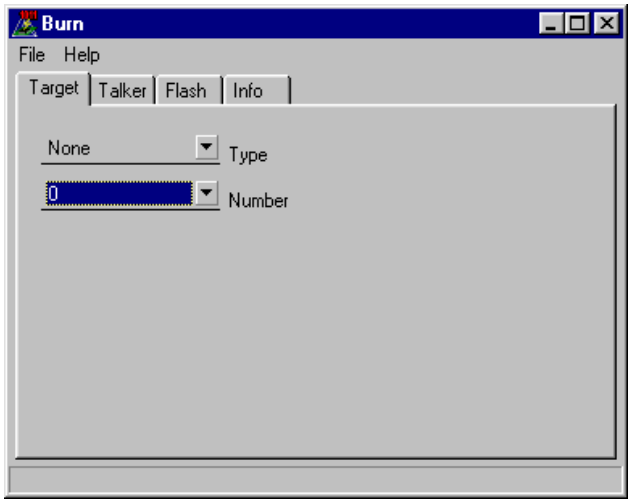
SBC62, SBC67, and SBC6711) products. Refer to the hardware manual for your DSP target to determine whether FLASH ROM is available on your system.

The utility supports both the 29F010 and 29F040 ROMs and provides special support for users with JTAG links between the host and the target in order to support programming.

The PROM utility contains only two menus, File and Help plus four tabbed page controls labelled Target, Talker, Flash, and Info. Currently, the Help menu is used to invoke help contents and the application About box, which indicates the revision of the application. The File menu provides access to the Options dialog which controls the timeout on JTAG loads and flash erase byte values (discussed later). In addition to the ability to convert Ascii hex text files into binary files, suitable for consumption by the Burn utility. The following is a brief description of the functions available under each of the primary tabs.

Target Page

The Target page contains two controls used to select the DSP type being used and its target number, as shown below.



Click on the Number combo control and select the number of the DSP target you're using (usually 0). It is important to remember that for single-boards targets, zero corresponds to COM1 and target one corresponds to COM2, ect. Next, click on the Type combo control to display a list of supported targets, then select your DSP type.

After you have selected the DSP type, the applet will attempt to open the target device driver to establish communication with the target. The status of this attempt is displayed in the status bar at the bottom of the Burn applet window. If the driver is successfully opened (Target Open OK), click on the Talker tab to proceed. If the target driver does not successfully open, check the parameters being requested

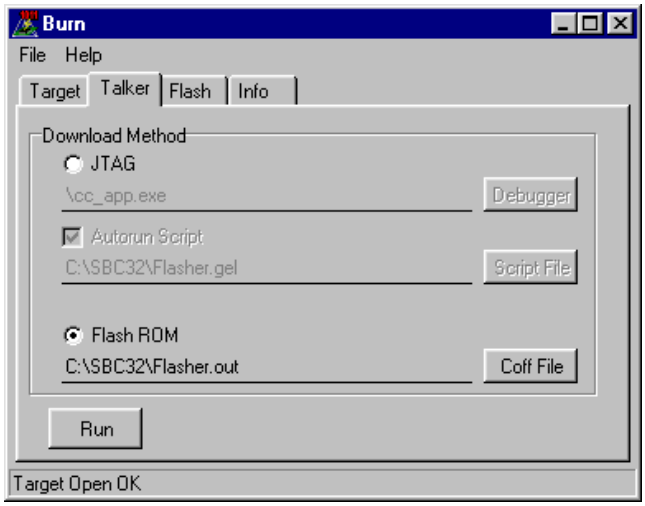
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

and the installation of your DSP. Also, close any other open applications (such as the terminal emulator.EXE, which could have previously opened the DSP device driver. Due to Windows restrictions, only a single application may open a com port device at a time.

Talker Page

To support all single-board DSP targets similarly, BURN must download a simple DSP support executable to the target before an embedded application may be burned into the Flash ROM. This executable is `FLASHER.OUT`, located in the root directory of the Zuma Toolset. This small, pre-written executable provides all of the Flash I/O access functions necessary to allow the BURN utility to interact with the target Flash ROM.

The Talker page contains a number of window controls used to specify the mechanism to use to deliver the `FLASHER.OUT` support executable into the target DSP. Two methods are currently supported, JTAG and downloading via the existing Talker in Flash ROM.



You must use the JTAG download method whenever the target DSP Flash ROM does not contain a viable (bootable) image of the Talker program. In the default factory condition, the target contains a bootable Talker, which is used by application programs through the DLL and device driver to download executable programs to the target DSP. If this Talker image has been erased or corrupted, you must use the JTAG download method.

If the Talker is still viable in Flash, then you may use either of the download method. The Talker method is the preferred method, especially in circumstances where a JTAG debugger is unavailable, such as when performing in-the-field software updates.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

JTAG Download. To download using JTAG, click the JTAG radio button and then click the Debugger button to browse to the location of your JTAG debugger software (usually `c:\composer\cc_app.exe` if you're using Code Hammer or `c:\ti\cc\bin\cc_app.exe` if you're using Code Composer Studio) and click `<Open>`. Next, click on the Script File button to browse to the location of the `Flasher.gel` file located in the root directory of the board-specific Zuma libraries (ie. `c:\SBC6711\Flasher.gel`) and click `<Open>`.

Under most circumstances, you should enable the Autorun Script check box as well. With this feature enabled, the specified Code Composer GEL script file is automatically executed as the debugger is started. A default script file has been provided and is located in the root directory of the Zuma Toolset installation. This GEL file automatically initializes the target DSP, downloads the `Flasher.out` support executable and launches it using the debugger Run Free command.

In rare circumstances, you may elect to disable the Autorun Script option. When this feature is disabled, BURN will automatically launch the debugger when you click the Run button, but you must manually load and run the `Flasher.out` executable. For example, this option is useful when using a debugger other than Code Hammer.

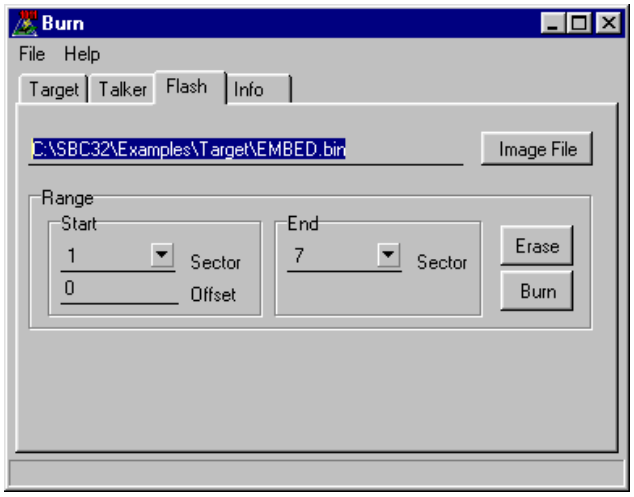
Flash ROM Download . To download using the Talker in flash ROM, click on the Flash ROM radio button and then click on the Coff File button to browse to the location of the `Flasher.out` file. This file is usually located in the root of the board-specific Zuma libraries directory (ie `c:\SBC6711\Flasher.out`).

Downloading the Flash Support Code. Close Code Composer, if you had been using the debugger, click on Debug|Run Free before exiting. To download the flash support executable, `Flasher.out`, click on the Run button. The program will deliver the `Flasher.out` executable image into the Target using the selected method. When the support code has been download and is running, the status bar will display Flash Info read OK. If you have difficulty running the support executable, refer to the "Common Problems" section for troubleshooting suggestions.

Flash Page

When the support code has been downloaded and is running, the status bar will display "Flash Info read OK". Next, you may burn binary images into the Flash ROM of the target DSP. Click on the Flash tab to change to the Flash Burn page (seen below).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Click on the Image File button to select the binary image file that is to be burned into the DSP flash. Note that BURN is incapable of burning DSP COFF executables (.OUT files) directly. You must use the other tools in the Zuma Toolset (specifically CodeWright, TIDeps and the TI HEX Conversion Utility) to create a viable binary image before attempting to burn an application into Flash ROM. Example embed-able projects for each II single-board target are available on the Innovative web site at www.innovative-dsp.com.

Controlling Region of Erasure. Adjust the Start Sector and End Sector combo boxes within the Start and End group boxes to control the location within the target flash in which the image is to be placed. Note that this controls only the range of sectors erased prior to attempting to burn the application into Flash.

The number of sectors actually needed is determined by the size of the binary image file. The BURN applet performs a range check to insure that your image will actually fit into the erased sectors. You must be sure to erase a sufficient number of sectors to contain your entire application image.

Re-burning the Talker Image. To burn or re-burn the factory Talker image, select 0 as both the starting and ending Sector. On all targets except C32-based targets, select 0 as the Offset. On C32 targets, the Offset should be set to 0x1000.

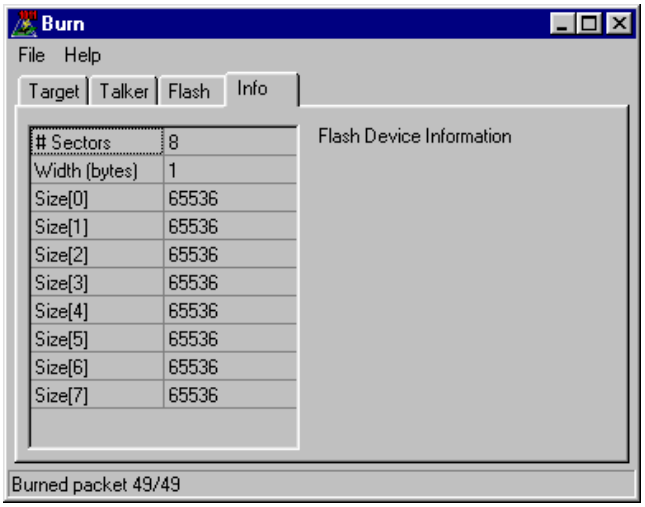
Warning: Avoid burning application code other than the factory Talker image into sector zero. On some targets, notably the SBC62, it may be impossible to initialize the JTAG debugger on a target board containing an invalid boot image. This situation can only be corrected by shipping the target DSP back to the factory for rework.

Burning a User-Written Application. To burn a user-written application, select 1 as the starting Sector and adjust the ending sector to the uppermost available sector, less any reserved sectors used as data storage within your DSP application program. The Offset field should be set to zero for all target board types.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Info Page

Detailed autoselect information read from the Flash ROM is available on this Info Tab. This information is utilized internally by the BURN applet. You need not enter information on this tab.



Example Burn Sequence

The following is a typical example of burning a binary file (`.bin`) into the target card's Flash ROM.

1. Run the Flash ROM programming utility (BURN.EXE) from the `Start Menu/Programs/<Target Board>/Burn`.

2. Click on the Target Tab.

   - Then click on the Number combo box and select the number of the DSP target you are using (usually 0).

   - Next, click on the Type combo box and select the target DSP type. (At the bottom of the display, "Target Open OK" should be displayed)

3. Click on the Talker Tab.

   - In this example, from the two download methods, the Flash ROM method is selected. Click on the `<Coff File>` button and browse to the root directory of the Zuma Toolset and find the `Flasher.out` file.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- Then, click on the <Run> button. (At the bottom of the display, “Flash info read OK” should be displayed)

4. Click on the Flash Tab.

   - Then, click on the <Image File> button and browse to the binary file that is to be downloaded to the target DSP flash.
   - Next, adjust the Start Sector combo box to 1 and the End Sector combo box to 7.
   - Then, click on the <Burn> button to start the download. The dowloading progress should be displayed. (At the bottom of the display, “Burn Completed” should be displayed)

5. Testing the burned .bin file.

SBC32:

- To run the application from the target DSP, turn the power off, disconnect the JTAG & serial port, and turn the power back on. Alternately with the serial cable attached, you may click on the Boot Embedded Application speed button on the terminal emulator to launch the embedded application, rather than the embedded Talker.

SBC6x:

- To run the application from the target DSP, turn the power off, install jumper (JP24) and turn the power back on. Alternately with the serial cable attached and JP24 installed, you may click on the Reset Dsp speed button on the terminal emulator to launch the embedded application, rather than the embedded Talker.

SBC6711:

- To run the application from the target DSP, turn the power off, install jumper (JP11) and turn the power back on. Alternately with the USB cable attached and JP11 installed, you may click on the Reset Dsp speed button on the terminal emulator to launch the embedded application, rather than the embedded Talker.

Common Problems when Embedding Code

There are several problems frequently encountered when ROMing code. If you encounter one of the symptoms listed below, attempt the corrective action listed before calling Innovative technical support:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Symptom	Possible Corrective Actions
Inadvertently overwritten the Talker in sector 0.	Re-burn the Talker (per above).
Code Hammer cannot initialize the target.	Verify that the I/O address specified in the Code Composer setup is correct and that the JTAG board is properly connected to the DSP board. Configure and run the Innovative JTAGDIAG.EXE utility to reset the debugger hardware. Code Composer Studio users should also run the XDS510 Reset Utility (using the same I/O address as entered into JTAGDIAG.EXE) to initialize the Studio Debugger. The target may be held in reset. Verify that the boards device driver is installed and operational. Code Composer versions prior to 4.01 are incapable of communicating with C67xx processors. Also, v4.01 is better able to recover from invalid images burned into sector 0 on SBC6711 DSPs. Contact I.I. for upgrade information.
The embedded application will not boot.	The target is being held in reset. For single-board targets, disconnect serial port #1 to the host and reboot. If serial port #1 must be connected, insure that the DTR output from the PC is de-asserted (since it controls target reset). Further, insure that the SBC's CTS serial input line is de-asserted since the Talker cannot launch a co-existing embedded application if CTS is asserted during cold-boot. On the SBC62, insure that the boot jumper JP24 is installed. You burned the application at an incorrect offset or starting at a sector other than 1. The application burn image was created improperly. Inspect the `.A0` file and insure that the target image has an appropriate boot record. Be sure to convert the `.A0` file into binary using the Options dialog for consumption by the Burn applet. On the SBC62 and SBC54 targets, be sure to locate the .const and cinit sections into the ROM so that they may be resurrected at boot time.
The application works properly from within Code Composer Studio but does not run (or is erratic) when embedded.	The Target may still be held by the JTAG. Run the JTAG Diagnostic utility and then select the `<Reset>` button to release the target. The `.bss` section of your application is not automatically initialized to zero. Modify your linker command file to zero-fill the `.bss` section. Your embedded application may be too large. Use the Coff Dump utility to inspect the size of the .OUT file to verify that it can fit into the seven residual 16 kByte sectors in the AMD 29F010 (64 kByte sectors in the 'F040).

TABLE 1. Common Problems when Embedding Code in Flash ROM

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CHAPTER 6

Developing Target Code

Introduction

The Innovative Integration (I.I.) Zuma Toolset allows users of I.I. DSP processor boards to develop complete executable applications suitable for use on the target platform. The environment suite consists of the TI Optimizing C Compiler, Assembler and Linker, the Code Composer debugger and code authoring environment as well as I.I.'s custom Windows applets (such as the `UNITERMINAL.EXE` terminal emulator).

Code Composer Studio is the default package used to automate executable build operations within Innovatives Zuma Toolsets, simplifying the edit-compile-test cycle. Source is edited, compiled, and built within Code Composer Studio, then downloaded to the target and tested within either the Code Composer Studio debugger or via the Zuma terminal emulator.

On C6x platforms, such as Innovatives M6x, SBC6x, SBC6711, and Quatro6x, Code Composer Studio may be used for both code authoring and code debugging. Details of constructing projects for use on Innovative DSP platforms using Code Composer Studio are provided in this chapter.

Do not confuse the creation of target applications (code running on the target DSP processor) with the creation of host applications (code running on the host platform). The TI tools generate code for the TI DSP processors, and are a separate toolset from that needed to create applications for the host platform (which would consist of some native compiler for the host processor, such as Microsoft's Visual C++ or Borland Builder C++ for IBM compatibles). To create a completely turn-

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

key application with custom target and host software, *two* programs must be written for *two* separate compilers. While I.I. supports the use of Microsoft C/C++ for generation of host applications under Windows with sample applications and libraries, we do not supply the host tools as part of the Development Environment. For more information on creating host applications, see the section in this manual on host code development.

This section supplies information on the use of the development environment in creating custom or semicustom target DSP software. It is not intended as a primer on the C language. For information on C language basics, consult one of the C primer books available at your local bookstore. The definitive reference to the C language is The C Programming Language, by B. Kernighan and D. Ritchie (Prentice Hall. Englewood Cliffs, NJ. 1988).

Components of Target Code (.c, .asm, .cmd)

In general, DSP applications written in TI C require at least two files: a `.c` file (or "source" file) containing the C source code for the application, and a `.cmd` file (or "linker command" file) which contains the target-specific build data needed by the linker. There may also be one or more `.asm` assembler source files, if the user has coded any portions of the application in assembly language.

Edit-Compile-Test Cycle using Code Composer Studio

Nearly every computer programming effort can be broken down into a three step cycle commonly known as the edit-compile-test cycle. Each iteration of the cycle involves editing the source (either to create the original code or modify existing code), followed by compiling (which compiles the source and creates, or builds, the executable object file), and finally downloading and testing the result to see if it functions in the desired fashion. In the Innovative Intergration development system these stages are accomplished within the Code Composer program.

By using Code Composer Studio, these stages of the programming cycle are accomplished entirely within the integrated Studio environment. The project features of Code Composer Studio support component file editing and compilation stages, along with allowing the executable result to be downloaded and tested on the target hardware. This fully integrated programmers environment is more user-friendly then the basic command line interface, which comes standard with the TI tools.

A Simple Code Composer Studio Project

The following sequence illustrates the creation of a project to build the "`Hello World!`" program from within Code Composer Studio.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

First, start Code Composer Studio. Select `Project | New` from the Project menu and you will see the following dialog:

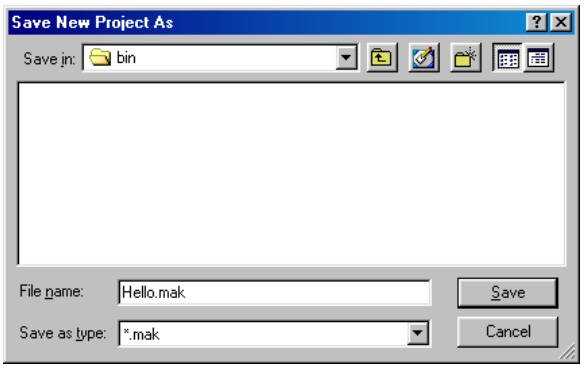


FIGURE 12. Creating a New Project in Code Composer Studio

Browse to the directory in which you would like to create the new project (your working directory) and then type the name of the new project. In this example, the working directory is c:\ti\bin and the project name is hello.mak. In the standard developers package, you may browse into the `<I.I. Target Board>\EXAMPLES\TARGET` directory.

Next, open the `Project | Add Files` to Project dialog box to add files to the project. Add the `HELLO.C file from the C:\<I.I. Target Board>\Examples\Target` directory and the `GENERIC.CMD` file from the `C:<I.I. Target Board>` directory to the project. Remember to choose the correct file type then when you have selected the file to add, click <**Open**> button.

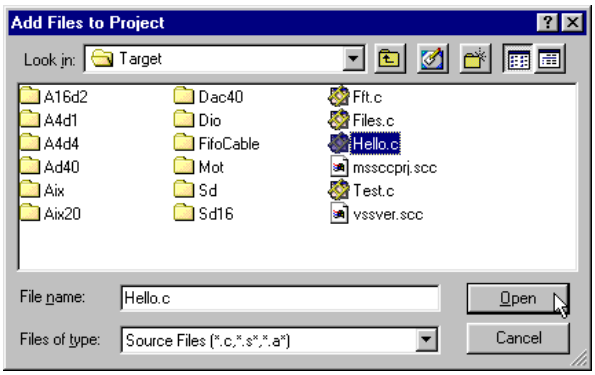


FIGURE 13. Adding Files to a Code Composer Studio Project

SBC6x Only: When building an application to be burned into the Flash ROM, the `EMBED.CMD` file (located `C:\<I.I. Target Board>\Examples\Target)` should be added to the project instead of `GENERIC.CMD` (refer to Chapter 7 “Building Flash Programmable Application”).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

It is imperative that you add an appropriate command file to the Code Composer Studio project. The `generic.cmd` command file describes the memory map of the target hardware, without which the linker will be unable to place executable sections into appropriate memory regions for debugging. That is, the memory map for the target DSP specified in the `generic.cmd` file will be used to link the project output file. If you wish, you may copy the contents of the `generic.cmd` file (located in the root of the Zuma toolset) into your working directory, rename it appropriately and add the modified cmd file to your project instead.

The library files will be required, but do not add them directly into the project like the hello.c and generic.cmd files. Rather, manually type the desired libraries needed to link the project into the Project | Options | Linker tab when instructed to do so later within this chapter.

Next, you may optionally open the files in the project by double-clicking on their names within the Project window.

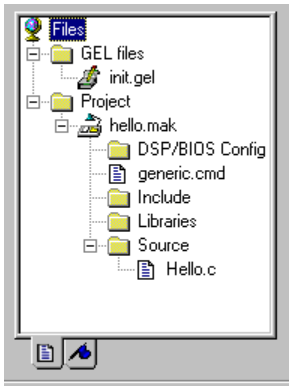


FIGURE 14. Code Composer Studio Project Window

Next, you must configure the project compiler settings so that when Hello.c is compiled, the appropriate memory model and switches are used.

Build Options (M62, Q62, SBC62 Boards)

Click on `Project | Build Options` to open the Build Options dialog box, then click on the Compiler tab to show the current compiler options. The required Build options are described in the following text, all other settings are optional and are described in the Code Composer Studio documentation.

Configure the compiler options to use the following settings: In the category column, choose Basic and click on the Generate Debug Info combination box and select “Full Symbolic Debug”. Then click on the Opt Level combination box and select “-02: Function”. Next, in the category column, choose Advanced, click on the Endianness combination box and select “Big Endian”. Then click on the Mem-

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ory Models combination box and select “Far Calls & Aggregate Data”. In the category column, choose Feedback, and click on the Banners combination box and select “No Banner”. In the category column, choose Files and in the Obj Directory box type “C:\M6x\Examples\Target”. Again in the category column, choose Preprocessor and in the Include Search Path box type C:\M6x\Include\Target”. When finished, the compiler dialog screen should look very similar like one below.

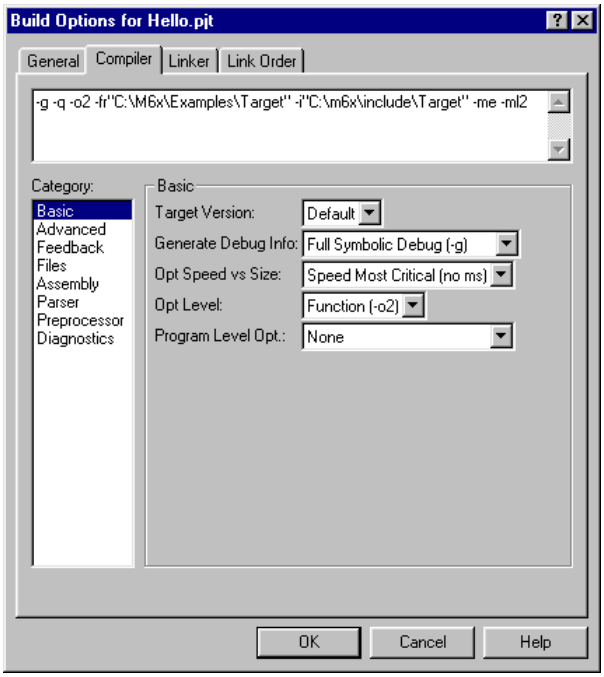


FIGURE 15. Code Composer Studio Compiler Build Options

Now, click on the Linker tab and configure the linker build options. In the category column, choose Basic. In this screen, make sure the Exhaustively Read Libraries has been selected. Then in the Output Filename box and Map Filename box type the project name. Click on the Autoint Model combination box and select “Run-time Autoinitialization”. Next, set the Heap Size to 0x400 bytes and the Stack Size to 0x800 bytes, the Heap & Stack size may change depending on your application’s needs. In the Library Search Path, type “..\..\..\lib\Target. Now, add stdio.lib; periph.lib; dsp.lib; and rts6201e.lib into the Include Libraries edit box (in that order). Then on the Include Libraries edit box line, add either init.lib for non-DSB Bios applications or biosinit.lib for DSP Bios applications, between the periph.lib and dsp.lib entries. When finished, the linker dialog screen should look like one below.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

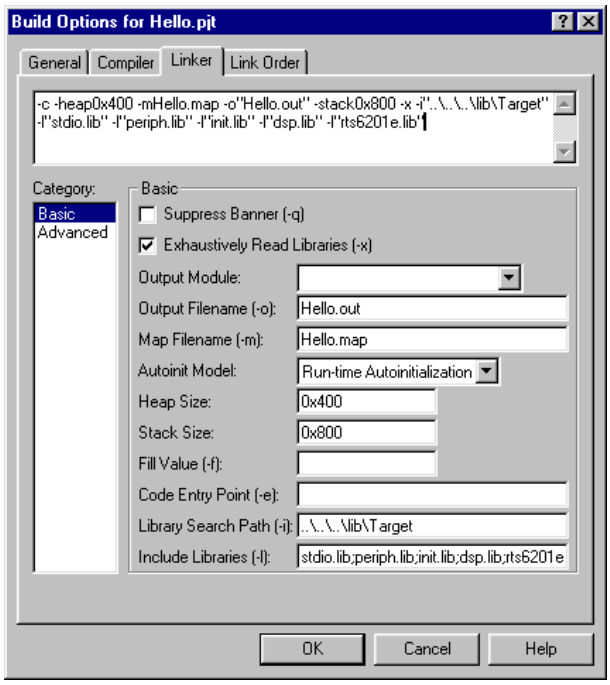


FIGURE 16. Code Composer Studio Linker Build Options

Once all the build options have been set, rebuild your project by clicking Project|Rebuild All in the Code Composer Studio menu bar. If errors are encountered in one or more source files, they are listed in the output window. You may visit and repair each error by either double clicking on each error in the Output window.

