TMS320F240 DSP Controllers Evaluation Module Technical Reference

Literature Number: SPRU248B
July 1999

IMPORTANT NOTICE

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

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

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

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

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

Read This First

About This Manual

This manual describes the hardware and software installation and general operation of the TMS320C24x evaluation module. In this document, the TMS320C24x DSP controllers evaluation module is also referred to as the ’C24x EVM, the evaluation board included in the ’C24x EVM is the ’C24x evaluation board, and the TMS320F240 device on the evaluation board is also referred to as the ’F240 device. The TMS320C24x is a subset of the TMS320C2xx family of DSPs.

Notational Conventions

This document uses the following conventions:

- Program listings and program examples are shown in a `special typeface`.

  Here is a segment of a program listing:

```
OUTPUT  LDP     #6          ;select data page 6
        BLDD    #300, 20h   ;move data at address 300h to 320h
        RET
```

- In syntax descriptions, **bold** portions of a syntax must be entered as shown; *italic* portions of a syntax identify information that you specify. Here is an example of an instruction syntax:

  BLDD *source, destination*

- A lowercase h after a number indicates hexadecimal notation.

  For example:

xxxx	xxxh
0000	A/11h

Information About Cautions

This book contains cautions.

> **This is an example of a caution statement.**
>
> **A caution statement describes a situation that could potentially damage your software or equipment.**

The information in a caution is provided for your protection. Please read each caution carefully.

Related Documentation From Texas Instruments

This subsection describes related TI™ documents that can be ordered by calling the Texas Instruments Literature Response Center at (800) 477–8924. When ordering, please identify the book by its title and literature number.

TMS320C24x DSP Controllers Reference Set, Volume 1: CPU, System, and Instruction Set (literature number SPRU160) describes the TMS320C24x 16-bit, fixed-point, digital signal processor controller. Covered are its architecture, internal register structure, data and program addressing, and instruction set. Also includes instruction set comparisons and design considerations for using the XDS510 emulator.

TMS320C24x DSP Controllers Reference Set Volume 2: Peripheral Library and Specific Devices (literature number SPRU161) describes the peripherals available on the TMS320C24x digital signal processor controllers and their operation. Also described are specific device configurations of the ’C24x family.

TMS320C240, TMS320F240 DSP Controllers (literature number SPRS042) data sheet contains the electrical and timing specifications for these devices, as well as signal descriptions and pinouts for all of the available packages.

TMS320C1x/C2x/C2xx/C5x Code Generation Tools Getting Started Guide (literature number SPRU121) describes how to install the TMS320C1x, TMS320C2x, TMS320C2xx, and TMS320C5x assembly language tools and the C compiler for the ’C1x, ’C2x, ’C2xx, and ’C5x devices. The installation for MS-DOS™, OS/2™, SunOS™, and Solaris™ systems is covered.

TMS320C1x/C2x/C2xx/C5x Assembly Language Tools User's Guide (literature number SPRU018) describes the assembly language tools (assembler, linker, and other tools used to develop assembly language code), assembler directives, macros, common object file format, and symbolic debugging directives for the 'C1x, 'C2x, 'C2xx, and 'C5x generations of devices.

TMS320C2x/C2xx/C5x Optimizing C Compiler User's Guide (literature number SPRU024) describes the 'C2x/C2xx/C5x C compiler. This C compiler accepts ANSI standard C source code and produces TMS320 assembly language source code for the 'C2x, 'C2xx, and 'C5x generations of devices.

TMS320C2xx C Source Debugger User's Guide (literature number SPRU151) tells you how to invoke the 'C2xx emulator and simulator versions of the C source debugger interface. This book discusses various aspects of the debugger interface, including window management, command entry, code execution, data management, and breakpoints. It also includes a tutorial that introduces basic debugger functionality.

TMS320C2xx Simulator Getting Started (literature number SPRU137) describes how to install the TMS320C2xx simulator and the C source debugger for the 'C2xx. The installation for MS-DOS™, PC-DOS™, SunOS™, Solaris™, and HP-UX™ systems is covered.

TMS320C2xx Emulator Getting Started Guide (literature number SPRU209) tells you how to install the Windows™ 3.1 and Windows™ 95 versions of the 'C2xx emulator and C source debugger interface.

XDS51x Emulator Installation Guide (literature number SPNU070) describes the installation of the XDS510™, XDS510PP™, and XDS510WS™ emulator controllers. The installation of the XDS511™ emulator is also described.

JTAG/MPSD Emulation Technical Reference (literature number SPDU079) provides the design requirements of the XDS510™ emulator controller, discusses JTAG designs (based on the IEEE 1149.1 standard), and modular port scan device (MPSD) designs.

TMS320 DSP Development Support Reference Guide (literature number SPRU011) describes the TMS320 family of digital signal processors and the tools that support these devices. Included are code-generation tools (compilers, assemblers, linkers, etc.) and system integration and debug tools (simulators, emulators, evaluation modules, etc.). Also covered are available documentation, seminars, the university program, and factory repair and exchange.

Digital Signal Processing Applications with the TMS320 Family, Volumes 1, 2, and 3 (literature numbers SPRA012, SPRA016, SPRA017) Volumes 1 and 2 cover applications using the ’C10 and ’C20 families of fixed-point processors. Volume 3 documents applications using both fixed-point processors, as well as the ’C30 floating-point processor.

TMS320 DSP Designer’s Notebook: Volume 1 (literature number SPRT125) presents solutions to common design problems using ’C2x, ’C3x, ’C4x, ’C5x, and other TI DSPs.

TMS320 Third-Party Support Reference Guide (literature number SPRU052) alphabetically lists over 100 third parties that provide various products that serve the family of TMS320 digital signal processors. A myriad of products and applications are offered—software and hardware development tools, speech recognition, image processing, noise cancellation, modems, etc.

TMS320C2xx Emulator Getting Started Guide (literature number SPRU209) tells you how to install the Windows™ 3.1 and Windows™ 95 versions of the ’C2xx emulator and C source debugger interface.

Microprocessor Development Systems Customer Support Guide (literature number SPDU08A) describes the registration, customer support services, service and warranty, and software license agreements. Precautions and safety considerations are also covered in this document.

Related Technical Articles

“A Greener World Through DSP Controllers”, Panos Papamichalis, *DSP & Multimedia Technology*, September 1994.

“Application Guide with DSP Leading-Edge Technology”, Y. Nishikori, M. Hattori, T. Fukuhara, R.Tanaka, M. Shimoda, I. Kudo, A.Yanagitani, H. Miyaguchi, et al., *Electronics Engineering*, November 1995.

“Approaching the No-Power Barrier”, Jon Bradley and Gene Frantz, *Electronic Design*, January 9, 1995.

“Digital Signal Processing Solutions Target Vertical Application Markets”, Ron Wages, *ECN*, September 1995.

“Digital Signal Processors Boost Drive Performance”, Tim Adcock, *Data Storage*, September/October 1995.

“DSPs Advance Low-Cost ‘Green’ Control”, Gregg Bennett, *DSP Series Part II*, *EE Times*, April 17, 1995.

"Easing JTAG Testing of Parallel-Processor Projects", Tony Coomes, Andy Fritsch, and Reid Tatge, *Asian Electronics Engineer*, Manila, Philippines, November 1995.

"Fixed or Floating? A Pointed Question in DSPs", Jim Larimer and Daniel Chen, *EDN*, August 3, 1995.

"Function-Focused Chipsets: Up the DSP Integration Core", Panos Papamichalis, *DSP & Multimedia Technology*, March/April 1995.

"Real-Time Control", Gregg Bennett, *Appliance Manufacturer*, May 1995.

"The Digital Signal Processor Development Environment", Greg Peake, *Embedded System Engineering*, United Kingdom, February 1995.

"The Growing Spectrum of Custom DSPs", Gene Frantz and Kun Lin, *DSP Series Part II*, *EE Times*, April 18, 1994.

"The Wide World of DSPs, " Jim Larimer, *Design News*, June 27, 1994.

"Third-Party Support Drives DSP Development for Uninitiated and Experts Alike", Panos Papamichalis, *DSP & Multimedia Technology*, December 1994/January 1995.

"Toward an Era of Economical DSPs", John Cooper, *DSP Series Part I*, *EE Times*, Jan. 23, 1995.

Trademarks

If You Need Assistance . . .

❑ World-Wide Web Sites

TI Online	http://www.ti.com
Semiconductor Product Information Center (PIC)	http://www.ti.com/sc/docs/pic/home.htm
DSP Solutions	http://www.ti.com/dsps
320 Hotline On-line™	http://www.ti.com/sc/docs/dsps/support.htm

❑ North America, South America, Central America

Product Information Center (PIC)	(972) 644-5580		
TI Literature Response Center U.S.A.	(800) 477-8924		
Software Registration/Upgrades	(214) 638-0333	Fax: (214) 638-7742	
U.S.A. Factory Repair/Hardware Upgrades	(281) 274-2285		
U.S. Technical Training Organization	(972) 644-5580		
DSP Hotline	(281) 274-2320	Fax: (281) 274-2324	Email: dsph@ti.com
DSP Modem BBS	(281) 274-2323		

DSP Internet BBS via anonymous ftp to ftp://ftp.ti.com/pub/tms320bbs

❑ Europe, Middle East, Africa

European Product Information Center (EPIC) Hotlines:

Multi-Language Support	+33 1 30 70 11 69	Fax: +33 1 30 70 10 32	Email: epic@ti.com
Deutsch +49 8161 80 33 11	or +33 1 30 70 11 68		
English	+33 1 30 70 11 65		
Francais	+33 1 30 70 11 64		
Italiano	+33 1 30 70 11 67		
EPIC Modem BBS	+33 1 30 70 11 99		
European Factory Repair	+33 4 93 22 25 40		
Europe Customer Training Helpline		Fax: +49 81 61 80 40 10	

❑ Asia-Pacific

Literature Response Center	+852 2 956 7288	Fax: +852 2 956 2200
Hong Kong DSP Hotline	+852 2 956 7268	Fax: +852 2 956 1002
Korea DSP Hotline	+82 2 551 2804	Fax: +82 2 551 2828
Korea DSP Modem BBS	+82 2 551 2914	
Singapore DSP Hotline		Fax: +65 390 7179
Taiwan DSP Hotline	+886 2 377 1450	Fax: +886 2 377 2718
Taiwan DSP Modem BBS	+886 2 376 2592	

Taiwan DSP Internet BBS via anonymous ftp to ftp://dsp.ee.tit.edu.tw/pub/TI/

❑ Japan

Product Information Center	+0120-81-0026 (in Japan)	Fax: +0120-81-0036 (in Japan)
	+03-3457-0972 or (INTL) 813-3457-0972	Fax: +03-3457-1259 or (INTL) 813-3457-1259
DSP Hotline	+03-3769-8735 or (INTL) 813-3769-8735	Fax: +03-3457-7071 or (INTL) 813-3457-7071
DSP BBS via Nifty-Serve	Type “Go TIASP”	

❑ Documentation

When making suggestions or reporting errors in documentation, please include the following information that is on the title page: the full title of the book, the publication date, and the literature number.

Mail: Texas Instruments Incorporated
Technical Documentation Services, MS 702
P.O. Box 1443
Houston, Texas 77251-1443

Email: comments@books.sc.ti.com

Note: When calling a Literature Response Center to order documentation, please specify the literature number of the book.

Contents

Figures

Tables

Examples

Chapter 1

Introduction

The TMS320C24x evaluation module (’C24x EVM) is a digital signal processor (DSP) development package that allows you to evaluate the ’C24x family of DSP controllers. The ’C24x EVM contains a standalone evaluation board that enables you to explore the architecture and operation of the ’C24x CPU and its peripherals. The board is built around the TMS320F240 (’F240) DSP controller, which is optimized for digital motor control and power conversion applications. The evaluation board connects to the parallel port of your PC through the XDS510PP™ emulator. The emulator and evaluation board, together with the ’C24x EVM C source debugger, allow for real-time verification of your ’C24x code. The four 34-pin connectors enable the use of expansion boards for a prototyping area or peripherals such as predrivers, power amplifiers, and user interfaces.

This chapter describes the ’C24x EVM, its features, design details, and external interfaces.

Topic **Page**

1.1 EVM Contents

There are six packages included in the ’C24x EVM. These packages contain the following hardware, software, and documentation:

- ’C24x evaluation board package
  - ’C24x evaluation board
  - ’C24x EVM software diskette (one diskette)
  - 5-pin DIN-to-2.1-mm power supply adapter cable
  - *TMS320C24x DSP Controllers Evaluation Module Technical Reference*
- ’C24x EVM documentation package
  - *TMS320C24x DSP Controllers Reference Set, Volume 1: CPU, System, and Instruction Set*
  - *TMS320C24x DSP Controllers Reference Set, Volume 2: Peripheral Library and Specific Devices*
- ’C24x EVM C source debugger package
  - TMS320C2xx C source debugger software (one diskette)
  - *TMS320C2xx C Source Debugger User’s Guide*
- ’C1x/C2x/C2xx/C5x assembler package
  - Assembly language tools software (two diskettes)
  - *TMS320C1x/C2x/C2xx/C5x Assembly Language Tools User’s Guide*
  - *TMS320C1x/C2x/C2xx/C5x Code Generation Tools Getting Started Guide*
  - *Microprocessor Development Systems Customer Support Guide*
- XDS510PP emulation controller package
  - XDS510PP emulator
  - PC™ parallel port cable
  - *XDS51x Emulator Installation Guide*
- XDS workstation power supply package
  - Power supply (Input: 100–250 V, 50–60 Hz; Output: 5 V dc, 3.3 A)
  - Power cord

1.2 Features of the TMS320C24x Evaluation Board

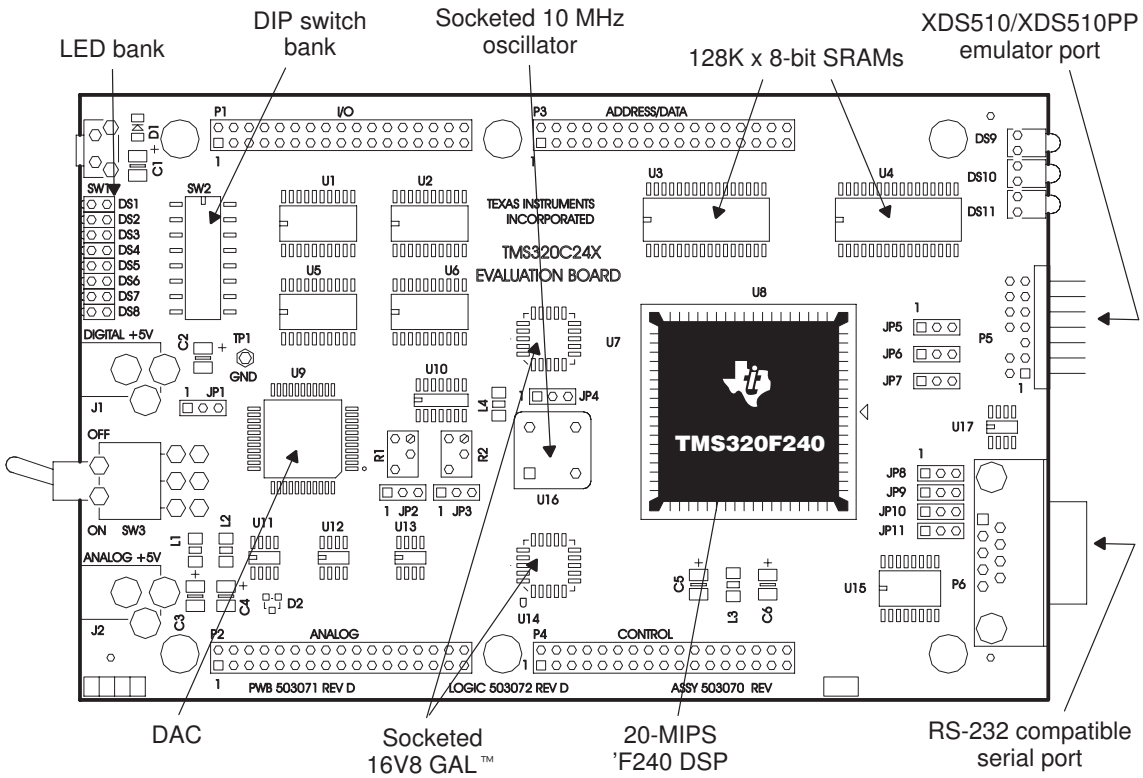
The following features of the ’C24x evaluation board that are useful in developing digital motor control and automotive applications:

- ’F240 fixed-point DSP controller
- 128K words of external on-board SRAM
- On-board, 4-channel, 12-bit digital-to-analog converter (DAC)
- RS-232 compatible serial port
- XDS510™/XDS510PP emulation port
- Bank of eight I/O memory-mapped DIP switches
- Bank of eight I/O memory-mapped LEDs

1.3 Functional Overview of the TMS320C24x Evaluation Board

A diagram of the ’C24x evaluation board is shown in Figure 1–1. This section gives a brief overview of the major components on the ’C24x evaluation board.

Figure 1–1. TMS320C24x Evaluation Board Diagram



The ’C24x evaluation board is built around the ’F240 DSP controller. This device operates at 20 MIPS with an instruction cycle time of 50 ns. It is optimized for digital motor control and power conversion applications. Some key features of the ’F240 device are listed below:

- The event manager (EV):
  - Three 16-bit, 6-mode, general-purpose timers
  - 12 pulse-width modulation (PWM) channels
  - A quadrature encoder pulse interface (QEP)
  - Four capture units

- Dual 10-bit, 8-channel analog-to-digital converters (ADCs)
- Synchronous and asynchronous communications peripherals
- A programmable phase-locked-loop (PLL) clock module
- 544 words of dual access RAM (DARAM)
- 16K words of on-chip Flash memory

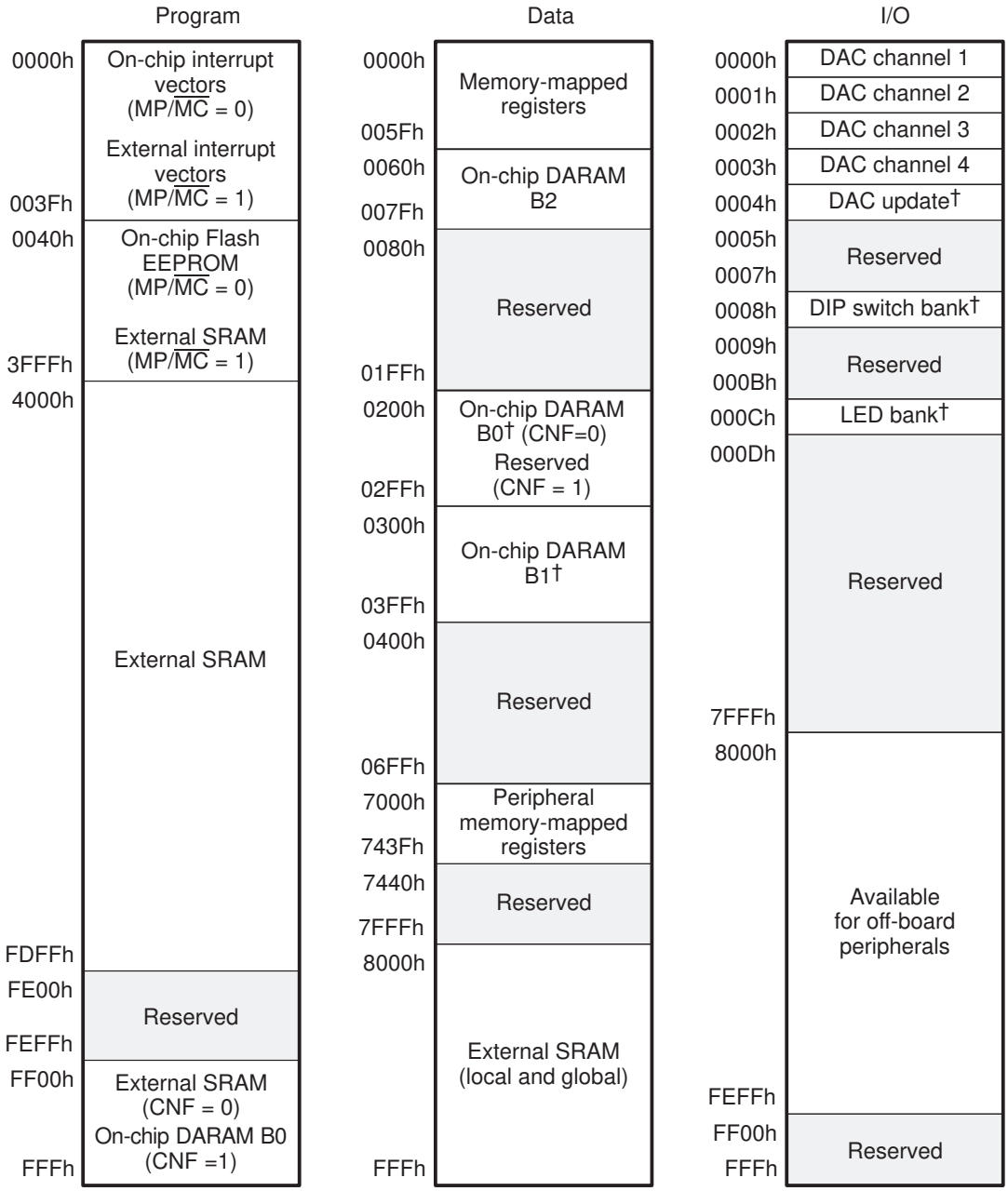
An on-board, 4-channel, 12-bit DAC (U9) has been included on the ’C24x evaluation board for code development purposes. The four DAC channel registers and the DAC update register are mapped into the I/O space of the ’F240 device. The DAC module requires that wait states be generated for proper operation. See section 5.3, *Digital-to-Analog Converter* on page 5-5, for more information on configuring the DAC for proper use.