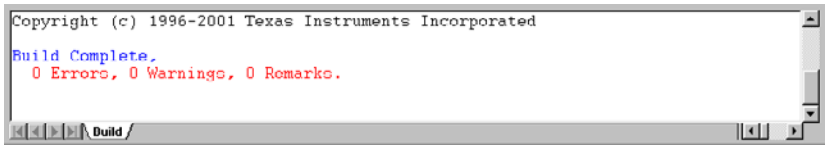


FIGURE 17. Code Composer Studio Build Results Window

Build Options (M67, Q67, SBC67 Boards)

Click on `Project | Options` to open the Build Options dialog box, then click on the Compiler tab to show the current compiler options. The required Build options are described in the following text, all other settings are optional and are described in the Code Composer Studio documentation.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Configure the compiler options to use the following settings: In the category column, choose Basic and click on the Target Version combination box, and select “670x”. Click on the Generate Debug Info combination box and select “Full Symbolic Debug”. Then click on the Opt Level combination box and select “-02: Function”. Next, in the category column, choose Advanced, click on the Endianness combination box and select “Big Endian”. Then click on the Memory Models combination box and select “Far Calls & Aggregate Data”. In the category column, choose Feedback, and click on the Banners combination box and select “No Banner”. In the category column, choose Files and in the Obj Directory box type “C:\M6x\Examples\Target”. Again in the category column, choose Preprocessor and in the Include Search Path box type C:\M6x\Include\Target”. When finished, the compiler dialog screen should look very similar like one below.

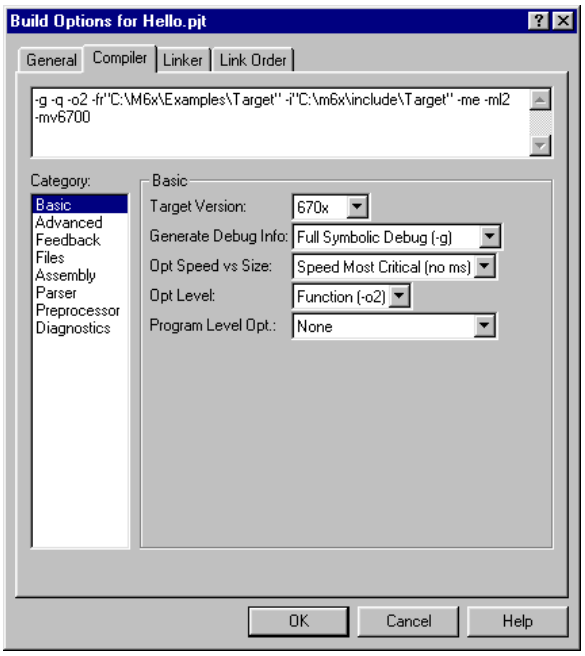
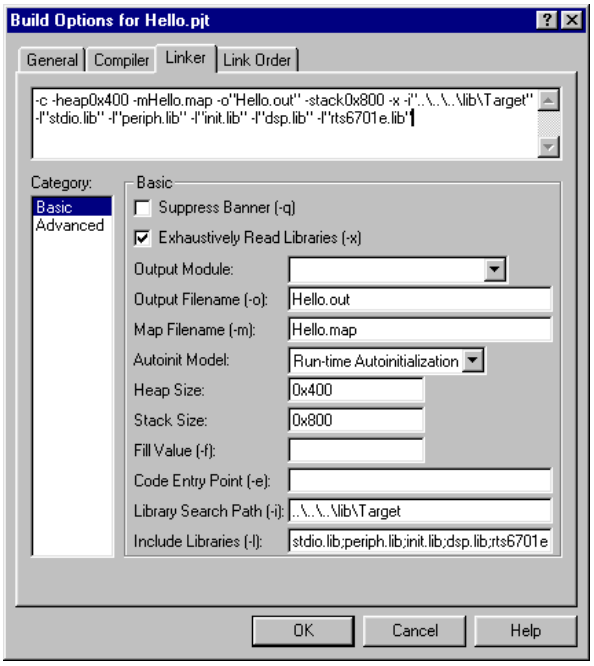


FIGURE 18. Code Composer Studio Compiler Build Options

Now, click on the Linker tab and configure the linker build options. In the category column, choose Basic. In this screen, make sure the Exhaustively Read Libraries has been selected. Then in the Output Filename box and Map Filename box type the project name. Click on the Autoint Model combination box and select “Run-time Autoinitialization”. Next, set the Heap Size to 0x400 bytes and the Stack Size to 0x800 bytes, the Heap & Stack size may change depending on your application’s needs. In the Library Search Path, type “..\..\..\lib\Target. Now, add stdio.lib; periph.lib; dsp.lib; and rts6201e.lib into the Include Libraries edit box (in that order). Then on the Include Libraries edit box line, add either init.lib for non-DSB Bios applications or biosinit.lib for DSP Bios applications, between the periph.lib and dsp.lib entries. When finished, the linker dialog screen should look like one below.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Once all the build options have been set, rebuild your project by clicking Project|Rebuild All in the Code Composer Studio menu bar. If errors are encountered in one or more source files, they are listed in the output window. You may visit and repair each error by either double clicking on each error in the Output window.

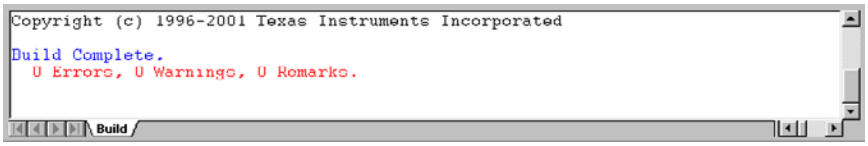


FIGURE 19. Code Composer Studio Build Results Window

Build Options (SBC6711 Boards)

Click on `Project | Build Options` to open the Build Options dialog box, then click on the Compiler tab to show the current compiler options. The required Build options are described in the following text, all other settings are optional and are described in the Code Composer Studio documentation.

Configure the compiler options to use the following settings: In the category column, choose Basic, click on the Target Version combination box, and select “671x”. Then click on the Opt Level combina-

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

tion box and select “-02: Function”. Next, in the category column, choose Advanced, click on the Endianness combination box and select “little Endian”. Then click on the Memory Models combination box and select “Far Calls & Aggregate Data”. Again in the category column, choose Feedback, and click on the Banners combination box and select “No Banner or File Names”. Under the Files category the Obj Directory can be added. When finished, the compiler dialog screen should look very similar like one below.

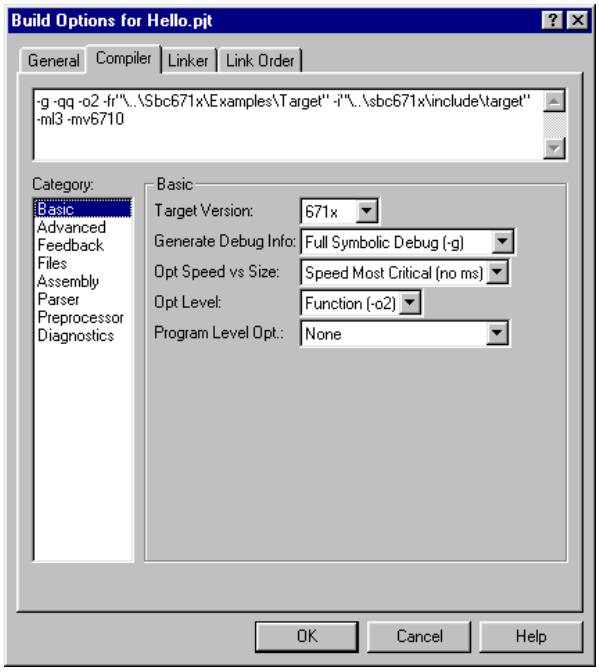


FIGURE 20. Code Composer Studio Compiler Build Options

Next, click on the Linker tab and configure the linker build options as follows. In the Output Module combination box select “Relocatable Executable”. Then, click on the Autoint Model combination box and select “Run-time Autoinitialization”. Next, set the Heap Size to 0x400 bytes and the Stack Size to 0x800 bytes, the Heap & Stack size may change depending on your application’s needs. Make sure the Exhaustively Read Libraries has been selected. Now, add periph.lib, init.lib, stdio.lib, dsp.lib, and rts6700.lib into the Include Libraries edit box (in that order). Then on the Include Libraries edit box line, add either init.lib for non-DSB Bios applications or biosinit.lib for DSP Bios applications, between the periph.lib and stdio.lib entries. When finished, the linker dialog screen should look like one below.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

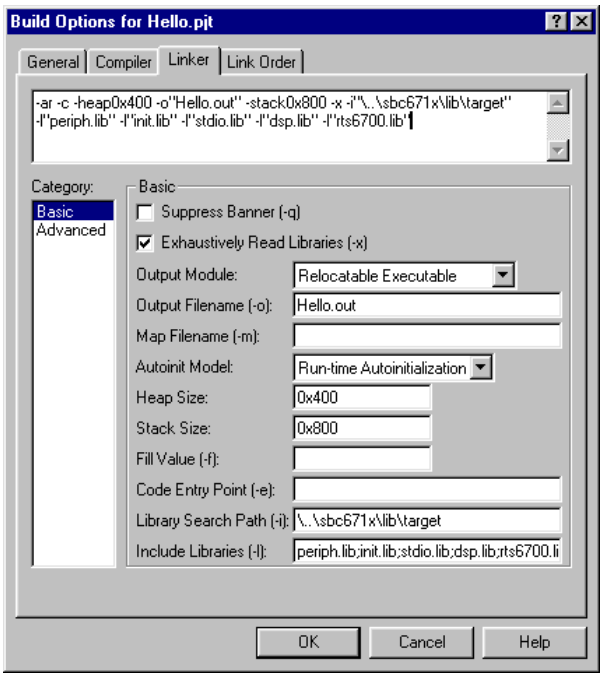


FIGURE 21. Code Composer Studio Linker Build Options for Non-Bios Project

Once all the build options have been set, rebuild your project by clicking Project|Rebuild All in the Code Composer Studio menu bar. If errors are encountered in one or more source files, they are listed in the output window. You may visit and repair each error by either double clicking on each error in the Output window.

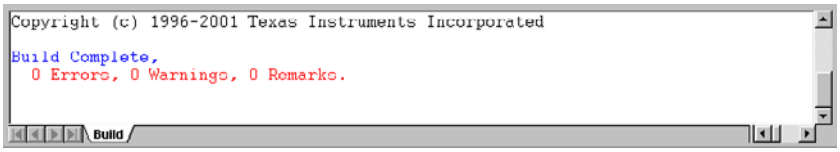


FIGURE 22. Code Composer Studio Build Results Window

Build Options (M44 Boards)

Click on `Project | Options` to open the Build Options dialog box, then click on the Compiler tab to show the current compiler options.

To configure the compiler options type the following command line in the compiler option box: "**`-gq -v40 -mn -o2 -x2 -frC:\M44\Examples\Target`**" When finished, the compiler dialog screen should look exactly like one below.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

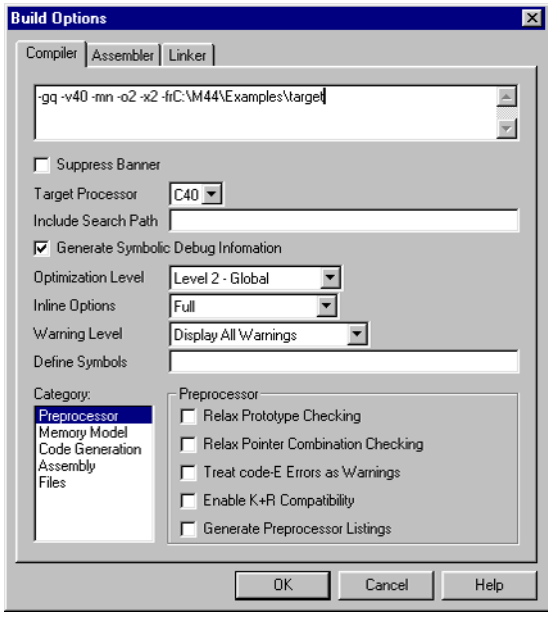


FIGURE 23. Code Composer Studio Compiler Build Options

Next, click on the Assembler tab and configure the assembler build options. In the assembler option box type the assembler build option "**-r**". When finished, the assembler dialog screen should look like one below.

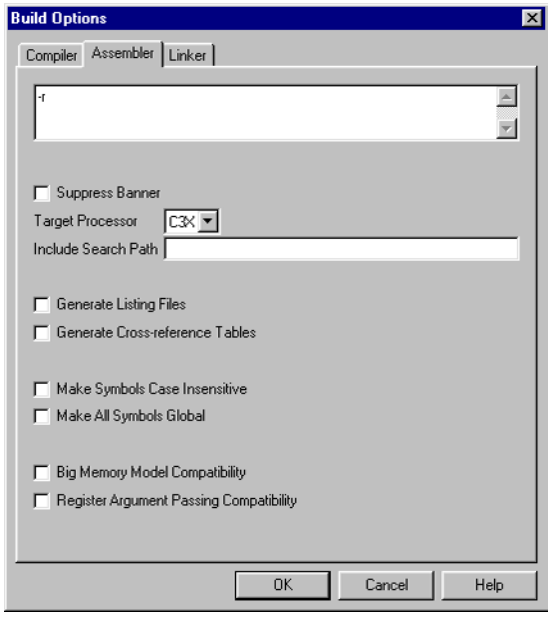


FIGURE 24. Code Composer Studio Assembler Build Options

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Finally, click on the Linker tab and configure the linker build options as follows. In the Output Module combination box, select “Relocatable Executable”. In the Output Filename box, type in the name of the output file. Then in the Map Filename combination box, select “ROM Autoinitialization Mode”. Next, set the Heap Size to 0x400 bytes and the Stack Size to 0x200 bytes, the Heap & Stack size may change depending on your application’s needs. Make sure the Exhaustively Read Libraries have been selected. Now, add stdio.lib; periph.lib; dsp.lib; prts40.lib; and rts30.lib into the Include Libraries edit box (in that order). When finished, the linker dialog screen should look like one below.

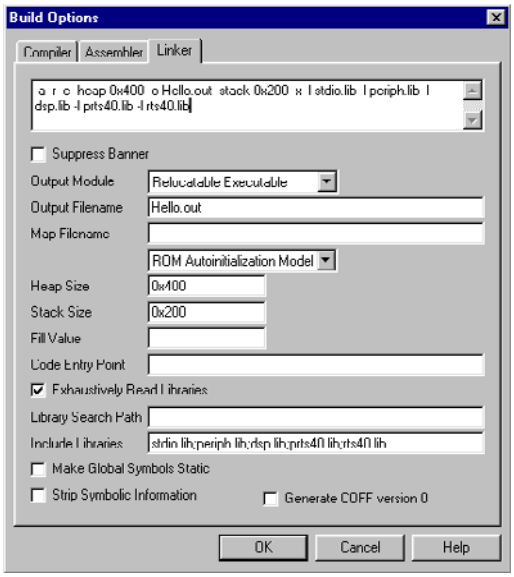


FIGURE 25. Code Composer Studio Linker Build Options

Once all the build options have been set, rebuild your project by clicking Project|Rebuild All in the Code Composer Studio menu bar. If errors are encountered in one or more source files, they are listed in the output window. You may visit and repair each error by either double clicking on each error in the Output window.

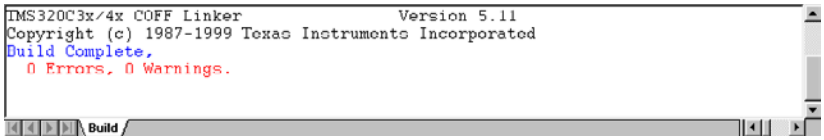


FIGURE 26. Code Composer Studio Build Results Window

Build Options (SBC32 Boards)

Click on `Project | Options` to open the Build Options dialog box, then click on the Compiler tab to show the current compiler options.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To configure the compiler options type the following command line in the compiler option box: "**`-gq -v32 -o2 -x2 -frC:\SBC32\Examples\Target`**" When finished, the compiler dialog screen should look exactly like one below.

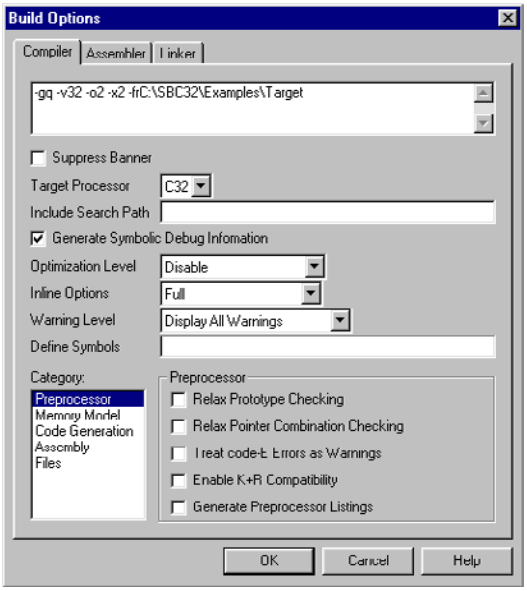


FIGURE 27. Code Composer Studio Compiler Build Options

Next, click on the Assembler tab and configure the assembler build options. In the Target Processor combination box, select "C3X". Then type the last assembler build option "**`-r`**" in the assembler option box. When finished, the assembler dialog screen should look like one below.

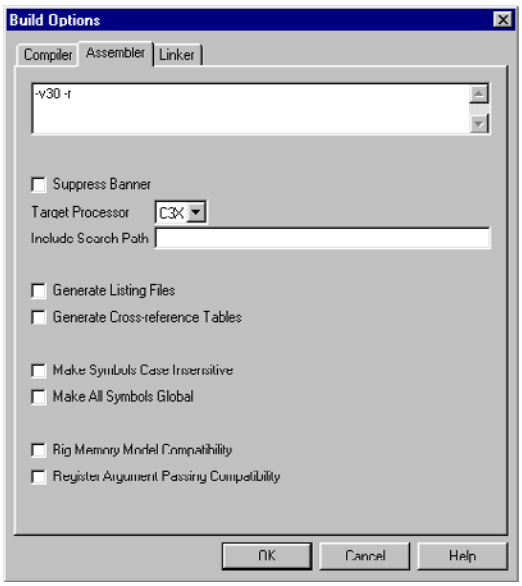


FIGURE 28. Code Composer Studio Assembler Build Options

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Finally, click on the Linker tab and configure the linker build options as follows. In the Output Module combination box select “Relocatable Executable”. In the Output Filename box, type in the name of the output file. Then in the Map Filename combination box, select “ROM Autoinitialization Mode”. Next, set the Heap Size to 0x400 bytes and the Stack Size to 0x200 bytes, the Heap & Stack size may change depending on your application’s needs. Make sure the Exhaustively Read Libraries have been selected. Now, add stdio.lib; periph.lib; dsp.lib; and rts30.lib into the Include Libraries edit box (in that order). When finished, the linker dialog screen should look like one below.

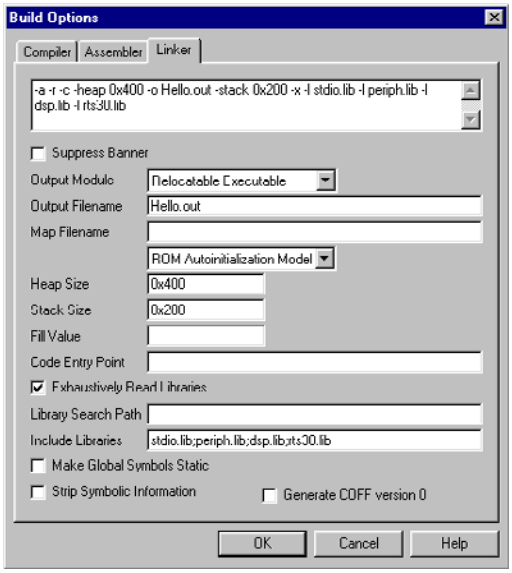


FIGURE 29. Code Composer Studio Linker Build Options

Once all the build options have been set, rebuild your project by clicking Project|Rebuild All in the Code Composer Studio menu bar. If errors are encountered in one or more source files, they are listed in the output window. You may visit and repair each error by either double clicking on each error in the Output window.

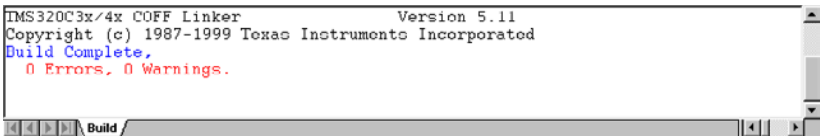


FIGURE 30. Code Composer Studio Build Results Window

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Automatic projectfile creation

When a project is created, opened, modified, built or rebuilt, the Code Composer Studio dependency generator automatically generates a project makefile (named `<project file>.pjt`, located in the project directory), which is capable of rebuilding the project's output file from its components.

This file is automatically submitted to the internal make facility whenever you click on build or rebuild within Code Composer Studio. The make facility automatically constructs the output file by recompiling the out-of-date source files including the dependencies contained within those source files.

Rebuilding a Project

It is sometimes necessary to force a complete rebuild of an output file manually, such as when you change optimization levels within a project. To force a project rebuild, select `Project | Rebuild All` from the Code Composer Studio menu bar.

Running the Target Executable

The `hello` program is very simple, only printing the single line "Hello, World" to the terminal emulator before waiting to echo any keystrokes and exiting. Bring up the "Hello, World" source file edit screen. Scroll down the source file by using cursor down button until you reach the call to `printf()`, which looks like the following:

```
printf("Hello, World\n");
```

Change the output string to read "`Hello, Brave New World\n`". You can now compile the new version by executing Build from the Project menu (or by clicking on its toolbar icon). This causes Code Composer Studio to start the compiler, which produces an assembly language output. The compiler then automatically starts the assembler, which produces a `.obj` output file (`hello.obj`). Code Composer Studio then invokes the TI Linker using the `generic.cmd` file, which is located in the root board directory. This rebuilds the executable file using the newly revised `hello.obj` . If no errors were encountered, this process creates the downloadable COFF file `hello.out`, which can be run on the target board. At this point, the program may be run using the terminal emulator applet, which may be invoked using the UniTerminal shortcut located within the target board program group created during the Zuma Libraries installation process. In the terminal emulator, download the `hello.out` file. The program runs and outputs the message "Hello, Brave New World" to the terminal emulator window.

If errors are encountered in the process, Code Composer Studio detects them and places them in the build output window. If the error occurred in the compiler or assembler (such as a C syntax error), the cursor may be moved to the offending line by simply double-clicking on the error line within the build output window, and the error message will be displayed in the Code Composer Studio status bar. If the linker returns a build error, the build output window shows the error file. From this information, the linker failure can be determined and corrected. For example, if a function name in a call is misspelled, the linker will fail to resolve the reference during link time and will error out. This error will be displayed on the screen in the build output window.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Note: Be sure to start the terminal emulator **BEFORE** starting Code Composer, to avoid resetting the DSP target in the midst of the debugging session. If the terminal emulator is not yet running and you wish to run the Hello object file, perform the following steps.

1. Execute Debug | Run Free to logically disconnect the DSP from the debugger software.
2. Terminate the Code Composer Studio application.
3. Invoke the terminal emulator "UniTerminal" application.
4. Restart the Code Composer Studio application.

This outlines the basics of how to recompile the existing sample programs within the Code Composer Studio environment.

Anatomy of a Target Program

While not providing much in the way of functionality, the `hello` program does demonstrate the code sequence necessary to properly initialization the target. The exact coding, however, is very specific to the I.I. C Development Environment, target boards, and is explained in this section in order to acquaint developers with the basic syntax of a typical application program.

Here we examine the M6x version of the `hello` program example. Although the source is not necessarily identical to that of `hello` for the other targets, it is typical of the overall structure of the typical application program designed under the development environment.

```
/*
*       HELLO.C
*       Test file/program for target board.
*/

#include "periph.h"
#include "stdio.h"

main()
{
        int key;
        enable_monitor();
        clrscr ();

        printf("Hello World!\n");
        printf("\nEchoing keystrokes...\n");
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
        do
                {
                key = getchar();
                putchar(key);
                }
        while(key != ESC);

        monitor();
}
```

The two lines of the program that being with a "#" are #include statements, which include the header files for the peripheral and standard I/O libraries. These include prototypes for all the library routines as well as variable definitions and #define statements for the peripheral memory-mapping addresses. These #defines are especially important for those who wish to perform direct peripheral access, rather than using the peripheral libraries.

The `enable_monitor()` function will setup the standard monitor I/O interface and reset the terminal window. The next two lines perform basic standard I/O functions; clearing the terminal emulation screen and printing "Hello World!" & "Echoing keystrokes...". These two lines are where custom code should be inserted.

The following `getchar()`/`putchar()` sequence simply echoes keys typed at the terminal emulator back to the terminal display, until the Esc key is pressed. When Esc is pressed, the `monitor()` function effectively terminates the program, except that interrupts are still active and interrupt handlers (if they had been installed) would still execute properly.

The `hello` program is very simple, but it contains the basic components of a typical DSP application, as well as the initialization needed to interact with the terminal emulator.

Use of Library Code

Library routines can be compiled and linked into your custom software simply by making the appropriate call in the source and adding the appropriate library to the linker command file. Refer to the library reference in this manual for library location information on each function.

In general, user software needs to #include the relevant library header file in source code. The header files define prototypes for all library functions as well as definitions for various data structures used by the library functions. The file **stdio.h** should be included by programs using the standard I/O library, and the file `periph.h` should be included if a program uses functions in the peripheral library. The function definitions in the peripheral library reference note which library a particular function resides in, as well as the header file, which that be included for the function.

Compiling/Assembling/Linking Outside Code Composer Studio

Under certain circumstances, it may not be possible to use Code Composer Studio macro definitions to compile inside the editor. `COMPILE.BAT`, `ASSEMBLE.BAT`, and `LINK.BAT` are provided in the

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

%II_BOARD% directory and may be executed by typing their names followed by the source file on which they are to operate. For example, the file `mycode.c` can be compiled by typing:

```
compile mycode
```

at the DOS prompt. This causes the `COMPILE.BAT` script to start, which runs the compiler and generates the file `mycode.obj`, assuming no errors occurred. The `COMPILE.BAT` script also searches for the file `mycode.cmd` in the current directory. If the linker command file is found, then the linker is automatically run and the entire executable linked. If the command file is not found, processing stops with the generation of `mycode.obj`.

Assembly source (`mycode.asm`) may be assembled by typing:

```
assemble mycode
```

where the assembler is called and an object file generated.

Linking can also be performed. In this case the input file is not source code, but a linker command file (`mycode.cmd`):

```
link mycode
```

This line causes the linker to build the executable `mycode.out`, again assuming no errors have occurred during the process. Also, note that the `COMPILE.BAT` script will automatically link the executable if a linker command file of the same name exists.

In all the above cases, if any errors occur, an error file (`mycode.err`) is generated by the software tools. The `mycode.err` file contains the full console output of each of the tools. Any error that is generated by the tools will be recorded in this file.

The Next Step: Developing Custom Code

In building custom code for an application, Innovative Intergration recommends that you begin with one of the sample programs as an example and extend it to serve the exact needs of the particular job. Since each of the example programs illustrates a basic data acquisition or DSP task integrated into the target hardware, it should be fairly straightforward to find an example which roughly approximates the basic operation of the application. It is recommended that you familiarize yourself with the sample programs provided. The sample programs will provide a skeleton for the fully custom application, and ease a lot of the target integration work by providing hooks into the peripheral libraries and devices themselves.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CHAPTER 7

Developing Host Code

This section describes the Innovative Integration Windows host software development environment. The environment provides complete support for generating 32-bit Windows-compatible software, which is capable of controlling and communicating with Innovative Integration's DSP co-processor and data acquisition cards. Virtual device drivers (Windows 9x VxD or NT/2K Kernel Mode Driver) and dynamic link libraries (DLL) are included to provide an easy-to-use, portable low-level interface for the target hardware. Sample applications show how to call the DLL functionality and present basic interface examples with guidelines for on processor card control requirements and data movement.

Host software development is directly supported under either the Borland C++ Builder environment or the Microsoft MSVC environment for generating 32-bit Windows applications. Example application programs included in the development package are supplied with Borland project files, making program modification and regeneration as simple as possible. These programs utilize DspComponent, a VCL C++ component class provided in the Development package, to gain access to DLL features. Examples with similar functionality, utilizing an ActiveX wrapper derived from DspComponent are available for Microsoft Visual C++ users. See the online DspComponent.hlp file for a full description of the DspComponent.

Please Note: Only Windows application development is currently supported by the Developer's Package. Foreign operating systems, such as Unix and OS9 are not currently supported.

Dynamic Link Library

All target interactions takes place through calls to the supplied target DSP dynamic link library (DLL). This library supplies low-level functions for basic target board

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

control, including processor reset/run state, message passing via the board-specific mailbox registers, application downloading, and busmaster memory locking and access control.

The function calls available under the DLL are documented in the Host32.hlp Windows help file, provided in the Zuma Toolset. Sample applications (described below) provide working examples on how to interact with the card via host software.

Sample Host Programs

The DLL is capable of interacting with up to sixteen target DSP boards simultaneously by default (contact II if more targets are required). The DLL maintains a board-specific structure of information for each target, known as the `cardinfo` structure. An prototype of the cardinfo structure is located in the `\INCLUDE\HOST\` subdirectory in the `CARDINFO.H` file. An example is shown below.

```
//
//    cardinfo.h   --   definition of CARDINFO structure
//

#ifndef __CARDINFO_H__
#define __CARDINFO_H__

#include "ii_iostr.h"       // Common IO Driver/DLL Structures
#include "mailbox.h"        // Definition of MAILBOX structures

//
//  BoardInfo structure
//
typedef struct _BoardInfo
{
    ULONG        ProcessorCount;
    ULONG        DLL_Version;      // Version ID numbers
    ULONG        DrvVersion;
    ULONG        TalkerVersion;
    ULONG        CellSize;         // Target memory cell size, in bytes
    ULONG        CtlReg;           // Shadow of control register
    ULONG       FlashSectorSize; // Size of flash sectors, in bytes
    ULONG       FlashDeviceId;   // Flash device ID
    ULONG        QuietMode;        // Don't Display Messages if true
} BoardInfo;

//
//  InterruptInfo structure
//
typedef struct _InterruptInfo
{
    ULONG        IRQ;              // IRQ of attached interrupt
    HANDLE       Ring0Event;       // Ring 0 event handle
    HANDLE       Ring3Event;       // Ring 3 event handle
    void         (*Vector)(void *); // Virtual ISR function pointer
    void *      Context;         // Virtual ISR context pointer
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
 } InterruptInfo;

//
//  SerialInfo structure
//
typedef struct _SerialInfo
{
  LONG           In;         // Buffer for last character received
  LONG           ReadFlag;   // True when character received
  LONG           MbValue;    // Multi-byte value
  LONG           MbCtr;      // Multi-byte read state
  ULONG          RTS_state;  // Current state of the RTS output
  LONG          Bcr;    // Bus control register value for Flash access
  LONG           Reading;    // TRUE if currently reading a character
  OVERLAPPED     RxOverlap;  // Info used in asynch input
  OVERLAPPED     TxOverlap;  // Info used in asynch output
  COMMTIMEOUTS   Timeouts;   // Info for set/query time-out parameters
  DCB            Dcb;        // Device control block
} SerialInfo;

//
//  CARDINFO  structure
//
typedef struct _cardinfo

{
   ULONG           Target;         // Number of current target
   HANDLE          Device;         // Handle to Driver for device
   BoardInfo       Info;           // Board Info
   MAILBOX *       Mail;           // Talker Mailbox Array
   IoPortBlock     Port;           // Primary Port Block Information
   IoPortBlock     OpReg;          // Secondary Port Block Information
   MemoryBlock     DualPort;       // Shared Memory Area Information
   MemoryBlock     BusMaster;      // BusMaster Memory Information
   nterruptInfo    Interrupt;      // Interrupt Information
   SerialInfo      Serial;         // Serial Port I/O (SBC's)
} CARDINFO;

#endif
```