The ’C24x evaluation board supports a total of 128K words of external on-board memory. The two 128K x 8 bit SRAMs (U3 and U4) on the evaluation board are partitioned in the following manner:

- 64K words external program memory
- 32K words external local data memory
- 32K words external global data memory

The on-board SRAMs interface with the external address and data buses of the ’F240 device. The 15-ns access time of the SRAMs allows the ’F240 device to access external program and data memory spaces with zero wait states. The ’C24x evaluation board memory maps are shown in Figure 1–2.

Figure 1–2. TMS320C24x EVM Memory Map



† These registers are mapped to multiple memory locations.

The ’C24x evaluation board has an on-board, RS-232 compatible, DB-9 serial port for asynchronous communication. The DB-9 serial port (P6) interfaces to the serial communications interface (SCI) peripheral on the ’F240 device through an RS-232 transceiver. The serial port can be configured for various communications protocols with software and hardware handshaking. See section 5.4, *RS-232 Serial Port*, on page 5-8, for more information on configuring and using the on-board serial port.

Four 34-pin connectors give access to all relevant signals on the ’C24x evaluation board. All EV, serial peripheral interface (SPI), and SCI signals are brought out to the I/O connector (P1). All analog signals, including the four DAC output channels, the 16 ADC input channels, and the ADC reference voltages, are brought out to the analog connector (P2). The external address and data bus signals can be found on the address/data connector (P3). And the external memory interface control signals are brought out to the control connector (P4).

The emulation port (P5), which is compatible with the IEEE1149.1 standard, allows the ’C24x evaluation board to act as an XDS emulator target board. The XDS510PP emulator included with the EVM operates as the main interface between the debugger and the ’C24x evaluation board.

1.4 Hardware and Software Requirements

You must meet the following hardware and software requirements to use the ’C24x evaluation board.

1.4.1 Hardware Requirements

In addition to the EVM contents, the following hardware is required to use the ’C24x evaluation board:

- **Host** A ’386 or higher IBM PC/AT™ or 100% compatible PC, 486, or Pentium PC with a 1.44M-byte 3.5-inch floppy disk drive
- **Ports** Supports 4-bit standard parallel ports (SPP4s), 8-bit bidirectional standard parallel ports (SPP8s), and enhanced parallel ports (EPPs). The EVM does not sup port extended capabilities ports (ECPs).

Note:

Your parallel port should be configured for EPP mode to obtain the maximum data transfer rate. Be sure to consult your PC documentation for specific instructions on configuring the parallel port.

- **Memory** Minimum of 4M bytes
- **Monitor** Color VGA

1.4.2 Software Requirements

In addition to the software provided, the following applications are required to use the ’C24x evaluation board:

- Windows™ 3.1 or Windows 95
- ASCII editor

Chapter 2

Hardware Installation

This chapter provides instructions on how to connect the XDS510PP emulator and evaluation board to your host PC and how to supply power to the target board and emulator.

Before installing your evaluation board, verify that your PC hardware and software platforms meet the minimum requirements described in section 1.4.

Topic **Page**

2.1 Connecting the TMS320C24x Evaluation Board and XDS510PP Emulator

Follow the steps below to ensure proper connections among the host PC, XDS510PP emulator, and 'C24x evaluation board. See Figure 2–1 for a connection diagram.

1) Turn off the power to the PC.

2) Connect one end of the DB25 printer cable to the parallel port of your PC.

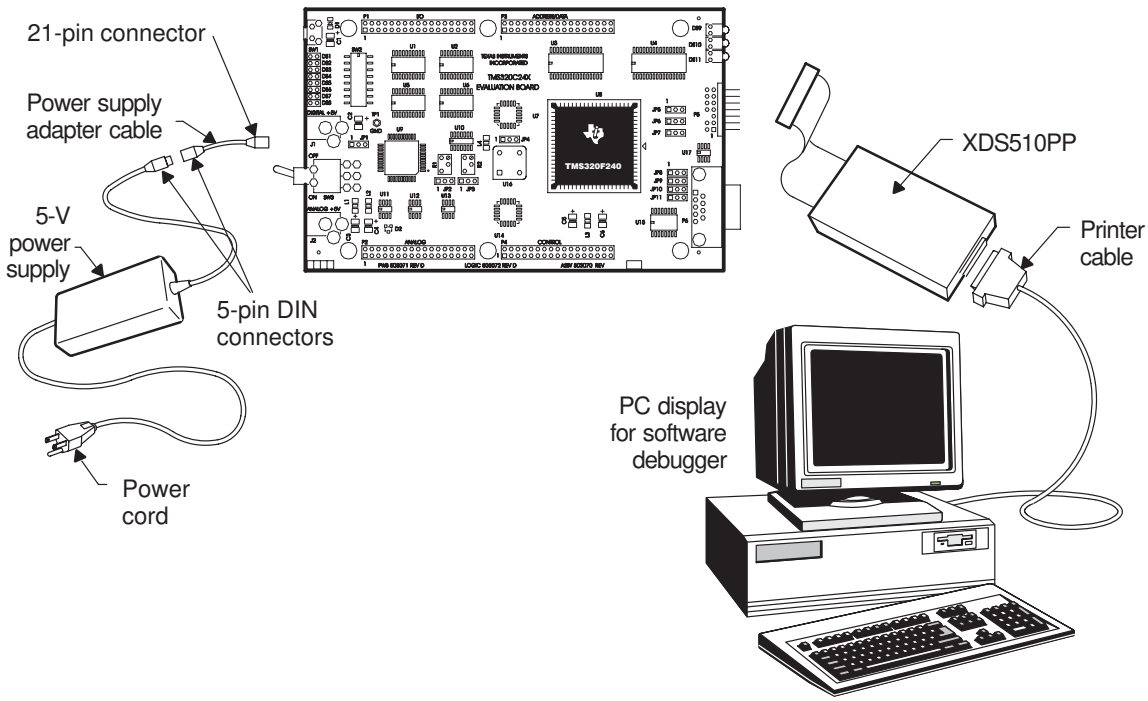
3) Connect the other end of the DB25 printer cable to the XDS510PP emulator.

4) Connect the 14-pin female header on the XDS510PP emulator to the 14-pin right-angle emulation port on the evaluation board (P5).

Note:

Pin 6 of the 14-pin XDS510PP emulator header is filled in and the corresponding pin 6 of the emulation port on the evaluation board is missing to ensure proper connection.

Figure 2–1. Connecting the TMS320C24x Evaluation Board and XDS510PP Emulator



2.2 Supplying Power to the TMS320C24x Evaluation Board and XDS510PP Emulator

Follow the steps below to supply power to the host PC and evaluation board. See Figure 2–1 for a connection diagram.

1) Make sure the power switch on the evaluation board is in the OFF position.
2) Use the black power cord to plug the power supply into the wall outlet.
3) Connect the 5-pin DIN end of the power supply adapter cable into the 5-pin DIN connector on the power supply.
4) Connect the 2.1-mm end of the power supply adapter cable into the digital power supply jack (J1) on the evaluation board.
5) Turn on the power to the PC and the evaluation board.

Chapter 3

Installing and Configuring the Code Development Tools

This chapter provides instructions for installing the tools code provided in the ’C24x EVM package. These include assembly language tools, C source debugger, and sample applications code. See the appropriate user’s guide for complete documentation.

Topic **Page**

3.1 Installing the Assembly Language Tools

Perform the following steps to install the assembly language tools. See the *TMS320C1x/C2x/C2xx/C5x Code Generation Tools Getting Started Guide* for additional information on installing and configuring the tools. These instructions assume that C is your fixed or hard disk and that A is your external, 3 1/2-inch drive.

1) In the kit, locate the box labeled *TMS320C1x/C2x/C2xx/C5x Assembler, PC System*. This box contains the assembly language tools. Open the box, and remove the two assembly language tools diskettes.

2) Create a directory on your hard disk drive for the assembly language tools by entering the following from a system prompt.

```
MD C:\C24XTOOLS
```

3) Copy all of the files from the two diskettes to the *C24XTOOLS* directory on your hard disk drive. Put diskette 1 into drive A, and enter the following from a system prompt:

```
COPY A:\DOS32\*.*  C:\C24XTOOLS
```

Repeat this step for disk 2 of the assembly language tools.

4) Modify the PATH statement to identify the assembly language tools directory. Edit your *autoexec.bat* file and find the line that includes PATH=. At the end of the line, type:

```
;C:\C24XTOOLS
```

See Example 3–1 for an example of how to modify your *autoexec.bat* file.

Example 3–1. Sample autoexec.bat File Modified to Include the Assembly Language Tools Directory in the PATH Statement.

```
DATE
TIME
ECHO OFF
PATH=C:\WINDOWS;C:\C24XTOOLS
CLS
```

Note:

Do not copy the OS/2 tools onto your hard disk drive. These tools are contained in the os2v2 directories of the product disks. The copy command properly installs the MS-DOS tools onto your hard disk.

3.2 Installing the TMS320C24x EVM Applications Code

Follow the steps below to install the sample applications code. See the *appscode.txt* file for more information on the specific applications included with your EVM. This file can be found on the *'C24x EVM Software Diskette*.

1) Locate the diskette labeled *'C24x EVM Software Diskette* in the 'C24x evaluation board box.

2) Create a directory on your hard disk for the assembly language tools by entering the following from a system prompt.

```
MD C:\C24XCODE
```

3) To install the sample applications code, copy all the files in the *APPSCODE* directory on the diskette to the *C24XCODE* directory on your hard disk. Put the *'C24x EVM Software Diskette* into drive A and enter the following from a system prompt:

```
COPY A:\APPSCODE\*.* C:\C24XCODE
```

4) Modify the PATH statement to identify the assembly language tools directory. Edit your *autoexec.bat* file and find the line that includes PATH=. At the end of the line, type:

```
;C:\C24XCODE
```

See Example 3–2 for an example of how to modify your *autoexec.bat* file.

Example 3–2. Sample autoexec.bat File Modified to Include the Assembly Language Tools Directory in the PATH Statement

```
DATE
TIME
ECHO OFF
PATH=C:\WINDOWS;C:\C24XTOOLS;C:\C24XCODE
CLS
```

3.3 Installing the Debugger and Configuring Your System

This section gives you instructions on how to install the ’C24x EVM C source debugger and how to configure your system. It includes instructions on installing the debugger software, configuring your system for the XDS510PP emulator, and verifying the installation of the debugger and emulator.

3.3.1 Installing the TMS320C24x EVM C Source Debugger Software

Follow the instructions below to install the ’C24x EVM C source debugger software.

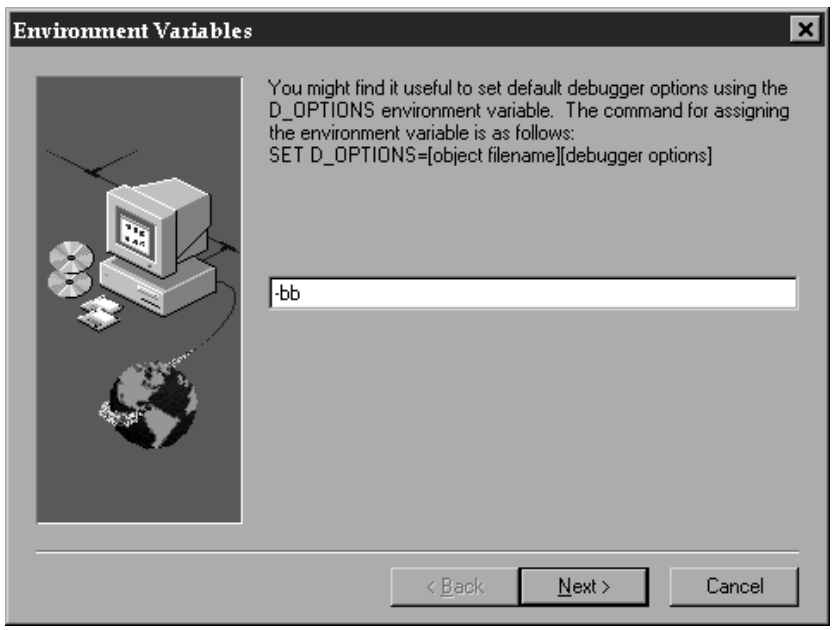
1) Locate the diskette labeled *TMS320C24x EVM C Source Debugger* in the *TMS320C24x EVM C Source Debugger* book. Open the box and remove the C source debugger diskette.

2) Insert the diskette into drive A:\, click on the *Start* button from the task bar in Windows95, select *Run...* (or in Windows 3.x.x select the *File* menu in the program manager, click *Run...*) and type:

```
A:\INSTALL.EXE
```

3) Follow the instructions in the installation utility until you reach the dialog box labeled *Environment Variables* shown in Figure 3–1.

Figure 3–1. D_OPTIONS Environment Variable Dialog Box



An environment variable is a special symbol used by the debugger to find or obtain certain types of information. The D_OPTIONS environment variable is convenient for specifying the debugger options commonly used when invoking the debugger.

SET D_OPTIONS=[object filename] [debugger options]

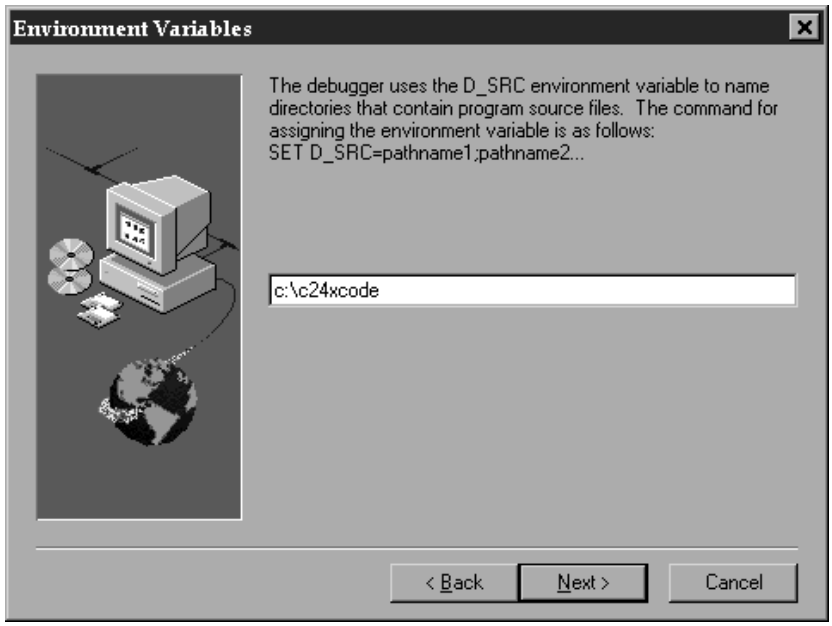
For example, the –bb option can be used to increase the debugger window by two sizes. Table 3–1 lists the options that you can identify with the D_OPTIONS environment variable.

Table 3–1. Options for Use With D_OPTIONS

Option	Description
–b[b]	Increase the size of the debugger window
–c	Clear the .bss section
–f *filename*	Identify a new board configuration file
–i *pathname*	Identify additional directories
–min	Select the minimal debugging mode
–n *processor name*	Identify processor for debugging
–s	Load the symbol table only
–t *filename*	Identify a new initialization file
–v	Load without the symbol table

See the *TMS320C2xx C Source Debugger User's Guide* for more information on debugger options and defining your target system to the debugger.

Figure 3–2. D_SRC Environment Variable Dialog Box



4) Define the directories that contain program source files in the D_SRC environment variable dialog box shown in Figure 3–2. For example, the *C24XCODE* directory must assigned to the D_SRC environment variable, because this directory contains the 'C24x EVM applications programs included with the EVM.

5) Complete the installation utility by following the instructions listed in the remaining dialog boxes. A file folder labeled *TMS320C24x EVM debugger* is created by the utility and contains three icons. The *'C24x debugger* icon is used to invoke the debugger. The *XDS510PP reset* icon resets the emulator and the *readme.txt* file gives additional information on configuring your system. Continue with section 3.3.2, *Configuring Your System for the XDS510PP Emulator*, after completing the debugger installation.

3.3.2 Configuring Your System for the XDS510PP Emulator

The *xds510pp.ini* file is used to set driver parameters for the XDS510PP emulator and is located in the *C24XHLL* directory. The three parameters defined in this initialization file are *mode*, *port*, and *speed*. The following are the default values for each parameter:

```
port = 378
mode = spp4
speed = 10
```

Each parameter in the *xds510pp.ini* file must be modified to reflect the configuration your system. A brief description of each parameter is given in Table 3–2. A more detailed description of each parameter is given in the paragraphs that follow.

Table 3–2. XDS510PP Initialization Parameters

Parameter	Description
Port	Selects the parallel port of your PC. The default address is 378h.
Mode	Defines the parallel port mode. The XDS510PP emulator supports the following modes: SPP4 4-bit unidirectional mode SPP8 8-bit bidirectional mode EPP Enhanced parallel port mode The default mode is SPP4.
Speed	Determines the amount of delay between port accesses. The default value is 10.

The *port* parameter defines the address of your parallel port. The default address on most PCs is 378h. Other commonly used addresses are 278h and 3BCh. Consult your PC documentation to determine the address of your parallel port. The port detection utility included with the debugger software, *portchk.exe*, can also be used to determine the parallel port address. This utility searches for parallel ports and tries to determine if they are bidirectional. See Appendix D, *Parallel Port Utility Programs*, for more information on the *portchk.exe* utility.

The *mode* parameter defines the configuration of the parallel port for the debugger. The 4-bit unidirectional (SPP4) mode works on all machines, but is the slowest. This mode is necessary for older PCs that do not have full 8-bit bidirectional parallel port capability. Newer PCs typically have 8-bit bidirectional (SPP8) and 8-bit enhanced parallel port (EPP) capability. EPP mode is the recommended choice and provides the highest throughput.

Note:

Your parallel port must be configured for EPP mode to obtain the maximum data transfer rate. In many cases, this requires entering the BIOS setup utility of your PC. Consult your PC documentation for specific instructions on configuring the parallel port.

The *speed* parameter determines the amount of delay between parallel port accesses. The amount of delay increases as the parameter value increases.

Set this parameter to 10 initially. Once the XDS510PP is operating correctly, you can reduce this number.

Two utility programs, *smcmode.exe* and *nscmode.exe* are included with the debugger software. These programs may help you in setting the parallel port mode for certain PC hardware configurations. Appendix D, *Parallel Port Utility Programs*, contains instructions on their use. See the *XDS51x Emulator Installation Guide* included in the emulation controller package, or the *readme.txt* file included in the *C24XHLL* directory, for more information regarding the configuration of your system for the XDS510PP emulator.

3.3.3 Verifying the Emulator Driver Installation

The emulator device driver enables the 'C24x EVM C source debugger to communicate with the XDS510PP emulator. See Appendix D, *Parallel Port Utility Programs*, for more information on this utility. Follow these steps to ensure that the emulator driver has been installed correctly:

1) Reset the XDS510PP emulator by double clicking on the *XDS510PP Reset* icon in the *TMS320C24x EVM Debugger* file folder. After resetting the XDS510PP emulator, you should see a message similar to the following in a system prompt window:

```
EMURST FOR THE XDS510PP VERSION 1.0
XDS510PP IS RESET, HARDWARE VERSION 1
```

   - ❑ If this message appears, continue with section 3.3.4, *Verifying the Debugger and Emulator Installation*.
   - ❑ If you see the following message the emulator has not been reset successfully:

```
EMURST FOR THE XDS510PP VERSION 1.0
COMMUNICATIONS ERROR, OR POD HAS NO POWER
```

     Continue with step 2 to troubleshoot your system. When you see the first message in step 1, you have resolved the problem and can go to section 3.3.4, *Verifying the Debugger and Emulator Installation*.