The `cardinfo` structure is accessed within Host application programs in order to gain access to board-specific parameters which are maintained by the DLL. For example, in order to ascertain the size of the shared memory area on a specific target card a host program could use:

```
/* send bus mastering physical address to target processor */

dsp = (CARDINFO*)target_cardinfo(target);

size = dsp->Dualport.Size;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DspComponent

In the last few years, several different Rapid Application Development (RAD) environments have been developed to improve programmer productivity. Microsoft first produced Visual Basic, and Borland followed with Delphi (Object Pascal) and most recently, C++ Builder (which uses C++). What these tools have in common is the concept of a "component", a packaged piece of software than encapsulates the behavior of a button or other Windows control.

These can be arranged on forms with a visual editor provides a way to "drop" components into a window and add code to process events generated by components. For example, a button component generates a "Button Press" event, which can be set to automatically call a handler function by the RAD tool. The handler code will be called whenever the button is pressed.

DspComponent is a non-visual component provided by Innovative Integration, which provides a component interface to the board DLL that is included in each Zuma tool set. This allows the application programmer to avoid learning the details of connecting to the DLL, in favor of a drop-in package that is integrated into the language. This allows the developer to concentrate on the user interface and the DSP target programming, and not on the mechanics of Windows DLLs and drivers.

The DSPComponent Package consists of two components, an ActiveX and a Borland C++ Builder VCL component. Both perform the same function, and have essentially the same feature set, so that unless indicated a reference to the TIIDspComp component for C++ Builder applies also for the IIBoardX ActiveX control. The control was ported to ActiveX to allow its use in MSVC and Delphi.

The TIIDspComp component provides a high level interface to the Innovative Integration board DLL included as part of the Zuma tool set for a particular target board. This interface is common across all targets, and the same control can be used for all Innovative boards by picking the Target type at design time or even at run time. Mailbox access to the target is fully supported to the limits of the board selected, as simply as reading or writing to an array in the program. Interrupts in both directions, from host to target and target to host are supported. An interrupt from the target, if enabled, triggers a "software event" that can be handled just as any other event, such as a button click can. Resetting the board and downloading COFF files is simplified into a single call that returns a status code for failure conditions.

Features:

- Drop in VCL Component for C++ Builder and Active X version for Visual C++ and Delphi.
- A BoardType property selectable at design time or run time allowing one component to handle any Innovative target board.
- *Enabled* property to open and close target DLL
- Simplified board Booting and COFF downloading methods.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- Interrupt support for both Target to Host and Host to Target interrupts.
- Target to Host interrupt handler by use of standard Event handler mechanism.
- Simplified mailbox property array (Mailbox[]) to access on-board mailboxes.
- Mailbox flag arrays to check for incoming mailbox data or free outgoing mailboxes without waiting for data.
- Shared memory properties to give access to board shared memory for high-speed data transfer to dualport or busmaster memory.
- Windows NT and Windows 95 support using Innovative's Zuma DLLs.

See the DspComponent subdirectory on your hard disk for detailed online help for DspComponent (Windows help: DspComponent.hlp) and for examples illustrating use of the component for streamlined Host application development.

Host Example Programs for the SBC671x Baseboard

Overview

The SBC671x Baseboard differs from other baseboards in that the normal means of communicating with the host is at a higher level than other cards. On other single-boards, the serial interface lends itself to low-level, character based communications well. The Dsp cards support some means of bulk data transfer and a separate low-level signalling and command interface (For example, Busmastering for bulk transfer and mailboxes for signalling and commands on the M62). For compatibility with other SBCs, the SBC671x also implements a simulation of mailboxes for transferring characters and low level data.

Message Packets. On this baseboard, using USB, the most natural way to transport information is by means of asynchronous packets of data that are transferred and decoded by the destination. These messages may be of varying sizes, allowing large messages to be efficient in sending bulk data while allowing small command messages to be mixed into the message stream. When delivered to the destination, the messages can be parsed and can result in any kind of processing desired.

By having the receipt of a message trigger the generation of additional messages, a message protocol can be developed to allow the transfer of data or the execution of control functions on demand from the other side of the link. This is even more natural if the applications are written in an event-driven style. The arrival of messages are the events to which the application responds.

Commands. The SBC671x also supports a second, more limited method of communication called *commands*. These are implemented via the USB VendorCommand mechanism. The host side can send a 32-bit word with a Write command, or send a 32-bit word and receive a 32-bit word reply with a Read command synchronously from the target. The requirement that the host initiate the transfer limits the

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

use of commands, but the fact that the host will wait for the response before proceeding makes commands useful for synchronizing the target and host applications.

Interrupts. As with all baseboards, the SBC671x supports target-to-host and host-to-target interrupts.

On other baseboards, the standard host example was named “Scope” and demonstrated data transfer via the bulk interface signalled by interrupts. Because of the differences in the interface, the example programs for the SBC671x demonstrate the use of the forms of communication between the target and host and give an example of how to construct a command protocol between two applications, the target application and the host application. The two applications are very similar, the major difference being the platform used for development.

MessageTest

The MessageTest.exe example resides in the Examples\Host\Message directory. The target application matched with it is located in the Dsp subdirectory. The two applications share a header file, `Messages.h`, that contains the ID codes for the commands that are passed between the target and host. The IDs are entries in an enumeration, which assures that the IDs will be unique in each application. By using a single source file, we avoid the problem of two lists of IDs losing sync and producing difficult-to-detect errors during development.

The application uses the VCL DspComponent to provide the component interface to the application.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

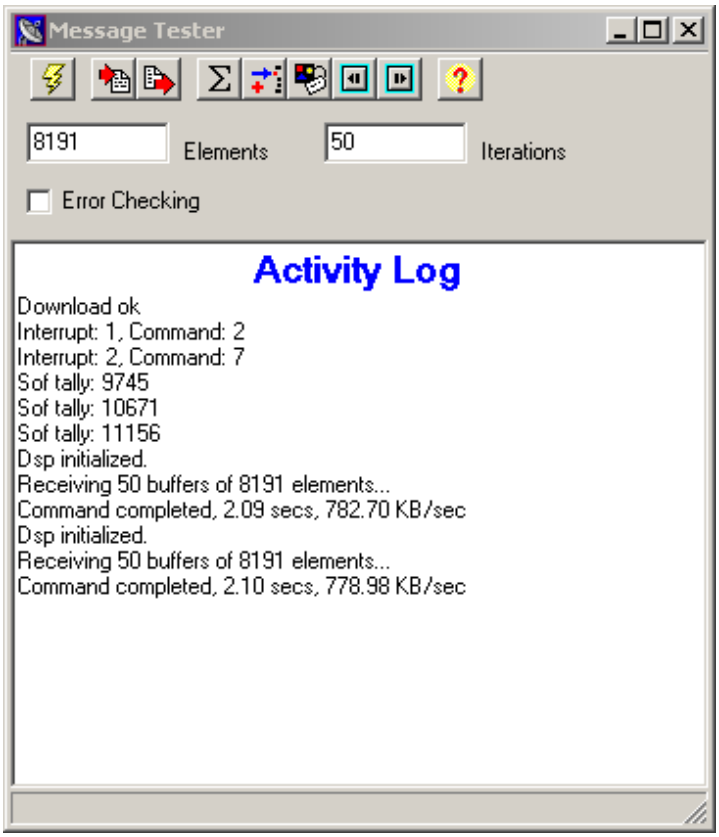


FIGURE 31. The MessageTest Example

Most of the area of the window is a visual log of the results of the functions performed when the buttons across the top of the screen are pressed. The remaining controls provide arguments to some of the other functions below. The code for this window can be found in `MessageTestMain.cpp`.

Download. The leftmost button is the only one active when the application begins. When pressed, it brings up the browser window, allowing the user to select which `.OUT` file will be downloaded to the target. There are two versions of `MsgTest.OUT` available, residing in the Examples\Host\Message\Dsp and Examples\Host\Message\Bios directories. Both have identical functionality, however one uses DSP/BIOS and the other does not. The browser defaults to the non-BIOS version. Once the file is selected, TForm1::DownloadButtonClick() is called. This function will attempt to prepare the target processor and download the target application onto it. It first attempts to load the baseboard DLL, printing a specific message if the DLL cannot be found. If the DLL is present, the DSP is booted and the target code is downloaded and executed. If the program downloads without error, the other buttons are enabled, allowing the communication tests to be run.

Message Receive Test. The second button performs a message receive rate test. (See TForm1::RcvButtonClick()). A single message to the target is used to trigger the sending of *Iterations* messages back, each containing data of size *Elements* in words in addition to the header information. These parameters are

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

entered from the edit controls on the main window. In addition, if the ErrorChecking box is checked, known data is inserted in the packets and checked for accuracy when it arrives at the host.

Messages are sent by calling the DspComponent function XmtMessage(). It takes a command code, and a pointer to the user-defined parameter region and its size in words. Note that since the message is asynchronously sent off, the host application cannot wait for the reply to arrive here. This portion of the processing must end with the posting of a request for action by the target application. When the target sends the reply messages, they will be processed by the message receiver code.

For this code, view the function TForm1::DspRcvMessage(). This function is an event handler for the Dsp Component's OnRcvMessage event. Whenever a message is received from the host, this handler is called to allow applications to process the message. The command ID of the message is used to select the processing function for the message. In this case, the TForm1::HandleTargetDataMessage() function is called to check the packet for errors, count the bytes transferred, and when all the messages have arrived, add to the log display the data rate for incoming data.

Message Transmission Test. The third button performs a similar test for message transmission. When the button is pressed, the TForm1::XmtButtonClick() function sends *Iterations* messages to the target, each containing data of size *Elements* in words in addition to the header information. If the ErrorChecking box is checked, known data is inserted in the packets for checking by the target on arrival.

SOF Counter Commands. The next two buttons demonstrate the use of commands. TForm1::ReadSofButtonClick() uses a Read Command to obtain a counter from the target that counts the number of SOF packets detected on the USB. These packets are sent about 1000 times a second. TForm1::InitDspButtonClick() uses a Write Command to clear the SOF counter to 0.

Revision Code Message. The next button uses a command message an reply message to create a simple protocol that in this case retrieves a revision code from the target. TForm1::RevisionButtonClick() sends a message that requests the revision code. The target program creates a reply message, which is processed by TForm1::HandleRevisionReply(). This processing function adds the revision code to the log. The non-BIOS example will report "Version: 1.01", while the DSP/BIOS example will return "Version: 1.02".

Target to Host Interrupts. The next button calls TForm1::HostInterruptButtonClick(), which sends a message to the target that requests an interrupt to the host. This interrupt results in the OnInterrupt event on the DspComponent. TForm1::DspInterrupt() is the handler for this event and prints a message in the log.

Host to Target Interrupts. The final test button demonstrates Host to Target interrupts. In TForm1::TargetnterruptButtonClick(), first the interrupt is generated by calling the Dsp Component Interrupt() method. After this, a message is posted to the target requesting the count of interrupts on the target. When this message arrives, TForm1::HandleMailboxTallyReply() posts the count to the log display.

The Target Application

The target application is involved in all of the transactions of the Message Test application. Each command or message must be processed and a reply might possibly be returned if required by the protocol. In this section we will run through how the target can perform these actions. See the file `MsgTest.c` to follow along with the description.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The main() Routine. The main routine is relatively simple, as most of the work involved in communication with the host is performed by the Zuma library. The interface to the library code is a number of callback function pointers that will be called when processing is required. These act very much like events do on the host side application. These handlers are:

Event Variable	Handler Function	Purpose
OnWriteCommand	WriteCommandHandler()	Process Write Commands from Host
OnReadCommand	ReadCommandHandler()	Process Read Commands from Host
OnProcessMessage	ProcessMessageHandler()	Process Messages from Host

After installing these handlers, the host communications are started with a call to init_communications(). Next, a handler for mailbox interrupts from the host is installed to process interrupts from the host. Finally, the main function remains in a loop calling the library function ProcessRcvMessage(). This function needs to be called to allow messages to be processed. It should be called in any long loop or idle polling process to process messages that may arrive in a timely manner.

ReadCommandHandler(). This handler is installed to process Read Commands from the host. The system passes in the command ID, along with other parameters. The data to be returned is passed back by the data argument. For example, the Get SOF command has the ID of *ccGetSofCnt*. The switch statement loads the data argument with get_sof_tally()'s return value. This sends the SOF count back as the response.

WriteCommandHandler(). This handler processes Write Commands from the host. Here the data argument is a parameter sent by the host. As an example, the *ccInit* command resets system parameters and the SOF counter.

ProcessMessageHandler(). This handler processes messages from the host. In a simple case, the *ccGetRevision* ID is sent to obtain the revision ID. The processing for this message is a simple call to XmtMessage(), which transmits the message to the host with the revision code inserted as part of the message. In a more complex version, the *ccGenerate* command results in the posting of a number of messages to the host to perform the receive message test. Each message coming from the host results in a call to a processing function.

Host to Target Interrupts. When an interrupt is sent by the host, the handler installed for mailbox interrupts is called. MailboxInterruptHandler() acknowledges the interrupt and increments a counter. This counter is retrieved by the host by a message request in order to verify the delivery of the interrupt.

Target to Host Interrupts. One message sent by the host, *ccHostXrpt*, generates an interrupt to the target by calling the library function mailbox_interrupt(). The parameter is delivered through the mailbox to the host for logging.

VcMessage Test

The second example is implemented using Microsoft Visual C++. It also demonstrates the means of communication between the target and host applications. It uses the same target application as the VCL example, and uses the same header file, `Messages.h`, to assure that the commands IDs remain consistent in the two applications.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

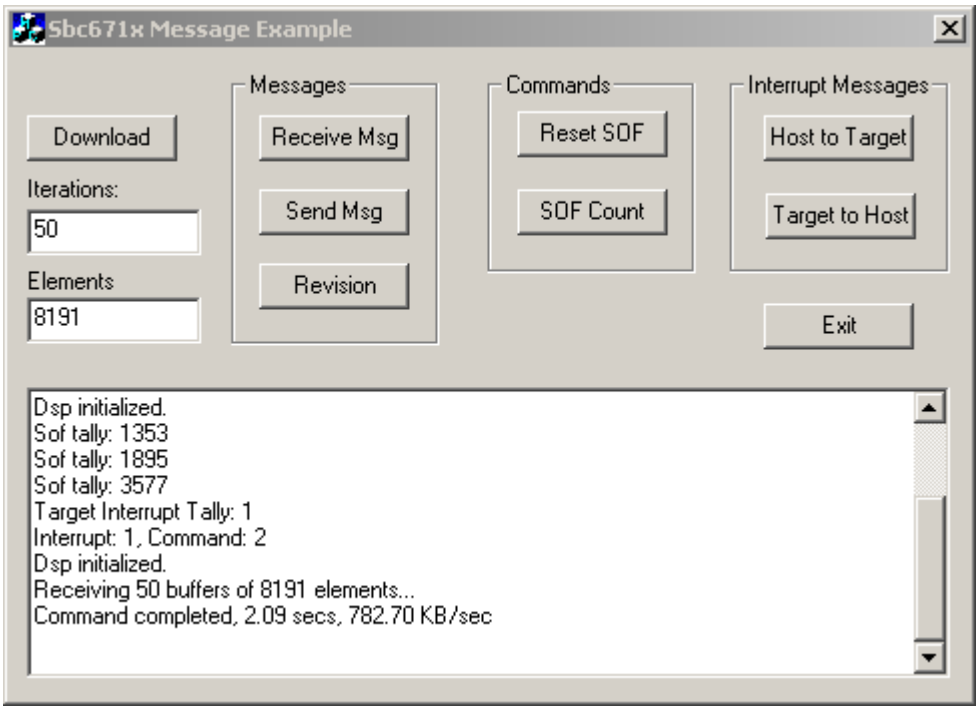


FIGURE 32. The VclMessage Example

To provide an object-oriented interface to the target, the VclMessage example uses the Dsp Component ActiveX control. This component allows the use of similar methods, properties, and events as the VCL component.

At the bottom of the window is a visual log of the results of the functions performed when the buttons in the top half of the window are pressed. The remaining controls provide arguments to some of the other functions.

The code for this application is located in Examples\Host\VcMessage directory. It is a dialog box based MFC application. The code for the window controls can be found in `VcMessageDlg.cpp`.

Download. The Download button is the only one active when the application begins. When pressed, it brings up the browser window, allowing the user to select which `.OUT` file will be downloaded to the target. There are two versions of `MsgTest.OUT` available, residing in the Examples\Host\VcMessage\Dsp and Examples\Host\VcMessage\Bios directories. Both have identical functionality, however one uses DSP/BIOS and the other does not. The browser defaults to the non-BIOS version. Once the file is selected, CVcMessageDlg::OnDownload() is called. This function will attempt to prepare the target processor and download the application. It first attempts to load the baseboard DLL, printing a specific message if the DLL can not be found. If the DLL is present, the DSP is booted and the target code is downloaded and executed. If the program downloads without error, the other buttons are enabled, allowing the communication tests to be run.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Message Receive Test. The first group of buttons collects tests involving Messages. The first button in this group performs a message receive rate test. (See CVcMessageDlg::OnRcvMessage()). A single message to the target is used to trigger the sending of *Iterations* messages back, each containing data of size *Elements* in words in addition to the header information. These parameters are entered from the edit controls on the main window.

Messages are sent by calling the DspComponent function XmtMessage(). It takes a command code, and a pointer to the user-defined parameter region and its size in words. Note that since the message is asynchronously sent off, the host application cannot wait for the reply to arrive here. This portion of the processing must end with the posting of a request for action by the target application. When the target sends the reply messages, they will be processed by the message receiver code.

For this code, view the function CVcMessageDlg::OnRcvMessageIiboardx1(). This function is an event handler for the Dsp Component's OnRcvMessage event. Whenever a message is received from the host, this handler is called to allow applications to process the message. The command ID of the message is used to select the processing function for the message. In this case, the CVcMessageDlg::HandleTargetDataMessage() function is called to count the bytes transferred, and when all the messages have arrived, add to the log display the data rate for incoming data.

Message Transmission Test. The SendMsg button performs a similar test for message transmission. When the button is pressed, the CVcMessageDlg::OnSendMsg() function sends *Iterations* messages to the target, each containing data of size *Elements* in words in addition to the header information.

Revision Code Message. The Revision button uses a command message an reply message to create a simple protocol that in this case retrieves a revision code from the target. CVcMessageDlg::OnGetRevision() sends a message that requests the revision code. The target program creates a reply message, which is processed by CVcMessageDlg::HandleRevisionReply(). This processing function adds the revision code to the log. The non-BIOS example will report "Version: 1.01", while the DSP/BIOS example will return "Version: 1.02".

SOF Counter Commands. The central group contains two buttons that demonstrate the use of commands. CVcMessageDlg::OnGetSof() uses a Read Command to obtain a counter from the target that counts the number of SOF packets detected on the USB. These packets are sent about 1000 times a second. CVcMessageDlg::OnResetSof uses a Write Command to clear the SOF counter to 0.

Target to Host Interrupts. The right group of buttons contains buttons that test interrupts. The Target to Host button calls CVcMessageDlg::OnTargtoHost(), which sends a message to the target that requests an interrupt to the host. This interrupt results in the OnInterrupt event on the DspComponent. CVcMessageDlg::OnDspInterruptIiboardx1() is the handler for this event and prints a message in the log.

Host to Target Interrupts. The final test button demonstrates Host to Target interrupts. In CVcMessageDlg::OnHostToTarget(), first the interrupt is generated by calling the Dsp Component Interrupt() method. After this, a message is posted to the target requesting the count of interrupts on the target. When this message arrives, CVcMessageDlg::HandleMailboxTallyReply() posts the count to the log display.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CHAPTER 8

Creating Target Software

Software is created for theSBC6711 target DSP by using one or more of the tools included in the Developer's Package, either alone or in concert with each other. These tools are used to generate a downloadable executable COFF format file, which can be run on the target DSP board with the aid of the utilities included in the package.

This section of the *Developer's Package Manual* details the use of the individual tools in the package to create executables for the target. In addition, step-by-step instructions are provided on how to use the C compiler and Code Composer Studio to write, compile, test, and debug custom C applications on the target. Sample C applications are also discussed.

C Code Development

C Compiler

The Texas Instruments C compiler is an ANSI C compatible compiler, which produces optimized assembly code for the TMS320 family of processors. A complete set of manuals is included on the installation CD provided with the Developers Package.

In addition to the manuals from TI, refer to the Kernighan and Ritchie C Handbook (available at cost from I.I.) for generic C questions and syntax. The TI manuals primarily describe the use of the compiler with the TMS320C6x family and are not intended as C primers for the beginner.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

C Library Reference

Complete source code to the entire suite of ANSI C libraries is provided to aid in code development. Refer to the *TMS320C6x C User Guide*, Runtime-Support Functions chapter for a complete list of TI C functions.

The Innovative Integration Developer's System also includes extensive high-level libraries useful in interacting with the various peripherals on the target board. The following sections describe by peripheral type the functions provided in the peripheral library.

Note: See the online Zuma.hlp for a complete listing of available target peripheral library functions, as the listing in this manual is a subset of the available functions.

Zuma Toolset Libraries

The Zuma toolset provides both target peripheral libraries and host DLLs, as well as numerous example programs to illustrate their usage.

The peripheral libraries for the target provide support for the on-board peripherals and terminal I/O functions. The libraries are provided in three linkable `.LIB` files: `PERIPH.LIB`, `STDIO.LIB` and `DSP.LIB`. `STDIO.LIB` consists of all the console terminal emulation and communications routines listed in the following section, while `PERIPH.LIB` contains all other peripheral driver routines. `DSP.LIB` contains commonly requested C-callable digital signal processing functions, plus common math and queue management extensions. Source code for the routines is also provided, arranged by function in the **`\PERIPH`**, **`\STDIO`** and **`\DSP`** subdirectories of the root II_BOARD directory, as follows:

Directory	Library Source
\DSP	Standard Digital Signal Processing Routines.
\PERIPH\ANALOG	Drivers for the A4D4 instrumentation-grade analog I/O module and the complementary TERM mux module, SD and SD16 high-performance audio modules, AD40, DAC40, AIX, AIX20, A4D1, and RF high-speed modules, and the Servo16 servo control module.
\PERIPH\BUS	Drivers for the SL811 USB Interface and the TL16C750 UART.
\PERIPH\DIGITAL	Digital I/O, 8254 Timer control, etc. Drivers for DIO module. Drivers for MOT motion control module.
\PERIPH\FIFO	Fifo Port control functions.
\PERIPH\FLASH	Flash ROM programming drivers.
\PERIPH\MISC	Miscellaneous processor control and data conversion functions.
\PERIPH\RTS	Modified boot-up routines for non-BIOS applications.
\PERIPH\RTSBIOS	Modified boot-up routines for DSP/BIOS applications.
\STDIO	Console and terminal emulation functions.
\TALKER	Start-up umbilical software.

TABLE 2. Zuma Toolset Source Directories

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The toolset also contains various support files arranged as described below.

Directory	Library Source
\EXAMPLES\HOST	Example host PC programs illustrating use of the DLL to control the DSP board from within Borland Builder & MSVC programs.
\EXAMPLES\TARGET	Example target DSP programs illustrating use of the target peripheral libraries to perform common DSP tasks.
\INCLUDE\HOST	Header files used by Host Visual C programs.
\INCLUDE\TARGET	Header files used by target C programs.
\LIB\HOST	Linkable library files for Host C++ programs.
\LIB\TARGET	Linkable library files for target C and assembler programs.

TABLE 3. Zuma Toolset Support Subdirectories

STDIO Console Terminal Driver. The Developer's Package contains a full-featured terminal emulator application (`uniterminal.exe`), suitable for both user interface purposes as well as debugging use. The peripheral library provides a complete set of standard I/O routines, which can communicate directly with this terminal emulator. The source for the standard I/O routines is given in the `\STDIO` subdirectory under the installation directory. In general, the standard I/O library functionality is identical to that of the K&R standard I/O library. However, some target-specific functions are provided to allow higher level functionality such as cursor positioning, text attribute control, and graphical data plotting. The following target programming section gives details on how to use the standard I/O peripheral library to interact with the terminal emulator.

Digital Peripheral Drivers. The digital peripheral drivers control the DSP internal timers and the digital I/O lines, allowing high-level access to timebase control functions and digital I/O activity without doing direct hardware programming. The following target programming section gives details on how to use the digital peripheral library to program the digital peripherals. Source code for the functions is given in the **`\PERIPH\DIGITAL`** directory.

BUS Peripheral Drivers. The **`\PERIPH\BUS`** directory contains code to support high-level access to the SL811 USB interface and the TL16C750 UART which handles RS-232 communications. Both of these interfaces utilize interrupt-driven input and output. Source code is provided for all of these functions. Strategic communications support functions have been coded as inline functions. Source for the inline functions is located in **`\INCLUDE\TARGET\BUS`**.

FLASH ROM Programming Drivers. The flash ROM directory contains code to support high-level access to the onboard Flash ROM. The code supports byte, word, device, and sector programming along with device and sector erasure. Source code for the functions is given in the **`\PERIPH\FLASH`** directory.

Miscellaneous Peripheral Drivers. The MISC directory contains code to support high-level access to the internal registers, byte packing and unpacking, interrupt vector support, and other functions. Source code for the functions is given in the **`\PERIPH\MISC`** directory.

RTS Peripheral Drivers. The RTS peripheral drivers provide board-specific versions of the functions called by the TI C Compiler during coldstart initialization of the C runtime engine. These files have been modified as necessary in order to provide a complete initialization of the target onboard hardware immediately prior to calling `main()` within application code. Additionally, the RTS functions include a

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

modified version of the millisecond timer function required to support the TI C timekeeping functions (listed in `time.h`). Source code for the functions is given in the **\PERIPH\RTS** directory.

RTSBIOS Peripheral Drivers. The functions contained in the **\PERIPH\RTSBIOS** directory are variations of those found in **\PERIPH\RTS** which have been modified for use with DSP/BIOS.

USB Peripheral Drivers. The BUS directory contains code that implements a high-level, queue based streaming interface with the host using the ScanLogic USB chip to allow data transfer to the host at rates beyond that supported by a standard RS232 serial port. While source code for the functions is given in the **\PERIPH\BUS** for the system, the complexity of USB is such that altering this code is not recommended by Innovative Integration. On the host side a device driver is included that will detect the target and allow the host to communicate with the target.

Further details of the USB interface are given in the section on the Message Interface for the SBC6711.

Note that the USB interface requires Windows 98 or Windows 2K or Windows XP to function.

Digital Peripheral Drivers. The digital drivers support access to all baseboard and add-on digital I/O functions.

The DIO peripheral drivers provide control functions for the optional DIO plug-in module. The functions provide high-level C access to the DIO module's 32 additional digital I/O lines plus either interrupt-driven or polled use of the DIO's onboard DUART (Dual-channel Universal Asynchronous Receiver Transmitter). The target programming section gives details on how to use the digital peripheral library to program the digital peripherals. Source code for the functions is given in the **\PERIPH\DIGITAL\DIO** directory.

The MOT peripheral drivers provide control functions for the optional MOT plug-in module. The functions provide high-level C access to the MOT module's four, precision motion-control axes. Each of the axes features independent encoder inputs and either digital or 16-bit analog output. Digital output may be either pulse or direction positive and negative pulse to support stepper motor amplifier inputs. The target programming section gives details on how to use the MOT peripheral library to program these peripherals. Source code for the functions is given in the **\PERIPH\DIGITAL\MOT** directory.

Analog Peripheral Drivers. The Analog peripheral drivers provide control functions for the optional analog plug-in modules: A4D4, A4D1, AIX, SD, SD16 and A16D2 modules. The functions provide high-level C access to the A4D4's analog input and output channels and their associated gain amplifiers. Additionally, the driver supports control of the optional TERM break-out panel, a companion to the A4D4 module, in order to support muxing of each of the A4D4 modules 8:1 to allow input from up to 32 simultaneous channels per A4D4 module. Source code for the functions is given in the **\PERIPH\ANALOG\A4D4** directory.

The AIX peripheral drivers provide control functions for the optional AIX plug-in module. The functions provide high-level C access to the AIX module's four, 2.5 MHz, 16-bit analog input channels. Source code for the functions is given in the `\PERIPH\ANALOG\AIX` directory.

The A4D1 peripheral drivers provide control functions for the optional A4D1 plug-in module. The functions provide high-level C access to the module's four, 10 MHz, 14-bit analog input channels and single 15 MHz analog output channel. Source code for the functions is given in the `\PERIPH\ANALOG\A4D1` directory.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The A16D2 peripheral drivers provide control functions for the optional A16D2 plug-in module. The functions provide high-level C access to the module's sixteen, 12.5 kHz, 16-bit analog input channels and two 200 kHz analog output channels. Source code for the functions is given in the **\PERIPH\ANALOG\A16D2** directory.

The SD peripheral drivers provide control functions for the optional SD plug-in module. The functions provide high-level C access to the SD module's four, audio-grade, 24-bit analog input and 20-bit output channels. Source code for the functions is given in the **\PERIPH\ANALOG\SD** directory.

The SD16 peripheral drivers provide control functions for the optional SD16 plug-in module. The functions provide high-level C access to the SD16 module's sixteen, audio-grade, 18-bit analog input and output channels. Source code for the functions is given in the **\PERIPH\ANALOG\SD16** directory.

The target programming section gives details on how to use the analog peripheral library to program these analog peripherals.

Digital Signal Processing Library. The DSP directory contains code to support high-level access to the common signal processing functions such as FFT's, filters and compression. Additional routines are provided for common functions such as matrix manipulation, curve fitting and general purpose queue management. Source code for the functions is given in the **\DSP** directory.

Texas Instruments C Libraries

Several libraries are included with the system that provide support for floating point and extended math functions, DSP oriented procedures and initialization examples. Chapter 5 in the *TMS320 Floating Point DSP Optimizing C Compiler User's Guide* describes the libraries.

The following libraries are available:

Library	Operation
ASSERT.H	Defines the assert macro for runtime error message reporting.
CTYPE.H	Declares functions that test and convert characters.
LIMITS.H	Defines range limits for characters and variable types.
FLOAT.H	Defines floating point range limits.
MATH.H	Defines trigonometric, exponential and hyperbolic math functions.
ERRNO.H	Defines errno variable for catching range errors in function calls.
STDARG.H	Defines macros to aid in variable argument functions.
STDDEF.H	Defines two new types and macros used within runtime functions.
STDLIB.H	Declares many common library functions such as string conversion, sorting and searching functions, program exit functions and some integer-arithmetic that is not a standard part of C.
STRING.H	Declares functions for string manipulations.
TIME.H	Declares macros and types useful for time manipulations.

TABLE 4. Texas Instruments Standard Library Functions

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Target Hardware Interaction

All peripherals are memory mapped into the DSP address space, using the locations given in the following table. The table also lists the wait states applied to access each peripheral.

The development system provides routines to access all integrated SBC peripherals. This section describes how to program the peripherals using the supplied library functions under C or via direct memory accesses to the supplied peripheral register map. In general, direct memory access delivers higher performance than using the C function library since it avoids the overhead of the function calls necessary to access the library. However, the libraries have been crafted to utilize inline code where possible to mitigate this effect. In the peripheral descriptions that follow, each device's access methods are called out for both high level and direct memory access. In the case of C functions, the function names and argument variables are called out. In the case of direct memory access operations, the relevant addresses are listed along with the functions they perform and accompanying `Periph` structure elements which may be used from C to simplify access. These elements are defined in the header file `periph.h`.

Function	Address	C Language Mneumonic	Memory Space
Flash ROM	0x90000000..901fffff	Periph->Prom[0x080000]	CE1
SL811 USB interface	0x90200000..90200fff	Periph->Usb	
TL16C750 UART ctl	0x90201000..90201fff	Periph->Uart	
DDS timer data	0x90203000..90203fff	Periph->DdsData	
DIO direction control	0x90300000..90300fff	Periph->DioDir	
Misc status reg	0x90301000..90301fff	Periph->Status	
Misc control reg	0x90302000..90303fff	Periph->Control	
Interrupt control	0x90303000..90307fff	Periph->Interrupt[5]	
Fifo Port	0x90308000..9030bfff	Periph->Fport	
DSP timer source ctl	0x9030c000..9030dfff	Periph->DspTimerSrc[2]	
Omnibus timer src	0x9030e000..9030efff	Periph->ModuleTimerSrc	
Omnibus DDS src	0x9030f000..9030ffff	Periph->ModuleDdsSrc	
Clock Link src	0x90310000..90310fff	Periph->ClockLinkSrc	
Sync Link master ctl	0x90311000..90311fff	Periph->SyncLinkMaster	
Sync Link src ctl	0x90312000..90312fff	Periph->SyncLinkSrc	
Watchdog data	0x90313000..90313fff	Periph->Watchdog	
DDS timer update	0x90314000..90314fff	Periph->DdsUpdate	
Serial ROM ctl	0x90315000..90315fff	Periph->SRomData	
Serial ROM enable	0x90316000..90316fff	Periph->SRomEnable	
Baseboard timer data	0x90317000..90319fff	Periph->TimerCount[3]	
Baseboard timer ctl	0x9031a000..9031cfff	Periph->TimerCtl[3]	
Interrupt sense ctl	0x9031d000..9031dfff	Periph->InterruptSense	
Interrupt type ctl	0x9031e000..9031efff	Periph->InterruptType	
Interrupt ack[5]	0x9031f000..90323fff	Periph->InterruptAck[5]	
Dio data	0x90324000..90324fff	Periph->DioData	

TABLE 5. SBC6711 External Peripheral Memory Map

This document does not describe peripheral hardware specifications and other hardware issues. Refer to the *SBC6711 Hardware* section of this manual for hardware information.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Digital Input/Output

The digital input/output (I/O) buffers provide a means for generating 32 bits of direct digital input or output to and from external hardware. This I/O can be clocked from either the DSP processor or from external TTL sources, allowing external devices to automatically latch data into the I/O buffers for the DSP to read.

Input/output direction for either half of the 32-bit port may be programmed on the fly using the on-board logic. The port may be configured in software for input or output in groups of eight bits.

Memory Mapped Digital I/O Access. The following table shows the memory locations used to interact with the digital I/O buffers. Three C language routines are supplied to interact with the digital I/O port.

Function	C Language Mnemonic
Digital I/O Data Register	Periph->DioData
Digital I/O Direction Control (4 bits)	Periph->DioDir
Digital I/O Latch Control (1 bit)	Periph->DioDir

TABLE 6. Digital I/O Access Memory Location

The `Periph->DioData` location is used to access the data lines of the digital I/O port. Results of read and write accesses depend on the I/O direction of the port (see below for information on setting the port direction). If the port is configured for input, a read access latches new read data from the external pins and the new data is read into the DSP. If the port is configured for output, the most recently latched output data is read from the DSP (output data does not change). Write accesses to an input port cause no change to the port status, while write accesses to an output port cause the new data to be latched and output to the external I/O pins.

The `Periph->DioDir` location controls the direction of each byte of the digital I/O port. The four least significant bits of this register are used to configure each of the bytes of the digital I/O port for either input or output, as follows:

Dio.DirectionRegister Bit #	Value	Direction
0	0	DX[0..7] output (default)
	1	DX[0..7] input
1	0	DX[8..15] output (default)
	1	DX[8..15] input
2	0	DX[16..23] output (default)
	1	DX[16..23] input
3	0	DX[24..31] output (default)
	1	DX[24..31] input

TABLE 7. Digital I/O Direction Configuration

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The `Periph->DioDir` location also controls the method of latching data into each byte of the digital I/O port. Bit four of this register is used to configure the latch method as either internal (triggered by CPU accesses) or external (triggered by an external TTL pulse)

C Language Digital I/O Functions

Data may be read or written to the digital I/O port using the following routines in the DIGITAL support library.

Function Name	Description
dig_dir	Sets the direction of all four bytes of the onboard 32-bit digital I/O port.
read_dig	Returns current state of all 32-bits of digital I/O port.
write_dig	Sets current state of all 32-bits of digital output port currently configured for output.
write_dig_bit	Returns current state of specific bit from digital I/O port.
read_dig_bit	Sets current state of specific bit of digital output port currently configured for output.
set_dig_trigger	Sets the latch method of all four bytes of the onboard 32-bit digital I/O port.

TABLE 8. Digital I/O Library Functions

Timers

The timers provide the capability to generate hardware timebases, which can be used to trigger processor interrupts, analog signal conversions, or as direct outputs to external hardware. There are a total of six timebase sources built in to the SBC6711: two 32-bit timers internal to the ‘C6201 processor plus, three onboard 24-bit timers and one AD9850 direct digital synthesizer. The supplied library functions initialize the timers to a free-running, pulse generation mode suitable for generating convert pulses to the analog hardware.

The timers are initialized by code in the `timebase()` routine each time it is called. Normally, no other function calls are necessary to use the timers. However, when supplying an external TTL signal to the DSP TCLK0/1 inputs in order to provide an external timebase to analog circuitry, it will be necessary to create and use a custom version of `timebase()` which tristates the TCLK output driver to avoid contention with external sources. Please note that certain hardware setups might be required depending on the application. See the *Hardware* section of this manual for more details on how to set up the target board.

C Language Timer Functions. The following functions give high-level access to the timer hardware.

Function Name	Operation
timebase	Configures a specified timer channel (0..5) for periodic counting at a specified frequency using a specified source clock rate.

TABLE 9. C Language Timer Function

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The `timebase()` function can be used to set a particular timebase to a particular frequency. For example, the following call sets internal timer channel 1 to generate a 1000 Hz output pulse stream, assuming the system CPU clock as the timebase.

```
// tbTClk1 from Enums.h

timer(tbTClk1, 1000.0,  MHZ);
```

Memory Mapped Timer Access. It is possible to directly access to the internal timer hardware controls via memory mapped registers at specific addresses. It may be necessary to use these addresses to set the timers to a custom mode. In general, unless custom functionality is required of the timers, it is recommended that the user exclusively access the timers via the `timebase()` routine rather than programming the control and period registers manually.

For information about the DSP internal timers, please see the *TMS320C6x User's Guide*. For information about the 9850 DDS timer device, see the data sheet. For an example of direct timer channel control, refer to the source code for the `timebase()` function, located in the **`PERIPH\DIGITAL`** subdirectory.

STDIO Communication

The C stdio terminal emulation is provided in the Peripheral Library. The stdio library communicates with the host `UNITERMINAL.EXE` program via the USB interface to provide stdio support to DSP applications running on the SBC6711. The stdio interface may be used for real-time, intrusive software debugging or to create a basic user interface for OEM applications.

The following list shows the available Peripheral Library calls and their operation. Refer to the Zuma help file for complete information on the functions.

Function Name	Operation
putchar()	Emits an 8-bit character to the terminal emulator.
getchar()	Gets an 8-bit character from the terminal emulator's keyboard buffer.
gets()	Inputs a string into a target buffer.
puts()	Displays a string from a target buffer.
sprintf()	Formats a string into a memory buffer pointed to by buffer.
printf()	Prints a formatted string to the terminal.
scanf()	Inputs a formatted string from the terminal into a buffer.
sscanf()	Converts a formatted string in memory into a buffer.
stdio_reset()	Resets the terminal emulator display.
fopen()	Opens a file on the Host PC, returning the file handle.
fclose()	Closes a previously opened Host PC file.
fread()	Reads file contents into a target buffer.
fwrite()	Writes a target buffer into a Host PC file.
fseek()	Repositions the Host PC file pointer.
ferase()	Erases the specified Host PC file.
kbd_hit()	Returns a nonzero value if characters are currently available in the monitor keyboard buffer.
kbd_key()	Returns 16-bit IBM scancode for pending keystroke from the terminal emulator's keyboard buffer.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

gotoxy()	Moves the terminal cursor.
wherexy()	Returns the terminal cursor position.
clreol()	Clears to end of current line.
clrscr()	Clears the terminal screen.
type()	Types formatted, null terminated string to console.
bold()	Enables bold text attribute in terminal emulator.
normal()	Enables standard text attribute within terminal emulator.
get_attribute()	Returns the current character display attributes.
set_attribute()	Sets the current character display attributes.
cursor()	Enables/disables the cursor.
get_busmaster_addr()	Obtains the base of the host busmaster memory from the terminal emulator.
plot()	Plots a Host PC file as a graph.
view()	Plots a target buffer as a graph.

TABLE 10. STDIO Driver Functions

Using Interrupts

The SBC supports four external and numerous internal hardware interrupts. These include EI0, EI1, EI2, EI3 plus TINT0, TINT1 (internal timer/counters), internal comm port transmit and receive and DMA.

Interrupts on the TMS320 may be handled by writing either high-level C or assembly language procedures within your application files which employ the following interrupt-specific function names:

`void interrupt isr_fcn()` for C handlers or

`isrfcn` for assembly language

where `isr_fcn` can be any legal C function name. For each interrupt, a procedure must be coded which will be executed upon acknowledgment of the appropriate by the DSP. This is described in Chapter 6 the TMS320C6x C Compiler Users Manual.

Consider the following code example:

```
//
// EXAMPLE.C
//
main()
{
    EnableInterrupts();// Enable unmasked xrpts
    timebase(5, 1000.0, MHZ);// Internal timer 1 at 1kHz
    // install interrupt handler on TINT1
    InstallIntVector(ms_isr, TCLK1_INTERRUPT);
    EnableInterrupt(TCLK1_INTERRUPT);
    .// Bulk of application
    .
    DisableInterrupt(TCLK1_INTERRUPT);// Disable TINT0 xrpt
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
//
// ISR for timer 0 - Tally a variable
//
uint32 milliseconds
void interrupt ms_isr()
{
    // Internal timer 1 is used to synthesize a timebase
    milliseconds++;
}
```

In this code, the internal timer 1 is configured to output a pulse every millisecond which drives TCLK1 on the ‘C6x. The vector is installed into the jump table with a call to `InstallIntVector()` and the bit associated with TCLK1 in the interrupt enable register, is enabled. Finally, `main()`calls `EnableInterrupt(TCLK1_INTERRUPT)` which sets the appropriate bit in the interrupt enable register so that all unmasked interrupts can be processed.

Each time the counter expires, the routine `ms_isr()` executes. In this example, the variable `milliseconds` is incremented during each interrupt service cycle.

Each DSP application should include a copy of the default interrupt vector table, which is defined in `vectors.asm`. This assembly file is located in the **`PERIPH\RTS`** directory and when compiled into a `.obj` file and linked into the application, will cause all entries in the vector table to be initialized with a default handler. If an application needs to make use of interrupts, those vectors which are affected need to be changed with `InstallIntVector()` at run time.

See the target example programs provided on your distribution disks for further examples of the use of interrupts.

The SBC6711 Message Interface

The USB interface built onto the SBC6711 provides a means for the rapid transport of data from the target to the host, which many applications will find useful. In order to achieve this speed, however, the USB system requires considerable interaction from the target side and considerable support on the host side as well. In order to isolate users from the details of this mechanism, Innovative Integration has layered a software interface over the raw USB software, thus insulating the user from most of the details of interacting with the USB system itself.

NOTE: USB support is only provided for Windows 98 and Windows 2K systems. Do not attempt to use these functions in Windows 95 or WinNT installations!

Overview of the Message Interface

USB allows each target to define a number of logical targets that constitute separate unidirectional data streams between the target and host. These targets are termed “endpoints”. The endpoints available on the SBC6711 are arranged into a single logical channel, which provides bi-directional movement of data

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

from host to target, independent. This channel can be opened simultaneously from multiple applications.

The channel has the same efficiency from input as output. Measured performance is 360,000 Bytes/sec under Win9x and 800 KB/sec under Win2K.

On the target, the message channel is opened using init_communications(). This function creates eight, messages for use in each direction. Each message is capable of containing a header plus up to 32KB of data. These messages are managed by the target USB driver via autonomous queue structures.

To send data from the target to the host, the target application obtains an available (empty) message from the system, fills it with data, then relinquishes it to the system for transfer to the host. When the system completes the transfer, the message is returned to the pool of available messages for reuse.

To receive data sent from the host to the target, the target application obtains an available (full) message from the system, extracts its data, then relinquishes it to the system for subsequent reuse.

All messages contain a header consisting of a channel number, command code and payload size. The channel number and command code are message-specific. The payload size indicates the number of bytes contained in the message body, from 0..32KB. The body of the message is untyped and may contain information of any variety, message-specific.

Target API for Messaging

The API for the Messaging is intended to resemble the familiar event-driven interface familiar to most Windows programmers. Note that both read and write functions will ‘block’ to assure all data is transferred.

Function Name	Operation
init_communications	Sets up the Messaging interface and enumerates the USB connection, initializes the UART, and installs a shared interrupt handler.
quit_communications	Shuts down the Messaging interface and detaches the USB connection
ProcessRcvMessage	Call periodically from within app idle loop to check for received messages
XmtMessage	Sends a message from target to the host
OnProcessRcvMessage	Callback ftn from ProcessRcvMessage invoked when a receive message is available for target processing

TABLE 11. Messaging System Target Functions

Refer to the Zuma help file for additional details on these functions.

Host USB Support

USB Device Driver Installation. Supporting USB on the host requires a device driver be loaded to link the specifics of the SBC interface to the native USB support provided in Windows 98 or Windows 2K. This driver, `SBC6711.SYS`, must be installed before host applications will be able to provide support for USB data transfer.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

This driver is automatically installed during the SBC6711 Zuma Peripheral Library installation. When USB enumeration is initiated, by turning on or resetting the SBC, the SBC USB driver will automatically be loaded into memory.

If your system shows the "Add New Hardware" expert, select the option to load the driver from a disk supplied by the hardware manufacturer. To select the driver, browse to the directory where the USB driver and INF file are located on the hard disk and select the `SBC6711.INF` file provided. This will install the driver and in all future cases will automatically install the driver when a SBC6711 is detected on the USB chain.

Host API for Bulk Transport

On the host, the API for the Messaging is very suimilar to the target API.

Function Name	Operation
SB6711_rcv_message_install	Install target to host message processing function.
SB6711_rcv_message_deinstall	Remove target to host receive message handler.
SB6711_xmt_message	Enqueue a message for subsequent transmission to the target DSP.
SB6711_vendor_read	Execute a application-specific read command
SB6711_vendor_write	Execute an application-specific write command

TABLE 12. Messaging System Host Functions

Refer to the Host DLL Reference section of this manual for additional details on these functions.

Example Programs for the Bulk Transport Interface

MessageTest. The MessageTest program is intended to illustrate message operation and vendor command operation plus test the throughput of the USB BUS. It uses the BULK interface to the USB device. It automatically use overlapped I/O, managed by multiple threads within the SBC6711 DLL. This application was built using Borland C++ Builder v 5.0.

VcMessage. The VcMessage program is a port of the MessageTest example to MSVC. It supports the same basic features, but uses the ActiveX import version of the DspComponent C++ class. The program was built using MSVC++6.0.

The SBC6711 RS-232 Interface

The SBC6711 implements a single asynchronous serial port channel with RS232 drivers compatible with standard PC serial port interfaces. The serial port is implemented using a Texas Instruments TL16C750 UART and Maxim MAX3223 RS232 transceiver.

The SBC6711 Zuma Toolset provides software support which allows the user to easily send and receive RS232 serial data. This high-level interface frees the user from the burden of low-level programming of

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

the UART device. An interrupt driven mechanism is installed, along with internal send & recieve queues, which handles RS232 data transfers. The following table lists the available user functions for managing serial communications. Note that all read and write functions will 'block' to assure all data is transferred.

Function Name	Operation
init_communications	Sets up the Messaging interface and enumerates the USB connection, initializes the UART, and installs a shared interrupt handler.
init_comm_serial_only	Initializes the UART and installs the interrupt handler, leaving the USB idle. (Intended for applications that will not be using the USB.)
SerialSetBaud	Set the UART baud rate generator to a specified rate.
SerialPortReset	Resets the 16C750.
SerialGetChar	Read an 8-bit character from the input queue.
SerialPutChar	Transmits an 8-bit character.
SerialRead	Reads a buffer of 8-bit characters.
SerialWrite	Transmits a buffer of 8-bit characters.
SerialInQDepth	Returns the number of unread characters in the input queue.
SerialOutQRoom	Returns the amount of available space in the output queues.

TABLE 13. RS232 Communication Functions

Refer to the Zuma help file for additional details on these functions.

Example Program for the RS232 Interface

SerialTest. The SerialTest program is intended to demonstrate a simple loopback test using the RS232 serial port. This example consists of two components. The first is `SerialTest.HT` which is a script file that will properly configure HyperTerminal. Characters typed in the HyperTerminal window will be transmitted via the PC's serial port. Any characters received over the PC's serial port will be displayed in the HyperTerminal window. The SBC6711 executing `SerialTest.OUT` will initialize the UART and begin echoing back any received characters. The init_comm_serial_only() function is used to so that the interrupt handler is configured for RS232-only transfers, leaving the USB idle. Therefore, this application can be embedded and run from flash memory in the absence of a USB connection. When in this mode, the SBC6711 can still be reset via the USB.

Building Flash Programmable Applications

The SBC's flash EEPROM and automatic power-up bootload support provide the means to allow customers to program applications into the target's EEPROM memory. An application program installed in this non-volatile memory may be optionally bootstrap loaded automatically on power-up or board reset, allowing the SBC to operate in an isolated environment with no requirement for connection to a host computer.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The SBC6711 Zuma Toolset contains software support for the SBC's bootload capability, and allows the user to easily generate custom applications and program them into the SBC's flash EEPROM for automatic start-up. The program build process is supported under the Code Composer Stuido environment, and the Zuma Toolset includes a flash programming utility, which is used to embed the finished application into flash EEPROM.

The program build process for creating an application intended for programming into the flash memory is identical to that used to generate COFF files for download using a JTAG debugger or the UNITERMINAL application. However, after the executable .out file has been created, in must be converted into binary, boot-loadable format in order to be placed into flash ROM using the supplied BURN.EXE applet. This operation is accomplished using the PromImage.exe applet, provided in the Zuma toolset.

Follow the sequence below in order to embed an application into SBC6711 flash memory.

1. Build and debug the embedded application using Code Composer Studio.
2. Use the PromImage applet to convert the application COFF file into proprietary boot-loadable format.
3. Run the Burn applet to embed the boot load image into sector one of Flash memory.
4. Install the boot jumper onto the Sbc. Reset the Sbc and the application will load and run from flash.

Example Target Programs for the SBC6711

The following section details the example target software included with the Developer's Package. These programs are provided as models for custom user software, and it is highly recommended that the user examine these examples before beginning a first development effort for the target. Full source code is provided for user inspection and reuse in modified or custom applications.

These examples will run on a standard SBC6711 cards with no additional hardware required.

HELLO

`HELLO` is a very simple introduction to basic program components and use of the C stdio library for the target card. When run with the host terminal emulator active, the program simply initializes the target hardware and stdio interface and prints the message "Hello World!" via the stdio library to the terminal emulator screen. The program then drops into an infinite dwell loop.

`HELLO` may be rebuilt from with the Code Composer Studio environment by loading the `HELLO.MAK` project from the **`\target\examples`** directory, and rebuilding the project by selecting Project/Rebuild All from the menu. Refer to the Code Composer Studio documentation for more information on the application's project management and make facilities.

For correct program functionality, it is necessary to run the `HELLO` application via the host terminal emulator program. If the terminal emulator is not active and communicating with the target SBC card

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

on which HELLO is running, the application will appear to hang at the first instance of a stdio function call (usually a `getint()` or `putint()` call). This is due to the fact that all stdio calls use the SBC bus mailbox interface and are handshaken with the host terminal emulator application. Any such calls will hang if the terminal emulator is not active to complete the communication link.

TEST

TEST is board level hardware test program, capable of exercising the major peripherals on the SBC to double-check proper hardware functionality. As such, it contains routines for exercising each of the peripherals on the SBC, including:

1. Digital I/O.
2. Internal timers.
3. External timers.
4. Communications Ports.

Since the TEST program aims to be all-encompassing in that it tries to test as much of the board-level functionality as possible, it serves as a poor example for complicated operations such as A/D multi-channel sampling and display. However, since the code included for TEST is broken down into functional pieces, which are called separately for each subsystem to be tested, it is possible to factor out individual tests for use in other programs.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CHAPTER 9

DOS Environment Requirements

Innovative Integration (I.I.) Developers Packages and the TI C Compiler make use of environment variables in order to locate header files monitor script files, etc. Be sure to set the following environment variables when installing either the C compiler or I.I. libraries. Note that several of these environment variables may be automatically set when running the SETUP program on the distribution disks. However, when upgrading from previous versions or when mixing development components from I.I. or other sources, problems can arise.

Use the table below to insure that you specify all needed environment variables.

Environment Variable Name	Products Affected	Suggested Settings
II_BOARD	Dev Pkg Applets	**set II_BOARD=**<*board dir*> ie set II_BOARD=c:\SBC6711
C_DIR	All TI C Compilers All I.I. peripheral libraries	**set C_DIR=**%ii_board%;%ii_board%\include\target;%ii_board%\lib\target;%compiler dir%\rtdx\include;%compiler dir%\rtdx\lib;%compiler dir%\cgtools\include;%compiler dir%\cgtools\lib;%compiler dir%\bios\include;%compiler dir%\bios\lib; ie set C_DIR=C:\SBC6711;C:\SBC6711\include\target;C:\SBC6711\lib\target;C:\ti\c6000\rtdx\include;C:\ti\c6000\rtdx\lib;C:\ti\c6000\cgtools\include;C:\ti\c6000\cgtools\lib;C:\ti\c6000\bios\include;C:\ti\c6000\bios\lib; Specified order is critical!
A_DIR	All TI Assemblers	Same as C_DIR above.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

D_DIR	TI Debuggers	**set D_DIR=**<*debugger dir*> ie set D_DIR=c:\ti\c600\evm6x\lib
D_SRC	All Debuggers	**set D_SRC=**<*source code dir1*>;<*dir2*>;...;<*dir n*> ie set D_SRC=c:\SBC6711\stdio;c:\SBC6711\dsp; c:\SBC6711\periph\analog;c:\SBC6711\periph\digi-tal;… ;c:\SBC6711\periph\bus
PATH	All II products All TI Tools	**set path=**%path%;%ii_board%;%ii_board%\lib\host;%compiler dir%\cgtools\bin;%compiler root%\bin ie set path=%PATH%;C:\SBC6711;C:\SBC6711\lib\host;C:\ti\c6000\cgtools\bin;C:\ti\bin

TABLE 14. Required Disk Directory Structure for II Development Tools.

The Innovative Integration, TI, C Development System for the SBC6711 requires the following environment variables be set properly for correct operation:

set ii_board=C:\SBC6711

set C_DIR=C:\SBC6711;C:\SBC6711\include\target;C:\SBC6711\lib\target;C:\ti\c6000\rtdx\include;C:\ti\c6000\rtdx\lib;C:\ti\c6000\cgtools\include;C:\ti\c6000\cgtools\lib;C:\ti\c6000\bios\include;C:\ti\c6000\bios\lib;

set A_DIR=%C_DIR% (same as c_dir)

set d_src=C:\SBC6711;C:\SBC6711\stdio;C:\SBC6711\periph\bus;C:\SBC6711\periph\digital;C:\SBC6711\periph\misc;C:\SBC6711\dsp;C:\SBC6711\periph\analog;C:\SBC6711\periph\flash;C:\SBC6711\periph\bus\polled

set path=%PATH%;C:\SBC6711;C:\SBC6711\lib\host;C:\ti\c6000\cgtools\bin;C:\ti\bin

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CHAPTER 10

SBC6711 Hardware

SBC6711 Hardware Functions

The SBC6711 is a stand alone digital signal processor (DSP) card based around the Texas Instruments TMS320C6711 processor. Implementing a modular I/O expansion system, the SBC6711 is particularly well suited to data acquisition and control tasks. The SBC6711 is also supported by a collection of I/O bus function cards, which provide hardware interfacing to real-world equipment.

The SBC6711's features include:

1. TMS320C6711 digital signal processor.
2. External one wait-state SDRAM memory pools.
3. FLASH memory.
4. FIFOPort communications expansion capability.
5. OMNIBUS I/O expansion compatible (two available slots).
6. 32 bits of digital I/O.
7. Synclink and clocklink trigger and clock sharing bus.
8. Universal Serial Bus (USB) port.
9. JTAG hardware emulation support.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The following figure gives a block diagram of the SBC6711.

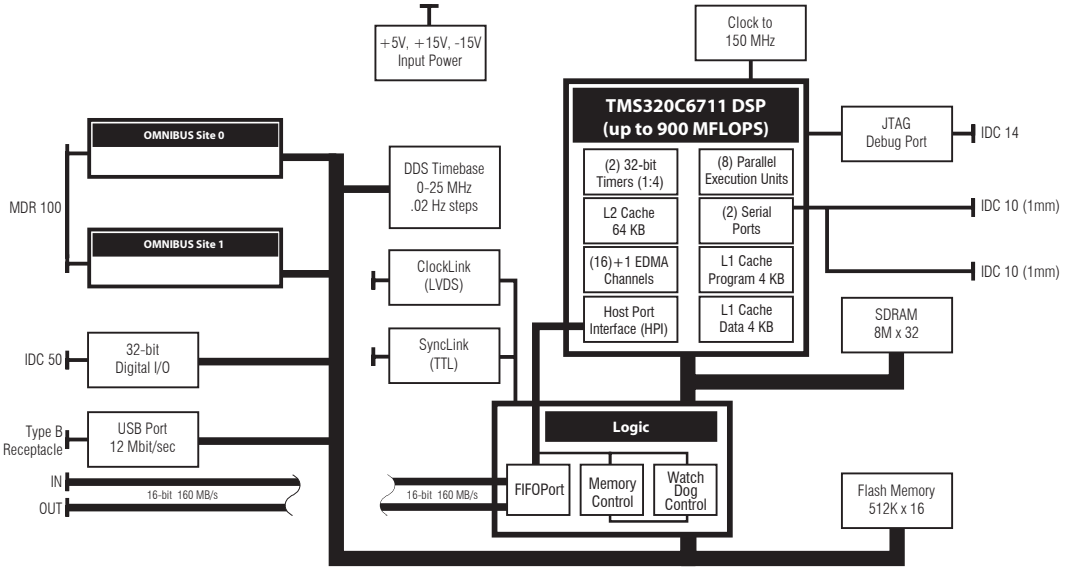


FIGURE 33. SBC6711 Block Diagram

Memory Map

The SBC6711 processor operates in 'C6711 ROM boot mode. In this mode, applications initially boot from external flash EEPROM and run from the on-chip memory. As part of the boot process, the 'C6711 cache memory controller is enabled so that code runs using the SDRAM and cache memory.

The following figure gives the processor memory map of the SBC6711 for external peripherals and memory. Please note that this table ignores any on-chip resources (see TI peripheral manual for TMS320C6711 for on-chip peripherals).

Function	Address	Memory Space	R/W
SDRAM	0x80000000	CE0	R/W
FLASH Memory	0x90000000	CE1	R/W
USB Register Set	0x90200000	CE1	R/W
UART Register Set	0x90201000	CE1	R/W
DDS	0x90203000	CE1	W
Digital IO direction control	0x90300000	CE1	W

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Function	Address	Memory Space	R/W
Status Register	0x90301000	CE1	R
Control Register	0x90302000	CE1	W
DSP Interrupt 4 source select	0x90303000	CE1	W
DSP Interrupt 5 source select	0x90304000	CE1	W
DSP Interrupt 6 source select	0x90305000	CE1	W
DSP Interrupt 7 source select	0x90306000	CE1	W
DSP NMI Interrupt source select	0x90307000	CE1	W
FIFOPort	0x90308000	CE1	R/W
FIFOPort Transmit FIFO PEN Mode	0x90309000	CE1	W
FIFOPort Status/Flags	0x9030A000	CE1	R/W
FIFOPort Receive FIFO interrupt level	0x9030B000	CE1	W
DSP Timer 0 clock input config	0x9030C000	CE1	W
DSP Timer 1 clock input config	0x9030D000	CE1	W
Timer outputs configuration	0x9030E000	CE1	W
DDS pin output configuration	0x9030F000	CE1	W
Clocklink output configuration	0x90310000	CE1	W
Synclink master control	0x90311000	CE1	W
Synclink output control	0x90312000	CE1	W
Watchdog timer reset	0x90313000	CE1	W
DDS frequency update	0x90314000	CE1	W
Logic SROM	0x90315000	CE1	R/W
Logic SROM write enable	0x90316000	CE1	W
Timer 0	0x90317000	CE1	R/W
Timer 1	0x90318000	CE1	R/W
Timer 2	0x90319000	CE1	R/W
Timer 0 configuration	0x9031A000	CE1	W
Timer 1 configuration	0x9031B000	CE1	W
Timer 2 configuration	0x9031C000	CE1	W
Interrupt polarity	0x9031D000	CE1	W
Interrupt type	0x9031E000	CE1	W
DSP Int4 status/ack	0x9031F000	CE1	R/W
DSP Int5 status/ack	0x90320000	CE1	R/W
DSP Int6 status/ack	0x90321000	CE1	R/W
DSP Int7 status/ack	0x90322000	CE1	R/W
Digital I/O data register	0x90324000	CE1	R/W
I/O Module Strobe 0	0xA0000000	CE2	R/W
I/O Module Strobe 1	0xA0010000	CE2	R/W
I/O Module Strobe 2	0xA0020000	CE2	R/W
I/O Module Strobe 3	0xA0030000	CE2	R/W
I/O Module Strobe 4	0xB0000000	CE3	R/W
I/O Module Strobe 5	0xB0010000	CE3	R/W
I/O Module Strobe 6	0xB0020000	CE3	R/W
I/O Module Strobe 7	0xB0030000	CE3	R/W

TABLE 15. SBC6711 External Memory Map

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SBC6711 Hardware Initialization Requirements

The SBC6711 design requires the following values to be written to its hardware control registers in order to provide access to on-board hardware:

Register	Address	Value
EMIF Global Control	0x01800000	0x88
CE1 Control	0x01800004	0x20810220
CE0 Control	0x01800008	0x30
CE2 Control	0x01800010	0x00400120
CE3 Control	0x01800014	0x00400120
SDRAM Control	0x01800018	0x53116000
SDRAM Refresh	0x0180001C	0x465
SDRAM Extension	0x0180001C	0x000544a7
Interrupt Polarity	0x019C0008	0x00000000

TABLE 16. SBC6711 Bus Control Register Initialization Values

These values are initialized automatically by C programs compiled under the SBC6711 Development Package software libraries. Be sure to include initialization of these values whenever software is developed outside the Development Package or when a JTAG hardware assisted debugger is employed for code downloading to the SBC6711 (i.e. when using Code Composer Studio or any other JTAG debugger package).

External Memory

The SBC6711 has two types of external memory: synchronous DRAM (SDRAM), and flash EEPROM. The EEPROM memory is 512K x 16 (1Mbytes) and the SDRAM is organized as 8Mx32 (32 Mbytes).

The flash EEPROM memory contains the SBC6711's start-up software. Shipped standard with Innovative's TALKER program installed for use with the SBC6711 Development Package, the EEPROM may be programmed with end user applications through the use of tools included in the Development Package. The EEPROM and boot process makes the SBC6711 a stand-alone processor board capable of remote control and data acquisition tasks. The EEPROM operates with several wait-states on the processor bus.

EEPROM memory is implemented on the SBC6711 with a single AMD 29F800 (compatible) flash device. In addition to its intended purpose as the boot device for application software, the 29F800 may also be used to store nonvolatile application-dependent data. The device is connected to the 'C6711 processor on the least significant sixteen bits of the processor bus, and responds to standard AMD programming sequences. The SBC6711 Development Package contains support software for erasing and programming the flash device.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The SDRAM memory provides a large, fast memory area for program and data storage. The ‘C6711 has an advanced cache controller that is used to control the SDRAM so that the programmer has a large virtual memory pool of very fast memory. The cache controller allows the SDRAM to operate at approximately 80% the efficiency of internal memory in many cases by caching instructions and data in on-chip memory for faster access. The 16 Mbyte SDRAM operates at one half the processor’s bus rate, for a maximum throughput of 300 Mbytes/sec.

SBC6711 OMNIBUS

The SBC6711 I/O bus provides a modular, high-speed expansion area, which is directly tied to the processor’s bus and is ideally suited for I/O hardware expansion. Direct memory-mapped accesses allow the processor to transfer data to and from I/O bus peripherals constructed as plug-in modules, which can be mixed and matched to suit the particular user’s functional requirements. Omnibus is an open standard that allows customers to design application specific modules to use with the SBC6711 and other Omnibus cards.

The OMNIBUS slots are accessed as memory-mapped peripherals with the SBC6711 providing four decoded chip select signals per slot, for a total of eight on the SBC6711. The following figure gives the memory map for the OMNIBUS slots, and shows the decode signal to slot mapping.

Function	Starting Address	Module Slot
OMNIBUS Strobe 0	0xA0000000	0
OMNIBUS Strobe 1	0xA0010000	0
OMNIBUS Strobe 2	0xA0020000	0
OMNIBUS Strobe 3	0xA0030000	0
OMNIBUS Strobe 4	0xB0000000	1
OMNIBUS Strobe 5	0xB0010000	1
OMNIBUS Strobe 6	0xB0020000	1
OMNIBUS Strobe 7	0xB0030000	1

TABLE 17. SBC6711 I/O Bus Memory Mapping

Each module site provides a 32-bit wide data bus connection to the processor’s data bus, with 12-bits of low order address signals for additional decoding beyond the four chip select signals available per slot. Since the Omnibus modules are 32-bit devices, and the DSP has byte addressing, the address mapping to the SBC6711 is such that Omnibus address A0 is address A2 on the DSP.

Each module also connects to a ‘C6711 serial port (serial port zero for slot zero, and serial port 1 for slots 1) to allow serial port driven I/O. Bus reset, RDY, R/W, and processor clock signals are available, as are power connections for digital 5V and analog +/-5V and +/-15V. Timebase connections include timer channels from both the 16-bit timers and the 9851 direct-digital synthesizer (DDS).

Each OMNIBUS slot has a 50 pin undedicated connector (JP4 on slot 0 and JP8 on slot 1) that provides access from the external I/O to and from a module installed in the slot. The slot’s I/O connector is in

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

turn pinned out to a single 100 pin MDR connector (JP5) for use in attaching cables from external hardware. Cables and breakout modules for the MDR 100 output connector are available from Innovative. Custom cables may also be made using the connectors specified in the appendix.

Connector pinouts for the module sites are provided in the appendices. Individual pin functions are noted in the tables, and in general the OMNIBUS pinout represents a direct connection to the 'C6711 local bus.

SBC6711 OMNIBUS Memory Mapping

Since the 'C6711 processor is a byte addressable machine which implements its address bus based on a 32-bit transfer width (i.e. the address bus starts at A2 and separate byte enable pins are supplied to control accesses to individual bytes within the 32-bit wide location denoted by the address bus), users must take care when writing software which performs OMNIBUS accesses.

The OMNIBUS specification requires 32-bit accesses and does not support byte or half-word (16-bit) accesses. No support is included in the specification for the 'C6711's byte enable pins. This means that software performing accesses must always perform 32-bit transactions with the OMNIBUS modules. When writing C code for the SBC6711, programmers should use only variables of type int or unsigned int (or their derived types), and all accesses should be word justified (the least significant nibble of the address must always be a multiple of four). Accesses generated using pointers to variables of type char, short, or long will cause erroneous non-32-bit accesses. Correct OMNIBUS module operation under these situations can not be guaranteed.

Please note that memory decoding within the OMNIBUS decode regions uses 32-bit addressing and that the memory map tables given in the *OMNIBUS Manual* should be treated appropriately. For example, the description of the OMNIBUS DIG module notes that the byte 3 direction control register for a module installed in site 0 is mapped to address IOMOD2 + 3. This address should be literally interpreted as 0xA002000C, where IOMOD2 is equal to 0xA0020000 and the offset adds decimal 12 (three 32-bit words of offset). IOMOD2 + 3 should NOT be interpreted as 0xA002003, since the offset is 3 32-bit words and not 3 bytes.

This addressing is most easily handled in C by using integer pointers and integer pointer arithmetic, which will always result in the required address alignment. For example, the following code defines a pointer and accesses the byte 3 direction control register with the documented offset:

```
unsigned int *pointer = 0xA0020000;

*(pointer + 3) = 0x0;       /* set byte 3 to output mode */
```

The actual accessed memory location is 0xA002000C, due to the way pointer math is handled in C.

OMNIBUS Power

The OMNIBUS interface provides five separate power supplies for use by modules along with two separate ground return connections. The following table lists the power supplies and their power ratings. A separate digital 5V supply is provided along with separate digital grounds to minimize the digital noise present on the analog power supplies.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Pin Name	Voltage	Current Rating (max.)
DVCC	5V (digital)	(System dependent)
AVCC	5V (analog)	500 mA
-AVCC	-5V	500 mA
+AV	+15V	(System dependent)
-AV	-15V	(System dependent)

TABLE 18. I/O Bus Power Ratings

Note: The SBC6711 implementation of the OMNIBUS expansion slots deviates from the standard specification in the way the power supplies are handled. In order to provide a more compact form factor for the host card, the SBC6711 does not supply discrete +/-12V power supplies as required by the OMNIBUS specification. Instead, it connects the +/-AV rails to the +/-12V OMNIBUS power pins, resulting in a 3V absolute overvoltage on those supply pins. Designers of custom OMNIBUS modules intended for use with the SBC6711 should keep these revised power supply values in mind when planning circuitry, which connects to these power supplies.

Please note that the AGND and DGND busses are separated on the SBC6711 and for proper ground referencing they must be tied together on modules which use the analog power supplies (any supply other than digital 5V, 12V, or –12V). Innovative Integration recommends that either a ferrite bead (Panasonic EXC-ELSA35V or equivalent) or a hard wire connection, depending on expected ground return current and frequency content from the analog power supplies being used on custom modules to connect the two ground busses. The current steering used by the separate grounds prevents high frequency digital noise on the DGND bus from polluting the clean AGND return.

FIFOPort I/O Expansion

The FIFOPort feature provides a single buffered bi-directional 16-bit interface, which allows external hardware or other Innovative processor and data acquisition boards to communicate with the SBC6711 at high data rates. The FIFOPort communicates with the ‘C6711 as 32-bit words, arranged as two 16-bit words of FIFOPort data. For transmit, the lower 16-bits is sent first followed by the upper 16-bit half. For receive, the first 16-bit word received is the lower 16-bits of the 32-bit output word, and the second 16-bit word is the upper half of the 32-bit word. A single 256x32 bit FIFO is provided to buffer incoming strobed parallel data, while a FIFOPort compatible output supports clocking data to external hardware or other FIFOPorts. The FIFOPort is memory mapped into the CE1 space at address 0x90308000.

The following diagram illustrates the FIFOPort’s operation. The FIFO buffer memory serves to clock incoming data and store it for use by the ‘C6711 processor. Data is formatted as a 16-bit wide data bus synchronous with a rising edge strobe signal, which acts as the load clock to the FIFOport receive logic. The output portion consists of the same two signals: output data plus the strobe signal for the receiving end of the port.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

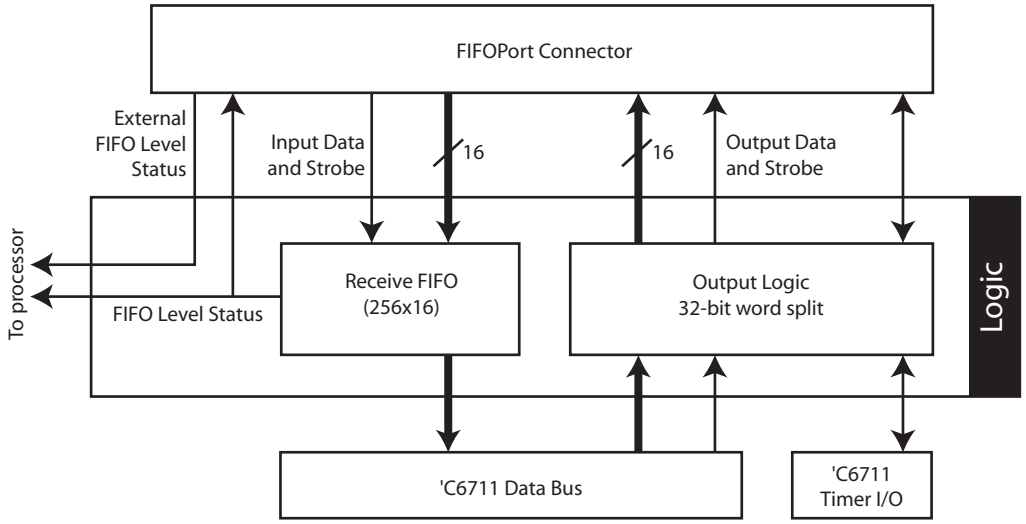


FIGURE 34. FIFOPort Block Diagram

The FIFOPort also provides external access to the receive programmable almost full flags to allow hardware to monitor the FIFO's level status. The port can also receive FIFO level status from external hardware to allow the 'C6711 processor to monitor level status of FIFOs located off the SBC6711 card. Both the onboard receive FIFO level status and the off board FIFO status lines may be polled or may generate interrupts to the 'C6711 processor.

Transmitting and Receiving FIFOPort Data

Data is transmitted and received on the FIFOPort by means of processor address location 0x90308000 as 32-bit words. EMIF read and write accesses (either due to CPU or DMA activity) accesses the FIFOPort control logic in the FPGA when this address is accessed. The FIFOPort read and write paths are fully independent, allowing overlapped operation and separate DMA channels.

In the case of a write access, the 32-bit word written by the DSP results in the logic generating two writes on the FIFOPort bus. The 32-bit word is sent as two 16-bit words, low half first, using an active high output strobe. The upper 16-bit half of the data is automatically written immediately after the first half. These 16-bit data should be latched by external hardware on the rising edge of the FIFOPort output strobe. Write accesses do not affect the current state of the receive FIFO.

In the case of a read, the DSP reads 32-bit words from the FIFO. Each 32-bit word in the FIFO has been assembled from two 16-bit words received over the FIFOPort link. The lower 16-bits is received first, followed by the upper 16-bits. Odd length transmissions need to be padded by one 16-bit word so that the FIFO completes the 32-bit word. If the data item being read in the current cycle is not the last item stored in the buffer, the next data item is clocked out by the FIFO and held ready for the next read access by the processor. Read accesses do not generate output strobes to the external connector.

If the receive FIFO grows empty, the last data item's value will be output on any subsequent read accesses.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Monitoring FIFO Status

The FIFOPort provides a FIFO level monitoring feature which allows software to read the receive FIFO's level indicators as well as FIFO level data from external hardware (if connected). The receive FIFO's empty, full, and programmable almost full flags can be read at any time by the CPU. Alternatively the interrupt selection matrix may be programmed to notify the CPU of level events via an interrupt (see Interrupts section for more information). The same functionality is provided for the external FIFO, allowing the CPU to read back or be interrupted by any of six different level state conditions.

The FIFO level status register resides at address 0x9030A000 in the processor memory map with the bit configuration shown in the following figure.

Bit	Function
0	FIFOPort Tx level int state.
1	FIFOPort Rx level int state.
7..2	not used
8	FIFOPort Rx full. '1' = full
9	FIFOPort Rx quarter-full. '1' = 25% full or greater
10	FIFOPort Rx half-full. '1' = full
11	FIFOPort Rx not empty. '1' = not empty
14..12	not used. Read back as 0's
15	FIFOPort Tx busy. This bit = '1' if the transmitter is busy.
31..16	0x1303 readback

TABLE 19. FIFOPort Level Status Register

The receive FIFO level bits are read directly from the logic controlling the FIFOPort, while the transmit FIFO bits are read from the level input pins on the FIFOPort connector. If no external status is being reported by the hardware connected to the FIFOPort, then these bits will read as zeros (onboard 10K pull-up resistors hold the transmit full, empty, and almost-full input pins high). If external FIFO level reporting is not desired, the level inputs may be used for application specific bit inputs to report other hardware status conditions or trigger interrupts on the SBC6711 processor. Note that this is in addition to the dedicated timer I/O pins and the processor interrupt input pin on the FIFOPort connector, which remain available regardless of the use of the FIFO status inputs.

With the appropriate programming, the FIFO levels may also be monitored using processor interrupts. The FIFO status bits are available as sources to the processor interrupt selection matrix. This technique is typically used to drive DMA transfers to and from the FIFOPort. Where one FIFO status interrupt triggers one or more transfers using DMA synchronization, or for CPU interrupts where the target CPU in a transfer wants to be interrupted when data (or space) is available in the FIFO. This would be typical of "one-shot" FIFO transfers, where a single full FIFO's worth of data is transferred at once and the receiving processor needs to be notified when the FIFO reached the full state so that a read operation on the other side of the FIFO may commence. For more information on using the FIFO levels to trigger interrupts to the 'C6711 processor, see the Interrupts section.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

FIFOPort Reset

The receive FIFO may be cleared and its condition reset at any time by accessing the FIFOPort control register at address 0x90302000, bit '0'. Writing a '1' to bit 0 of the control register puts the FIFOPort in reset. Write a '0' to this bit to release the FIFOPort from reset. When in reset, the FIFO levels are cleared and the flags change to reflect the FIFO empty status, and the programmable level control variables are reset to default values (see below for more information).

Controlling the FIFOPort Programmable Level Flag

The FIFOPort provides a programmable level flag which can be used to make an interrupt to the DSP when the FIFO level is at or above the specified level. This feature is particularly suitable to DMA block transfers on the FIFOPort because it maximizes the transfer rates on both sides of the FIFO by keeping the buffer partially filled. This level interrupt is also useful in packet transfers, where the threshold level is set to the packet size. When used in this method, each time a packet is received an interrupt will be signaled.

The receive level flag has a level register which holds the level threshold at address 0x9030B000. This register initializes to 128 words, which is half-full for the SBC6711 FIFOPort FIFO. If the level exceeds this register, the level flag will go true ('1') and remain true so long as the level is above the threshold. The threshold value is 0 to 255 32-bit words. The level register is only reset by a card reset.

The transmit level flag is also programmable. This register provides level information back to the transmitting card so that it may pace its FIFOPort transmissions. This register is programmable only by the transmitting card. For the transmitting card to change the threshold level, it must first lower (set to '0') the TX PEN line indicating that the next word will be used to set the level register. This value is from 0 to 255 and is aligned on the lower incoming byte. This value is sent using a normal FIFOPort write while the PEN is low. The default value for the TX threshold is 128 32-bit words, and is reset only on a card reset. When the SBC6711 is transmitting to another SBC6711 or similar card, this procedure should be followed by each transmitter to insure that the FIFOPort transfers are paced by the FIFO level so that overflow conditions do not occur. The transmit PEN bit is mapped to 0x90309000, bit 0 and is inactive ('1') by default at power-up.

FIFOPort Flags

The FIFOPort also has one bit which may be written to signal the other card that the FIFOPort is ready for use, and one incoming bit from the other card that may be read. The outgoing flag indicating the FIFOPort is ready for use may be written at 0x9030A000, bit 0. The incoming flag indicating the other card is ready may be read at 0x9030A000, bit 0. The bits are '1' by default, so active low logic is expected.

Software can use these two bits as ready indication to the transmitting card. This allows the receiving card to be set up and ready to receive the expected data when the flag is set to the transmitting card.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Designing External Hardware for use with the FIFOPort

Use caution when designing external hardware, which is to be connected to the FIFOPort. The signals present on the interface connector are extremely high speed and failure to handle them appropriately can cause functional problems with the FIFOPort. Innovative Integration does not recommend driving cables directly as capacitive load and ringing issues can cause corruption of the transmitted data. FIFOPort connector pinouts are given in the appendices at the end of this manual.

The following diagrams give timing information for the FIFOPort circuitry. This data is derived from device specifications and is not factory tested.

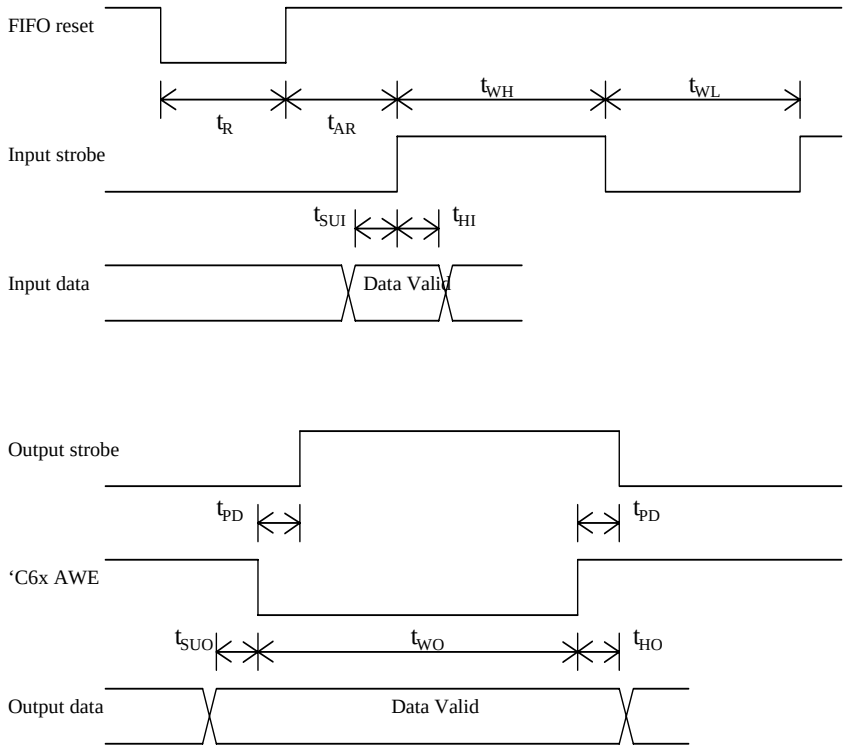


FIGURE 35. FIFO Port Timing

Parameter	min. (ns)	max. (ns)
t_{WH}	7	
t_{WL}	7	
t_{SUI}	5	
t_{HI}	0	
t_R	10	

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameter	min. (ns)	max. (ns)
t_{AR}	5	
t_{PD}		7
t_{SUO}	10[1]	
t_{HO}	0[1]	
t_{WO}	10[1]	

TABLE 20. FIFO Port Timing Parameters

Notes [1] : Dependent on EMIF programming for CE0 space as well as processor cycle frequency. These values are determined from recommended EMIF register values.

All FIFOPort signals are TTL compatible CMOS inputs and outputs. All input pins may be safely driven with 5V logic devices. All FIFOPort signals are driven with LVTTL (3.3V) logic, which meet TTL requirements for 2.4V high and 0.7 volt low.

Innovative also sells an active LVDS cable for linking FIFOPorts together that supports full rate data transmissions. Each direction of this cable is buffered using LVDS receiver/drivers for high signal integrity. This cable may also be used with custom equipment interfacing to the FIFOPort. Contact the sales department at Innovative for further details.

'C6711 Serial Ports

The 'C6711's on-chip serial ports are pinned out to connectors JP18 (port 0) and JP19 (port 1) for use with external hardware. The serial ports are also connected to the OMNIBUS sites for use with modules designed to interface to the processor serially.

Pinouts for the serial port connectors are given in the appendices. Innovative Intergration recommends buffering these ports with off board hardware in order to preserve signal integrity.

RS232 Serial Port

The SBC6711 implements a single asynchronous serial port channel with RS232 drivers compatible with standard PC serial port interfaces. The serial port is implemented using a Texas Instruments TL16C750 UART and Maxim MAX3223 RS232 transceiver. The following table gives the address mapping for the UART device.

16C750 Register	C6x Address
TXD	0x90201000
RXD	0x90201000
DLL (DLAB=1)	0x90201000

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DLM (DLAB=1)	0x90201004
IER	0x90201004
IIR	0x90201008
FCR	0x90201008
LCR	0x9020100C
MCR	0x90201010
LSR	0x90201014
MSR	0x90201018
SCR	0x9020101C

TABLE 21. UART Control Registers

For hardware and programming details on the 16C750 UART, see the data sheet for the device (available on the Texas Instruments web site at www.ti.com). The 16C750 data sheet contains a complete description of the registers used to configure and control the UART.

RS232 serial port signals are present on connector JP22. For more information, refer to the Connector Pinouts section in the appendices.

The 16C750 external reset signal is automatically asserted on power up and may be reasserted under software control via the SERIAL_RESET bit of the miscellaneous control register.

RS232 Interrupts

The 16C750 supports an interrupt output to the ‘C6711 processor along with a fully programmable interrupt masking system. The interrupt generation hardware allows for fully event driven RS232 communications.

The Zuma Toolset from Innovative provides a high-level interface to facilitate RS232 data transfers. These functions implement an interrupt-driven mechanism which uses internal queues to manage bi-directional data flow. This eliminates the need for the user to hande low-level interaction with the UART. The SBC6711 Development Package contains complete source code for the UART interrupt handler.

To receive an interrupt from the UART it is necessary to properly configure the SBC6711 interrupt control logic such that the interrupt output from the UART is connected to the desired interrupt input pin on the ‘C6711 processor. For more information on programming the interrupt control logic on the SBC6711, refer to the Interrupts section.

USB Port

A Universal Serial Bus (USB) compatible host interface port is included on the SBC6711 to support high data rate host communications requirements. Implemented using a ScanLogic SL811HST USB interface device, the port allows for communications on a USB network at up to 12 Mbits/sec (USB full speed bit rate). In-system real data rates will be determined by the USB data packet sizing, host bus rate and other limitations. Most systems can achieve from 4-8 Mbits/sec under Win2000 (Win98 is slower).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The USB port supports USB specification 1.1 at 12 Mbit/sec or 1.5Mbit/sec data rates. Data rate is automatically sensed by the controller chip during the USB initialization. The SL811 provides an efficient dual-port memory interface between the DSP and serial engine with address auto-incrementing that allows the SBC6711 to achieve high data rates without unnecessary burden.

The USB interface is accessed starting at address 0x90200000. The following table gives the SL811device register mappings.

SL811 Register	Access Direction	Address
DATA_OUT	R	0x90200000
DATA_IN	W	0x90200000
Reserved	R	0x90200001
ADDRESS	W	0x90200001

TABLE 22. USB Control Registers

The SL811 uses an indirect addressing method whereby the address of the device register to be accessed is written to the ADDRESS register. Then a read or write operation performed on the DATA_OUT or DATA_IN registers, respectively, to retrieve or store the requisite data. The control logic automatically paces accesses to the UBSB device so that access-to-access timings are respected without software intervention. The SL811 USB controller has a maximum read rate of approximately 4.5M bytes/sec due to the timing limitations of the chip itself.

The SBC6711 provides a byte wide interface between the SL811HST and the ‘C6711 processor. Data read from the device is right justified into the least significant eight bits of the data bus, while the device latches the least significant eight bits of the data bus on each write to the device. The remaining most significant 24 bits are undefined on reads and should be masked off.

Please note that the USB interface does not operate at the slow USB data rate of 1.5 Mbits/sec. Host ports and hubs to which the SBC6711 are to be interfaced must support the full USB rate in order to communicate with the SBC6711.

Resetting the SBC6711 over USB port

The SBC6711 may be reset over the USB interface. This allows the host computer to initialize the SBC6711 for use and is especially convenient during software development. This feature uses the standard USB reset signalling mechanism to trip a reset to the SBC6711. During application development, this allows the PC to grab control of the SBC6711 in case of program runaway.

After development is completed, the application booting from FLASH memory may disable resets from the USB bus so that the SBC6711 continues to run even when the PC issues a USB reset. This prevents PC resets to the USB bus during device identification cycles from resetting the SBC6711. This feature may be disabled by writing a ‘0’ to bit 6 of the control register at 0x90302000. By default, USB reset is enabled after reset or power-on.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