2) Check the following items to troubleshoot your system:
   - ❑ Be sure power is supplied to the 'C24x evaluation board and the power switch on the board (SW3) is in the ON position.
   - ❑ Verify that the parallel port is firmly connected to the parallel port on your PC and to the XDS510PP emulator pod.
   - ❑ Verify that the 14-pin header on the XDS510PP emulator pod is firmly connected to the emulation port (P5) on the evaluation board.

3) If the problem is not resolved, verify that the *xds510pp.ini* file has been configured correctly (see section 3.3.2, *Configuring Your System for the XDS510PP Emulator*, on page 3-6). Be sure that the *mode* and *port* parameters in this file match the settings of your PC. You can use the port detection utility (*portchk.exe*) to determine how your parallel port is configured.

4) Run the SMC port configuration utility (*smcmode.exe*) to determine whether your PC uses an SMC parallel port controller device. Enter the following line at a system prompt to execute the utility:

```
C:\C24XHLL\smcmode.exe
```

See Appendix D, *Parallel Port Utility Programs*, for more information on this utility.

- ❑ If an SMC device is detected but not configured for the desired mode, then use the *smcmode.exe* utility with the –m option to configure the device for the mode desired. For example, to configure the device for mode 1, EPP and SPP mode, (the recommended mode) type the following at a system prompt:

```
C:\C24XHLL\smcmode.exe -m 1
```

- ❑ If an SMC device is detected and configured for the desired mode, then reset the SMC device by typing the following at a system prompt:

```
C:\C24XHLL\smcmode.exe -r
```

See Appendix D, *Parallel Port Utility Programs* for more information on how to reset an SMC device.

5) Run the NSC port configuration utility (*nscmode.exe*) to determine whether your PC uses an NSC parallel port controller device. Enter the following line at a system prompt to execute the utility:

```
C:\C24XHLL\nscmode.exe
```

- ❑ If an NSC device is detected but not configured for the desired mode, then use the *nscmode.exe* utility with the –m option to configure the device for the mode desired. For example, to configure the device for mode 2, EPP mode, (the recommended mode) type the following at a system prompt:

```
C:\C24XHLL\nscmode.exe -m 2
```

3.3.4 Verifying the TMS320C24x EVM C Source Debugger Installation

Follow these instructions to ensure that the debugger software and XDS510PP emulator have been installed correctly:

1) Reset the XDS510PP emulator by double clicking on the *XDS510PP Reset* icon in the *TMS320C24x EVM Debugger* file folder.

Note:

The *XDS510PP Reset* utility program only resets the XDS510PP emulator, not the 'F240 device on the evaluation board. You should reset the emulator before invoking the debugger following a PC or evaluation board power-on condition.

2) Invoke the debugger by double clicking on the *C24x EVM Debugger* icon in the *TMS320C24x EVM Debugger* file folder. After invoking the debugger, you should see a display similar to the display in Figure 3–3.

Figure 3–3. TMS320C24x EVM C Source Debugger Display

- ❑ If you see a display similar to this one, you have correctly installed the debugger software and XDS510PP emulator. Continue with Chapter 4, *Assembling, Linking, and Running Your First Program*.

- ❑ If you see a display and the lines of code say *Invalid Address* or the fields in the MEMORY window are shown in red, the debugger may not be able to find the *emuinit.cmd* file. Check for the file in the *C24XHLL* directory or verify that the file resides in a directory specified by the D_SRC environment variable.

- The following message in the debugger display indicates another source:

```
CANNOT INITIALIZE THE TARGET !!
- Check the I/O configuration
- Check cabling and target power
```

Check the following items:

- Be sure power is supplied to the ’C24x evaluation board and the power switch on the board (SW3) is in the ON position.
- Verify that the parallel port is firmly connected to the parallel port on your PC and to the XDS510PP emulator pod.
- Verify that the 14-pin header on the XDS510PP emulator pod is firmly connected to the emulation port (P5) on the evaluation board.

Chapter 4

Assembling, Linking, and Running Your First Program

This chapter takes you through the process of assembling, linking, and running a simple assembly source program on the ’C24x evaluation board. The chapter describes a sample program that you can run on your own. This program, *leds.asm*, causese the I/O-mapped bank of LEDs to flash sequentially, showing you the program is running. After completing this chapter, you will be able to assemble, link and run your own programs on the evaluation board.

Topic **Page**

4.1 Assembling the Program

The 2-pass assembler, *dspa.exe*, translates assembly language source files into machine language object files. These translated files are in common object file format (COFF), discussed in the *TMS320C1x/C2x/C2xx/C5x Assembly Language Tools User's Guide*.

The assembly language source file, *leds.asm*, is located in the *C24XCODE* directory. Execute the following command from a system prompt in the *C24XCODE* directory to assemble *leds.asm* with the –v2xx and –s options:

```
dspa leds.asm -v2xx -s
```

Note:

The assembler does not execute out of the *C24XCODE* directory unless you have included the *C24XTOOLS* directory in the path statement of your *autoexec.bat* file and have rebooted your PC.

The –v2xx option tells the assembler to produce code for the 'C2xx family of devices. The –s option puts all defined symbols, including labels, in the object file's symbol table. See the *TMS320C1x/C2x/C2xx/C5x Assembly Language Tools User's Guide* for further instructions on invoking the assembler and a complete listing of all assembler options. The following message appears after successfully assembling the source file, *leds.asm*:

```
TMS320C1x/C2x/C2xx/C5x COFF Assembler   Version 6.60
Copyright (c) 1987-1995 Texas Instruments Incorporated
Pass 1
Pass 2
No Errors, No Warnings
```

The assembler produces an object file, *leds.obj*, from the assembly source file, *leds.asm*. The *leds.obj* is relocatable and is used by the linker, as shown in section 4.2, *Linking the Program*.

4.2 Linking the Program

The linker creates an executable COFF object module that can be downloaded to the ’F240 device on the evaluation board. The object file created by the assembler in the previous section, *leds.obj*, and the command file *f240init.cmd* are used by the linker to generate a single output file, *leds.out*. The linker, *dsplnk.exe*, accepts several types of files as input, including object files, command files, libraries, and partially linked files. The *f240init.cmd* file defines the memory map of the ’F240 device and tells the linker where certain sections of code should reside in memory.

The *leds.out* file generated by the linker can be downloaded directly to the ’F240 device. Execute the following command from a system prompt in the *C24XCODE* directory to link the *leds.obj* and *f240lnk.cmd* files:

```
dsplnk leds.obj f240lnk.cmd –o leds.out
```

Note:

The linker does not execute out of the *C24XCODE* directory unless you have included the *C24XTOOLS* directory in the path statement of your *autoexec.bat* file and have rebooted your PC.

The preceding command creates an output file, *leds.out*, from the *leds.obj* and *f240lnk.cmd* input files. The –o option specifies the name of the output file, *leds.out*. See the *TMS320C1x/C2x/C2xx/C5x Assembly Language Tools User’s Guide* for further instructions on invoking the linker and a complete listing of all linker options. The following message appears after successfully creating the output file, *leds.out*:

```
TMS320C1x/C2x/C2xx/C5x COFF Linker        Version 6.60
Copyright (c) 1987–1995 Texas Instruments Incorporated
```

4.3 Running the Program

Use the C source debugger to download the *leds.out* file to the evaluation board. After invoking the debugger, type the following command at the prompt in the command window of the debugger:

```
load c:\C24XCODE\leds.out
```

Note:

Be sure that the MP/$\overline{MC}$ jumper is in position 1–2 (its default position). If it is not, the debugger will unsuccessfully attempt to load the program into the internal Flash memory.

Press the F5 function key to run the program after it has been loaded. The debugger screen looks similar to the one in Figure 4–1 when the program is running. The command window indicates that the program is running and the I/O mapped bank of LEDs (DS1–DS8) on the evaluation board flashes sequentially.

Figure 4–1. TMS320C24x EVM C Source Debugger Display

Chapter 5

TMS320C24x Evaluation Board Components

This chapter describes the major components on the ’C24x evaluation board and the hardware interfaces between the ’F240 DSP and peripherals. These include external memory, the digital-to-analog converter (DAC), LEDs, and DIP switches. This chapter also provides peripheral register definitions and discussed operation of the oscillator, DAC, LEDs, and DIP switches.

Topic **Page**

5.1 External Memory

The ’C24x evaluation board has a total of 128K words of external, on-board memory. The two 128K x 8-bit SRAMs (U3 and U4) are partitioned in the following manner:

- 64K words external program memory
- 32K words external local data memory
- 32K words external global data memory

The on-board SRAMs interface to the external address and data buses of the ’F240 device. One of the two socketed 16V8 generic-array-logic (GAL) devices (U7) on the ’C24x evaluation board has been factory programmed to control the write-enable ($\overline{WE}$) and output-enable ($\overline{OE}$) functions of the SRAMs and the memory space decode logic. The 15-ns access time of the SRAMs allows the ’F240 device to access external program and data memory spaces with zero wait states. The ’C24x evaluation board memory maps are shown in Figure 1–2 on page 1-6.

5.1.1 External Program Memory Space

The ’C24x evaluation board has 64K words of on-board external program memory. When the ’F240 device is operating in microprocessor mode, the entire program memory space of the ’F240 device resides in the 64K words of external program memory. The MP/$\overline{MC}$ jumper (JP6) must be in position 1–2 for the ’F240 device to operate in microprocessor mode.

When the ’F240 device is operating in microcomputer mode, only program memory accesses above 4000h reside in external program memory. All program memory accesses at or below 4000h reside in the 16K words of on-chip Flash memory. The MP/$\overline{MC}$ jumper (JP6) must be in position 2–3 for the ’F240 device to operate in microcomputer mode.

5.1.2 External Data Memory Space

External data memory accesses for the ’F240 device are only valid for the upper 32K words (8000h–FFFFh) of local data memory space. The lower 32K words (0000h–7FFFh) of local data memory space are reserved for on-chip peripheral registers and memory-mapped registers of the CPU.

Addresses in the upper 32K words (8000h–FFFFh) of local data memory space can be partitioned in order to create a second distinct data memory space known as global data memory. The global memory allocation register (GREG) determines the size of the global data memory space, which is between 256 and 32K words. The GREG is connected to the eight least significant bits (LSBs) of

the internal data bus and is one of the CPU memory-mapped registers. This register is mapped to data memory location 0005h. Table 5–1 shows the allowable GREG values and shows the corresponding address ranges set aside for local and global data memory spaces.

Note:

Choose only the GREG values listed in Table 5–1. Other values lead to fragmented memory maps.

The ’C24x evaluation board has 32K words of local data memory and 32K words of global data memory for a total of 64K words of on-board external data memory. It is important to realize that the two data memory spaces actually reside in two separate physical external memory spaces. For example, if you write a value to 8000h in local data memory space, you do not overwrite a value at 8000h in global memory space. However, if you want to read the value at 8000h in global data memory space, you must first modify the GREG to make that address valid in that space.

Table 5–1. External Local and Global Data Memory Configurations

GREG Value		External Local Memory		External Global Memory	
High Byte	**Low Byte**	**Range**	**Words**	**Range**	**Words**
XXXX XXXX	0000 0000	8000h–FFFFh	32 768	–	0
XXXX XXXX	1111 1111	8000h–FEFFh	32 512	FF00h–FFFFh	256
XXXX XXXX	1111 1110	8000h–FDFFh	32 256	FE00h–FFFFh	512
XXXX XXXX	1111 1100	8000h–FBFFh	31 744	FC00h–FFFFh	1 024
XXXX XXXX	1111 1000	8000h–F7FFh	30 720	F800h–FFFFh	2 048
XXXX XXXX	1111 0000	8000h–EFFFh	28 672	F000h–FFFFh	4 096
XXXX XXXX	1110 0000	8000h–DFFFh	24 576	E000h–FFFFh	8 192
XXXX XXXX	1100 0000	8000h–BFFFh	16 384	C000h–FFFFh	16 384
XXXX XXXX	1000 0000	–	0	8000h–FFFFh	32 768

Note: X = Any value

5.2 Analog-to-Digital Converter

The ’F240 device has an on-chip, dual, 10-bit analog-to-digital converter (ADC) module. The ADC module consists of two 10-bit ADCs with two internal sample-and-hold circuits. Eight analog inputs are provided for each ADC unit via an 8-to-1 analog multiplexer. The maximum conversion time for each ADC unit is 6.6 μs. See the *Dual 10-Bit Analog to Digital Converter (ADC) Module* section in the *TMS320C240 DSP Controllers Reference Set, Volume 2*, for more details on the on-chip ADC module.

This module requires an external analog supply voltage of 5 V dc. The analog supply jumper (JP1) on the ’C24x evaluation board allows you to select the analog power source. When the JP1 jumper is in position 1–2, the digital and analog power planes on the evaluation board are connected, allowing you to use one common power supply. When the JP1 jumper is in position 2–3, the power planes are not connected and require separate analog and digital power supplies. See section 5.6, *Jumpers*, for more details.

The reference voltages for the ADC module must also be supplied from an external source. The $V_{ref\ hi}$ (JP2) and $V_{ref\ lo}$ (JP3) jumpers allow you to set the upper and lower references anywhere between 0 and 5 V dc. When the JP2 and JP3 jumpers are in position 1–2, $V_{ref\ hi}$ is connected directly to the analog power plane and $V_{ref\ lo}$ is connected directly to the analog ground plane. When jumpers JP2 and JP3 are in position 2–3, the upper and lower references can be varied anywhere between 0 and 5 V dc by adjusting the appropriate potentiometers (R1 and R2). Turning the screw clockwise increases the reference voltage and turning the screw counterclockwise decreases the reference voltage.

The analog connector gives you access to the $V_{ref\ hi}$ and $V_{ref\ lo}$ pins and the 16 on-chip ADC channels. All analog signals must be input to the ’F240 device through the analog connector.

5.3 Digital-to-Analog Converter

The ’C24x evaluation board has an on-board, quad, 12-bit, double-buffered digital-to-analog converter (DAC) module for code development. The four on-board DAC channels and the DAC update register are mapped into the I/O space of the ’F240 device. Table 5–2 lists the register names, the register addresses, and a brief functional description for each DAC register.

Table 5–2. DAC Registers

Register Name	Register Address	Description
DAC0	0000h	Input data register for DAC0
DAC1	0001h	Input data register for DAC1
DAC2	0002h	Input data register for DAC2
DAC3	0003h	Input data register for DAC3
DAC update	0004h	DAC update register

The DAC module requires that wait states be generated for proper operation. The ’F240 device must be programmed to generate one software wait state for I/O space accesses and the 20-MHz CPUCLK signal must be output on the CLKOUT pin of the device. The CPUCLK signal is used by the GAL (U14) to generate the additional hardware wait states required by the DAC module. The following code illustrates how to configure the ’F240 device to generate the appropriate number of wait states:

```
LDP    #00E0h              ;Set Data Page Pointer to 00E0h
SPLK   #00BBh,CKCR1        ;CLKIN(OSC)=10MHz,CPUCLK=20MHz
SPLK   #00C3h,CKCR0        ;CLKMD=PLL Enable,SYSCLK=CPUCLK/2
SPLK   #40C0h,SYSCR        ;CLKOUT=CPUCLK

LDP    #0000h              ;Set Data Page Pointer to 0000h
SPLK   #4h,GPR0            ;Set wait state generator for:
OUT    GPR0,WSGR           ;Program Space, 0 wait states
                           ;Data Space, 0 wait states
                           ;I/O Space, 1 wait state
```

In the preceding code, the clock control registers (CKCR0, CKCR1) are configured to generate a 20-MHz CPUCLK signal from an input clock frequency of 10 MHz. The system control register (SYSCR) is configured so that the CPUCLK signal is output on the CLKOUT pin of the device. See section 5.9, *Oscillator*, on page 5-26 for more information on configuring the clock control registers.

The wait state generator register (WSGR) is also programmed to generate one software wait state for I/O space accesses. The preceding code assumes that GPR0 is an uninitialized register in data memory and is defined in the *.bss* section of your code. It also assumes that the *.bss* section is mapped into RAM block B2 of the ’F240 device, which requires setting the data page pointer to 0. The GPR0 register serves only as temporary storage because the OUT instruction does not support immediate addressing.

The digital value to be converted must be written to the appropriate DAC input data register. A value must then be written to the DAC update register to start the conversion for all four DACs. The code that follows illustrates how to write four digital values to the four DAC input registers and then start the digital-to-analog conversion by writing a dummy value to the DAC update register.

```
LDP    #0000h             ;Set data page pointer to 0
SPLK   #03FFh,DAC0_VAL    ;Load 03FFh into DAC0_VAL register
SPLK   #07FFh,DAC1_VAL    ;Load 07FFh into DAC1_VAL register
SPLK   #0BFFh,DAC2_VAL    ;Load 0BFFh into DAC2_VAL register
SPLK   #0FFFh,DAC3_VAL    ;Load 0FFFh into DAC3_VAL register

OUT    DAC0_VAL,0000h     ;Write 03FFh to the DAC0 register
OUT    DAC1_VAL,0001h     ;Write 07FFh to the DAC1 register
OUT    DAC2_VAL,0002h     ;Write 0BFFh to the DAC2 register
OUT    DAC3_VAL,0003h     ;Write 0FFFh to the DAC3 register

OUT    DAC3_VAL,0004h     ;Start DAC conversions by writing a
                          ;value to the DAC UPDATE register
```

The value stored in DAC3_VAL is written to the DAC update register (0004h); however, any value can be written to this register to start the conversion. This code assumes that DAC0_VAL, DAC1_VAL, DAC2_VAL, and DAC3_VAL are uninitialized registers in data memory and are defined in the *.bss* section of your code. It also assumes that the .bss section is mapped into RAM block B2 of the ’F240 device, which requires setting the data page pointer to 0. These registers serve only as temporary storage because the OUT instruction does not support immediate addressing.

See Table 5–3 for information on the expected voltage level at each DACOUT pin on the ’C24x evaluation board. This table assumes that the $V_{ref\ hi}$ and $V_{ref\ lo}$ pins on the ’F240 device are set to 5 V dc and 0 V dc, respectively.

Table 5–3. DAC Output

DAC Register	Value in Register	Output Pin	Voltage at Pin
DAC0	03FFh	DACOUT0	1.25
DAC1	07FFh	DACOUT1	2.50
DAC2	0BFFh	DACOUT2	3.75
DAC3	0FFFh	DACOUT3	5.00

Remember that the DAC module is considered a write-only device by the ’F240. Therefore, the correct values written to the DAC registers are not reflected when reading the registers. The last value stored on the external data bus is read, instead.

Note:

The default *emuinit.cmd* file for the ’F240 device defines the DAC registers as write-only memory.

5.4 RS-232 Serial Port

The ’C24x evaluation board has an RS-232 compatible DB-9 serial port for asynchronous communication. The DB-9 serial port (P6) interfaces to the SCI peripheral on the ’F240 device through an RS-232 transceiver. See the *TMS320C24x DSP Controllers Reference Set, Volume 2* for more information on using the SCI peripheral. This section focuses on the interface between the ’C24x evaluation board and a host processor.

Five RS-232 signals can be used to implement various communications protocols with software and hardware handshaking on the ’C24x evaluation board. These signals are:

- Receive data (RX)
- Transmit data (TX)
- Clear to send (CTS)
- Request to send (RTS)
- Data terminal ready (DTR)

You need an RS-232 cable to connect the serial port on the evaluation board to a host. The pinout for the evaluation board and host serial ports is provided in Table 5–4. Make sure your cable conforms to these specifications.

Table 5–4. Evaluation Board and Host RS-232 Serial Port Connections

	Evaluation Board Serial Port			Host Serial Port		
Function	**Pin (DB-9)**	**Signal**		**Signal**	**Pin (DB-9)**	**Pin (DB-25)**
Carrier detect (not used)	1	NC		DCD	1	8
Transmit data to host	2	SCITX/IO	→	RX	2	3
Transmit data to evaluation board	3	SCIRX/IO	←	TX	3	2
Reset evaluation board via host†	4	HOSTRESET	←	DTR	4	20
Signal ground	5	GND	–	GND	5	7
Data set ready (not used)	6	NC		DSR	6	6
Request to send‡	7	$\overline{\text{BIO}}$/IOPC3	←	RTS	7	4
Clear to send	8	XF/IOPC2	→	CTS	8	5
Ring indicator (not used)	9	NC		RI	9	22

Note: NC = No connection
† This line should only be connected if you are implementing the host reset function.
‡ This line should only be connected if you are implementing hardware handshaking.

5.4.1 Implementing a Communications Protocol with Software Handshaking

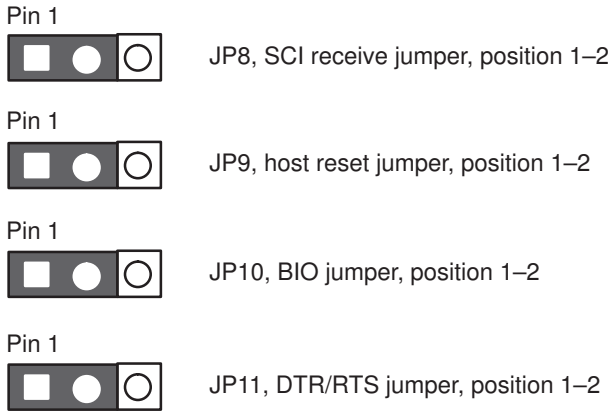
The RX and TX data lines are the only two lines required to implement a communications protocol with software handshaking (for example, Xon/Xoff). The RX and TX lines interface to the SCI peripheral on the ’F240 device through an RS-232 transceiver and are used to transmit and receive data to and from the host processor.

The SCITXD/IO pin on the ’F240 device is responsible for transmitting asynchronous data to the host processor. This pin is connected directly to both the RX line of the DB-9 serial port (via the RS-232 transceiver) and pin 27 of the I/O connector (P1).

The SCIRXD/IO pin on the ’F240 device is responsible for receiving asynchronous data from the host processor. The SCIRXD/IO pin connection is jumper selectable. When the SCI receive jumper (JP8) is in position 1–2, the SCIRXD/IO pin is connected to the TX line of the DB-9 serial port (via the RS-232 transceiver). The SCIRXD/IO pin is connected to pin 28 of the I/O connector (P1) when this jumper is in position 2–3.

Four jumpers (JP8–JP11) are used to configure the RS-232 serial port on the ’C24x evaluation board. When implementing a communications protocol with software handshaking, the serial port should be configured so that only the RX and TX lines are active. The jumper settings to configure the evaluation board for such a communications protocol are shown in Figure 5–1. See Figure 5–3 to understand how each jumper affects the hardware configuration of the serial port on the ’C24x evaluation board.

Figure 5–1. Software Handshaking Jumper Configuration



Since the data terminal ready (DTR) line is not used to implement a communications protocol with software handshaking, this line can be used by the host processor to reset the ’C24x evaluation board remotely. To implement this function, the DTR line must be selected as the active line by placing the DTR/RTS jumper (JP11) in position 1–2. The HOSTRESET signal on the evaluation board must also be selected by placing the host reset jumper (JP9) in position 2–3.

The HOSTRESET signal on the evaluation board drives the socketed 16V8 generic-array-logic (GAL) device responsible for controlling the reset logic (U14). The host processor must pull the DTR line low to cause a reset on the evaluation board.

The jumper settings to configure the evaluation board for a communications protocol with software handshaking and the host reset function are shown in Figure 5–2. See Figure 5–3 to understand how each jumper affects the hardware configuration of the serial port on the evaluation board.

Figure 5–2. Software Handshaking and Host Reset Jumper Configuration

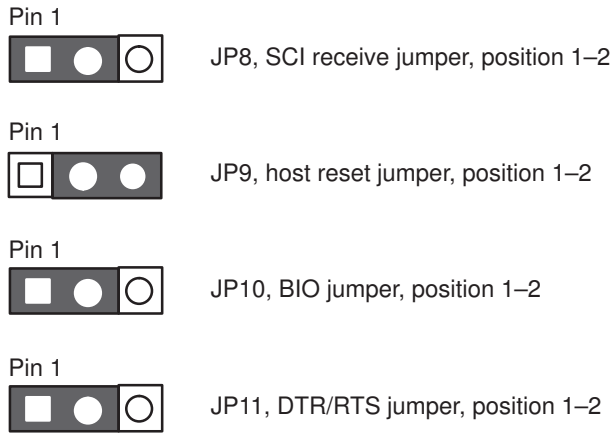
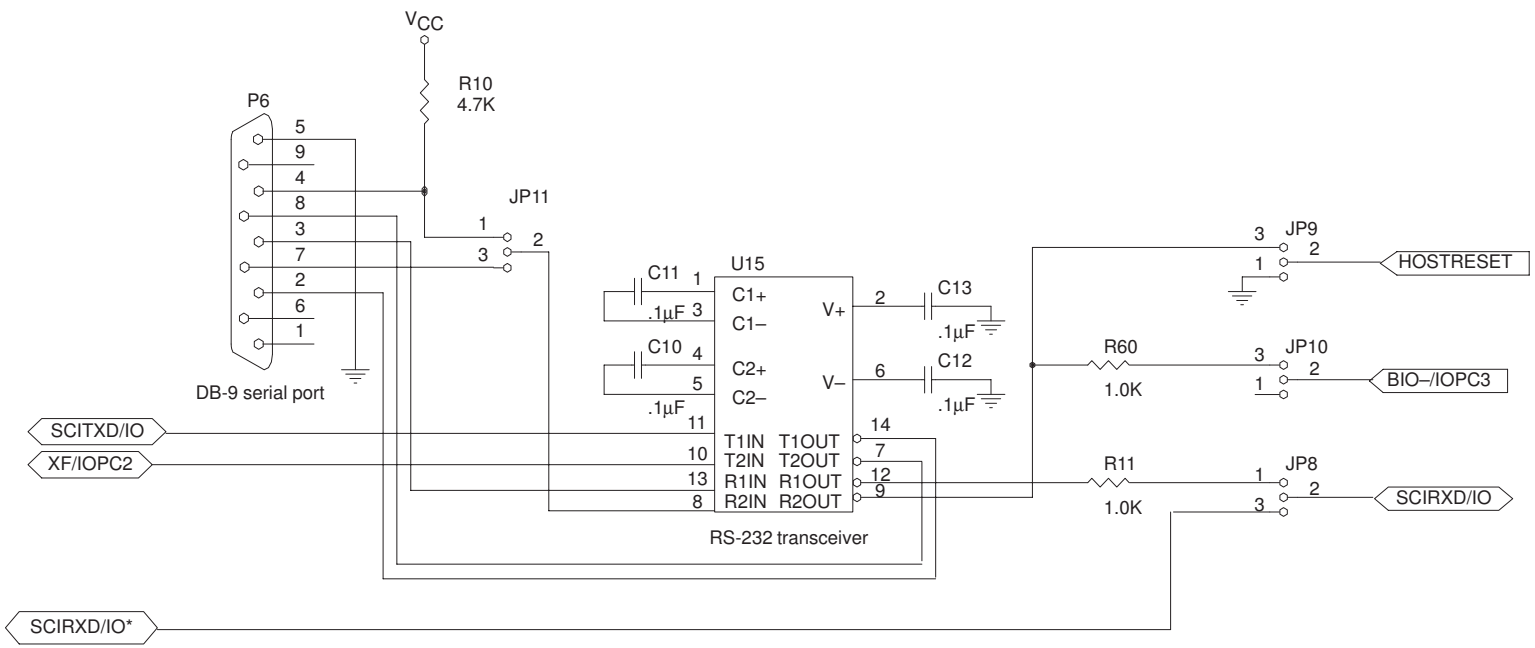


Figure 5–3. RS-232 Serial Port Hardware Interface on the TMS320C24x Evaluation Board



5.4.2 Implementing a Communications Protocol with Hardware Handshaking

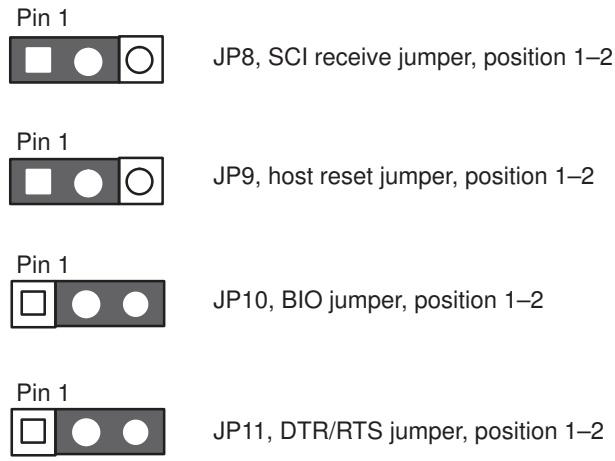
The CTS and RTS lines can be used in conjunction with the RX and TX lines to implement a communications protocol with hardware handshaking. The CTS line is used by the ’F240 device to notify the host processor that it is ready for data to be transmitted. The RTS line is used by the host processor to check whether the ’F240 device is ready to receive data.

The XF/IOPC2 pin on the ’F240 device generates the CTS signal for the host processor. This pin is connected directly to the CTS line of the DB-9 serial port (via the RS-232 transceiver) and pin 27 of the I/O connector (P1).

The DTR/RTS and BIO jumpers (JP11 and JP10) must be properly configured so that the BIO/IOPC3 pin on the ’F240 device can receive the RTS signal from the host processor. The DTR/RTS jumper (JP11) must be in position 2–3 to select the RTS line as the active line and the BIO jumper (JP10) must be in position 1–2 to connect the BIO/IOPC3 pin on the ’F240 to the RTS line of the DB-9 serial port (via the RS-232 transceiver).

See Figure 5–3 to understand how each jumper affects the hardware configuration of the serial port on the evaluation board. The jumper settings to configure the evaluation board for a communications protocol with hardware handshaking are shown below in Figure 5–4.

Figure 5–4. Hardware Handshaking Jumper Configuration



5.5 Connectors

The four 34-pin connectors give you access to all relevant signals on the ’C24x evaluation board. All event manager (EV), serial peripheral interface (SPI), and serial controller interface (SCI) signals are on the I/O connector (P1). All analog signals, including the four DAC output channels, 16 ADC output channels, and analog reference voltages are on the analog connector (P2). The external address and data bus signals are on the address/data (P3) connector and the external memory interface control signals are brought out to the control connector (P4). Sections 5.5.1 through 5.5.5 on pages 5-15 to 5-19 provide more information regarding each of the 34-pin connectors.

The relative position and 0.1-inch spacing of the four 34-pin connectors also allow you to easily connect external boards, sometimes referred to as daughter cards, to the evaluation board. The universal power supply included with the EVM has two independent 5-V dc power supplies capable of driving a total of 3.3 A. Daughter cards can include a 5-pin DIN connector to separate the power supplies, or you can use the included DIN-to-2.1-mm adapter to supply power to both the ’C24x evaluation board and the daughter card.

The 14-pin emulation port (P5) allows the evaluation board to act as a target system for either the XDS510PP (included in the EVM) or the XDS510 emulator. See section 5.5.5, *Emulation Port*, for more information regarding this connector.

5.5.1 I/O Connector

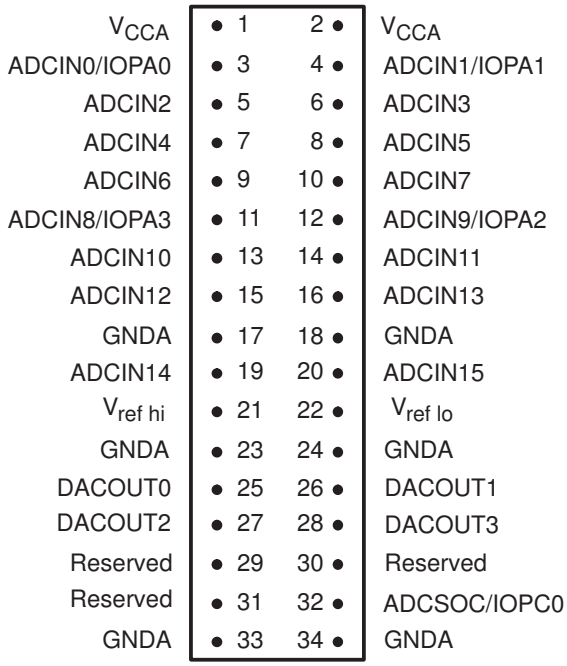
The I/O connector (P1) gives you access to all EV, SPI, and SCI signals from the ’F240 device. This connector can also supply digital power and ground to any peripherals or daughter cards. A diagram of the I/O connector is shown in Figure 5–5. See Table B–1 in Appendix B for the specific I/O connector pin assignments and a brief description of each signal brought out to the connector.

Figure 5–5. I/O Connector (P1) Diagram

5.5.2 Analog Connector

The analog connector (P2) gives you access to all on-chip ADC signals and all on-board DAC signals. This connector can also supply analog power and ground to any peripherals or daughter cards. The source of the analog power and ground is determined by the position of analog jumper (JP1). A diagram of the analog connector is shown in Figure 5–6. See section 5.6, *Jumpers*, on page 5-20, for more details on configuring the analog power source. See Table B–2 in Appendix B for the specific analog connector pin assignments and a brief description of each signal brought out to the connector.

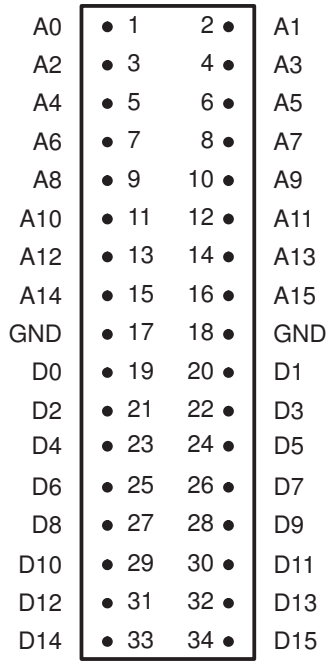
Figure 5–6. Analog Connector (P2) Diagram



5.5.3 Address/Data Connector

The address/data connector (P3) gives you access to the external data bus and address bus of the ’F240 device. A diagram of the address/data connector is shown in Figure 5–7. See Table B–3 in Appendix B for the specific address/data connector pin assignments and a brief description of each signal brought out to the connector.

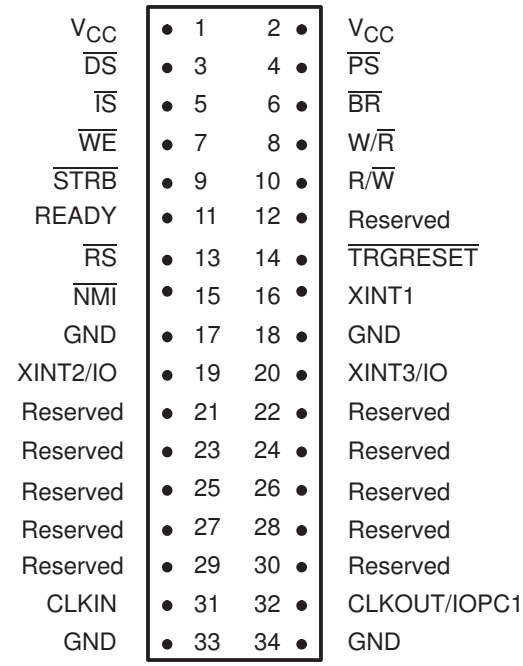
Figure 5–7. Address/Data Connector (P3 Diagram)



5.5.4 Control Connector

The control connector (P4) gives you access to all control signals for the ’F240 device. This connector can also supply digital power and ground to any peripherals or daughter cards. A diagram of the control connector is shown in Figure 5–8. See Table B–4 in Appendix B for the specific control connector pin assignments and a brief description of each signal brought out to the connector.

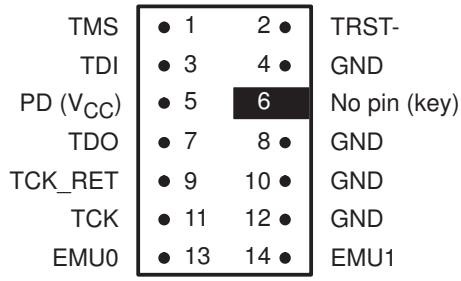
Figure 5–8. Control Connector (P4) Diagram



5.5.5 Emulation Port

JTAG target devices, such as the ’F240 DSP, support emulation through a dedicated emulation port. This port is based on the IEEE 1149.1 standard and is accessed by the emulator. The evaluation board, which serves as the target system, has a 14-pin header (two rows of seven pins) with the connections shown in Figure 5–9 to communicate with either the XDS510 or XDS510PP emulator. Table B–5 in Appendix B describes the emulation signals.

Figure 5–9. 14-Pin Emulation Port Signals



5.6 Jumpers

There are eight jumpers on the ’C24x evaluation board for you to configure. These jumpers configure the analog supply and reference voltages, clock source, Flash programming/watchdog disabling, external memory, and off-board communications.

Table 5–5 summarizes the function of each jumper on the ’C24x evaluation board. Each entry in the table gives the jumper name, jumper number, and the resulting function when it is in a given position. Use this table as a reference when configuring the jumpers on the ’C24x evaluation board.

Table 5–5. TMS320C24x Evaluation Board Jumpers

Jumper Name	Jumper	Position	Description
Analog supply jumper	JP1	1–2	The analog supply voltage and ground are tied to the digital supply voltage and ground.
		2–3	A separate off-board analog voltage must be supplied through the analog voltage connector (J2).
$V_{ref\ hi}$ jumper	JP2	1–2	The on-chip ADC $V_{ref\ hi}$ pin is tied directly to the analog supply voltage.
		2–3	The on-chip ADC $V_{ref\ hi}$ pin can be varied between 0 V dc and 5 V dc by adjusting the $V_{ref\ hi}$ potentiometer (R1).
$V_{ref\ lo}$ jumper	JP3	1–2	The on-chip ADC $V_{ref\ lo}$ signal is tied directly to the analog ground.
		2–3	The on-chip ADC $V_{ref\ lo}$ signal can be varied between 0 V dc and 5 V dc by adjusting the $V_{ref\ lo}$ potentiometer (R2).
Clock-in jumper	JP4	1–2	The on-board 10-MHz oscillator is used for the input clock signal.
		2–3	An off-board clock source is used for the input clock signal.
Flash/watchdog jumper	JP5	1–2	The Flash array cannot be programmed or the watchdog disabled.
		2–3	The Flash array can be programmed and the watchdog disabled.

Table 5–5. TMS320C24x Evaluation Board Jumpers (Continued)

Jumper Name	Jumper	Position	Description
MP/$\overline{\text{MC}}$ jumper	JP6	1–2	The ’F240 device runs in microprocessor mode. (All program memory accesses are off chip.)
		2–3	The ’F240 device runs in microcomputer mode. (Program memory accesses at 4000h and below are on chip; accesses above 4000h are off chip.)
Oscillator bypass jumper	JP7	1–2	The on-chip oscillator is bypassed. (The on-chip oscillator is not required when using an external oscillator to generate the clock signal for the ’F240 device.)
		2–3	The on-chip oscillator is enabled. (The on-chip oscillator is required when using a crystal to generate the clock signal for the ’F240 device.)
SCI receive jumper	JP8	1–2	The SCIRXD/IO pin is connected directly to the on-board RS-232 transceiver (U15).
		2–3	The SCIRXD/IO pin is connected directly to the I/O connector (P1).
Host reset jumper	JP9	1–2	The $\overline{\text{HOSTRESET}}$ signal is inactive.
		2–3	The $\overline{\text{HOSTRESET}}$ signal is active and is connected to the serial port line specified by the DTR/RTS jumper (via the RS-232 transceiver).
BIO jumper	JP10	1–2	The $\overline{\text{BIO}}$/IOPC2 pin is connected to pin 20 of the I/O connector (P1).
		2–3	The $\overline{\text{BIO}}$/IOPC2 pin is connected to pin 20 of the I/O connector (P1) and the serial port line specified by the DTR/RTS jumper (via the RS-232 transceiver).
DTR/RTS jumper	JP11	1–2	The DTR line of the serial port (P6) is selected.
		2–3	The RTS line of the serial port (P6) is selected.

5.7 LEDs

There are eleven LEDs on the ’C24x evaluation board. These LEDs display such information as board power, the status of the XF and BIO pins, and the status of certain bits mapped into the I/O memory space.