USB Interrupts

The SL811 supports an interrupt output to the ‘C6711 processor along with a fully programmable interrupt masking system. The flexible interrupt generation hardware allows for fully event driven USB message and state processing.

The Zuma Toolset from Innovative provides a number of high-level USB management features that give high data rates without managing low-level interaction with the USB controller. These functions implement an event-driven interface with the processor responding to state changes and packet reception by enabling selected interrupts from within the SL811’s interrupt control system. The SL811 data sheet contains a complete description of the relevant registers used to control the masking of individual interrupts and the required handshake responses once an interrupt is received from the interface.

The SBC6711 Development Package contains complete source code for the USB interrupt handler.

To receive an interrupt from the SL811 it is necessary to properly configure the SBC6711 interrupt control logic such that the interrupt output from the USB interface is connected to the desired interrupt input pin on the ‘C6711 processor. For more information on programming the interrupt control logic on the SBC6711, refer to the Interrupts section.

DMA Support and the USB Interface

The SL811 USB controller supports auto-incrementing data addressing for DMA interfacing. This feature allows DMA to be used for many of the data transfer functions from the USB interface to the DSP memory.

The SL811 interrupt pin is available as another input source to the SBC6711 interrupt control logic, allowing it to be connected to one of the ‘C6711’s interrupt input pins. The DMA controller in the DSP may then service this interrupt source for the processor. Configured in this way, each pulse on the interrupt pin can cause a read or write transfer to/from the SL811of a packet of data.

When DMA transactions are to be performed between the processor and the USB interface, one endpoint address should always be the USB data register (used as the source address for reads and the target address for writes). The SL811 should be in auto-incrementing address mode during this data transaction. The interrupt should be configured as a non-acknowledged interrupt (see interrupts section) that has a dedicated DSP interrupt. The other endpoint address can exist anywhere in processor memory, although it will most likely be in either on-chip or external (if populated) RAM. Bear in mind when performing DMA accesses that the USB interface is only 8-bit, so the DMA controller should be programmed appropriately depending on how the resultant data is to be handled (i.e. whether or not the source/destination data is packed as bytes into 32-bit words or is stored as single bytes in each 32-bit word).

USB Software Development

The SBC6711 Development Package contains extensive driver support for the USB environment. Users are strongly encouraged to make use of the standard software drivers in order to speed product

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

development. For more information regarding SL811 programming and the USB standard, see the SL811 data sheet and the Universal Serial Bus specification.

Timers

The SBC6711 provides a total of five counter/timers as well as the DDS used for timebase generation. These timers are independent and may be used as on board timebase generation for use in timing data acquisition, servo controls, real-time counters, and many other applications. The counter/timer functionality is divided into two devices: two 32-bit timer channels on the 'C6711 processor and three 24-bit timer channels in the logic. A 32-bit direct digital synthesizer (DDS) channel in the AD9851 device is also provided. This section discusses the AD9851 synthesizer in detail: for more information on the on-chip timers, see the *TMS320C6000 Peripherals Reference Guide*.

On-chip Timers

The on-chip DSP timers are available for use as software timebases and interrupt generators. These timers may be driven by several sources as defined by software-controlled registers in the logic. The sources available to drive the DSP timers are listed in the following tables. DSP on-chip timer 0 control register is located at 0x9030C000, and DSP timer 1 control register is located at 0x9030D000. The timers may also be cascaded to form one 64 bit counter/timer by setting bit 4 in DSP timer control register 1.

Bit	Timer 0 Input
0	External Clock Input (J1)
1	Timer Clock (80 MHz)
2	Module0 Interrupt
3	DDS Output

FIGURE 36. DSP On-Chip Timer 0 Clock Source Control Register

Bit	Timer 1 Input
0	External Clock Input (J1)
1	Timer Clock (80 MHz)
2	Module0 Interrupt
3	DDS Output
4	Timer 0 Output

FIGURE 37. DSP On-Chip Timer 1 Clock Source Control Register

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Note that the ‘C6711 does not support timer operation during JTAG emulation halt when the timers are programmed to use an external source. When clocking from an external signal, halting the DSP from a JTAG debugger will also halt the on-chip timers. This does not apply when the timers are run from the processor clock (timers will run when the CPU is halted).

AD9851 Direct Digital Synthesizer

The AD9851 direct digital synthesizer (DDS) is a precision programmable clock source which is capable of generating frequencies in the range of 0 to 25 MHz with a resolution of 0.019 Hz/step. Unlike a digital counter-timer chip, which uses a digital counter to divide down a high input clock rate, the DDS uses phase-locked-loop synthesizer technology to tune a sine wave oscillator based on a 32-bit digital word. This method realizes a linear output frequency over input range rather than the nonlinear one associated with counter-timer chips, whose resolution drops dramatically as the period register used to program them falls. The DDS should be used when a precise and accurate clock is required by the application.

The AD9851 is mapped into memory as shown in the table below. The device is interfaced using the parallel I/O method, with one address to write data, one to trigger frequency/phase updates, and one to control the reset pin of the device.

Function	I/O Space Address
AD9850 Reset	0x90302000, bit 1
AD9850 Frequency Update	0x90314000
AD9850 Write Clock	0x90203000

TABLE 23. AD9851 Control Registers

The write clock address latches frequency/phase data into the AD9851 one byte at a time. The least significant eight bits of the processor bus carry the byte wide data. The frequency update address causes the output frequency and phase of the DDS clock to update to the values contained in its input latches. The reset address causes an active high reset pulse to be generated to the AD9851. By default, the DDS reset is true at power-on or reset, so a ‘0’ must be written to the DDS reset control bit in the control register before configuration and use. All three of these registers are write-only.

The output of the DDS may be used for a variety of functions including driving interrupts, as a timebase to the DSP or modules, or as an output. Refer to the register descriptions for interrupt use, or timebase pin definition registers.

The SBC6711 Development Package includes support routines, which make it easy to set the AD9851’s output frequency as discussed in the previous sections of this manual.

The DDS has very fine resolution allowing the application to tune the timebase to many frequencies. The absolute accuracy of the DDS timebase is approximately 200 ppm for room temperature applications. This absolute accuracy may vary with temperature and time. Calibration may be required in applications requiring higher precision timebases. Designers may also be concerned with jitter, which is approximately 100 ps at 20 MHz.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

External Timer/Counters

There are three external 24-bit timer/counters that are resident in the logic that are separate, and in addition to the DSP on-chip timers. These three timer/counters have their control signals available on the digital I/O connector (JP3). Each timer has a separate clock and gate input, and a timer output pin. The counter values may be read directly by the DSP. Maximum input frequency for the timers is 75 MHz. The gate signal must be '1' to allow counting, which is pulled up on the board and can be driven to '0' to inhibit counting.

The timer/counters are software configurable for output mode (pulse or square wave) and timer input signal. The configuration registers are located at the memory addresses shown in the table below. The bit definitions are the same for each timer.

Timer Configuration Register	Address
0	0x9031A000
1	0x9031B000
2	0x9031C000

TABLE 24. External Timer/Counter Configuration Register Addresses

Bit	Clock Used
0	80 MHz on-board clock
1	Timer input clock (external on DIO connector)
2..5	not used
6	Counter reset: 0 = Reset (default) 1 = Run
7	Mode: 0 = Pulse (default) 1 = Square Wave

TABLE 25. External Timer/Counter Input Clock Control Registers Bit Definitions

The following table gives the address of the timer count/period register. When using the counter as a timer, write to this register to set the period, in clocks, of the timer. You must write the number of clocks minus one into this register. The timer output is programmable to be either a pulse or a square wave. When in pulse mode, you must write the number of clocks minus one, between each pulse. In square wave mode, write the number of clocks minus one for each half-period.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Timer Output Mode	Period Register
Pulse	Period = count +1
Square Wave	Period = 2(count +1)

TABLE 26. External Timer/Counter Period Formulas

Timer Count/ Period Register	Address
0	0x90317000
1	0x90318000
2	0x90319000

TABLE 27. External Timer/Counter Data Registers Addresses

Timer Output Control

The SBC6711 has a variety of timers, timebase controls, and external inputs that are useful as triggers along with timebases to the modules. The timer0, timer1 and DDS pins therefore have software programmable outputs so that any of the available triggers can drive the modules. Historically, Innovative has referred to these pins by the names timer0, timer 1, and DDS which reflected their dedicated function, but now the “timer0” pin can have any number of signals on it, not just the traditional timer0 function.

That being said, the register to program what signal is on the timer 0 and 1 pins delivered to the Omnibus is at memory location 0x9030E000 with the following bit assignments.

Omnibus Pin	Bit	Function
timer0	0	DSP timer0 Output
timer0	1	SyncLink
timer0	2	ClockLink In
timer1	16	DSP timer 0 Output
timer1	17	DSP timer 1 Output
timer1	18	SyncLink 1
timer1	19	ClockLink In

TABLE 28. Timer Pins Output Sources

The DDS output pin is also multi-sourced and is software controlled by a register at location 0x9030F000 with the following bit assignments.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Omnibus Pin	Bit	Function
DDS	0	DSP timer0 Output
DDS	1	DSP timer1 Output
DDS	2	External Clock Input
DDS	3	Timer Clock (80 MHz)
DDS	4	SyncLink0
DDS	5	SyncLink1
DDS	6	ClockLink In
DDS	7	DDS Output

TABLE 29. DDS Pin Output Sources

Digital I/O

The SBC6711 includes 32-bits of software programmable digital I/O for use in controlling digital instruments or acquiring digital inputs. The digital I/O port controls are mapped into memory space using two addresses: one to read/write the digital I/O data as a single 32-bit word, one for direction control and for each byte of the port and controlling the source of the clock edge used to latch input data into the digital I/O port register. The following table lists the addresses and their functions.

Function	Address
Digital I/O Data Register	0x90324000
Digital I/O Direction Control and input latch clock control	0x90300000

TABLE 30. Digital I/O Control Registers

The direction control register provides for software control of the drive direction of the port and the source of the input latch clock. The least significant four bits of the register control the four bytes available on the I/O port. Bit D0 sets the direction for the least significant eight bits if the port (port bits 0-7), D1 the next least significant bits (8-15), D2 the next least significant (16-23) and D3 the most significant (24-31). Each byte is individually controllable by writing a zero (to select output) or a one (to select input) to the respective bit in the direction control register. For example, if the value 0xC were written to the direction control register, bits 0-15 would act as inputs while bits 16-31 would act as outputs. All bytes default to input mode upon the board power-up or on reset.

Bit	Function
0	Direction Byte 0 0= Output (default), 1 = Input
1	Direction Byte 1 0= Output (default), 1 = Input
2	Direction Byte 2 0= Output (default), 1 = Input

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Bit	Function
3	Direction Byte 3 0= Output (default), 1 = Input
4	Input Latch Clock 0 = Latched on Digital Read 1 = Latched on External Digital Read Clock

TABLE 31. Digital IO Port Control Register Bit Definitions

The data register allows software to directly read data from port pins programmed for input, or write data to pins programmed for output. Read operations performed from the data register on port bytes programmed for output will return the current value of the digital I/O latch (i.e. the last value written to that portion of the port). For example, suppose that the direction controls were programmed to 0xC and the data register written with the data word 0x12340000. Since the most significant 16-bits are setup as outputs, those pins on the port connector would assume the value 0x1234. A subsequent read of the port would yield the value 0x1234xxxx, where xxxx is the value of the signals present on the digital I/O connector.

The input latch clock bit allows the user to select from either software read clocking or external hardware clocking. Writing a zero to the register selects software clocking, while writing a one selects external hardware clocking. If software clocking is selected, then the port latches programmed for input will clock in the digital data present on the external pins at the beginning of a read cycle executed on the port data register (30-50 ns before the data is returned to the processor, depending on processor clock speed). If external clocking is selected, then the port will latch data on the falling edge of the TTL signal EXT_DIG_RD_CLK* at the digital I/O connector. The data will be held for the processor to read until the next low-going edge of the EXT_DIG_RD_CLK* signal. In the external hardware clocking mode, read operations by the processor do not affect the contents of the digital I/O latch. The latched data may be reread as many times as is required, and only another EXT_DIG_RD_CLK* pulse will cause new data to be latched into the port.

Digital I/O Timing

The following diagram gives timing information for the digital I/O port when used in external readback clock mode (see above for details). This data is derived from device specifications and is not factory tested.

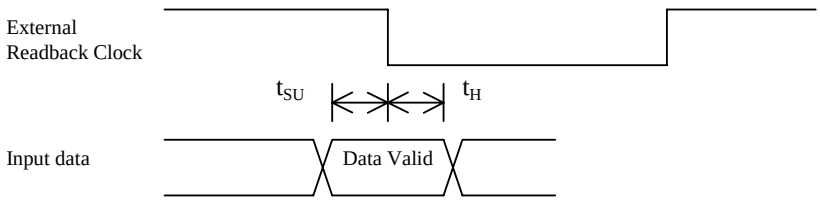


FIGURE 38. Digital I/O Port Timing

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameter	min (ns)
t_{SU}	0
t_H	10

TABLE 32. Digital I/O Port Timing Parameters

Digital IO Electrical Characteristics

The digital IO pins are TTL compatible pins driven by 5V logic. The logic chips used at 74FCT16952 bi-directional registered transceivers. The pins are 5V-tolerant. Output drive is +64 mA, with sink capability of -32 mA. This allows the SBC6711 to drive LEDs, opto-couplers and relays directly under normal circumstances. Be aware that the 74FCT16952 chips are not heat sunk on the SBC6711 as delivered from the factory, and sourcing or sinking large amounts of current may result in power dissipation requiring heat sinking.

Caution! Forced air cooling or heat sinking may be required when using multiple digital IO pins to control heavy loads.

Interrupts

The ‘C6711 processor implements five interrupt input pins which allow external hardware events to directly trigger software or DMA activity. Processor interrupt inputs are supported on the SBC6711 through a set of control registers and multiplexers that allows application software to dynamically select the source of the signal which will drive each particular interrupt input as well as the polarity and edge or level sensitivity. Additionally, the interrupt controller in the support logic allows interrupt sharing by peripherals on the SBC6711.

The following table shows the addresses of the control registers for each processor interrupt input. A value written to the appropriate control register causes the interrupt mux to select the interrupt source given in the next table (see below).

Functional	Address
NMI Interrupt Input Select	0x90307000
External Interrupt Input 4 Select	0x90303000
External Interrupt Input 5 Select	0x90304000
External Interrupt Input 6 Select	0x90305000
External Interrupt Input 7 Select	0x90306000

TABLE 33. External Interrupt Input Control Register Addresses

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Bit	Interrupt Source
0	External Interrupt Input to J2 (SMB connector)
1	Module 0 Interrupt 0
2	Module 0 Interrupt 1
3	Module 1 Interrupt 0
4	Module 1 Interrupt 1
5	External int on digital IO connector
6	FIFOPort Receive
7	FIFOPort Transmit
8	USB interrupt
9-12	Reserved
13	DDS
14	SyncLink 0
15	SyncLink 1
16	SyncLink 2
17	ClockLink
18	Uart
19..30	Not used
31	Acknowledged mode 0 = non-ack'd mode (default) 1 = ack'd mode

TABLE 34. External Interrupt Control Register Bit Definitions

The selection bits in each interrupt control register are identical on all interrupts. Set the corresponding bit true for each interrupt source that is used by that DSP interrupt. For example, if module 0 interrupt 0 is the interrupt source for the DSP external interrupt 4, then a 0x4 must be written to the DSP interrupt 4 select register at 0x90303000. Multiple interrupts sources may be enabled on shared interrupts. Dedicated interrupts should enable only one interrupt source. At reset, no interrupt sources are enabled.

Bit 31 is the mode selection for acknowledged interrupts mode described in the section on shared interrupts.

Conditioning for Interrupt Input Signals

Each interrupt source has polarity and edge/level selection so that nearly any interrupt source can be used by the DSP interrupts. The interrupt polarity is controlled by the interrupt source polarity register at 0x9031D000 with each interrupt source, as numbered in the interrupt selection register above, controlled by a bit in the register. Edge/level selection allows the DSP to use either the interrupt source edge or level as the trigger condition for the interrupt. Polarity selection is normally used to control the edge used, rising or falling, or the level, high or low, for an interrupt. The following table gives the correct settings for the possible interrupt conditions so that the logic interacts properly with the DSP. The SBC6711 support logic and software requires that the DSP is always programmed for rising edge interrupts.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Interrupt Type	Polarity	Edge/Level
Rising Edge	1	0
Falling Edge	0	0