The bank of eight LEDs on the evaluation board are mapped to 000Ch in the I/O memory space of the ’F240 device. Each of the eight LEDs can be turned on and off independently by setting or clearing the specific bit assigned to that individual LED. The following instructions can be used to turn on all eight LEDs by setting the eight LSBs in the LED register that is mapped at 000Ch in I/O space:

```
SPLK   #00FFh,LEDS_ON    ;Load value into the uninitialized
                         ;LEDS_ON register

OUT    LEDS_ON,000Ch     ;Write the value stored in LEDS_ON
                         ;to 000Ch in the I/O memory space
```

The following instructions can be used to turn off all eight LEDs by clearing the eight LSBs in the LED register mapped to 000Ch in I/O space:

```
SPLK   #0000h,LEDS_OFF   ;Load value into the uninitialized
                         ;LEDS_OFF register

OUT    LEDS_OFF,000Ch    ;Write the value stored in LEDS_OFF
                         ;to 000Ch in the I/O memory space
```

The LEDS_ON and LEDS_OFF variables in the preceding code load a value into an uninitialized register that must be defined in the *.bss* section of your code.

The bank of eight I/O-mapped LEDs are considered write-only devices by the ’F240. Therefore, the correct status of the LED register is not reflected when reading address 000Ch in I/O space. The last value stored on the external data bus is read instead.

Note:

The default *emuinit.cmd* file for the ’C24x evaluation board (*f240init.cmd*) defines the LED registers as write-only memory.

Table 5–6 summarizes the function of each LED on the ’C24x evaluation board. Each entry in the table gives the LED name, LED number, and the resulting status when the LED is in a given state. Use this table as a reference when verifying the status of certain pins and/or I/O memory-mapped bits on the ’C24x evaluation board.

Table 5–6. LEDs on the TMS320C24x Evaluation Board

LED Name	LED	Status	Description
Power LED	DS11	On	Digital power is supplied to the board, and the power switch is in the on position.
		Off	Digital power is not supplied to the board, or the power switch is in the off position.
XF LED	DS9	On	The XF signal is set (or a logical 1).
		Off	The XF signal is cleared (or a logical 0).
BIO LED	DS10	On	The BIO signal is set (or a logical 1).
		Off	The BIO signal is cleared (or a logical 0).
I/O LED-0	DS1	On	Bit 0 of the I/O-mapped LED bank is set (or a logical 1).
		Off	Bit 0 of the I/O-mapped LED bank is set (or a logical 1).
I/O LED-1	DS2	On	Bit 1 of the I/O-mapped LED bank is set (or a logical 1).
		Off	Bit 1 of the I/O-mapped LED bank is cleared (or a logical 0).
I/O LED-2	DS3	On	Bit 2 of the I/O-mapped LED bank is set (or a logical 1).
		Off	Bit 2 of the I/O-mapped LED bank is cleared (or a logical 0).
I/O LED-3	DS4	On	Bit 3 of the I/O-mapped LED bank is set (or a logical 1).
		Off	Bit 3 of the I/O-mapped LED bank is cleared (or a logical 0).
I/O LED-4	DS5	On	Bit 4 of the I/O-mapped LED bank is set (or a logical 1).
		Off	Bit 4 of the I/O-mapped LED bank is cleared (or a logical 0).
I/O LED-5	DS6	On	Bit 5 of the I/O-mapped LED bank is set (or a logical 1).
		Off	Bit 5 of the I/O-mapped LED bank is cleared (or a logical 0).
I/O LED-6	DS7	On	Bit 6 of the I/O-mapped LED bank is set (or a logical 1).
		Off	Bit 6 of the I/O-mapped LED bank is cleared (or a logical 0).
I/O LED-7	DS8	On	Bit 7 of the I/O-mapped LED bank is set (or a logical 1).
		Off	Bit 7 of the I/O-mapped LED bank is cleared (or a logical 0).

5.8 Switches

There are ten switches on the ’C24x evaluation board. These switches are used to control the supply of power, reset the evaluation board, and to set and clear certain bits mapped into the I/O memory space.

The bank of eight DIP switches on the evaluation board is mapped to 0008h in the I/O memory space of the ’F240 device. Each of the eight switches can independently set or clear the bit assigned to it. The following instruction can be used to determine the status of the eight DIP switches by reading the eight LSBs in the DIPSW register, which is mapped to 0008h in I/O space:

```
IN     SW_STATUS,0008h    ;Read the value in 0008h and store
                          ;value in the SW_STATUS register
```

For the preceding example, if all the DIP switches are in the *on* position, then the value stored in the SW_STATUS register is 00FFh. Remember, the SW_STATUS variable is an uninitialized register that must be defined in the *.bss* section of your code. Keep in mind that the bank of eight I/O mapped DIP switches are considered read-only devices by the ’F240. Writing to the DIPSW register has no effect.

Table 5–7 summarizes the function of each switch on the ’C24x evaluation board. Each entry gives a switch name, switch number, and the status of the switch when it is in a given state.

Table 5–7. Switches on the TMS320C24x Evaluation Board

Switch Name	Switch	Status	Description
Power switch	SW3	On	Both digital and analog power are supplied to the board.
		Off	No power is supplied to the board.
Reset switch	SW1	In	The device is in reset (that is, the $\overline{RS}$ signal is pulled low).
		Out	The device is not in reset (that is, the $\overline{RS}$ signal is pulled high).
I/O switch-0	SW2-1	On	Bit 0 of the I/O-mapped DIP switch bank is set (or a logical 1).
		Off	Bit 0 of the I/O-mapped DIP switch bank is cleared (or a logical 0).
I/O switch-1	SW2-2	On	Bit 1 of the I/O-mapped DIP switch bank is set (or a logical 1).
		Off	Bit 1 of the I/O-mapped DIP switch bank is cleared (or a logical 0).
I/O switch-2	SW2-3	On	Bit 2 of the I/O-mapped DIP switch bank is set (or a logical 1).
		Off	Bit 2 of the I/O-mapped DIP switch bank is cleared (or a logical 0).
I/O switch-3	SW2-4	On	Bit 3 of the I/O-mapped DIP switch bank is set (or a logical 1).
		Off	Bit 3 of the I/O-mapped DIP switch bank is cleared (or a logical 0).
I/O switch-4	SW2-5	On	Bit 4 of the I/O-mapped DIP switch bank is set (or a logical 1).
		Off	Bit 4 of the I/O-mapped DIP switch bank is cleared (or a logical 0).
I/O switch-5	SW2-6	On	Bit 5 of the I/O-mapped DIP switch bank is set (or a logical 1).
		Off	Bit 5 of the I/O-mapped DIP switch bank is cleared (or a logical 0).
I/O switch-6	SW2-7	On	Bit 6 of the I/O-mapped DIP switch bank is set (or a logical 1).
		Off	Bit 6 of the I/O-mapped DIP switch bank is cleared (or a logical 0).
I/O switch-7	SW2-8	On	Bit 7 of the I/O-mapped DIP switch bank is set (or a logical 1).
		Off	Bit 7 of the I/O-mapped DIP switch bank is cleared (or a logical 0).

5.9 Oscillator

The socketed 10-MHz oscillator generates the clock-in signal for the ’F240 device on the ’C24x evaluation board when the clock-in jumper (JP4) is in position 1–2. The on-chip phase-locked loop (PLL) clock module can then be programmed to multiply the 10-MHz input frequency by a factor of 1, 1.5, or 2. This generates an on-chip CPU clock frequency of 10, 15, or 20-MHz (the rated CPU clock frequency of the device). The following code illustrates how to program the PLL clock module to generate a CPU clock (CPUCLK) frequency of 20-MHz and a system clock (SYSCLK) frequency of 10 MHz:

```
LDP    #00E0h
SPLK   #00BBh,CKCR1     ;CLKIN(OSC)=10MHz,CPUCLK=20MHz
SPLK   #00C3h,CKCR0     ;CLKMD=PLL Enable,SYSCLK=CPUCLK/2
SPLK   #40C0h,SYSCR     ;CLKOUT=CPUCLK
```

See the *TMS320C24x DSP Controllers Reference Set, Volume 2* for more information on programming the PLL clock module.

An off-board clock signal can also be used to drive the ’F240 device on the evaluation board. Place the clock-in jumper (JP4) in position 2–3 and connect the external clock source to pin 31 of the control connector (P4).

5.10 GAL Devices

Two socketed 16V8 GAL devices reside on the ’F240 ’C24x evaluation board. One GAL device (U7) has been factory programmed with the decode logic necessary to access the on-board peripherals, including the external SRAMs, DAC, LED bank, and DIP switchbank. The other GAL device (U14) has been factory programmed to control the board-level reset logic and DAC wait-state generation.

The GAL equations programmed in the factory (see Appendix C) can be modified by reprogramming the appropriate plastic leaded chip carrier (PLCC) GAL 16V8 device. See the ’C24x evaluation board schematic diagrams in Appendix A to determine the pin connections for each GAL device.

Only experienced users should attempt to reprogram GAL devices on the ’C24x evaluation board. Improper programming of the devices may result in board damage. Modifications to the evaluation board are not supported by Texas Instruments and void all warranties.

Chapter 6

Code Development Tools

The ’C1x/C2x/C2xx/C5x assembly language tools software package and the ’C24x EVM C source debugger are the two primary tools used in code development and are both included in the ’C24x EVM. Each of the two tools are summarized in this chapter. See the appropriate document, either the *TMS320C1x/C2x/C2xx/C5x Assembly Language Tools User’s Guide* or the *TMS320C2xx C Source Debugger User’s Guide* for complete information on the operation and function of each software tool.

Topic **Page**

6.1 Assembly Language Tools

The following assembly language tools support the ’C24x family of fixed-point DSPs and are included in the ’C24x EVM kit:

- ❑ Assembler
- ❑ Archiver
- ❑ Linker
- ❑ Absolute lister
- ❑ Cross-reference lister
- ❑ Hex conversion utility

This section gives a brief description of each assembly language tool included in the EVM kit and shows how these tools fit into the general software tools development flow. See the *TMS320C1x/C2x/C2xx/C5x Assembly Language Tools User’s Guide* for detailed information on each of the assembly language tools and complete instructions for invoking them.

6.1.1 Assembly Language Tools Descriptions

The following provides a brief description of each assembly language software tool:

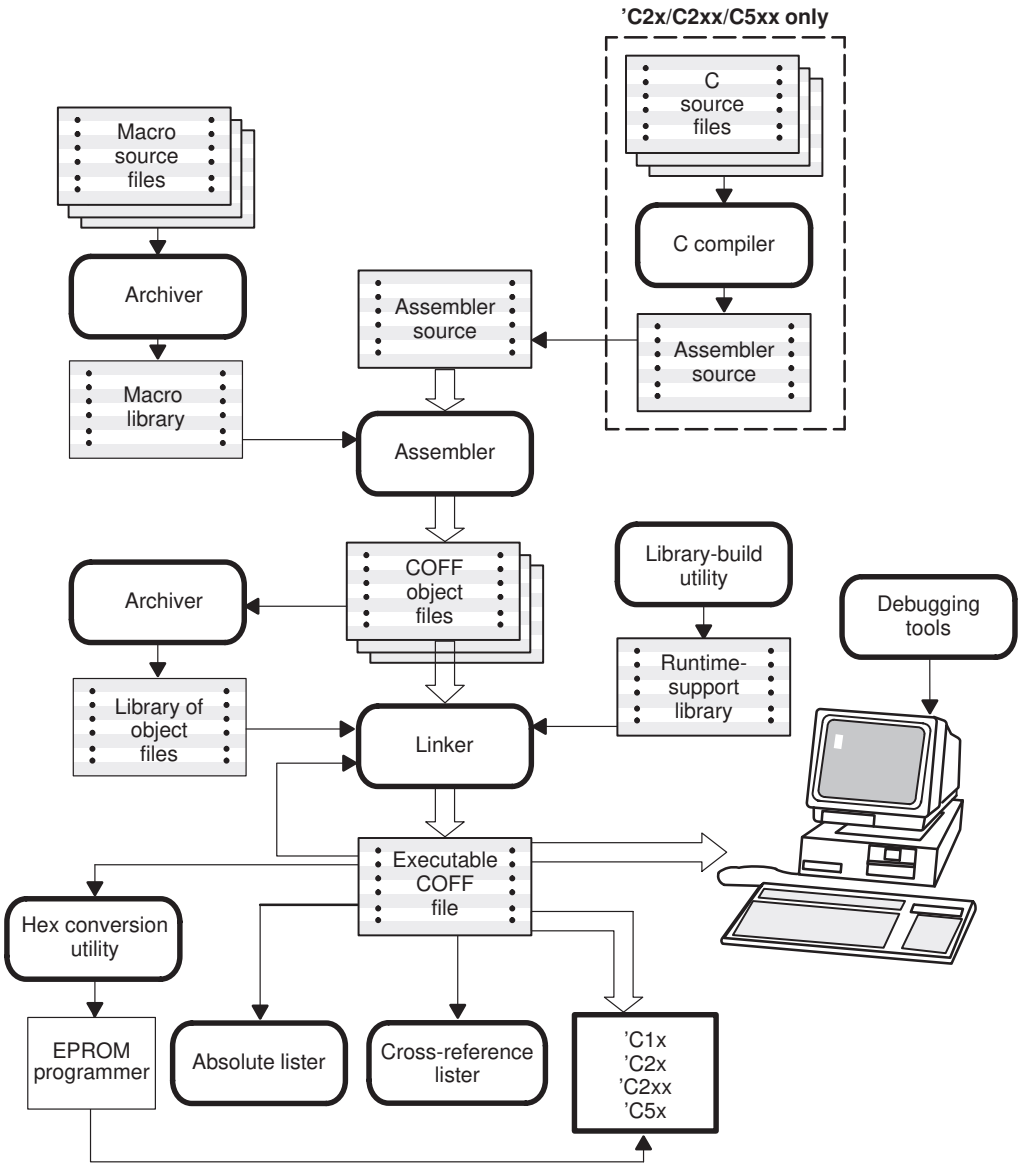
- ❑ The *assembler* translates assembly language source files into machine language object files. Source files can contain instructions, assembler directives, and macro directives. You can use assembler directives to control various aspects of the assembly process, such as the source listing format, data alignment, and section content.
- ❑ The *archiver* allows you to collect a group of files into a single archive file. For example, you can collect several macros into a macro library. The assembler searches the library and uses the members that are called as macros by the source file. You can also use the archiver to collect a group of object files into an object library. The linker includes in the library the members that resolve external references during the link.
- ❑ The *linker* combines object files into a single executable object module. As it creates the executable module, it performs relocation and resolves external references. The linker accepts relocatable object files (created by the assembler) as input. It also accepts archiver library members and output modules created by a previous linker run. Linker directives allow you to combine object file sections, bind sections or symbols to addresses or within memory ranges, and define or redefine global symbols.
- ❑ The *absolute lister* generates a file that can be reassembled to produce a listing of the absolute addresses of an object file.

- The *cross-reference lister* uses object files to produce a cross-reference listing showing symbols, their definitions, and their references in the linked source files.
- The ’C1x/C2x/C2xx/C5x debugging tools accept COFF files as input, but most EPROM programmers do not. The *hex conversion utility* converts a COFF object file into TI-Tagged, Intel, Motorola, or Tektronix object format. The converted file can be downloaded to an EPROM programmer.
- The *’C2x/C2xx/C5x C compiler* translates C source code into ’C2x/C2xx/C5x assembly language source code. The C compiler is *not* included in the ’F240 EVM package. Contact your local Texas Instruments distributor to purchase the C compiler.

6.1.2 Assembly Language Tools Overview and Development Flow

Figure 6–1 illustrates the assembly language development flow of the ’C1x/C2x/C2xx/C5x devices. The shaded portion highlights the most common development path; the other portions are optional.

Figure 6–1. TMS320C1x/C2x/C2xx/C5x Assembly Language Development Flow



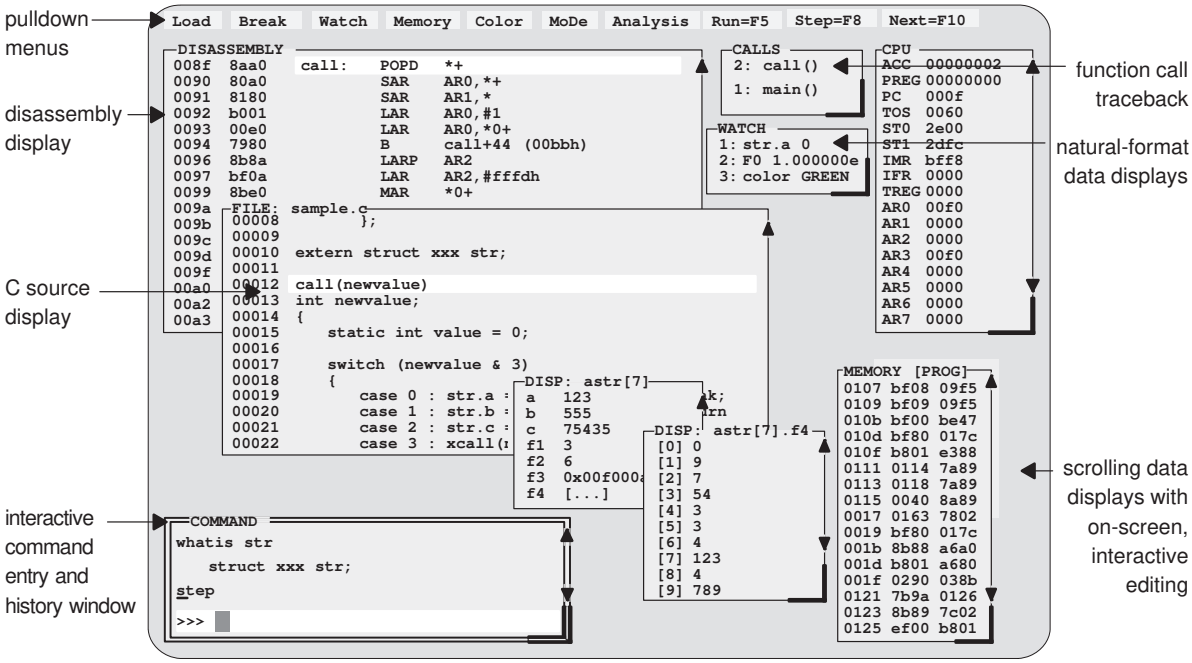
6.2 TMS320C24x EVM C Source Debugger

The ’C24x C source debugger is an advanced programmer’s interface that helps you to develop, test, and refine your programs in the language they were written. You can choose to debug your programs for the ’C24x family in C, assembly language, or both. Unlike many other debuggers, the ’C24x EVM debugger’s higher-level features are available even when you are debugging assembly language code.

Note:

The ’C1x/C2x/C2xx/C5x optimizing C compiler software package is required to write and debug your programs in C and is *not* included in the ’C24x EVM. The optimizing C compiler is a product of Texas Instruments and can be ordered through TI distribution channels.

Figure 6–2. The Basic Debugger Display



The ’C24x debugger is compatible only with ’C24x devices. This debugger is not compatible with devices outside of this family.

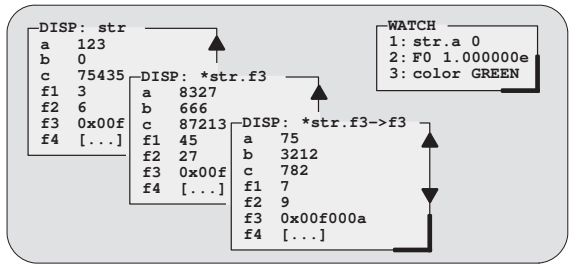
6.2.1 Key Features of the Debugger

The key features of the debugger are listed in the following bullets.

- **Multilevel debugging**. The debugger allows you to debug both C and assembly language code. If you are debugging a C program, you can choose to view just the C source, the disassembly of the object code created from the C source, or both. You can also use the debugger as an assembly language debugger.

- **Fully configurable, state-of-the-art, window-oriented interface.** The C source debugger separates code, data, and commands into manageable portions. Use any of the default displays or create your own by selecting the windows you want, sizing them, and moving them where you want.

- **Comprehensive data displays.** You can easily create windows for displaying *and editing* the values of variables, arrays, structures, pointers—any kind of data—in their natural format (float, int, char, enum, or pointer). You can even display entire linked lists.



- **On-screen editing.** Change any data value displayed in any window—just point the mouse, click, and type.

- **Continuous update.** The debugger continuously updates information on the screen, highlighting changed values.

- **Powerful command set.** Unlike many other debugging systems, this debugger does not force you to learn a large, intricate command set. The

'C2xx C source debugger supports a small but powerful command set that makes full use of C expressions. One debugger command performs actions that would take several commands in another system.

- **Flexible command entry.** There are a variety of ways to enter commands. You can type commands or use a mouse, function keys, or pulldown menus; choose the method that you like best. Want to reenter a command? No need to retype it—simply use the command history.


- **Create your own debugger.** The debugger display is completely configurable, allowing you to create the interface that is best suited for your use.
  - If you are using a color display, you can change the colors of any area on the screen.
  - You can change the physical appearance of display features, such as window borders.
  - You can interactively set the size and position of windows in the display.

  Create and save as many custom configurations as you like or use the defaults. Use the debugger with a color display or a black-and-white display. A color display is preferable; the various types of information on the display are easier to distinguish when they are highlighted with color.

- **Variety of screen sizes.** The debugger's default configuration is set up for a typical PC display, with 25 lines by 80 characters. If you use a sophisticated graphics card, you can take advantage of the debugger's additional screen sizes. A larger screen size allows you to display more information and provides you with more screen space for organizing the display—bringing the benefits of workstation displays to your PC.

- **All the standard features you expect in a world-class debugger.** The debugger provides you with complete control over program execution with features like conditional execution and single-stepping (including single-stepping into or over function calls). You can set or clear a breakpoint with a click of the mouse or by typing commands. You can define a memory map that identifies the portions of target memory that the debugger can access. You can choose to load only the symbol table portion of an object file to work with systems that have code in ROM. The debugger can execute commands from a batch file, providing you with an easy method for entering often-used command sequences.

Appendix A

TMS320C24x Evaluation Board Schematic Diagrams

This appendix contains the schematic diagrams for the ’C24x evaluation board.

Figure A–1. TMS320C24x Evaluation Board

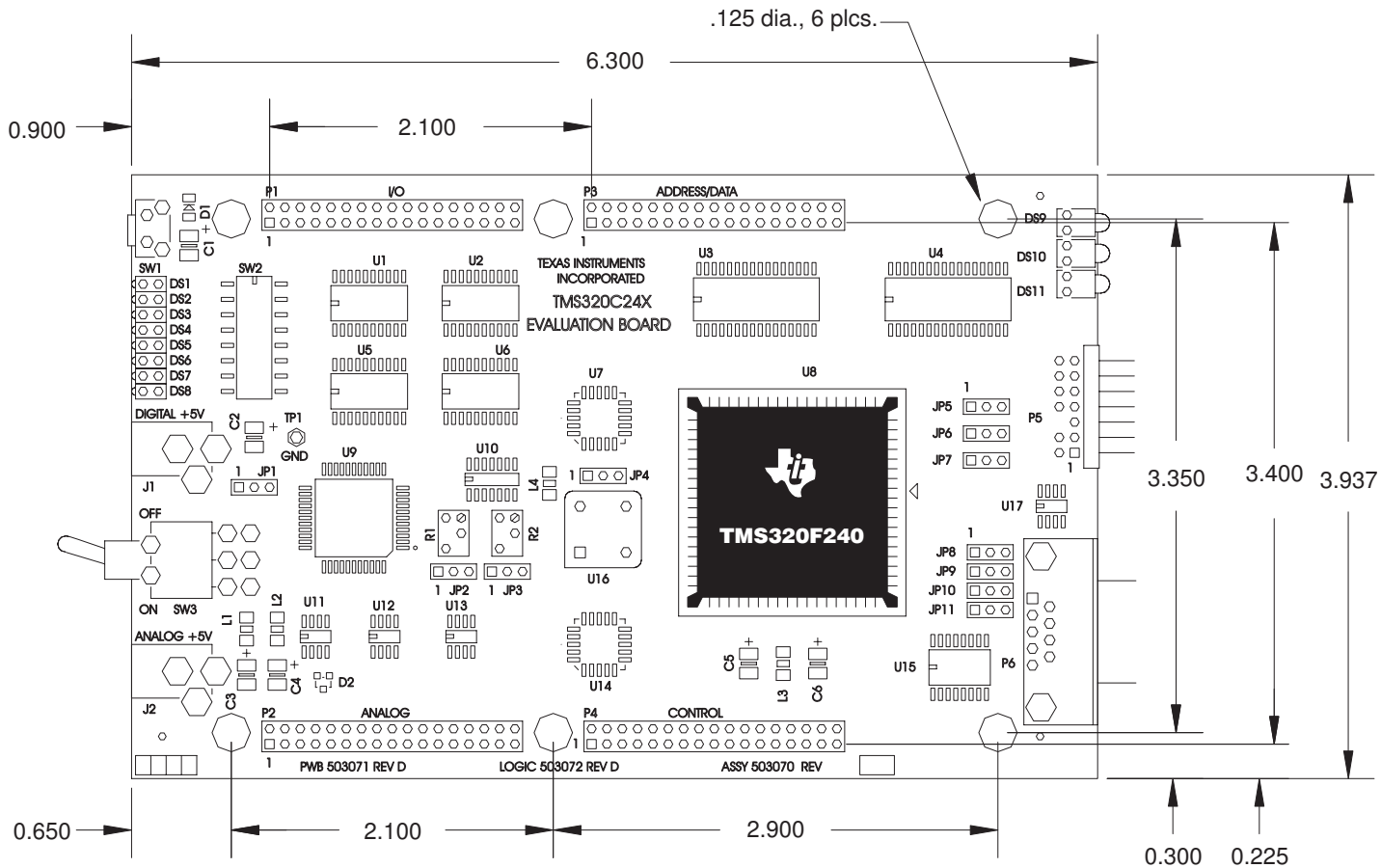


Figure A–2. Schematic Diagram of the TMS320C24x Evaluation Board

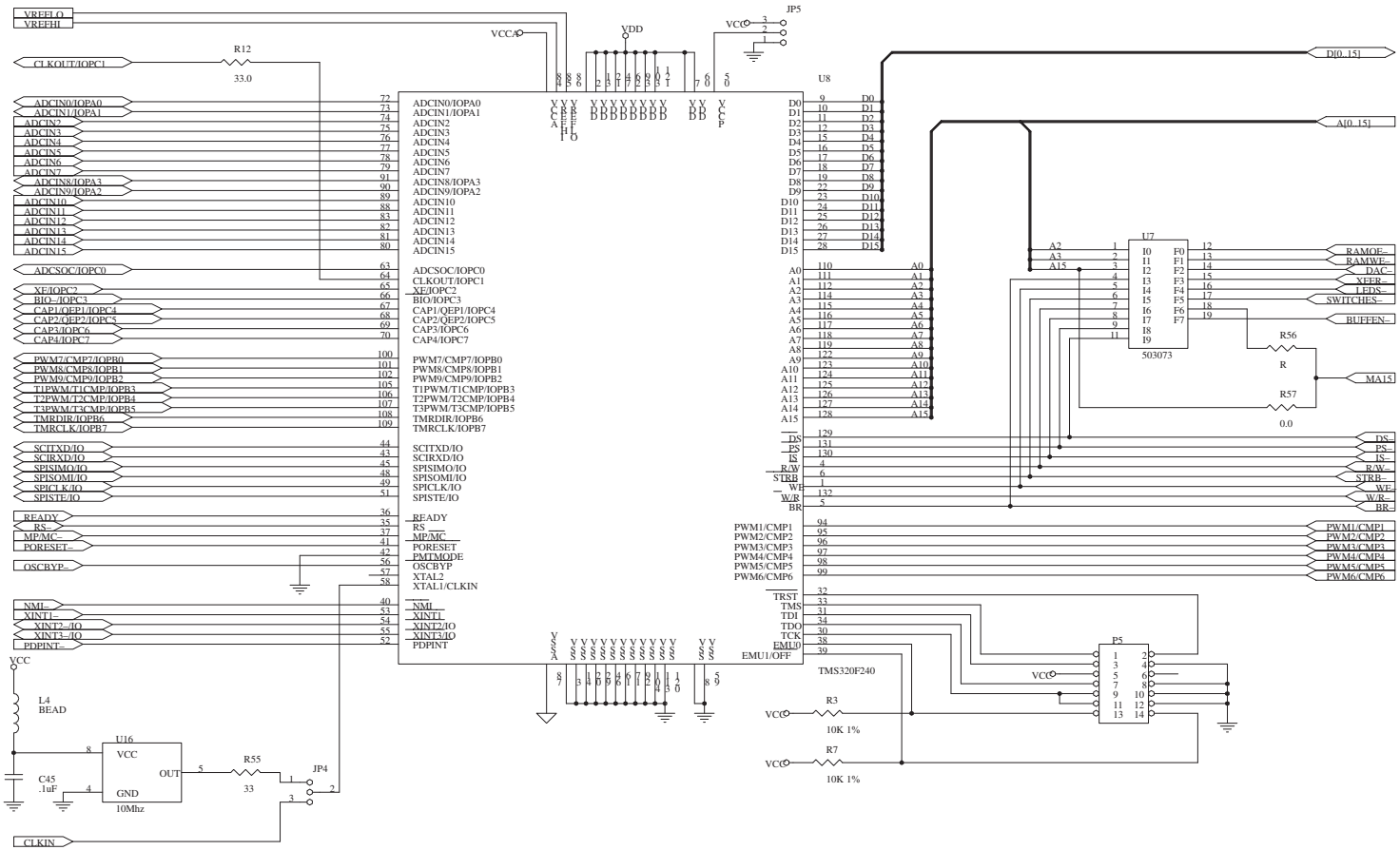


Figure A–2. Schematic Diagram of the TMS320C24x Evaluation Board (Continued)

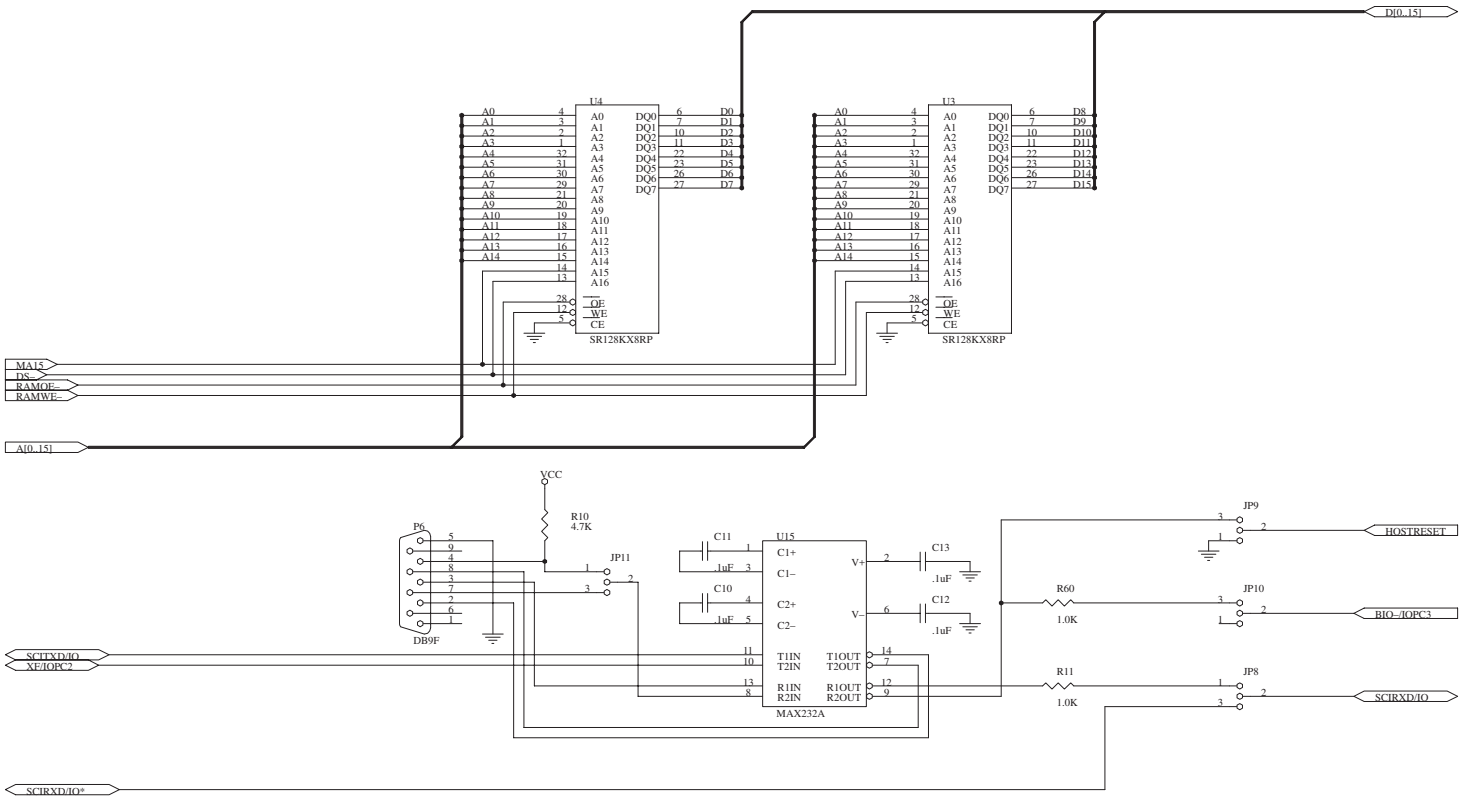


Figure A–2. Schematic Diagram of the TMS320C24x Evaluation Board (Continued)

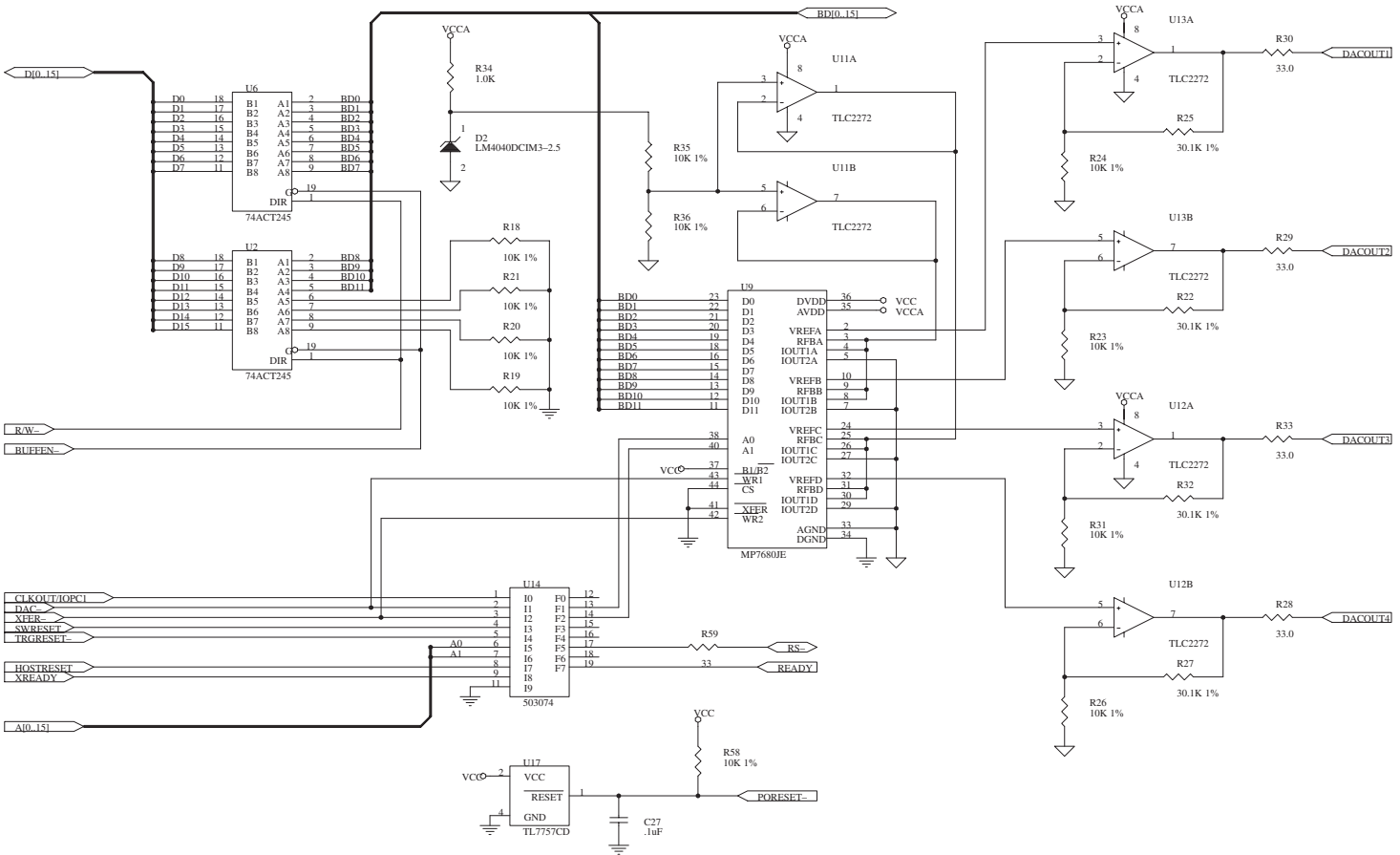


Figure A–2. Schematic Diagram of the TMS320C24x Evaluation Board (Continued)

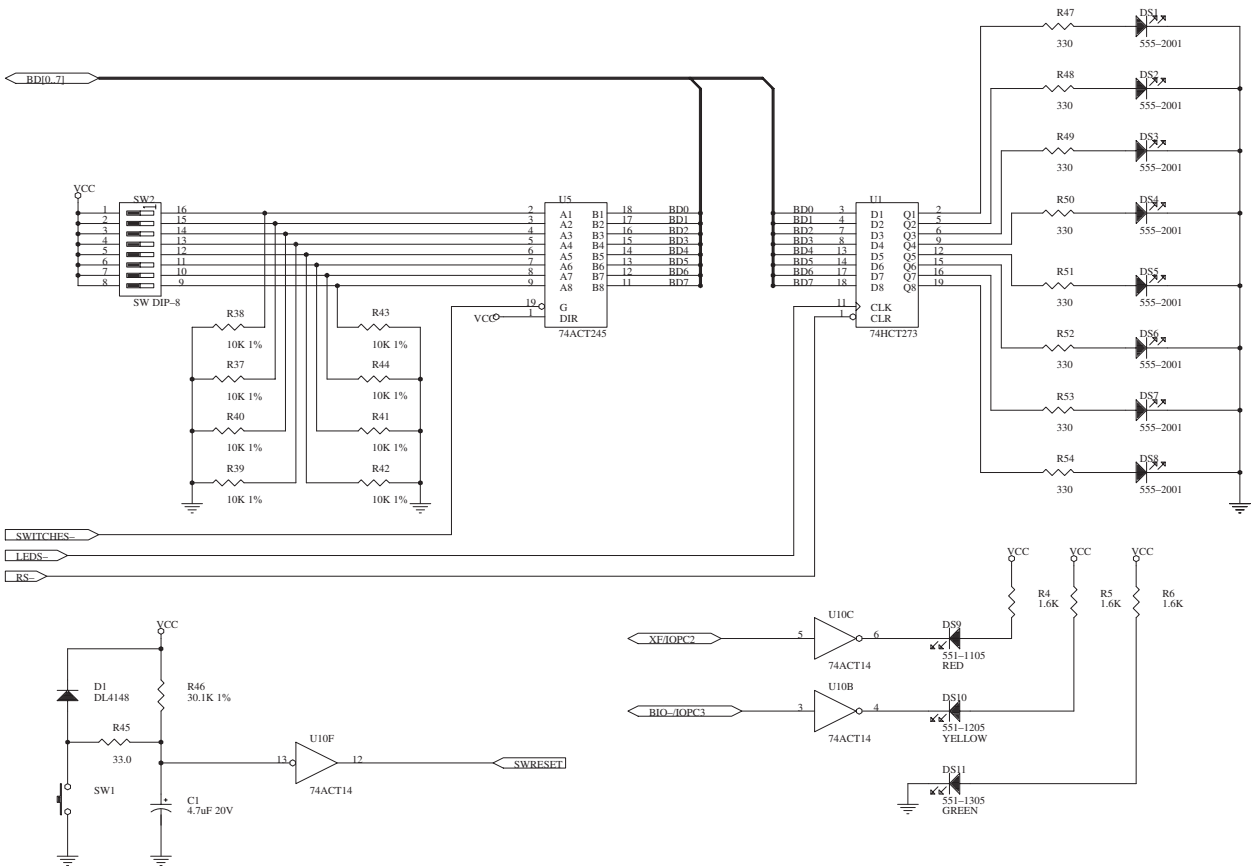


Figure A–2. Schematic Diagram of the TMS320C24x Evaluation Board (Continued)

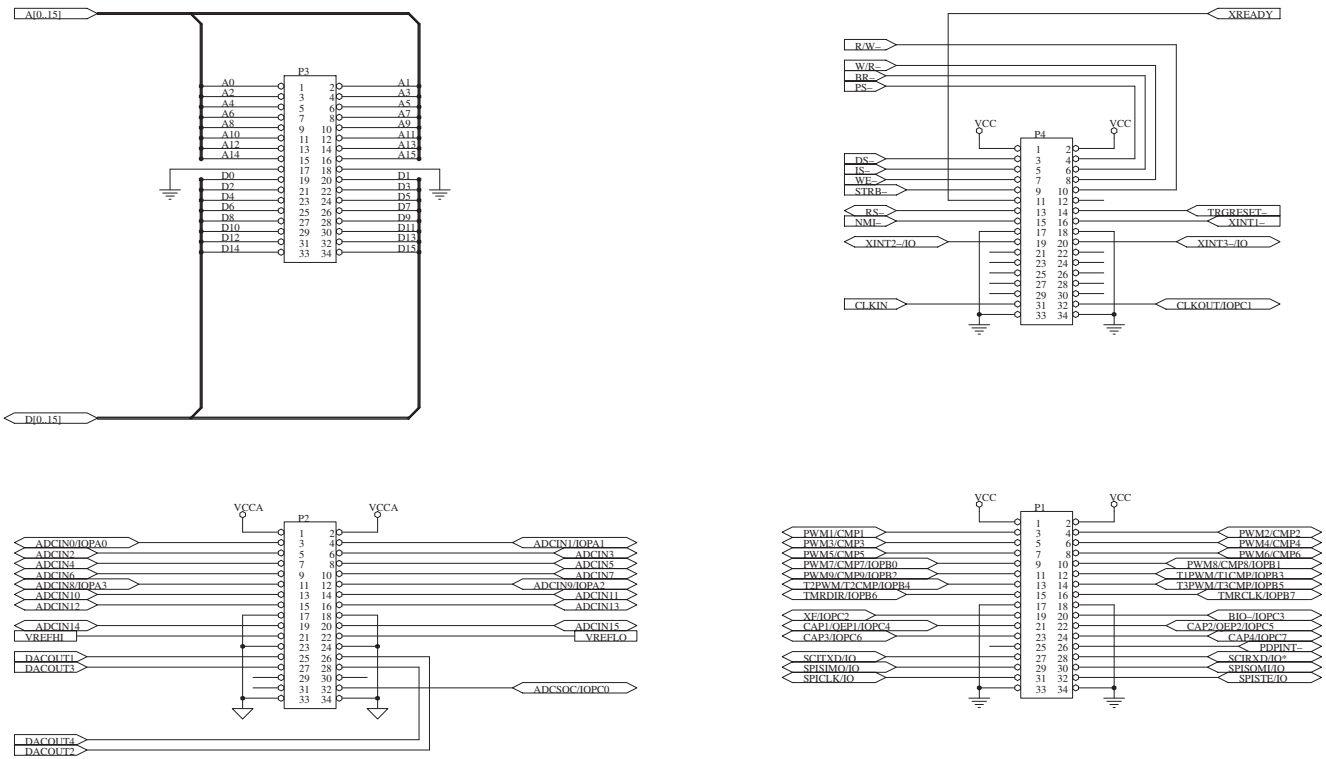
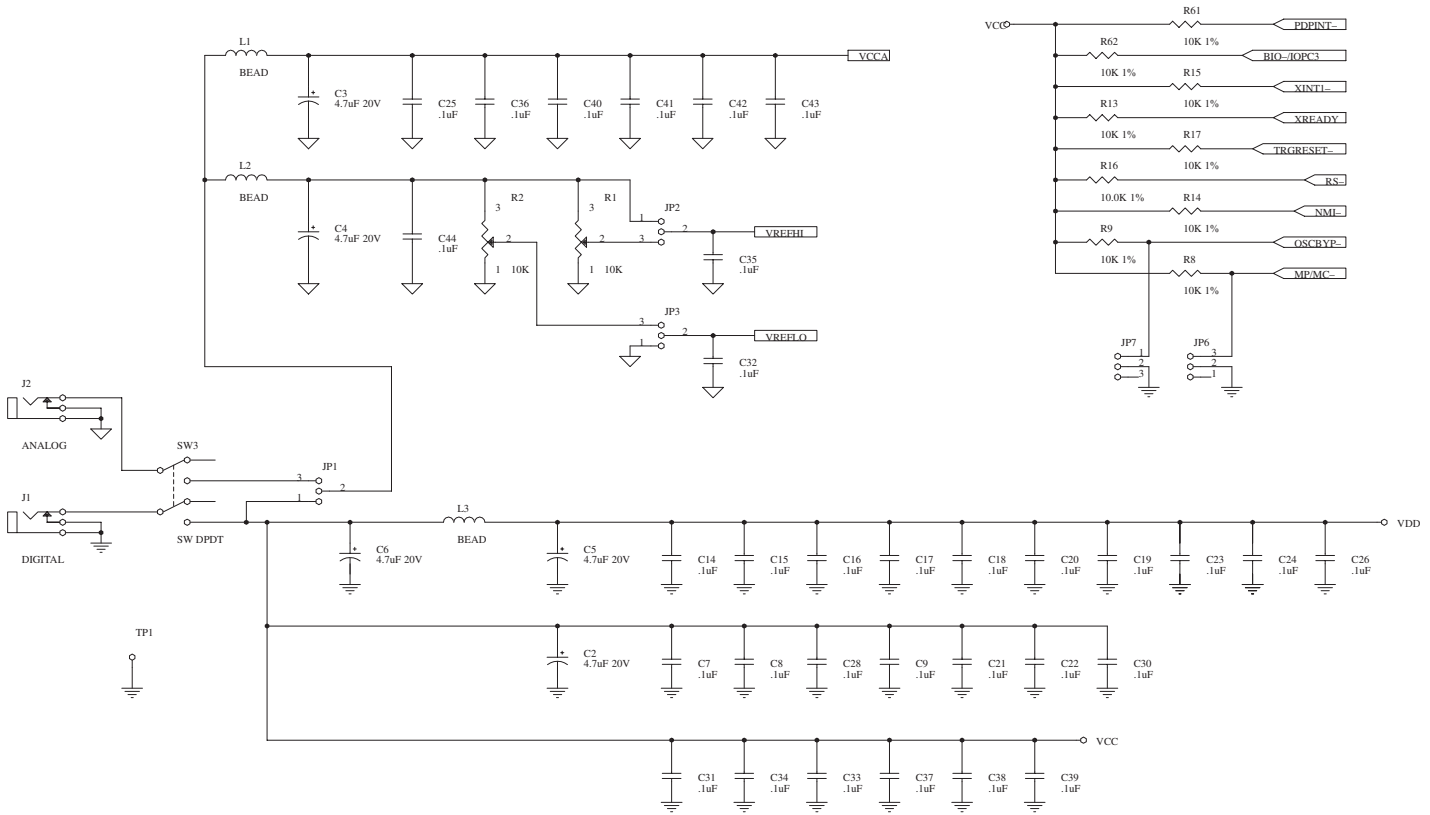


Figure A–2. Schematic Diagram of the TMS320C24x Evaluation Board (Continued)



Appendix B

Connector Signal Descriptions

This appendix describes the connector-accessible signals on the 'C24x evaluation board. The pin number, signal description, and the signal state relative to the target board are given in table format for each of the five connectors on the evaluation board.

Topic **Page**

B.1 I/O Connector Signal Descriptions

Table B–1. I/O Connector Signal Descriptions

Pin	Signal	Description	Type†
1, 2	V_{CC}	Digital power (5 V dc)	I
3	PWM1/CMP1	PWM/compare output generated by full compare units	O/Z
4	PWM2/CMP2	PWM/compare output generated by full compare units	O/Z
5	PWM3/CMP3	PWM/compare output generated by full compare units	O/Z
6	PWM4/CMP4	PWM/compare output generated by full compare units	O/Z
7	PWM5/CMP5	PWM/compare output generated by full compare units	O/Z
8	PWM6/CMP6	PWM/compare output generated by full compare units	O/Z
9	PWM7/CMP7	PWM/compare output generated by simple compare units	O/Z
	IOPB0	General-purpose I/O bit 0, port B	I/O
10	PWM0/CMP0	PWM/compare output generated by simple compare units	O/Z
	IOPB1	General-purpose I/O bit 1, port B	I/O
11	PWM9/CMP9	PWM/compare output generated by simple compare units	O/Z
	IOPB2	General-purpose I/O bit 2, port B	I/O
12	T1PWM/T1CMP	PWM/compare output generated by GPT1 compare unit	O/Z
	IOPB3	General-purpose I/O bit 3, port B	I/O
13	T1PWM/T2CMP	PWM/compare output generated by GPT2 compare unit	O/Z
	IOPB4	General-purpose I/O bit 4, port B	I/O
14	T2PWM/T3CMP	PWM/compare output generated by GPT3 compare unit	O/Z
	IOPB5	General-purpose I/O bit 5, port B	I/O
15	TMRDIR	Counter direction signal for GP timers	I
	IOPB6	General-purpose I/O bit 6, port B	I/O
16	TMRCLK	External clock input for GP timers	I
	IOPB7	General-purpose I/O bit 7, port B	I/O
17, 18	GND	Digital ground	
19	XF	External flag output pin	O
	IOPC2	General-purpose I/O bit 2, port C	I/O

† I = input, O = output, Z = high impedance

Table B–1. I/O Connector Signal Descriptions (Continued)

Pin	Signal	Description	Type†
20	$\overline{\text{BIO}}$	Branch control input	I
	IOPC3	General-purpose I/O bit 3, port C	I/O
21	CAP1/QEP1	Capture or QEP input 1	I
	IOPC4	General-purpose I/O bit 4, port C	I/O
22	CAP2/QEP2	Capture or QEP input 2	I
	IOPC5	General-purpose I/O bit 5, port C	I/O
23	CAP3	Capture input 3	O
	IOPC6	General-purpose I/O bit 6, port C	I/O
24	CAP4	Capture input	I
	IOPC7	General-purpose I/O bit 7, port C	I/O
25	NC	No connection	
26	$\overline{\text{PDPINT}}$	Maskable power-drive protection interrupt	I
27	SCITXD	SCI asynchronous serial port transmit data pin	O
	IO	General-purpose I/O	I/O
28	SCIRXD	SCI asynchronous serial port receive data pin	I
	IO	General-purpose I/O	I/O
29	SPISIMO	SPI slave in, master out pin	I/O
	IO	General-purpose I/O	I/O
30	SPISOMI	SPI slave out, master in pin	I/O
	IO	General-purpose I/O	I/O
31	SPICLK	SPI clock signal	I
	IO	General-purpose I/O	I/O
32	SPISTE	SPI transmit enable signal	I
	IO	General purpose I/O	I/O
33, 34	GND	Digital ground	

† I = input, O = output, Z = high impedance

B.2 Analog Connector Signal Descriptions

Table B–2. Analog Connector Signal Descriptions

Pin	Signal	Description	Type†
1, 2	V_{CCA}	Analog power (5 V dc)	I
3	ADCIN0	ADC input, channel 0	I
	IOPA0	General-purpose I/O bit 0, port A	I/O
4	ADCIN1	ADC input, channel 1	I
	IOPA1	General-purpose I/O bit 1, port A	I/O
5	ADCIN2	ADC input, channel 2	I
6	ADCIN3	ADC input, channel 3	I
7	ADCIN4	ADC input, channel 4	I
8	ADCIN5	ADC input, channel 5	I
9	ADCIN6	ADC input, channel 6	I
10	ADCIN7	ADC input, channel 7	I
11	ADCIN8	ADC input, channel 8	I
	IOPA3	General-purpose I/O bit 3, port A	I/O
12	ADCIN9	ADC input, channel 9	I
	IOPA2	General-purpose I/O bit 2, port A	I/O
13	ADCIN10	ADC input, channel 10	I
14	ADCIN11	ADC input, channel 11	I
15	ADCIN12	ADC input, channel 12	I
16	ADCIN13	ADC input, channel 13	I
17, 18	GNDA	Analog ground	I
19	ADCIN14	ADC input, channel 14	I
20	ADCIN15	ADC input, channel 15	I
21	$V_{ref\ hi}$	ADC analog high reference voltage	I
22	$V_{ref\ lo}$	ADC analog low reference voltage	I
23, 24t	GNDA	Analog ground	

† I = input, O = output, Z = high impedance

Table B–2. Analog Connector Signal Descriptions (Continued)

Pin	Signal	Description	Type†
25t	DACOUT0	DAC output, channel 0	O
26	DACOUT1	DAC output, channel 1	O
27	DACOUT2	DAC output, channel 2	O
28	DACOUT3	DAC output, channel 3	O
29, 30, 31	NC	No connection	
32	ADCSOC	External start of conversion signal for ADC	I
	IOPC0	General purpose I/O bit 0, port C	I/O
33, 34	GNDA	Analog ground	

† I = input, O = output, Z = high impedance

B.3 Address/Data Connector Signal Descriptions

Table B–3. Address/Data Connector Signal Descriptions

Pin	Signal	Description	Type†
1	A0	Bit 0 of external address bus	O/Z
2	A1	Bit 1 of external address bus	O/Z
3	A2	Bit 2 of external address bus	O/Z
4	A3	Bit 3 of external address bus	O/Z
5	A4	Bit 4 of external address bus	O/Z
6	A5	Bit 5 of external address bus	O/Z
7	A6	Bit 6 of external address bus	O/Z
8	A7	Bit 7 of external address bus	O/Z
9	A8	Bit 8 of external address bus	O/Z
10	A9	Bit 9 of external address bus	O/Z
11	A10	Bit 10 of external address bus	O/Z
12	A11	Bit 11 of external address bus	O/Z
13	A12	Bit 12 of external address bus	O/Z
14	A13	Bit 13 of external address bus	O/Z
15	A14	Bit 14 of external address bus	O/Z
16	A15	Bit 15 of external address bus	O/Z
17, 18	GND	Digital ground	
19	D0	Bit 0 of external data bus	I/O/Z
20	D1	Bit 1 of external data bus	I/O/Z
21	D2	Bit 2 of external data bus	I/O/Z
22	D3	Bit 3 of external data bus	I/O/Z
23	D4	Bit 4 of external data bus	I/O/Z
24	D5	Bit 5 of external data bus	I/O/Z
25	D6	Bit 6 of external data bus	I/O/Z
26	D7	Bit 7 of external data bus	I/O/Z

† I = input, O = output, Z = high impedance

Table B–3. Address/Data Connector Signal Descriptions (Continued)

Pin	Signal	Description	Type†
27	D8	Bit 8 of external data bus	I/O/Z
28	D9	Bit 9 of external data bus	I/O/Z
29	D10	Bit 10 of external data bus	I/O/Z
30	D11	Bit 11 of external data bus	I/O/Z
31	D12	Bit 12 of external data bus	I/O/Z
32	D13	Bit 13 of external data bus	I/O/Z
33	D14	Bit 14 of external data bus	I/O/Z
34	D15	Bit 15 of external data bus	I/O/Z

† I = input, O = output, Z = high impedance

B.4 Control Connector Signal Descriptions

Table B–4. Control Connector Signal Descriptions

Pin	Signal	Description	Type†
1, 2	V_{CC}	Digital power (5 V dc)	O/Z
2	V_{CC}	Digital power (5 V dc)	O/Z
3	$\overline{DS}$	Data memory select signal	O/Z
4	$\overline{PS}$	Program memory select signal	O/Z
5	$\overline{IS}$	I/O memory select signal	O/Z
6	BR	Bus request signal	O/Z
7	$\overline{WE}$	Write enable signal	O/Z
8	W/$\overline{R}$	Write-not-read signal	O/Z
9	$\overline{STRB}$	External memory access active strobe	O/Z
10	R/$\overline{W}$	Read-not-write signal	O/Z
11	READY‡	Memory ready to complete cycle	I
12	NC	No connection	
13	$\overline{RS}$‡	’F240 device reset	I/O
14	$\overline{TRGRESET}$‡	’C24x evaluation	I
15	$\overline{NMI}$‡	Nonmaskable interrupt	I
16	XINT1‡	Maskable external interrupt 1	I
17, 18	GND	Digital ground	I
19	XINT2	Maskable external interrupt 2	I
	IO	General-purpose I/O	I/O
20	XINT3	Maskable external interrupt 3	I
	IO	General-purpose I/O	I/O
21-30	NC	No connection	

† I = input, O = ouput, Z = high impedance
‡ Indicates that there is a 10-kΩ pullup resistor on this signal

Table B–4. Control Connector Signal Descriptions (Continued)

Pin	Signal	Description	Type†
31	CLKIN	Clock source input	I
32	CLKOUT	Clock output	O
	IOPC1	General-purpose I/O, ort C	I/O
33, 34	GND	Digital ground	

† I = input, O = ouput, Z = high impedance
‡ Indicates that there is a 10-kΩ pullup resistor on this signal

B.5 14-Pin Emulation Port Signal Descriptions

Table B–5. 14-Pin Emulation Port Signal Descriptions

Pin	Signal	Description	Type†
1	TMS	Test mode select	I
2	$\overline{\text{TRST}}$	Test reset	I
3	TDI	Test data input	I
4, 8, 10, 12	GND	Ground	
5	PD (V_{CC})	Presence detect. Indicates that the emulation cable is connected and that the target is powered up. PD is tied to V_{CC} on the evaluation board.	O
7	TDO	Test data output	O
9	TCK_RET	Test clock return. Test clock input to the emulator	O
11	TCK	Test clock. TCK is a 10.368-MHz clock source from the emulation cable pod.	I
13	EMU0	Emulation pin 0	I/O
14	EMU1	Emulation pin 1	I/O

† I = input; O = output

Appendix C

GAL Equations

The equations and associated test vectors for the two factory programmed GAL devices (U7 and U14) are included in this appendix. A brief description of each equation is provided.

Topic **Page**

C.1 TMS320C24x Evaluation Board Peripheral Decode Logic

This section contains the equations and associated test vectors for the GAL device U7. These are responsible for the peripheral decode logic on the 'C24x evaluation board. A brief description of each equation follows:

RAMOE Controls the read from external memory operation. This signal is active when either $\overline{DS}$ or $\overline{PS}$ signals are active (or = 0), R/$\overline{W}$ = 1, $\overline{STRB}$ = 0, and $\overline{BR}$ = 1. When this signal is active, the 'F240 DSP can read from external memory.

RAMWE Controls the write to external memory operation. This signal is active when either $\overline{DS}$ or $\overline{PS}$ signals are active (or = 0), $\overline{WE}$ = 0, and $\overline{BR}$ = 1. When this signal is active, the 'F240 DSP can write to external memory.

DAC Enables a value to be written to one of the four DAC channels. This signal becomes active when a write to one of the DAC channels mapped into I/O space (0000h–0003h) occurs.

XFER Enables the DAC to begin converting the digital values stored in each of the four DAC channels. This signal becomes active when a value is written to the DAC update register mapped into I/O space (0004h).

SWITCHES Enables the octal switch buffer. This signal becomes active when a read from the switch register mapped into I/O space (0008h) occurs.

LEDS Clocks the octal LED flip-flop. This signal clocks latches the value written to the LEDS register mapped into I/O space (000Ch).

BUFFEN Enables the I/O-mapped peripheral data bus buffer. This signal becomes active when a value is either written to or read from an I/O-mapped peripheral (DAC, switches, or LEDs).

Example C–1. Peripheral Decode GAL Equation Routine

```
/*
**     File:         503073a.tdl
**     Title:        'C24x Evaluation Board Peripheral Decode
**     Device:       16V8 GAL® (U7)
**     Company:      Texas Instruments
**     Modified:     25 April 1997
*/
503073(in     A2,           /* DSP address a2              */
              A3,           /* DSP address a3              */
              A15,          /* DSP address a15             */
              Rw,           /*     R/W-                    */
              !We,          /*     WE-                     */
              !Ps,          /*     PS-                     */
              !Ds,          /*     DS-                     */
              !Is,          /*     IS-                     */
              !Strb,        /*     STRB-                   */
              !Br;          /*     BR-                     */
       out
              !Ramoe,       /* SRAM Ouput Enable           */
              !Buffen;      /* I/O Buffer Enable           */
       io
              !Ramwe,       /* SRAM Write Enable           */
              !Dac,         /* DAC Register Write          */
              !Xfer,        /* DAC Output Transfer         */
              !Leds,        /* LED Write Strobe            */
              !Switches;    /* Switch Read Strobe          */
      )
{
```

Example C–1. Peripheral Decode GAL Equation Routine (Continued)

```
/* Uncomment next line for test vectors */
/* #define TEST_VEC */
/* Define Address Ranges */

   #define DAC            ( Is & !A15 & !A3 & !A2 )
   #define XFER           ( Is & !A15 & !A3 &  A2 )
   #define SWITCHES       ( Is & !A15 &  A3 & !A2 )
   #define LEDS           ( Is & !A15 &  A3 &  A2 )

/* Output enables   */

   Ramoe.oe         = 1;
   Ramwe.oe         = 1;
   Dac.oe           = 1;
   Xfer.oe          = 1;
   Leds.oe          = 1;
   Switches.oe      = 1;
   Buffen.oe        = 1;

/* Equations        */

   Ramoe     =   (( Ds | Ps ) & Rw & Strb & !Br );
   Ramwe     =   (( Ds | Ps ) & We & !Br );
   Dac       =   ( DAC & We );
   Xfer      =   ( XFER & We );
   Switches  =   ( SWITCHES & Rw & Strb );
   Leds      =   ( LEDS & We );
   Buffen    =   (( DAC | LEDS | SWITCHES ) & Strb );
```

Example C–1. Peripheral Decode GAL Equation Routine (Continued)

```
/* Part assignment */
   putpart(”g16v8”, ”503073a”,
      A2, A3, A15, Br, We, Strb, Rw, Is, Ps, GND,
      Ds, Ramoe, Ramwe, Dac, Xfer, Leds, Switches, _, Buffen, VCC );
#ifdef TEST_VEC
/* Test Vectors */
   test( Br, Strb, Rw, We, Ds, Ps, Is, A15, A3, A2 =>
      Ramoe, Ramwe, Dac, Xfer, Leds, Switches, Buffen )
   {
/* Test Ram */
( 1, 1, 0, 1, 1, 1, 1, 0, 0, 0 => 1, 1, 1, 1, 1, 1, 1 );
( 1, 0, 0, 0, 1, 0, 1, 0, 0, 0 => 1, 0, 1, 1, 1, 1, 1 ); /* Ram Data Write */
( 1, 0, 0, 0, 0, 1, 1, 0, 0, 0 => 1, 0, 1, 1, 1, 1, 1 ); /* Ram Prog Write */
( 1, 0, 1, 1, 1, 0, 1, 0, 0, 0 => 0, 1, 1, 1, 1, 1, 1 ); /* Ram Data Read  */
( 1, 0, 1, 1, 0, 1, 1, 0, 0, 0 => 0, 1, 1, 1, 1, 1, 1 ); /* Ram Prog Read  */
/* I/O Tests */
( 1, 1, 0, 1, 1, 1, 1, 0, 0, 0 => 1, 1, 1, 1, 1, 1, 1 );
( 1, 0, 0, 0, 1, 1, 0, 0, 0, 0 => 1, 1, 0, 1, 1, 1, 0 ); /* Write Dac’s   */
( 1, 0, 0, 0, 1, 1, 0, 0, 0, 1 => 1, 1, 1, 0, 1, 1, 1 ); /* Update Dac’s */
( 1, 0, 1, 1, 1, 1, 0, 0, 1, 0 => 1, 1, 1, 1, 1, 0, 0 ); /* Read Switches*/
( 1, 0, 0, 0, 1, 1, 0, 0, 1, 1 => 1, 1, 1, 1, 0, 1, 0 ); /* Write LED’s   */
}
#endif
}
```

C.2 TMS320C24x Evaluation Board Reset Logic and DAC Wait-State Generator

This section contains the equations and associated test vectors for the GAL device U14. These are responsible for the reset logic and the DAC wait-state generator on the ’C24x evaluation board. A brief functional description of each equation follows:

LA0 Controls address line 0 for DAC channel selection. If a value is written to one of the DAC channels, then the value of A0 on the external address bus is written to the A0 channel selection bit of the DAC. If the DAC channels are not accessed, then the value last written to the A0 channel selection bit of the DAC is latched.

LA1 Controls address line 1 for DAC channel selection. If a value is written to one of the DAC channels, then the value of A1 on the external address bus is written to the A1 channel selection bit of the DAC. If the DAC channels are not accessed, then the value last written to the A1 channel selection bit of the DAC is latched.

RSEN Enables a board-level reset. This signal becomes active when the on-board reset button is pressed, the HOSTRESET signal is activated via the serial port, or trgreset is activated via the control connector P4.

RS Controls the board-level reset function. This signal is always in the active state; however, the output of this signal is only enabled when the RSEN signal is active. Therefore, a board-level reset only occurs when the output of this signal is enabled (RSEN is active).

STATE Generates the wait states required to properly access the DAC. The DAC has an access time of 108 ns. Therefore, with a 20-MIPS ’F240 device, a simple state machine is used to generate the three wait states required for proper access to the DAC.

Example C–2. Reset Logic and DAC Wait-State Generator GAL Equation Routine

```
/*
** File:         503074c.tdl
** Title:        'C24x Evaluation Board Reset Logic & DAC
**               Wait-state Generator
** Device:       16V8 GAL® (U14)
** Company:      Texas Instruments
** Modified:     25 April 1997
*/
503074(in     Clk,
              A0,           /* DSP address a0                */
              A1,           /* DSP address a1                */
              !Dac,         /* DAC Write Strobe              */
              !Xfer,        /* DAC Write Strobe              */
              !TrgReset,    /* Target Board Reset            */
              !Reset,       /* Onboard Reset (Button)        */
              Xready,
              HostReset,    /* Onboard Reset (Serial Port)   */
              Oe;
       out    !Rs;
       reg    State0,
              State1,
              State2,
              Ready;        /* Uart Write Strobe             */
       io
              RsEn,
              La0,          /* Latched A0                    */
              La1;          /* Latched A1                    */
      )
{
```

Example C–2. Reset Logic and DAC Wait-State Generator GAL Equation Routine (Continued)

```
group State[Ready, State2, State1, State0];

  /*   States
                              +------- ready
                              |+----- state2
                              ||+---- state1
                              |||+--- state0
                              ||||      */

   #define IDLE           0b1111
   #define CYCLE1         0b0111       /* 50ns   */
   #define CYCLE2         0b0110       /* 100ns  */
   #define CYCLE3         0b0101       /* 150ns  */
   #define COMPLETE       0b1100
   #define START ( Dac | Xfer )

/* Uncomment next line for test vectors */

/* #define TEST_VEC */

/* Output enables */

   La0.oe       = 1;
   La1.oe       = 1;
   RsEn.oe      = 1;

   Rs.oe        = RsEn;

   State0.oe    = !Oe;

   State1.oe    = !Oe;

   State2.oe    = !Oe;

   Ready.oe     = !Oe;

/* Hook up the clocks */

   State0.ck    = Clk;

   State1.ck    = Clk;

   State2.ck    = Clk;

   Ready.ck     = Clk;
```

Example C–2. Reset Logic and DAC Wait-State Generator GAL Equation Routine (Continued)

```
/* Equations */
   La0       =   (  Dac & A0 ) | ( !Dac & La0 );
   La1       =   (  Dac & A1 ) | ( !Dac & La1 );
   RsEn      =   ( Reset | TrgReset | HostReset );
   Rs        = 1;
   switch(State)
   {
   case IDLE:
             if( START )
                    State = CYCLE1;
                else
                    State = IDLE;
             break;
   case CYCLE1:
             if ( !START )
                    State = IDLE;
                else
                    State = CYCLE2;
             break;
   case CYCLE2:
             if ( !START )
                    State = IDLE;
                else
                    State = CYCLE3;
             break;
   case CYCLE3:
             if ( !START )
                    State = IDLE;
                else if ( Xready )
                    State = COMPLETE;
                else
                    State = CYCLE3;
             break;
   case COMPLETE:
             State = IDLE;
             break;
   default:
             State = IDLE;
   }
```

Example C–2. Reset Logic and DAC Wait-State Generator GAL Equation Routine (Continued)

```
/* Part assignment */
   putpart(”g16v8”, ”503074c”,
      Clk, Dac, Xfer, Reset, TrgReset, A0, A1, HostReset, Xready, GND,
      Oe, State0, La0, La1, State1, RsEn, Rs, State2, Ready, VCC );
#ifdef TEST_VEC
   /* Test Vectors */
   test( Reset, TrgReset, Dac, A1, A0 => La0, La1 )
   {
      /* Put in known state */
      ( 1, 1, 0, 0, 0 => 0, 0 );
      /* Test Reset */
      ( 1, 0, 1, 0, 0 => 0, 0 );       /* Onboard Reset       */
      ( 0, 1, 1, 0, 0 => 0, 0 );       /* Target Reset        */
      /* Test Latch */
      ( 1, 1, 1, 0, 0 => 0, 0 );       /* Latch closed        */
      ( 1, 1, 1, 1, 1 => 0, 0 );       /* Holding old value   */
      ( 1, 1, 0, 1, 1 => 1, 1 );       /* Open Latch          */
      ( 1, 1, 1, 1, 1 => 1, 1 );       /* Close latch         */
      ( 1, 1, 1, 0, 0 => 1, 1 );       /* Holding old value   */
   }
#endif
}
```

Appendix D

Parallel Port Utility Programs

This appendix discusses several utility programs included with the 'C24x EVM C source debugger software. These programs help identify and initialize the parallel port on your PC. This is important because the XDS510PP emulator communicates through the parallel port. Some PCs come with a parallel port configuration utility on a separate diskette or built into the BIOS display. However, some PCs are shipped without any port configuration utilities. These users must use the programs described in this appendix to obtain the maximum data transfer rate.

Topic **Page**

D.1 Port Detection Utility Program (*portchk.exe*)

The port detection utility program (*portchk.exe*) identifies the parallel ports in your PC. The utility scans the default system addresses and attempts to configure each of the parallel ports detected.

To run the *portchk.exe* utility, type the program name at a system prompt followed by a carriage return. An example is shown below:

Example D–1. Running the portchk.exe Utility

```
C:\C24XHLL\portchk.exe

Bidirectional Parallel Port Detection Program Version 1.03
(c) Copyright 1994, Spectrum Digital Inc.

LPT1 @ 378-> Detected     Bidirectional
LPT2 @ 278-> Not Detected
LPT3 @ 3BC-> Not Detected
```

In this example, the *portchk.exe* utility indicates that a parallel port has been detected at address 378h. It has been configured for either SPP8 or EPP mode, both of which are bidirectional.

If no bidirectional ports are detected by the *portchk.exe* utility, it does not necessarily mean that there are no bidirectional parallel ports in your PC. The parallel ports may not be configured for bidirectional operation. To configure the parallel ports, run the SMC (*smcmode.exe*) and NSC port configuration (*nscmode.exe*) utilities. After configuring the parallel ports, run the *portchk.exe* utility again to verify the operation and presence of a bidirectional port.

D.2 SMC Port Configuration Utility (*smcmode.exe*)

Use the SMC port configuration utility (*smcmode.exe*) to configure the parallel port function of an SMC device. To run the *smcmode.exe* utility, type the program name at a system prompt, followed by a carriage return. An example is shown below:

Example D–2. Running the smcmode.exe Utility

```
C:\C24XHLL\smcmode.exe

SMC Port Configuration Program Version 1.10
(c) Copyright 1994, 1995, 1996 Spectrum Digital Inc.

SMC FDC37C666 Detected, Port is at 378
MODE 4 -> Port is in standard non bidirectional mode
Status = ff
Control = ff
```

In this example, the *smcmode.exe* utility indicates that a parallel port has been detected at address 378h and that it has been configured for standard nonbidirectional mode (SPP4).

If an SMC device is not detected, the utility reports this with a message similar to the following:

```
SMC Port Configuration Program Version 1.10
(c) Copyright 1994, 1995, 1996 Spectrum Digital Inc.

SMC FDC37C66X Not Detected
```

You can use the –h option on the command line to list all of the utility options available, as in the following example:

Example D–3. Displaying the smcmode.exe Utility Options

```
C:\C24XHLL\smcmode.exe -h

SMC Port Configuration Program Version 1.10
(c) Copyright 1994, 1995, 1996 Spectrum Digital Inc.

 smcmode [options]
   -h    help
   -r    reset status register
   -m x  set to mode x=0..4

Mode 0 -> Standard bidirectional mode
Mode 1 -> EPP and SPP
Mode 2 -> ECP mode
Mode 3 -> ECP and EPP mode
Mode 4 -> Non bidirectional mode
```

If an SMC device is present in the system, *mode 1* is recommended. The device can be configured for this mode by following these steps:

1) Configure the SMC chip for *mode1* by entering the following at a system prompt:

```
C:\C24XHLL\smcmode.exe -m 1
```

2) Run the *smcmode.exe* utility without any options to verify that the port is configured correctly.

3) Run the *portchk.exe* utility to verify that the port is again bidirectional.

When operating in EPP mode, an error condition can cause the parallel port to no longer appear in bidirectional mode. You can verify this by running the *portchk.exe* utility.

To return the parallel port back to its original configuration, the error must be cleared by resetting the SMC device. Do this by entering the following command at a system prompt:

```
C:\C24XHLL\smcmode.exe -r
```

Run the *portchk.exe* utility to verify that the port is again bidirectional.

If you use the SMC utility to configure the parallel port of your PC, run this utility every time you turn the PC on. This ensures that the parallel port is configured correctly for the XDS510PP emulator.

D.3 NSC Port Configuration Utility (*nscmode.exe*)

Use the NSC port configuration utility (*nscmode.exe*) to configure the parallel port function of the an NSC device. To run the n*scmode.exe* utility, type the program name at a system prompt, followed by a carriage return. An example is shown below:

Example D–4. Running the nscmode.exe Utility

```
C:\C24XHLL\nscmode.exe

NSC Port Configuration Program Version 1.10
(c) Copyright 1996 Spectrum Digital Inc.

No National Device Found
```

In this example, no National devices are detected.

If a National device is detected, the utility reports this with a message indicating the address of the parallel port and the mode the device is currently operating in.

You can use the –h option on the command line to list all of the utility options available as in the following example:

Example D–5. Displaying the nscmode.exe Utility Options

```
C:\C24XHLL\nscmode.exe -h

NSC Port Configuration Program Version 1.10
(c) Copyright 1996 Spectrum Digital Inc.

nscmode [options]
   -h    help
   -m x  set to mode x=0..4

Mode 0 -> Compatible mode
Mode 1 -> Extended mode
Mode 2 -> EPP mode
Mode 3 -> ECP
Mode 4 -> ECP and EPP mode
```

If a National device is present in the system, *mode 2* is recommended. Follow these steps to configure the NSC device for *mode 2*:

1) Enter the following command from a system prompt:

   ```
   C:\C24XHLL\nscmode.exe -m 2
   ```

2) Run the n*scmode.exe* utility without any options to verify that the port has been configured correctly.

3) Run the *portchk.exe* utility to verify that the port has been reset and is configured for bidirectional operation.

If you use the NSC utility to configure the parallel port of your PC, run this utility every time you turn the PC on. This ensures that the parallel port is configured correctly for the XDS510PP emulator.

Appendix E

Glossary

A

analog-to-digital converter (ADC): A successive-approximation converter with internal sample-and-hold circuitry used to translate an analog signal to a digital signal.

archiver: A software program that allows you to collect several individual files into a single file called an archive library. The archiver also allows you to delete, extract, or replace members of the archive library, as well as to add new members.

assembler: A software program that creates a machine-language program from a source file that contains assembly language instructions, directives, and macro directives. The assembler substitutes absolute operation codes for symbolic operation codes, and absolute or relocatable addresses for symbolic addresses.

B

.bss section: One of the default COFF sections. You can use the .bss directive to reserve a specified amount of space in the memory map that can later be used for storing data. The .bss section is uninitialized.

byte: A sequence of eight adjacent bits operated upon as a unit.

C

common object file format (COFF): A system of object files configured according to a standard developed by AT&T. These files are relocatable in memory space.

command file: A file created by the user which names initialization options and input files for the linker or the debugger.

COMMAND window: An area of the debugger display where you can enter commands and for the debugger to echo command entry, show command output, and list progress or error messages.

comment: A source statement (or portion of a source statement) that is used to document or improve readability of a source file. Comments are not assembled.

connector: A coupling device used to connect conductors of one circuit or transmission element with those of another circuit or transmission element.

constant: A fixed or invariable value or data item.

controller: Logic circuitry that decodes instructions, manages the pipeline, stores the status of operations, and decodes conditional operations.

CPU window: A window that displays the contents of ’C24x on-chip registers, including the program counter, status register, A-file registers, and B-file registers.

D

D_DIR: An environment variable that identifies the directory containing the commands and files necessary for running the debugger.

D_OPTIONS: An environment variable that you can use for identifying often-used debugger options.

D_SRC: An environment variable that identifies directories containing program source files.

debugger: A window-oriented software interface that helps to debug ’C24x programs running on a ’C24x emulator or simulator.

disassembly: The process of translating the contents of memory from machine language to assembly language. Also known as reverse assembly.

digital signal processor (DSP): DSPs process or manipulate digital signals, which are discrete or discontinuous electrical impulses.

E

emulator: A device or computer system that imitates another system. The imitating system accepts the same data, executes the same programs, and achieves the same results as the imitated system.

emurst: A utility that resets the emulator.

environment variable: A special system symbol that the debugger uses for finding directories or obtaining debugger options.

evaluation board: A hardware platform that allows the user to evaluate a specific device.

evaluation module: Tools and documentation provided to new users to evaluate a product.

event manager: A module in the ’C24x family that supports motor control applications.

F

Flash memory: Electronically erasable and programmable, nonvolatile (read-only) memory.

G

GREG: See *global memory allocation register (GREG)*.

general array logic (GAL): Programmable logic device created by Lattice Semiconductor Corporation which defines logical equations that model a systems inputs and outputs.

global-memory allocation register (GREG): An 8-bit memory-mapped register that specifies the size of the global memory space. At reset, the GREG is cleared.

H

hex conversion utility: A program that accepts COFF files and converts them into one of several standard ASCII hexadecimal formats suitable for loading into an EPROM programmer.

host: A device to which other devices (peripherals) are connected and that generally controls those devices.

I

interrupt: A signal sent to the CPU that (when not masked) forces the CPU into a subroutine called an interrupt service routine. This signal can be triggered by an external device, an on-chip peripheral, or an instruction (INTR, NMI, or TRAP).

J

Joint Test Action Group (JTAG): A group that designed a testability standard sanctioned by IEEE (IEEE Standard 1149.1).

jumper: A conductive tool used to maintain electrical continuity across equipment.

L

light-emitting diode (LED): A semiconductor chip that gives off visible or infrared light when activated.

linker: A software tool that combines object files to form an object module that can be allocated into TMS320C6200 system memory and executed by the device.

Least significant bit (LSB): The binary digit, or bit, in a binary number that has the least influence on the value of the number.

M

memory map: A diagram of target system memory space that is partitioned into functional blocks.

MEMORY window: A display area that shows the contents of memory.

Most significant bit (MSB): The binary digit, or bit, in a binary number that has the most influence on the value of the number.

O

object file: A set of related records treated as a unit that is the output of an assembler or compiler and is input to a linker.

options: Command parameters that allow you to request additional or specific functions when you invoke a software tool.

oscillator: A device that produces or maintains variations in the magnitude of a quantity above and below a reference value.

P

PC: Personal computer or program counter, depending on context and where it's used. In this book, in installation instructions, or in information relating to hardware and boards, PC means personal computer (as in IBM PC). In general debugger and program-related information, PC means program counter, which is the register that identifies the current statement in your program.

R

reference voltage: Voltage used as a standard for measurement, usually the nominal full scale of the computer.

S

serial communications interface: A high-speed asynchronous serial input/output (I/O) port. It allows a serial bit stream of programmed length (one to eight bits) to be shifted into and out of the device at a programmed bit-transfer rate.

serial-port interface: An on-chip full-duplex serial port interface that provides direct serial communication to serial devices with a minimum of external hardware, such as codecs and serial analog-to-digital (A/D) converters. Status and control of the serial port is specified in the serial port control register (SPC).

source file: A file that contains C code or assembly language code that is compiled or assembled to form an object file.

switch: A device that opens, closes, or changes the connection of a circuit.

T

target system: A board that has not been designed to work with a specific system or program; it uses an emulator to translate the instructions for its use.

W

wait state: A period of time that the CPU must wait for an external program, data, or I/O memory to respond when reading from or writing to that external memory. The CPU waits one extra cycle for every wait state.

wait-state generator: A program that can be modified to generate a limited number of wait states for a given off-chip memory space (lower program, upper program, data, or I/O).

window: Portion of a debugger's display that allows the user to enter commands, echoes commands already entered, and displays the status of the device.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X