High Level (1 = interrupt)	1	1
Low Level (0 = interrupt)	0	1

Interrupts must remain active at least 15 ns after changing states, in either edge or level mode.

Shared/Dedicated Interrupts

The SBC6711 has two interrupt modes that may be used with any processor interrupt : shared and dedicated mode. Since the DSP has only five interrupts externally, many applications need a method for sharing interrupts to support all the peripheral devices on the SBC6711 efficiently. Shared mode has been developed so that multiple interrupt sources may share a single interrupt to the DSP.

In dedicated mode, each interrupt to the processor is steered directly from the selection matrix, through the edge/level and polarity conditioning directly to the processor. This allows the interrupt source to directly connect to the DSP. Dedicated interrupts are not shared amongst interrupt sources and should be used for the devices requiring the highest rates of interrupt servicing. These devices might be, for example, FIFO level interrupts from an Omnibus module or FIFOPort interrupts. The dedicated mode does not have the burden of acknowledging the interrupts consumed so this mode is faster at interrupt servicing than the shared interrupts.

Shared interrupts allow multiple devices to share an interrupt to the processor. When the DSP receives a shared interrupt, it must read the interrupt status register associated with that interrupt to determine the interrupt source(s) requiring service. The DSP interrupt handling in this case should be capable of handling all the devices sharing this interrupt either alone or simultaneously to support the interrupt sharing. Upon completing the interrupt servicing, it is required that the DSP acknowledge the interrupt sources that were serviced. This prevents interrupts from being lost in the event that another interrupt source requires service in the meantime. The interrupt status/acknowledge registers are located at the address in the following table. Writing a ‘1’ to any of the bits indicates that the interrupt has been serviced.

The bits have the same order as the interrupt source selection bits previously shown above.

Interrupt	Address
DSP int 4	0x9031F000
DSP int 5	0x90320000
DSP int 6	0x90321000
DSP int 7	0x90322000
NMI	0x90323000

TABLE 35. External Interrupt Status and Acknowledge Register Addresses

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

For example, if the application requires the output from external interrupt input two to drive processor interrupt input five, the value two should be written to memory location 0x90320000. All interrupt control registers default to 0 on power-up or board reset. Note that the processor interrupt signals generated by the logic are active high (rising edge trigger), and the ‘C6711 interrupt polarity control register must be programmed to the value 0x0 to correctly receive interrupts.

External User Interrupt Usage

The SBC6711 supports two external user interrupts other than the Omnibus modules. One of these external interrupts is available on J2, an SMB connector. This input may be TTL or 50 ohm terminated as selected by jumper JP14 (jumper on = 50 ohm input). The second external input is available on the digital IO connector JP3 on pin 38. This input on JP3 is TTL compatible only and is referred to as the digital IO connector external interrupt in this document.

Other interrupts are also available in the Omnibus connectors if required, but are normally consumed by the Omnibus modules.

Watchdog Timer

The SBC6711 has a watchdog timer to prevent program runaway or unplanned behavior. When enabled, this timer will reset the card if it is not written to at address 0x90313000 within at least 27 mS, and each and every 27 mS interval thereafter. When the processor writes to this address with any value, the watchdog timer is reset and will not expire for another 27 mS. This feature may be disabled during software development by setting bit 4 of the control register to ‘0’ so that it does not interfere with debugging.

Multi-Card Timing Synchronization

The SBC6711 has several features to support multi-card synchronization and clock sharing. Synclink is a simple TTL bus that allows the SBC6711 to send or receive clock and trigger signals to other cards, while ClockLink is an LVDS input and output pair allowing the system to share high speed clock and trigger signals. A variety of DSP and data acquisition products from Innovative feature SyncLink and ClockLink to allow the construction of multi-card synchronized systems for channel expansion and common triggering.

The SyncLink bus has one master who is the originator of all clock and trigger signals, referred to as the master, while all other cards are referred to as slave devices. The slave cards can only receive clock/trigger signals from the master. The following table below shows the signals that may be transmitted over the SyncLink bus when it is the master. The SyncLink signal selection register resides at 0x90312000.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Bit	Signal Output as SyncLink 0
0	OMNIBUS timer 0
1	OMNIBUS timer 1
2	DDS Clock
3	Module 0 int 0
4	DIO Connector int
5	External Interrupt Signal (J2)
6..7	not used

TABLE 36. SyncLink 0 Signal Selection Register Bit Definitions

Bit	Signal Output as SyncLink 1
8	OMNIBUS timer 0
9	OMNIBUS timer 1
10	DDS Clock
11	Module 1 Int 0
12	DIO Connector Int
13	External Interrupt Signal (J2)
14..15	not used

TABLE 37. SyncLink 1 Signal Selection Register Bit definitions

Bit	Signal Output as SyncLink 2
16	Module 0 Trigger
17	Module 1 Trigger
18	DSP Trigger
19	External Int (J2)

TABLE 38. SyncLink 2 Signal Selection Register Bit definitions

The SyncLink master bit is controlled by the DSP at address 0x90311000. Write a '1' to bit 0 of this register to allow the SBC6711 to act as the SyncLink master. Only one master should be enabled between all the cards on the SyncLink bus to prevent conflicts.

The SyncLink signals are recommend for use below 1 MHz to a maximum of 8 TTL loads. Termination may be required depending on the cable and impedance of the loads to prevent signal ringing. More than eight loads may require additional buffering.

ClockLink is intended for higher speed signals that are usually card to card. The ClockLink allows the SBC6711 to share clocks up to 80 MHz over short distances, or farther for slower signals. Flat ribbon cable performs well, but twisted pair cable is recommended for the best signal integrity. Note that the transmit/receive signal wire pairs must be crossed to connect two cards together.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ClockLink may source a variety of signals as shown in the following table.

Bit	Signal Output as SyncLink 0
0	DSP timer 0
1	DSP timer 1
2	External Interrupt Signal (J2)
3	Timer Clock (80 MHz)
4	DDS Output

TABLE 39. ClockLink Signal Selection Register Bit Definitions

JTAG Test Bus

The SBC6711 implements a JTAG 1149.1-compatible scan path loop through the onboard ‘C6711, with connector compatible with the specification provided in the *TMS320C6000 User’s Guide*. When connecting a JTAG controller card cable (from an Innovative Integration Code Hammer debugger card, Texas Instruments XDS-510, or other vendor’s JTAG hardware), connector JP17 is used.

Miscellaneous Control

The miscellaneous control register implements several write-only control bits which control various features on the card. The following table and bit description give the definition of the register.

Bit	Function
0	Reset FIFOPort ‘1’ = reset (default). Resets the FIFOPort mechanism, including the level register for the other card.
1	Reset DDS ‘1’ = reset (default). This bit holds the DDS in reset when true. Set this bit to ‘0’ when the DDS is to be used.
2	Reset UART ‘1’=reset (default). This bit holds the 16450 UART in reset when true. Set this bit to ‘0’ when the UART is to be used.
3	DSP trigger ‘0’ = default. This bit is routed to the synclink and clocklink and may be used to a allow the DSP to provide a gate/trigger to the other cards.
4	Watchdog timer enable 0= disabled (default). This bit, when ‘1’, allows the SBC6711 to be reset by the watchdog timer.
5	Booted bit 0 = not booted (default). This bit flipped after boot to enable fast timing
6	USB reset enable 1 = enabled (default). This bit allows the USB bus reset to reset the SBC6711
7	DSP reset enable. 1= Asynchronously assert DSP reset. HW coldstart cycle will follow

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TABLE 40. Miscellaneous Control Register Definition

Power Requirements

The SBC6711 processor board requires an off-board power supply capable of delivering three power supplies: +5V DC, +15V DC, and –15V DC. Power is delivered to the card via the main power connector. Pinouts for the power connector are given in the appendices. Innovative Integration supplies linear power supplies suitable for use with AC mains powered installations (part number 32001 for US 120-volt input: call for information concerning European and Asian power supplies).

Caution: Double-check all power wiring before attempting to start the board with custom power supplies or connections. Incorrect power supply connections will severely damage the SBC6711 board. The board's hardware warranty is void if specified power connection requirements are not met by the end user.

Updating the SBC6711 logic

The SBC6711 logic may be updated in the field under normal circumstances without removing the card from the system. The FPGA logic is held in a reprogrammable EEPROM that can be written by the DSP so long as the DSP card is functioning normally prior to re-programming. In this case, an application provided with the Zuma Toolset allows the user to load new logic to the card over the USB link.

A few precautions to note about the logic burning must be observed to insure that the card is not left in an unrecoverable state. First, the power should remain on during the entire logic burning process as partial rewrites cause unknown behavior. Second, the card must communicate normally prior to burning the logic or a faulty logic image may be downloaded. Third, use only bit files provided by Innovative for the SBC6711 during the update process.

Warning: Should the card ever fail during the update, or if it is not functioning normally, contact technical support at Innovative. Do not attempt logic updates under this condition or the card could be rendered useless.

Updating the logic is only supported using the logic update program from Innovative included in the Zuma toolset.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CHAPTER 11

Appendices

Connector pinouts

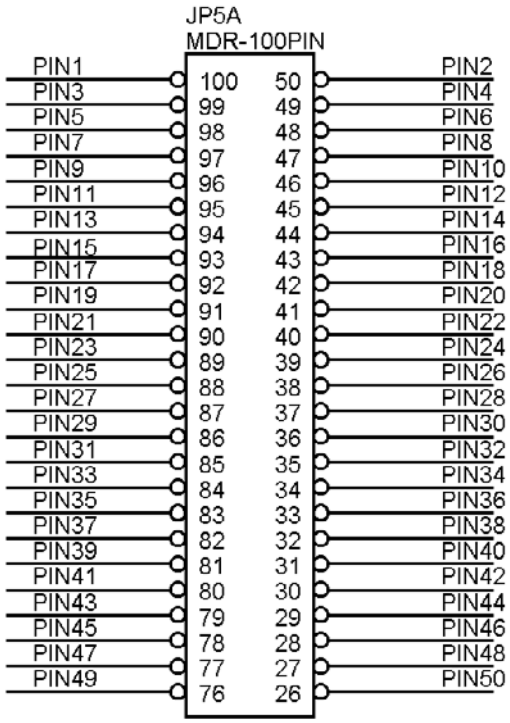
JP4, JP8, JP5 - OMNIBUS I/O Connectors

Connector types:	JP4, JP8: AMP .05 Subminiature D male, AMP 173280-3 JP5 : MDR connector, 3M N102A0-52E2VC
Number of pins:	JP11, JP16: 50 JP5: 100
Mating connector:	JP11, JP16: AMP 173279-3 JP5: 3M 101A0-4CZ3JL

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The following table shows the interconnections between the JP4 (OMNIBUS slot 0) and JP5 (OMNIBUS IO connector).

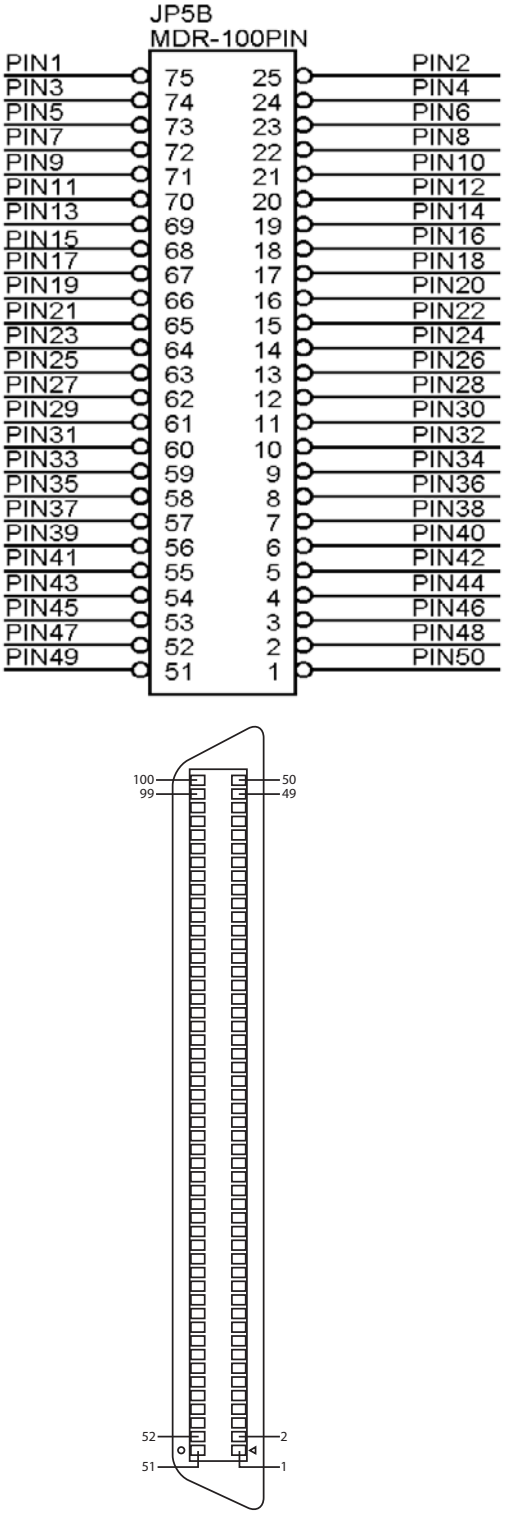
JP4, Module 0 Pin	JP5 Pin Numbers
1	100
2	50
3	99
4	49
5	98
6	48
7	97
8	47
9	96
10	46
11	95
12	45
13	94
14	44
15	93
16	43
17	92
18	42
19	91
20	41
21	90
22	40
23	89
24	39
25	88
26	38
27	87
28	37
29	86
30	36
31	85
32	35
33	84
34	34
35	83
36	33
37	82
38	32
39	81
40	31
41	80
42	30
43	79
44	29
45	78
46	28
47	77
48	27
49	76
50	26



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The following table shows the interconnections between the JP8 (OMNIBUS slot1) and JP5 (OMNIBUS IO connector).

JP8, Module 1 Pin	JP5 Pin Numbers
1	75
2	25
3	74
4	24
5	73
6	23
7	72
8	22
9	71
10	21
11	70
12	20
13	69
14	19
15	68
16	18
17	67
18	17
19	66
20	16
21	65
22	15
23	64
24	14
25	63
26	13
27	62
28	12
29	61
30	11
31	60
32	10
33	59
34	9
35	58
36	8
37	57
38	7
39	56
40	6
41	55
42	5
43	54
44	4
45	53
46	3
47	52
48	2
49	51
50	1



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

JP6, 7, 9, 10 – OMNIBUS Bus Connectors

Connector types: AMP .05 Subminiature D male

Number of pins: 50

Mating connector: AMP 173279-3

The following table gives the pin numbers and functions for the JP6 (OMNIBUS slot 0) and JP7 (OMNIBUS slot 0) connectors. The functions for JP9 are identical to those of JP6, except where noted.

Pin Number	JP6 Function	JP9 Function	Direction (from SBC6711)
1, 19	Digital +5V		O, power
2, 20	Digital ground		O, power
3-18	Data bus 0-15		I/O
21, 43, 40, 45, 39, 26, 27	Address bus 2-8		O
28	Reset (active low)		O
29	External interrupt 0	External interrupt 2	I
30	Bus ready (active low)		I (open-collector)
31	Processor ECLK/2 (37.5 MHz)		O
32	DSP timer channel 0		O
33	R/W*		O
34	DDS timebase		O
35-38	IOMOD0-3 decoded selects (active low)	IOMOD4-7 decoded selects (active low)	O
25	Analog –15V (OMNIBUS –12V)		O, power
23	Analog +15V (OMNIBUS +12V)		O, power
41,42	Analog ground		O, power
22,24	Analog -15V		O, power
44,46	Analog +15V		O, power
47,49	Analog +5V		O, power
48,50	Analog -5V		O, power

TABLE 41. OMNIBUS Bus Connectors

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The following table gives the pin numbers and functions for the JP7 (OMNIBUS slot 0) and JP10 (OMNIBUS slot 1) connectors. The functions for JP10 are identical to those of JP7, except where noted.

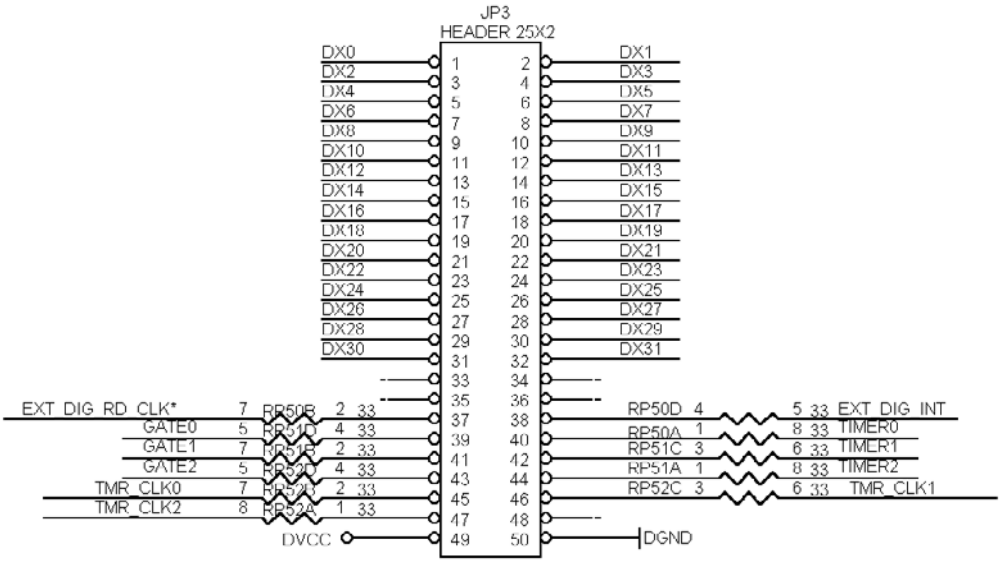
Pin Number	JP7 Function	JP10 Function	Direction (from SBC6711)
1, 3-6	Address bus 9-13		O
2, 19, 20, 49, 50	Digital ground		O, power
7-17	Reserved	Reserved	NA
18	Module trigger out		I
21	Processor timer channel 1		O
22	Module trigger in	External trigger 1	O
23,25	Analog +15V (OMNIBUS +12V)		O, power
24	CLKS0	CLKS1	I
26	CLKR0	CLKR1	I/O
27	FSR0	FSR1	I/O
28	CLKX0	CLKX1	I/O
29	External interrupt 1	External interrupt 3	I
30	DR0	DR1	I
31	FSX0	FSX1	I/O
32	DX0	DX1	O
33-48	Data bus 16-31		I/O

TABLE 42. I/O Module Bus Connectors

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

JP3 – Digital I/O Connector

Connector type:	0.1” double row shrouded header, center bump polarized, Thomas and Betts 609-5027
Number of pins:	50
Mating connector:	AMP 1-746285-0



The following table gives the pin numbers and functions for the JP3 connector.

Pin Number	JP3 Function	Direction (from SBC6711)
1-32	Digital I/O bit 0..31	I/O
33-36	Not used	-
37	External Digital Readback Clock(latch falling edge)	I
38	External Interrupt Input 0	I
39	Timer 0 gate (active high)	I
40	Timer 0 output	O
41	Timer 1 gate (active high)	I
42	Timer 1 output	O
43	Timer 2 gate (active high)	I
44	Timer 2 output	O
45	Timer 0 input clock	I
46	Timer 1 input clock	I
47	Timer 2 input clock	I
48	not used	-
49	DVCC (digital +5 V)	Power
50	DGND (digital ground)	Power

TABLE 43. Digital I/O Connector

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

JP2 – FIFOPort Connector

Connector type: 2mm header

Number of pins: 54

Mating connector: Samtec SQW style

The following table gives the pin numbers and functions for the JP2 connector.

Pin Number	JP23 Function	Direction (from SBC6711)
1	Digital 5V	Power
2, 44	Ground	Power
3-18	Input Data Bits 15-0	I
19	Reserved	NA
20	Input Strobe	I
21-24	not used	-
25-40	Output Data Bits 0-15	O
41	not used	-
42	Output Strobe	O
43	Digital 3.3V	Power
45	Half-full Flag Out	O
46	Almost-full Flag Out	O
47	Almost-full Level Control In	I
48	not used	-
49	Almost-full Level Control Out	O
50-51	not used	-
52	Almost-full Flag In	I
53-54	Not used	-

TABLE 44. FIFOPort Connector

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

JP13 - SyncLink/ClkLink

The SyncLink connector allows SBC6711 to synchronize to external hardware or other Innovative cards.

Connector type: 0.1” double-row shrouded header

Number of pins: 10

Mating connector: AMP 111810-1

The following table gives the pin numbers and functions for the JP13 connector.

Pin Number	JP13 Function	Direction (from SBC6711)
1	Clocklink Out +	O
2	Clocklink Out -	O
3	Clocklink In +	I
4	Clocklink In -	I
5	Synclink bus pin 2	I/O
6	Digital ground	Power
7	Synclink bus pin 1	I/O
8	Digital ground	Power
9	Synclink bus pin 0	I/O
10	Digital ground	Power

TABLE 45. SyncLink Connector

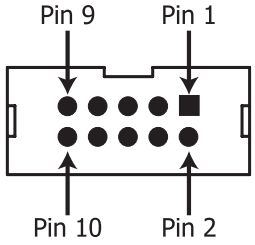


FIGURE 39. SyncLink Connector Pin Orientation

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

JP18, JP19 – Processor Serial Port Connectors

Connector type:	2 mm double row header
Number of pins:	10
Mating connector:	Samtec SQT style (for board-board applications)

The following table gives the pin numbers and functions for the JP6 and JP7 connectors. Pin functions of JP6 are identical to those of JP7 except where noted.

Pin Number	JP6 Function	JP7 Function	Direction (from SBC6711)
1	CLKS0	CLKS1	I
2	FSR0	FSR1	I/O
3	CLKR0	CLKR1	I/O
4	FSX0	FSX1	I/O
5	CLKX0	CLKX1	I/O
6	Digital 3.3V		Power
7	DR0	DR0	I
8	Digital 5V		Power
9	DX0	DX0	O
10	Digital Ground		Power

TABLE 46. Processor Serial Port Connector

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

JP17 – JTAG Debugger Connector

Connector type:	Shrouded header, pin 6 removed for key
Number of pins:	14
Mating connector:	AMP 746285-2

The following table gives the pin numbers and functions for the JP8 connector.

Pin Number	JP8 Function	Direction (from SBC6711)
1	TMS	I
2	TRST*	I
3	TDI	I
5	Digital +3V	Power
7	TDO	O
9,11	TCK	I
13	EMU0	I/O
14	EMU1	I/O
4, 6, 8, 10, 12	Digital ground	Power

TABLE 47. JTAG Debugger Connector

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

JP16 – Power Input Connector

Connector type:	6 pin locking power connector (Molex 43045-0602)
Number of pins:	6
Mating connector:	Molex 43025-0600 and contacts

The following table gives the pin numbers and functions for the JP22 connector.

Pin Number	JP22 Function	Direction (from SBC6711)
1	Digital +5V	Power
2,4	Digital ground	Power
3	Analog +15V	Power
5	Analog –15V	Power
6	MOD_PWR power control	O

TABLE 48. Power Input Connector

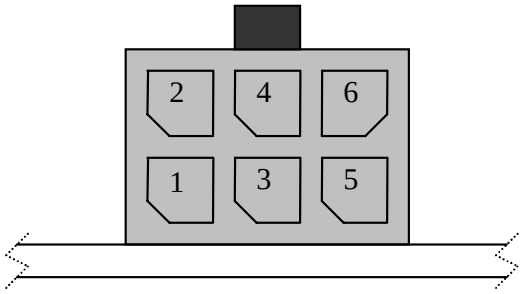


FIGURE 40. Power Connector Pin Positions (side view, from front of connector, showing connector keying and locking tab along with printed circuit board position)

Note: Matting connector may be numbered differently.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

JP22 – Asynchronous Serial Port Connector

Connector type:	Shrouded header
Number of pins:	10
Mating connector:	AMP 746285-1

The following table gives the pin numbers and functions for the JP22 connector.

Pin Number	Function	Direction (from SBC6711)
1, 2, 8, 10	Reserved	NA
3	RS232 Transmit Data	O
4	RS232 CTS	I
5	RS232 Receive Data	I
6	RS232 RTS	O
7	RS232 DTR	I
9	Digital ground	Power

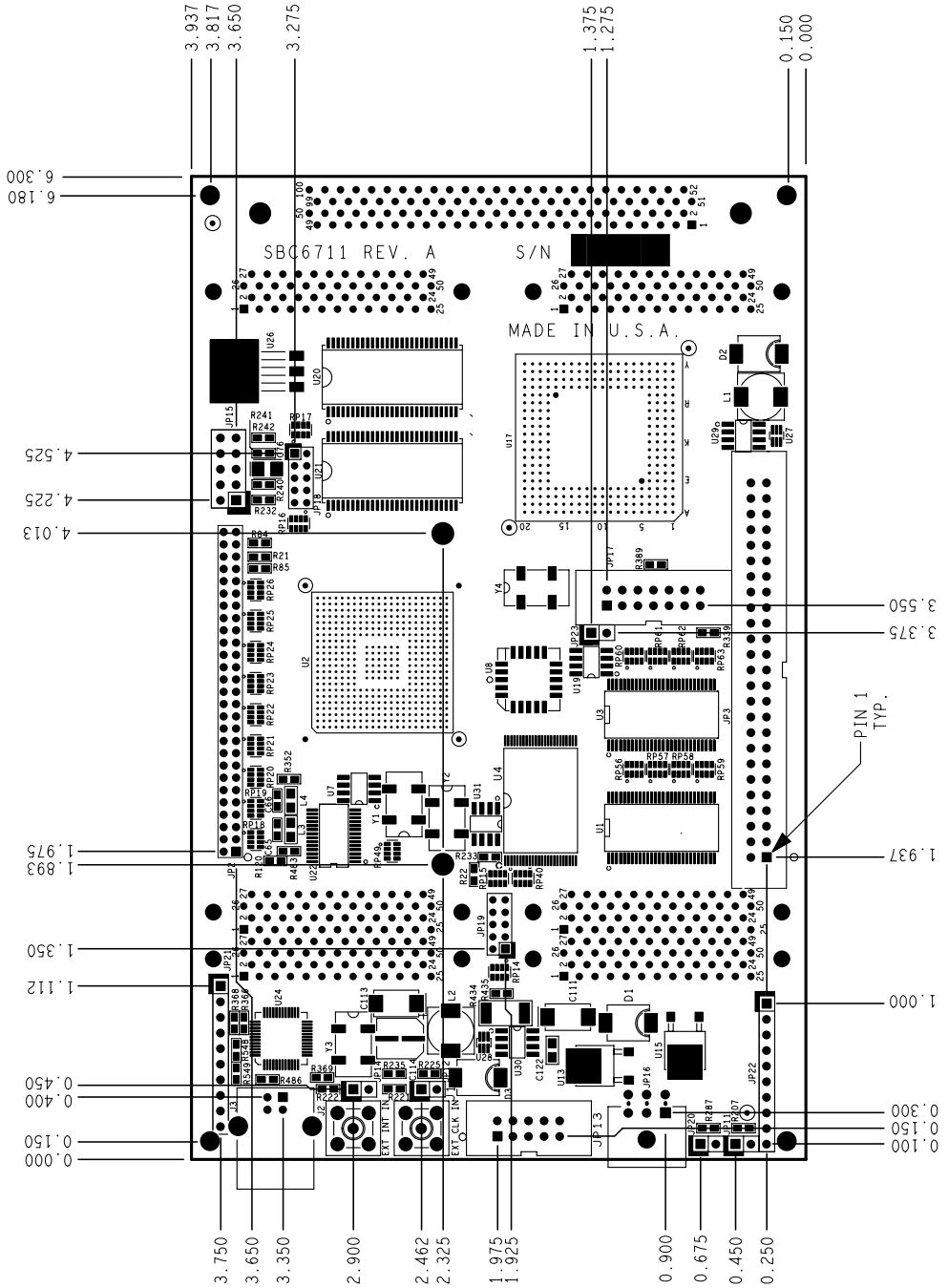
TABLE 49. Asynchronous Serial Port Connector

Board Layout

A dimentional schematic and the board layout is displayed on the following page. Please review these drawings to familiarize yourself with the circuit board's configuration and connections.

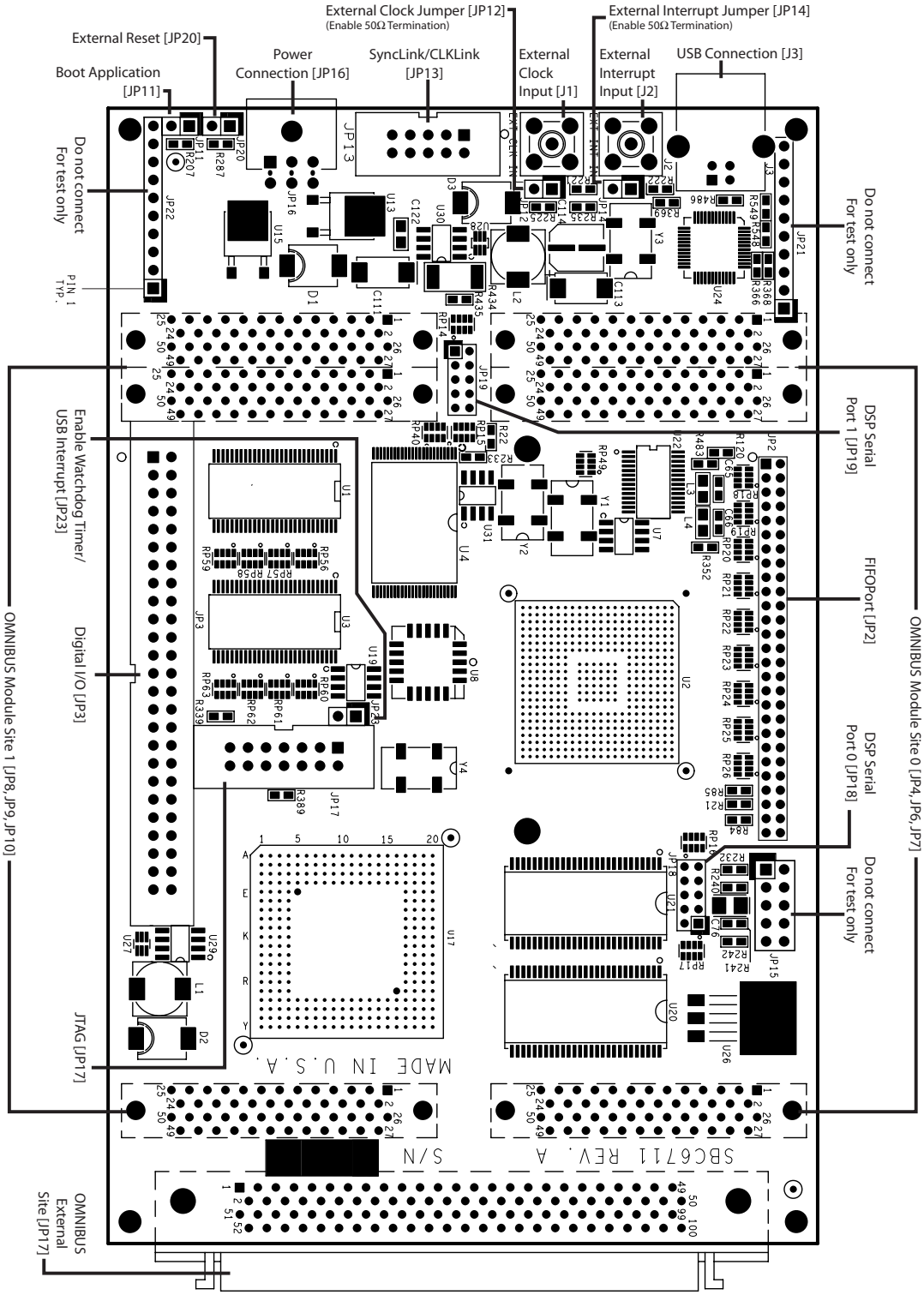
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Mechanical Drawing (Board Revision A & B)



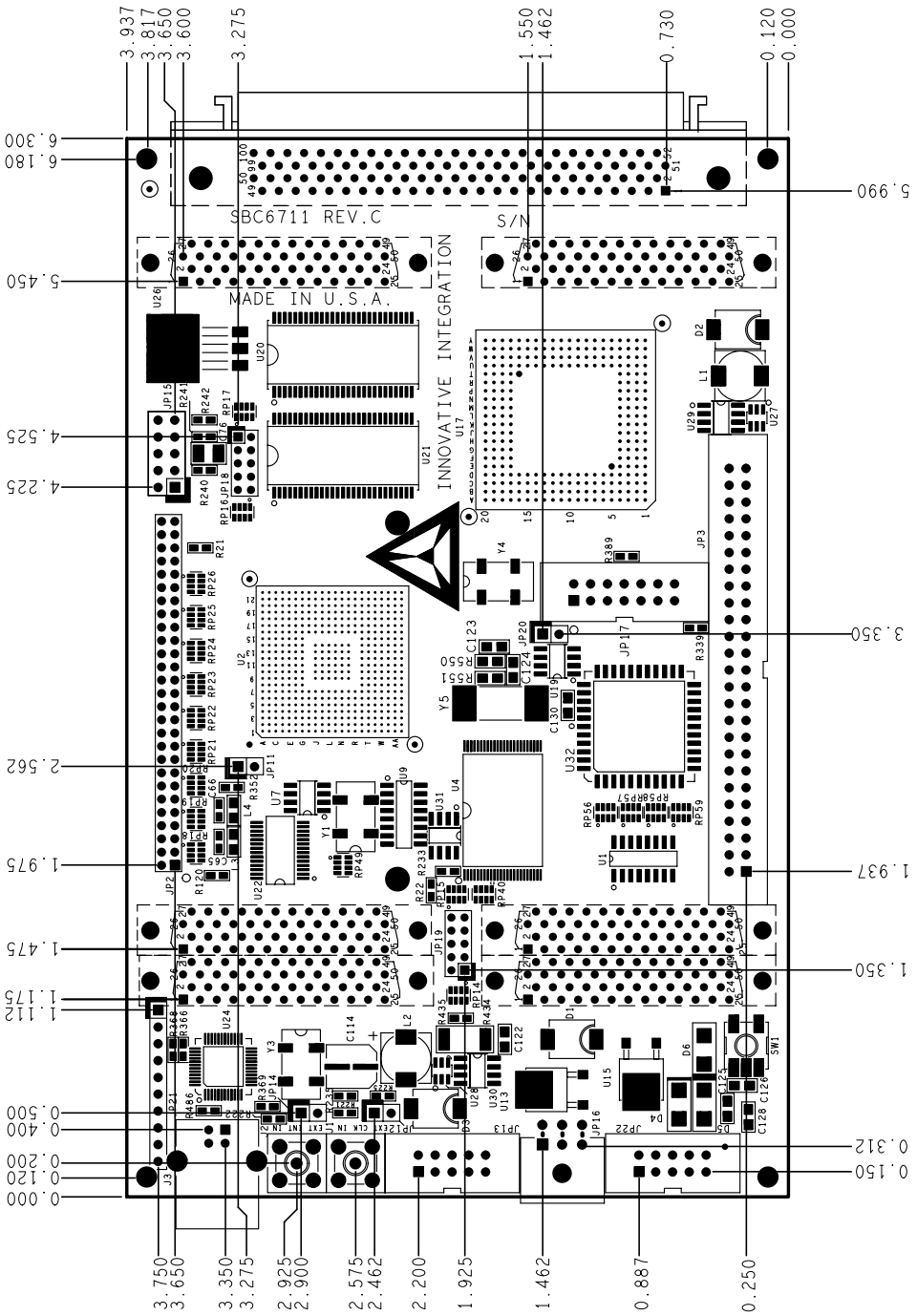
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Connector Location (Board Revision A & B)



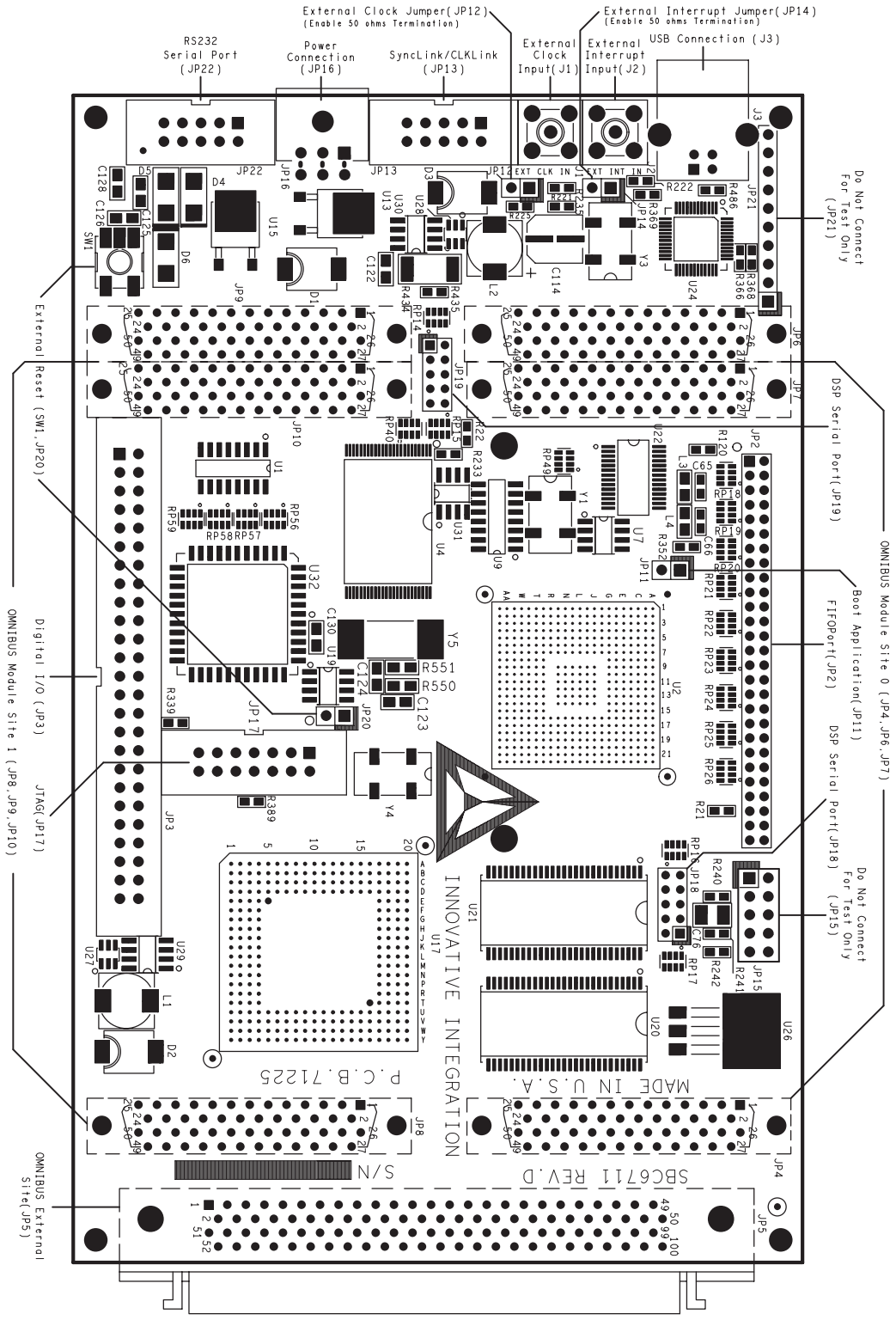
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Mechanical Drawing (Board Revision C & D)



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Connector Location (Board Revision C & D)



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP