TEXAS INSTRUMENTS

TMS320C4x Parallel Processing Development System

Technical Reference

1993 Digital Signal Processing Products

Texas Instruments

Printed in U.S.A., August 1993
2576298-9761 revision C

SPRU075A

Book Type
Two Lines
Volume #

Book Type
Volume #

Book Type
Two Lines

Book Type

Title
Two Lines
Subtitle
Line Two

year

Title
Two Lines
Subtitle

Title
Two Lines

Title
Subtitle
Line Two

Title
Subtitle

Title

year

TMS320C4x Parallel Processing Development System Technical Reference

SPRU075A
August 1993

Texas Instruments

IMPORTANT NOTICE

Texas Instruments Incorporated (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to current specifications 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.

Please be aware that TI products are not intended for use in life-support appliances, devices, or systems. Use of TI product in such applications requires the written approval of the appropriate TI officer. Certain applications using semiconductor devices may involve potential risks of personal injury, property damage, or loss of life. In order to minimize these risks, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards. Inclusion of TI products in such applications is understood to be fully at the risk of the customer using TI devices or systems.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI 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.

WARNING

This equipment is intended for use in a laboratory test environment only. It generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to subpart J of part 15 of FCC rules, which are designed to provide reasonable protection against radio frequency interference. Operation of this equipment in other environments may cause interference with radio communications, in which case the user at his own expense will be required to take whatever measures may be required to correct this interference.

Preface

Read This First

About This Manual

This technical reference describes the features, design, and operation of the TMS320C4x Parallel Processing Development System (PPDS) with four TMS320C40 devices installed. Use this manual in conjunction with the *TMS320C4x C Source Debugger User's Guide* for installation and setup instructions.

How to Use This Manual

This document contains the following chapters:

Chapter 1 Introduction
Gives an overview of the TMS320C4x Parallel Processing Development System (PPDS).

Chapter 2 How the TMS320C4x PPDS Works
Provides step-by-step procedures for installing and removing TMS320C4x devices; describes and explains the TMS320C4x Parallel Processing Development System (PPDS), its key components and how they operate, and the various interfaces.

Chapter 3 TMS320C4x PPDS PLD Equations
Describes the ABEL source files and reduced equations that were developed for programming the programmable logic devices (PLDs) that are used on the TMS320C4x PPDS.

Appendix A TMS320C4x PPDS Connectors
Gives the pin assignments and explains the major signals available for the expansion bus connector (P3) and the external communication port connectors P5–P12.

Appendix B TMS320C4x PPDS Schematics
Contains the schematics for the TMS320C4x PPDS.

Appendix C Glossary
Defines acronyms and key terms used in this book.

Notational Conventions

This document uses the following conventions.

- The TMS320C40 processor is referred to as the **'C4x**.
- Program listings, program examples, interactive displays, filenames, and symbol names are shown in a special typeface similar to a typewriter's. Here is an example:

```
# !bg_ & !busenable_ & busrdy_ & !busreq_
       & start_state & !strb0_
# !busenable_ & busrdy_ & !busreq_ & !priDMA_
       & start_state & !strb0_
# !bg_ & !busenable_ & busrdy_ & !busreq_
       & start_state & !stat2 & stat3
```

Information About Cautions

This is an example of a caution statement.

A caution statement describes a situation that could potentially damage your software or equipment.

CAUTION

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

Related Documentation From Texas Instruments

The following books describe the TMS320C4x devices and related support tools. To obtain a copy of any of these TI documents, call the Texas Instruments Literature Response Center at (800) 477–8924. When ordering, please identify the book by its title and literature number.

Details on Signal Processing is a quarterly newsletter that provides information about new TMS320 family products, updates on development tools and new documentation, and similar items.

TMS320 Family Development Support Reference Guide (literature number SPRU011) details the vast development support available from Texas Instruments for the TMS320 family of digital signal processors.

TMS320 Floating-Point DSP Assembly Language Tools User's Guide (literature number SPRU035) 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 'C3x and 'C4x generations of devices.

TMS320 Floating-Point DSP Optimizing C Compiler User's Guide (literature number SPRU034) describes the TMS320 floating-point C compiler. This C compiler accepts ANSI standard C source code and produces TMS320 assembly language source code for the 'C3x and 'C4x generations of devices.

TMS320 Third-Party Support Reference Guide (literature number SPRU052) describes a number of independent vendor-supplied products that augment the TMS320 support provided by Texas Instruments.

TMS320C4x Technical Brief (literature number SPRU076) has a condensed overview of the TMS320C40 processor, its development tools, and a listing of TMS320C4x third parties.

TMS320C4x User's Guide (literature number SPRU063) describes the 'C4x 32-bit floating-point processor, developed for digital signal processing as well as parallel processing applications. Covered are its architecture, internal register structure, instruction set, pipeline, specifications, and operation of its six DMA channels and six communication ports. Software and hardware applications are included.

TMS320C4x C Source Debugger User's Guide (literature number SPRU054) tells you how to invoke the 'C4x 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, and includes a tutorial that introduces basic debugger functionality.

If You Need Assistance. . .

If you want to. . .	Do this. . .
Request more information about Texas Instruments Digital Signal Processing (DSP) products	Write to: Texas Instruments Incorporated Market Communications Manager, MS 736 P.O. Box 1443 Houston, Texas 77251–1443
Order Texas Instruments documentation	Call the TI Literature Response Center: **(800) 477–8924**
Ask questions about product operation or report suspected problems	Call the DSP hotline: **(713) 274–2320** **FAX: (713) 274–2324**
Report mistakes in this document or any other TI documentation	Send your comments to: Texas Instruments Incorporated Technical Publications Manager, MS 702 P.O. Box 1443 Houston, Texas 77251–1443

Trademarks

AMP is a trademark of AMP Incorporated.

MS-DOS is a registered trademark of Microsoft Corp.

PAL® is a registered trademark of Advanced Micro Devices, Inc.

PC-DOS and OS/2 are trademarks of International Business Machines Corp.

Contents

Figures

Tables

Chapter 1

Introduction

The TMS320C4x Parallel Processing Development System (PPDS) is the first development system designed exclusively to evaluate and develop parallel-processing, floating-point applications. You can develop, benchmark, and evaluate code (in realtime) in a rich development environment, using the power and speed of the TMS320C4x PPDS.

Other development tools that are available from Texas Instruments to support TMS320C4x design include:

- Code generation tools, such as
  - The TMS320 floating-point DSP optimizing C compiler, which provides a library of parallel-processing, run-time support functions, and
  - Assembly language tools,
- The XDS510 parallel-processing in-circuit emulator, and
- The TMS320C4x C source debugger.

Additional development tools are available from third parties. Refer to the *TMS320C4x Technical Brief* or the *TMS320 Third-Party Support Reference Guide* for details.

Topic **Page**

1.1 Key Features of the TMS320C4x PPDS

Key features of the TMS320C4x PPDS include:

- Four on-board TMS320C40 parallel digital signal processors. Each TMS320C40 is supported by a local bus comprising:
  - 64K x 32-bit words of zero wait-state static RAM (SRAM)
  - 8K bytes of EPROM
- 128K x 32-bit words of one wait-state SRAM on a shared global bus.
- An expansion bus connector (P3) that provides an external interface to the shared global memory bus.
- Eight external communication connectors (P5–P12) that provide an interface for connecting off-board TMS320C40s and external peripherals to the PPDS's TMS320C40s.
- A JTAG test connector (P4) that provides an interface for connecting the XDS510 in-circuit emulator to the TMS320C4x PPDS.
- Support for TMS320C40s with speeds less than or equal to 32 MHz for shared memory access.
- Support for TMS320C40s with speeds less than or equal to 50 MHz when the shared memory is not used.

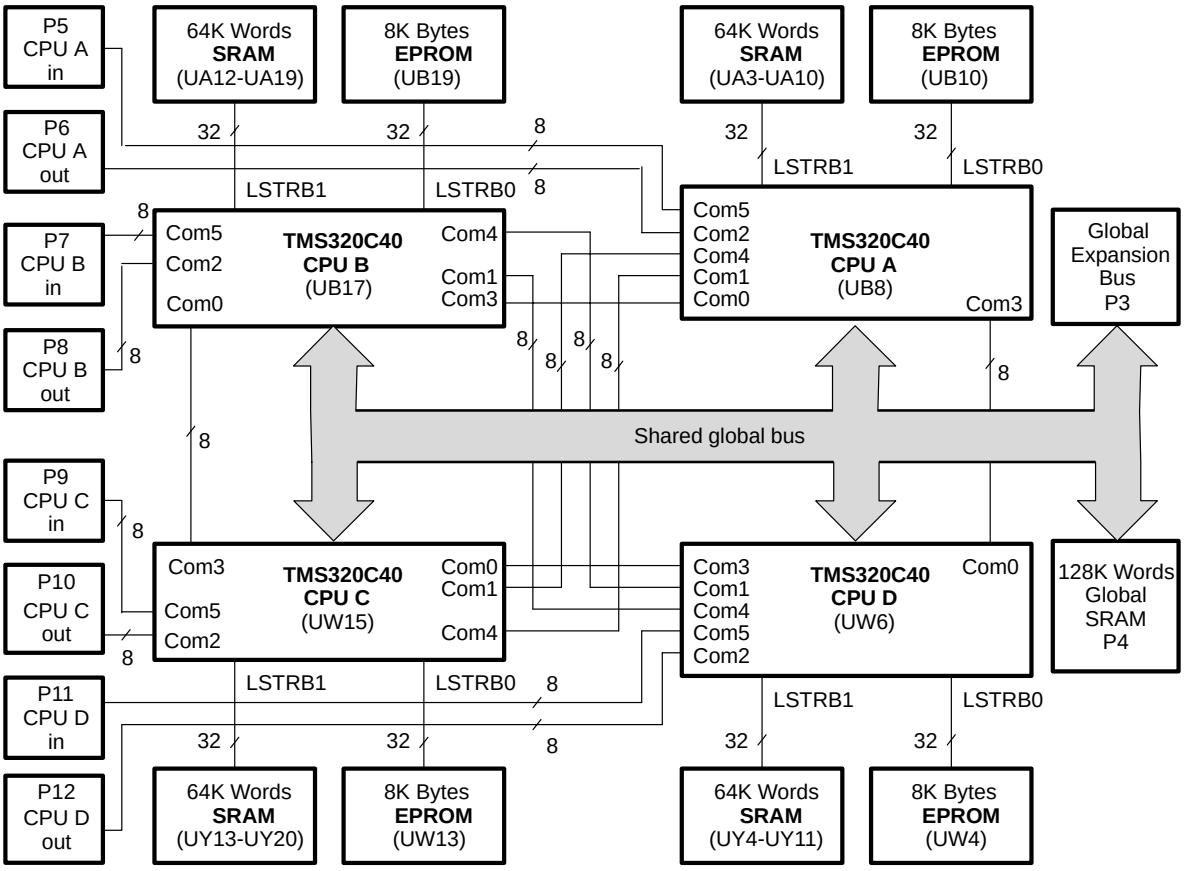
Note:

The PPDS shared-memory interface is designed for TMS320C40s running at speeds of **less than or equal to 32 MHz**. If the PPDS shared memory is not used, TMS320C40s with speeds as fast as 50 MHz can be used.

1.2 A Functional Overview of the TMS320C4x PPDS

Figure 1–1 shows the basic block diagram and interconnections of the TMS320C4x PPDS. The interconnects include the external global and communication port interfaces, local memory buses, and the shared global bus.

Figure 1–1. TMS320C4x PPDS Block Diagram



1.3 The TMS320C40 Parallel Processor

The TMS320C4x generation of floating-point processors is designed specifically to meet the needs of parallel processing and other realtime embedded applications.

Key features of the TMS320C40 include:

- Six communication ports (Com0–Com5) for high-speed interprocessor communication.
- Six-channel DMA coprocessor for concurrent I/O and CPU operation, maximizing sustained CPU performance by alleviating the CPU of burdensome I/O operations.
- High-performance CPU capable of 320 Mbytes-per-second throughput and 275 MOPS.
- Two identical external data and address buses that support shared memory systems and high data-rate, single-cycle transfers.
- On-chip program cache and dual-access, single-cycle RAM for increased memory access performance.
- Separate internal program, data, and DMA coprocessor buses for supporting massive concurrent I/O program/data throughput.
- JTAG interface for standard system connection.
- An on-chip analysis module that supports efficient, state-of-the-art parallel-processing debug.

Figure 1–2 shows the basic block diagram and interconnections for the TMS320C40. For more detailed information about the TMS320C4x generation of floating-point processors, refer to the *TMS320C4x User's Guide*.

Figure 1–2. TMS320C40 Block Diagram

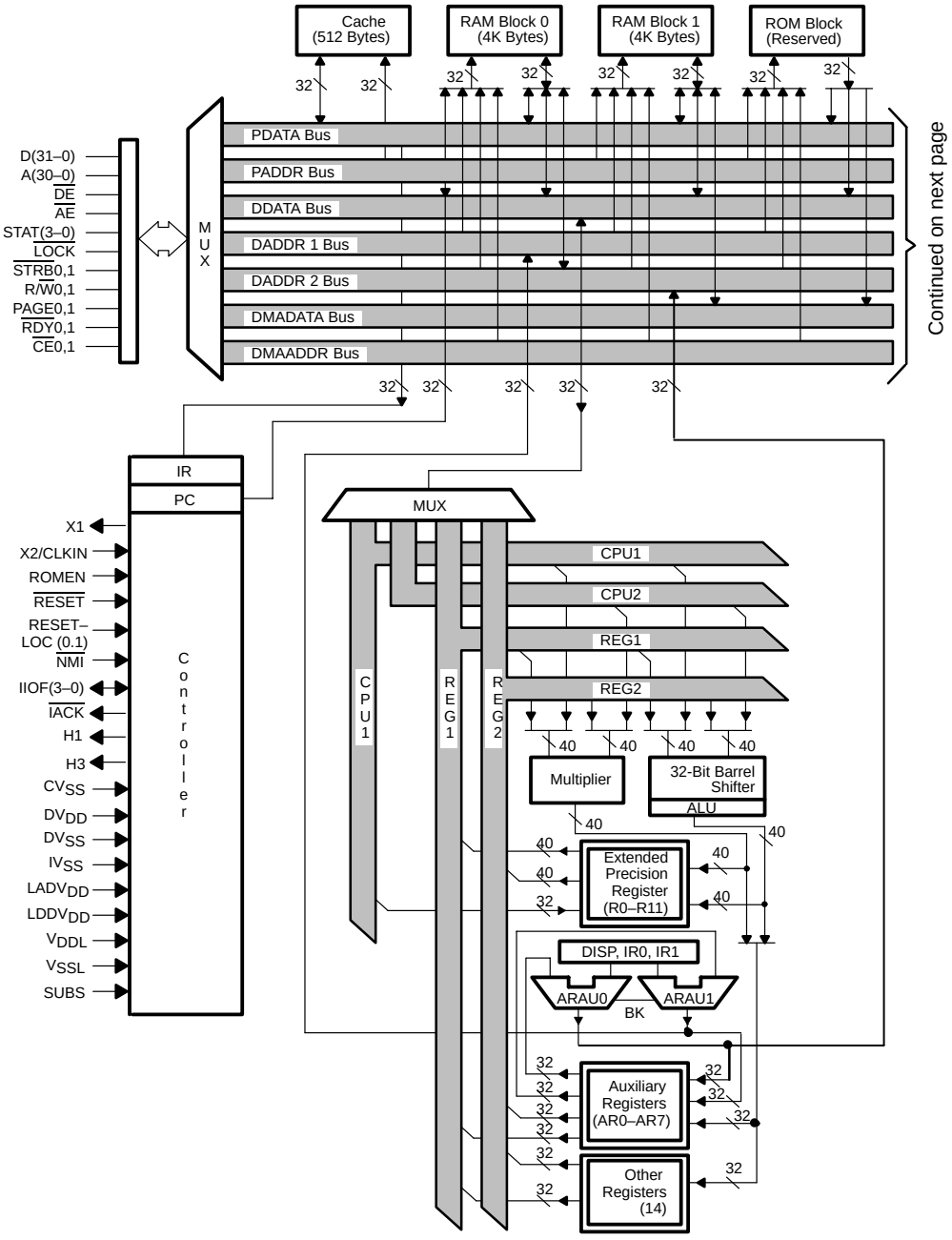
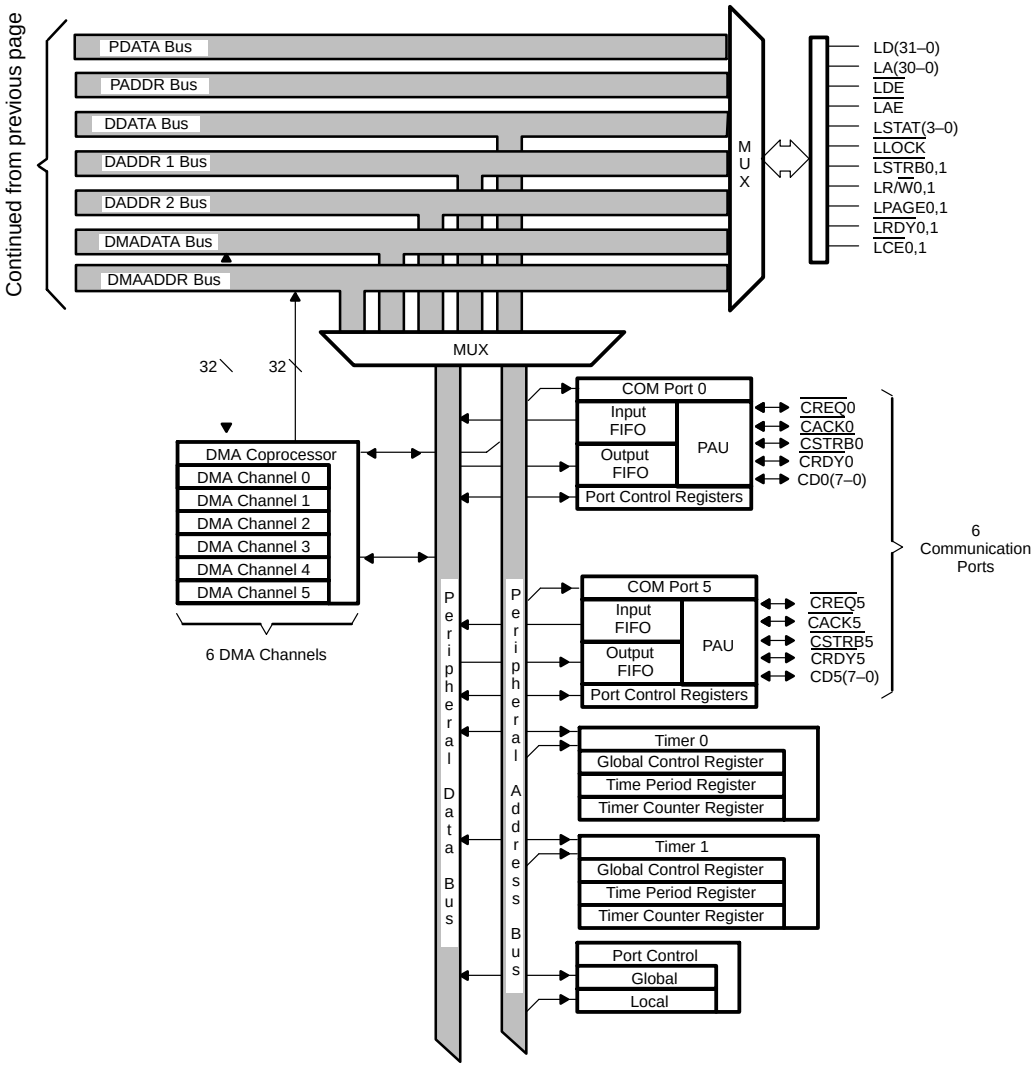


Figure 1–2. TMS320C40 Block Diagram (Concluded)



Chapter 2

How the TMS320C4x PPDS Works

This chapter tells you how to install and remove the board's TMS320C40 devices, describes the TMS320C4x Parallel Processing Development System (PPDS), lists the components and operations of the board, and outlines the board's interfaces.

Topic **Page**

2.1 TM320C40 Installation/Removal Procedures

You can install and remove TMS320C40 devices by using the AMP tool-actuated zero-insertion-force (TAZ) pin-grid array (PGA) hand tool, part number 854234-1. The tool has two handles:

- The long handle straddles the white housing boss of the PGA.
- The short handle engages the black socket cover.

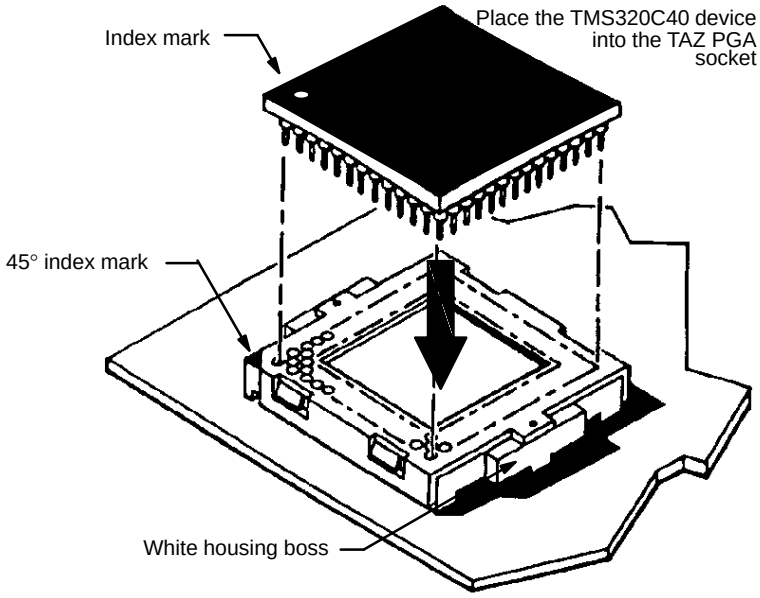
2.1.1 Installing a TMS320C40 Device

To install a TMS320C40 device, follow these steps:

Step 1: Align and place the TMS320C40 device into the PGA socket as shown in Figure 2–1. Make sure that the TMS320C40 device's index mark is pointing toward the 45° index mark of the PGA socket cover.

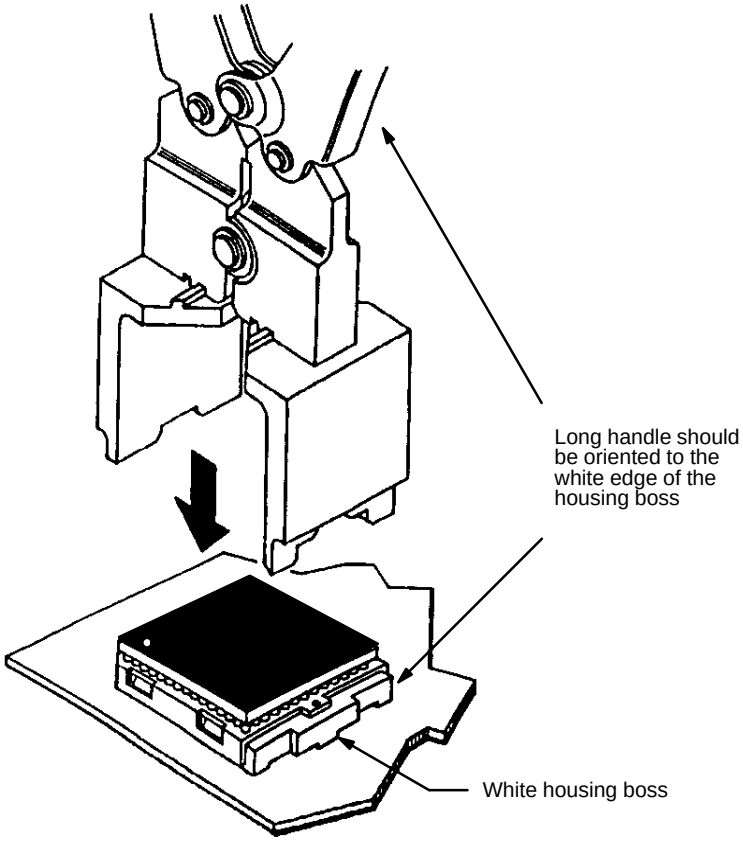
Do not force the TMS320C40 device into the PGA sockets.

Figure 2–1. Positioning a TMS320C40 Device for Installation



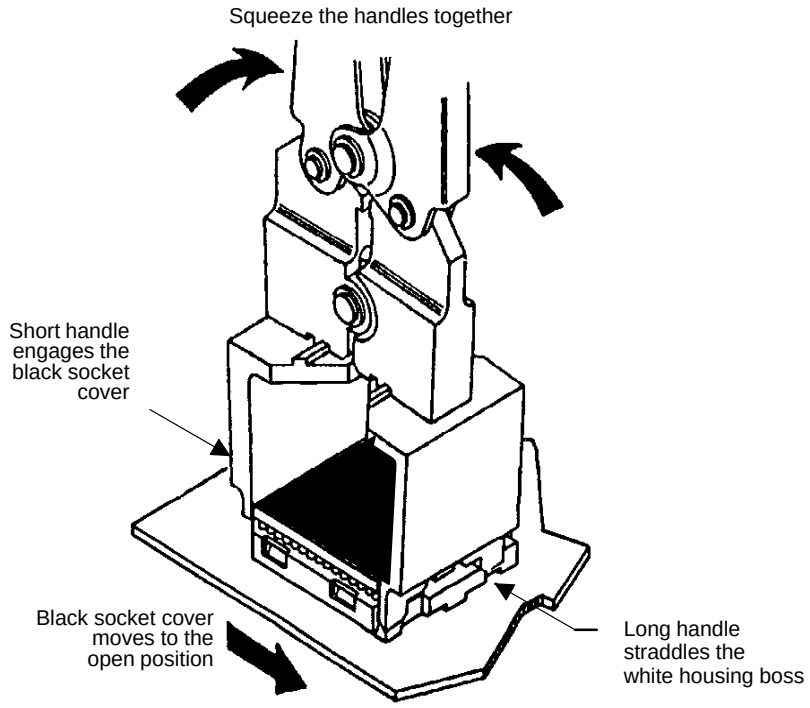
Step 2: Orient the long handle of the TAZ tool as shown in Figure 2–2.

Figure 2–2. Hand Tool Orientation



Step 3: Move the tool vertically until the long handle straddles the housing boss and the short handle engages the black socket cover.

Figure 2–3. Installing the TMS320C40 Device



Step 4: Squeeze the tool handles together until the black socket cover moves towards the white housing boss and clicks the TMS320C40 device into the closed position.

2.1.2 Removing a TMS320C40 Device

To remove a TMS320C40, follow these steps:

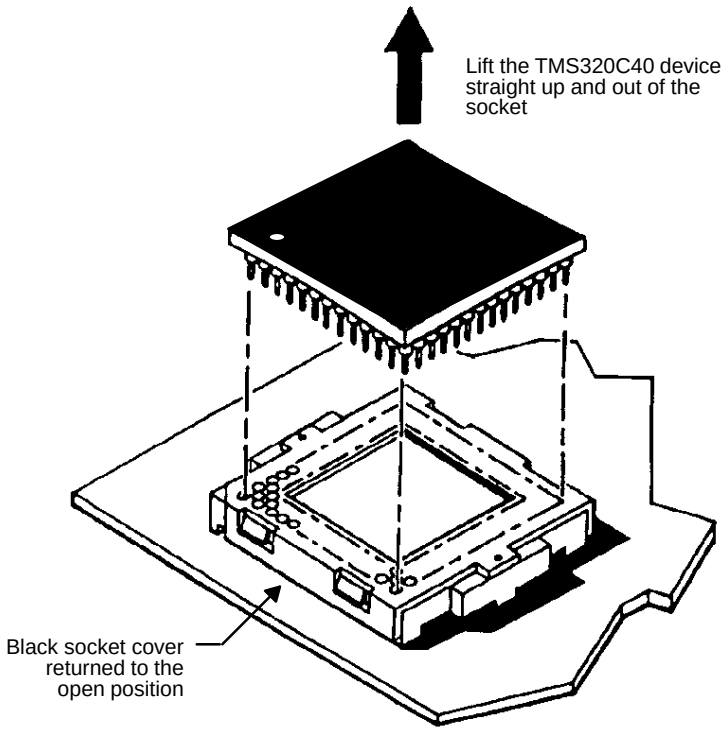
Step 1: Position the TAZ tool 180° opposite from the orientation shown in Figure 2–2 on page 2-3.

Step 2: Move the tool vertically until the long handle straddles the housing boss and the short handle engages the black socket cover.

Step 3: Squeeze the tool handles together until the black socket cover moves towards the white housing boss and clicks the TMS320C40 device into the open position.

Step 4: Lift the TMS320C40 device up and out of the PGA socket as shown in Figure 2–4.

Figure 2–4. Removing a TMS320C40 Device From a PGA Socket



2.2 The TMS320C4x Parallel Processing Development System

The TMS320C4x PPDS is a host-independent processor board that you can place upon your desktop. Figure 2–5 shows the board layout with four TMS320C40 devices installed; Figure 2–6 shows the various interconnects. Although they are not shown, the TMS320C4x PPDS is shipped with a dedicated desktop stand and a 7-ampere, 35-watt power supply. The typical power dissipation of the PPDS is 5 amperes.

Figure 2–5. The TMS320C4x PPDS Board Layout

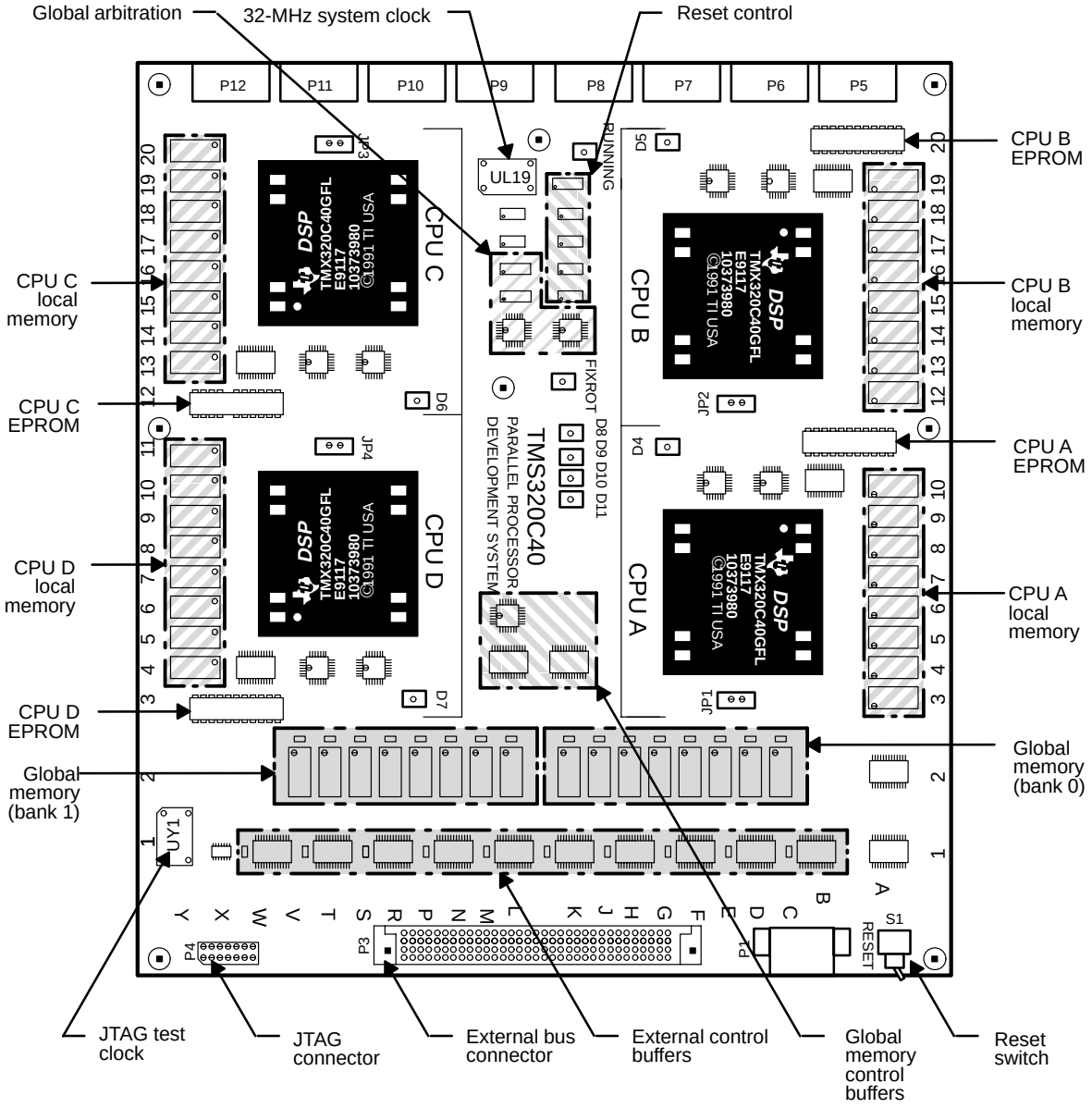
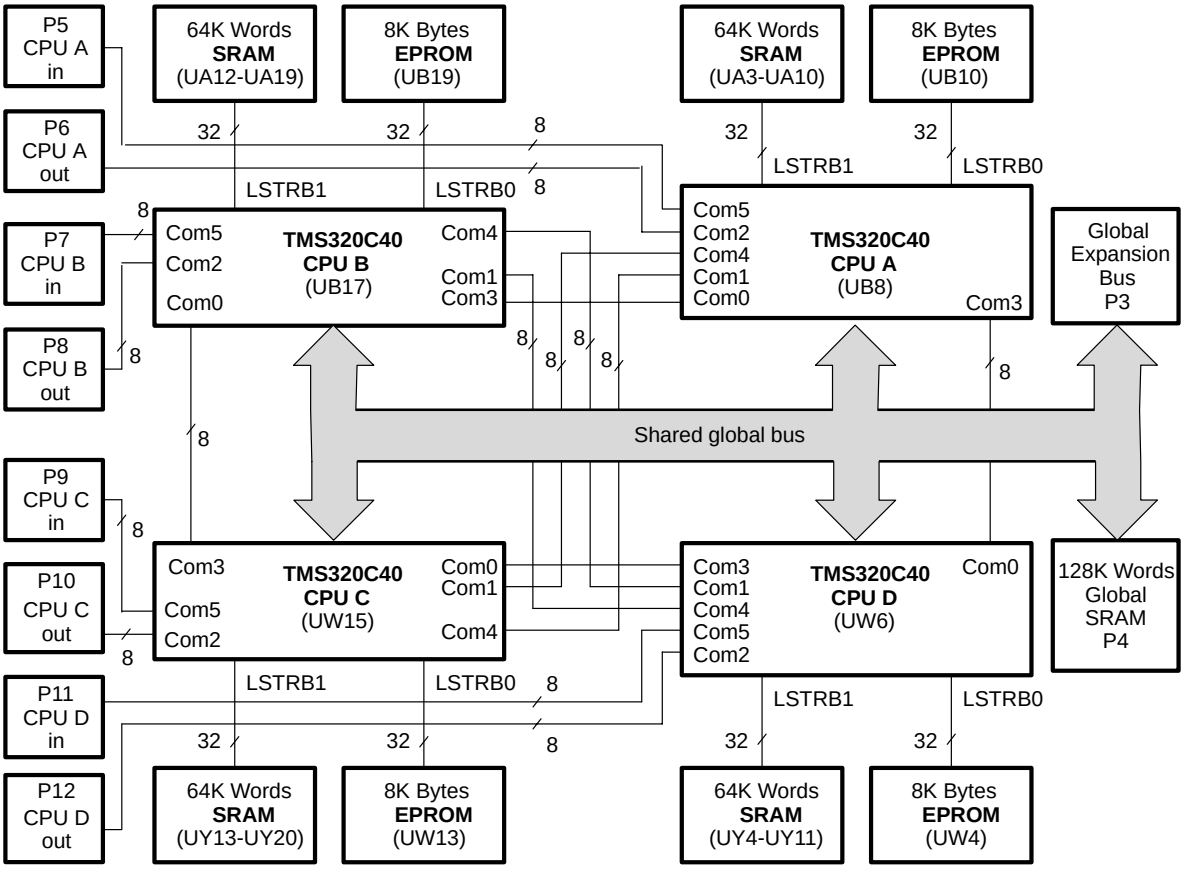


Figure 2–6. TMS320C4x PPDS Block Diagram



2.2.1 TMS320C40 Communication Ports

A parallel processing system supports optimum system performance by distributing tasks between two or more processors. Each TMS320C40 contains six identical high-speed communications ports (Com0–Com5); each communication port provides bidirectional communication. Figure 2–6 shows the interconnections.

- Com0, Com1, Com3, and Com4 communications ports connect each TMS320C40 directly to another, facilitating processor-to-processor communications.
- Each TMS320C40's Com2 and Com5 communication ports are routed to the TMS320C4x PPDS's external connectors P5–P12. These connectors make the power of the TMS320C40s available to external TMS320C40's A/D and D/A converters, SCSI controllers, frame grabbers, and other peripherals.
- You can use the communications ports to download code and data to the TMS320C4x PPDS or upload them to a host system.

2.2.2 Clock and Reset Logic

2.2.2.1 Clock Logic

The system clock circuit for the TMS320C40 is composed of a 32-MHz canned oscillator (UL19) and its associated control (UL18) and drivers (UL17). This circuitry provides the CLKIN input signal that is shared by each TMS320C40.

2.2.2.2 Reset Logic

The TMS320C40s share a common $\overline{\text{RESET}}$ signal. However, $\overline{\text{RESET}}$ is not generated automatically at power-up. Rather, after power-up, you must manually set the reset switch, S1 (see Figure 2–5 on page 2-6), to its most upward position and then back to its downward position.

Note:

During normal operation, the reset switch resides in the downward position, and the LED D2 (labeled *RUNNING*) is lighted.

2.2.3 LED Indicators

The PPDS has ten LEDs that allow you to see the status of the shared global bus signals.

- When an LED is lighted, it indicates that the corresponding signal is a logic level 1 (high).
- When an LED is not lighted, it indicates that the corresponding signal is a logic level 0 (low).

Table 2–1 lists the names of the LEDs and their respective functions.

Table 2–1. LED Indicators

LED	Name	Function
D2	RUNNING	When lighted, indicates that the PPDS is running. When not lighted, indicates that the PPDS is not running.
D3	FIXROT	GBC arbitration mode-control signal.
D4	APRIDMA	Priority DMA signal for CPU A.
D5	$\overline{\text{BPRIDMA}}$	Priority DMA signal for CPU B.
D6	$\overline{\text{CPRIDMA}}$	Priority DMA signal for CPU C.
D7	$\overline{\text{DPRIDMA}}$	Priority DMA signal for CPU D.
D8	ABG	$\overline{\text{BUSGRANT}}$ signal for CPU A.
D9	$\overline{\text{BBG}}$	$\overline{\text{BUSGRANT}}$ signal for CPU B.
D10	$\overline{\text{CBG}}$	$\overline{\text{BUSGRANT}}$ signal for CPU C.
D11	$\overline{\text{DBG}}$	$\overline{\text{BUSGRANT}}$ signal for CPU D.

2.2.4 Emulator Connections

Connector P4 provides the JTAG connections to the ’C4x XDS510 emulator. There are twelve scan path octals in addition to the four ’C40s in the PPDS JTAG scan chain. For this reason, the multiprocessor version of the XDS510 (available for PCs running OS/2 or for Sun SPARCstations) is required for debugging ’C40s on the PPDS.

The XDS510 emulator pod provides the JTAG clock. However, you have the option of installing a canned oscillator at UY1. The frequency of this oscillator must be between 5 and 16 MHz.

2.3 An Overview of the Local and Global Memory System

Each TMS320C40 has 64K words of local SRAM and 8K bytes of EPROM that allow each TMS320C40 to operate independently of the others; 128K x 32-bit words of one wait-state SRAM are shared globally by all of the TMS320C40s.

Table 2–2 shows the address space allocated for each TMS320C40's external memory map.

Table 2–2. TMS320C40 External Memory Map

Address Space	Function	Control Signals
0x0000 0000–0x0000 00FFF	On-chip ROM	Internal
0x0030 0000–0x003F FFFF	EPROM	LSTRB0, PAGE 0
0x0040 0000–0x3FFF FFFF	LCSR register	LSTRB0
0x4000 0000–0x7FFF FFFF	SRAM	LSTRB1
0x8000 0000–0x8000 FFFF	Shared SRAM	PAGE 0, STRB0
0x8001 0000–0x8001 FFFF	Shared SRAM	PAGE 1, STRB0
0x8002 0000–0xFFFF FFFF	Expansion	STRB1

2.3.1 Local Memory Bus

The 8K words of EPROM are mapped into PAGE0, the lower portion of the TMS320C40's memory space, by means of each TMS320C40's LSTRB0 pin.

Local SRAM is mapped to 64K-word boundaries in the LSTRB1 memory space. Each TMS320C40 uses its LSTRB1 pin to interface the 64K words of SRAM to the local bus.

2.3.2 Shared Global Memory Bus

The global bus consists of two 64K-word pages of SRAM (PAGE0 and PAGE1) that are shared by the four TMS320C40s.

- PAGE 0 is comprised of UC2, UD2, UE2, UF2, UG2, UH2, UJ2, and UK2.
- PAGE 1 is comprised of UL2, UM2, UN2, UP2, UR2, US2, UT2, and UV2.

The TMS320C40s interface to the shared SRAM by means of their STRB0 pins. The two pages are mapped on 64K-word boundaries in each TMS320C40's STRB0 memory space.

Note:

The PPDS shared-memory interface is designed for TMS320C40s running at speeds of **less than or equal to 32 MHz**. If the PPDS shared memory is not used, TMS320C40s with speeds as fast as 50 MHz can be used.

You can use the STRB1 signal on the global bus expansion connector (P3) to interface additional memory, TMS320C40s, or other peripherals to the shared global bus.

2.3.3 Local Control Synchronization Register (LCSR)

Each LCSR is mapped to the LSTRB0 memory space above address 0x003F FFFF. Each LCSR is mapped to location 0x0040 0000 in the corresponding TMS320C40's memory map. Write to this location to set or clear the LCSR bit fields.

For more information about the LCSR register, refer to subsection 2.4.5, page 2-13.

2.4 How the Local Memory Bus Works

Each TMS320C40 has 64K words of SRAM and 8K bytes of EPROM. These are the associated components:

- CPU A (UB8)
  - SRAM (UA3–UA10)
  - EPROM (UB11)
- CPU B (UB17)
  - SRAM (UA12–UA19)
  - EPROM (UB20)
- CPU C (UW15)
  - SRAM (UY13–UY20)
  - EPROM (UW12)
- CPU D (UW6)
  - SRAM (UY4–UY11)
  - EPROM (UW3)

2.4.1 The Boot Loader Program

During system initialization ($\overline{\text{RESET}}$ switch S1 is set), the TMS320C40's on-chip boot loader program loads the code stored in EPROM into local SRAM memory and then executes the loaded code by:

1) Configuring the local and global buses for proper bus-ready generation and control-signal memory mapping.

2) Reading four consecutive bytes from EPROM and packing these four bytes into a 32-bit word before writing to the local SRAM.

3) Executing the first code block that was loaded into SRAM.

For more information on the TMS320C40 on-chip boot loader program, see Section 13.2, *Boot Loader Description and External ROM Interfacing,* of the *TMS320C4x User's Guide*.

2.4.2 EPROM Contents

The local EPROM contains a boot loader table that provides the configuration for the TMS320C40's bus-ready generation and control-signal mapping. The boot loader program reads this configuration information from the EPROM to initialize the local and global bus interfaces. The EPROM also contains code that performs a read from the shared memory to initialize the shared-bus interface.

At the end of the boot loader sequence, the 'C40s will branch to themselves in an infinite loop.

2.4.3 The SRAM Interface

64K words of zero wait-state SRAM are interfaced to each TMS320C40's LSTRB1 pin. Only one page is required to interface the SRAM to the LSTRB1 pin.

Note:

An L prefix appended to a mnemonic denotes local memory. The absence of an L prefix denotes global memory. For example, LSTRB0 refers to data strobe 0 on the local memory bus.

The LSTRB1 PAGESIZE field (bits 19–23) of the local memory interface control register (LMICR) should be set to 01111_2. Additionally, the LSTRB1 SWW field (bits 6–7) should be set to 00 ($LRDY1_{int} = LRDY1_{ext}$). LRDY1 is tied externally to ground. This ensures zero wait-state accesses.

2.4.4 The EPROM Interface

Each EPROM byte is mapped to an entire 32-bit word in the TMS320C40's address space. Also, the EPROM is mapped to PAGE 0 of each TMS320C40's LSTRB0 pin. The EPROM, as used in the PPDS, requires three H1 cycles (two wait states) for valid access.

2.4.5 The Local Control Synchronization Register (LCSR)

Each TMS320C40 on the PPDS has one dedicated local control synchronization register (LCSR) that interfaces with the TMS320C40's local bus (LSTRB0). The LCSR assignment is as follows:

- CPU A — LCSR A (UB10)
- CPU B — LCSR B (UB19)
- CPU C — LCSR C (UW13)
- CPU D — LCSR D (UW4)

2.4.5.1 What is an LCSR?

The LCSRs are SN74ALS996 read-back latches that:

- Issue point-to-point and global software synchronization interrupts between the TMS320C40s.
- Control the priority DMA LEDs.
- Issue the arbitration mode for the GBC: fixed or rotating.

2.4.5.2 LCSR Bit Fields

Figure 2–7 shows the bit fields for LCSR A, Figure 2–8 shows the bit fields for LCSRs B through D, and Table 2–3 summarizes and describes the bit fields.

Figure 2–7. Bit Fields for LCSR A

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
priDMA	FIXROT	Reserved	INTD	INTC	INTB	INTA	GLOBINT
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Figure 2–8. Bit Fields for LCSRs B–D

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
priDMA	Reserved	Reserved	INTD	INTC	INTB	INTA	GLOBINT
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 2–3. LCSR Bit Field Summary

Bit No.	Name	Description
0	GLOBINT	When GLOBALINT = 1, a 0 is sent to each TMS320C40's IIOF1 pin.
1	INTA	When INTA = 1, a 0 is sent to CPU A's IIOF2 pin.
2	INTB	When INTB = 1, a 0 is sent to CPU B's IIOF2 pin.
3	INTC	When INTC = 1, a 0 is sent to CPU C's IIOF2 pin.
4	INTD	When INTD = 1, a 0 is sent to CPU D's IIOF2 pin.
5	Reserved	
6	FIXROT †	When fixrot = 0, sets bus arbitration mode to fixed priority. When fixrot = 1, sets bus arbitration mode to rotating priority.
7	priDMA	When priDMA = 1, the corresponding priDMA LED is off (unlit).

† FIXROT applies to CPU A only. It defines the arbitration mode to the GBC. Bit 6 is a reserved field for CPUs B–D.

Each TMS320C40 expects edge-triggered interrupts to appear at its IIOF pins. For proper point-to-point and global interrupt operation, the corresponding LCSR bit must be set or cleared before another interrupt can be sent to the same IIOF pin.

Note:

A TMS320C40 cannot send a point-to-point interrupt to itself. For example, INTA has no effect on CPU A, INTB has no effect on CPU B, etc. However, the global interrupt is sent to all the IIOF1 pins of all of the TMS320C40s, including the TMS320C40 that set the GLOBINT.

2.4.6 Setting the TMS320C40 Local Memory Interface Control Register (LMICR)

Each register can be programmed to control its respective memory interface by defining:

- ❑ Page sizes for the two strobes,
- ❑ When strobes are active,
- ❑ Wait states,
- ❑ Other operations that control the memory interface.

The TMS320C40's LMICR controls partitioning and speed of local bus accesses. Set the LMICR controls as follows:

- ❑ Set the LSTRB0 SWW field (bits 4–5) to 01_2 (for internal wait states).
- ❑ Set the LSTRB0 WTCNT field (bits 8–10) to the following values:
  - 001_2 for a minimum of one software wait state to ensure correct LCSR operation.
  - 010_2 for a minimum of two software wait states to ensure correct EPROM read operation.
- ❑ Set the LSTRB ACTIVE field (bits 24–28) to 11101_2 to map
  - STRB0 to the first gigaword of the TMS320C40 address space, and
  - STRB1 to the second gigaword of the TMS320C40 address space.
- ❑ For proper EPROM and LCSR operation, set the LSTRB0 PAGESIZE field (bits 14–18) to 10101_2 to insert an idle cycle between ROM and LCSR read accesses.
- ❑ Set the LSTRB1 SWW field (bits 6–7) to 00_2 ($LDY_{int} = LRDY_{text}$). LRDY1 is tied externally to ground to ensure zero wait-state accessing.
- ❑ Set the LSTRB1 PAGESIZE field (bits 19–23) to no less than 01111_2 to prevent extra cycles from being inserted between local SRAM accesses.

See Section 7.2, *Memory Interface Control Registers,* in the *TMS320C4x User's Guide* for additional information.

2.5 How the Global Memory Bus Works

Each TMS320C40 shares access to 128K × 32-bit words of one wait-state SRAM. The 128K words of SRAM are connected to each TMS320C40's STRB0 pin. The memory is separated into two pages of 64K words each.

Note:

The PPDS shared-memory interface is designed for TMS320C40s running at speeds of **less than or equal to 32 MHz**. If the PPDS shared memory is not used, TMS320C40s with speeds as fast as 50 MHz can be used.

2.5.1 The Global Bus Controller (GBC)

The global bus controller (GBC) logic controls access to the shared global bus. The GBC is composed of PLDs UL14, UK14, and UL6 and flip-flops UL15 and UL16.

The GBC offers fixed and rotating priority arbitration schemes for system flexibility: the priority scheme is user programmable. See Chapter 3, *TMS320C4x PPDS PLD Equations*, for more information about programming PLDs.

2.5.2 The TMS320C40 Global Memory Interface Control Register (GMICR)

The TMS320C40's GMICR controls partitioning and speed of global bus accesses. The GMICR and LMICR differ only as follows:

1) They occupy different locations in the memory map.

2) The local memory interface control signals have an L prefix.

Set the GMICR controls as follows:

- ❑ Set the STRB0 SWW field (bits 4–5) to 00_2 ($RDY0_{int}$ = $RDY0_{ext}$). The $\overline{RDY0}$ signal is generated by the shared memory interface logic.
- ❑ Set the STRB0 ACTIVE field (bits 24–28) to 10000_2 to make $\overline{STRB0}$ active for addresses 0x8000 0000–0x8001 0000.
- ❑ Set the STRB0 PAGESIZE field (bits 14–18) to 01111_2 to make the shared global memory page size 64K words.

2.5.3 Global Bus Arbitration

The GBC automatically performs shared global bus arbitration by suspending or allowing a TMS320C40 access to the global-bus. A TMS320C40 requests shared global bus access each time its CPU or DMA attempts to read from or write to the global bus. Once a TMS320C40 owns the global bus, it keeps it until another TMS320C40 requests the bus.

2.5.3.1 Interlocked Accessing

To allow multiple TMS320C40s to access global memory and shared data in a coherent manner, you can use the TMS320C40 interlocked instructions LDII or LDFI. These interlocked instructions guarantee that the TMS320C40 remains bus master after the completion of the read. After a TMS320C40 starts interlocked access to the shared bus, the SIGI, STII, or STFI instruction must not be executed, unless you want interlocked access to complete. Accordingly, interlocked access to memory should always end in an SIGI, STII, or STFI instruction to end interlocked operation.

For more information about TMS320C40 interlocked operations, refer to the following sections of the *TMS320C4x User's Guide*:

- Section 6.5, *Interlocked Operations*
- Section 7.7, *Interlocked Instructions Definition and Bus Timing*
- Chapter 11, *Assembly Language Instructions* (see the descriptions of the individual interlocked instructions)

2.5.3.2 Priority DMA Accessing

When a TMS320C40 uses interlocked access to the bus, it remains the bus master until interlocked access is complete. Each TMS320C40 has an LED indicator light, called priority DMA, that you can use to indicate priority access to the shared memory. To use this feature, your application software must set bit 7 of the TMS320C40's local data bus, which relates to bit D7 of the local control synchronization register (LCSR). Setting this bit turns the priority DMA LED off. Clearing D7 of the LCSR turns the LED on.

2.5.4 How Global Bus Arbitration Is Implemented

The FIXROT signal controls global bus arbitration. The FIXROT signal is generated by bit D6 of the LCSR register, which relates to bit 6 of TMS320C40 CPU A. Bit D6 toggles the arbitration priority mode between fixed or rotating arbitration for the shared bus.

- Clearing bit 6 selects a **fixed-mode scheme**.
- Setting bit 6 selects a **rotating-mode scheme**.

2.5.4.1 A Fixed-Priority Scheme

In a **fixed-priority** scheme, the priority does not change. When bit 6 is cleared, the TMS320C40s have fixed priorities for accessing the global bus: CPU A has the highest or first-serviced priority, CPU B has the second highest priority, CPU C has the next highest priority, and CPU D has the lowest or last-serviced priority.

2.5.4.2 A Rotating-Priority Scheme

In a **rotating-priority** scheme, the least recently serviced CPU (bus master) has the highest priority for the bus. The remaining CPUs rotate sequentially through the priority list, with the next lowest CPU from the just-serviced CPU becoming the highest priority on the next request.

Setting bit 6 places the CPUs in a rotating priority mode. The priority rotates every time a TMS320C40 gives up the bus and allows a new bus master to control the bus. However, after system reset, the default arbitration mode is the fixed-priority mode. You must set bit D6 of the LCSR register if you want the rotating-priority mode. After system reset, when the rotating mode is chosen, CPU A starts with the highest priority (first serviced), and CPU D has the lowest priority (last serviced).

2.5.5 Global Bus Controller (GBC) Parking Feature

Global bus parking is implemented for the following bus arbitration protocol: the current bus master retains control of the bus and can continue making accesses to global memory until another CPU requests bus access. In other words, the current bus master does not give up the bus unless another processor wants control. Bus parking reduces memory access latency when only one CPU requests access to the global bus.

2.5.6 When Bus Arbitration Occurs

Shared bus arbitration occurs when a TMS320C40 gives up the bus (that is, deasserts its bus request signal). A TMS320C40 bus master gives up the bus only if another processor requests the bus (bus grant deasserted), the bus master has finished its current bus access, and one of the following conditions exists:

- The bus master does not need to perform a shared memory access and is in the noninterlocked access mode ($\overline{\text{LOCK}}$ inactive)
- The bus master accesses a new page of shared SRAM ($\overline{\text{STRB0}}$ inactive)
- The bus master's next shared global bus (either SRAM or expansion connector) access is a write

The current bus master will not give up the bus in the following situations:

- The bus master is in interlocked access mode ($\overline{\text{LOCK}}$ active)
- The next bus master access is the second of back-to-back reads from the same page of shared memory ($\overline{\text{STRB0}}$ active)
- The next bus master access is a shared memory expansion connector read

2.5.7 Global Bus Arbitration and Transfer Timing

Figure 2–9 through Figure 2–12 show shared bus arbitration and data transfer timing controlled by the rotating-priority arbitration configuration. These figures represent a TMS320C40 requesting a shared-bus access when it is not the current bus master.

The following holds true for these figures:

- H1 is the output clock of the TMS320C40 that is requesting bus access.
- Clocks H1 and H3 are half the CLKIN signal rates.
- The GBC's input clock is CLKIN.

Access to shared memory requires wait states because of arbitration logic synchronizer delays and the 35-ns SRAMs. A new bus master's first memory access after an arbitration win requires a minimum of five H1 cycles, including the time beginning with the status lines becoming active, and ending with the read or write cycles. Subsequent reads or writes require two H1 cycles only.

Two-cycle memory accesses allow:

- Sufficient time for control signals to go active and inactive to complete read or write cycles for 35-ns memories.
- Processors to stop driving the bus (after a bus arbitration contest) before another processor starts driving the bus.
- Sufficient time for signal buffering between the processor address bus and memory. Buffer delays are less than 10 ns with commercially available parts.

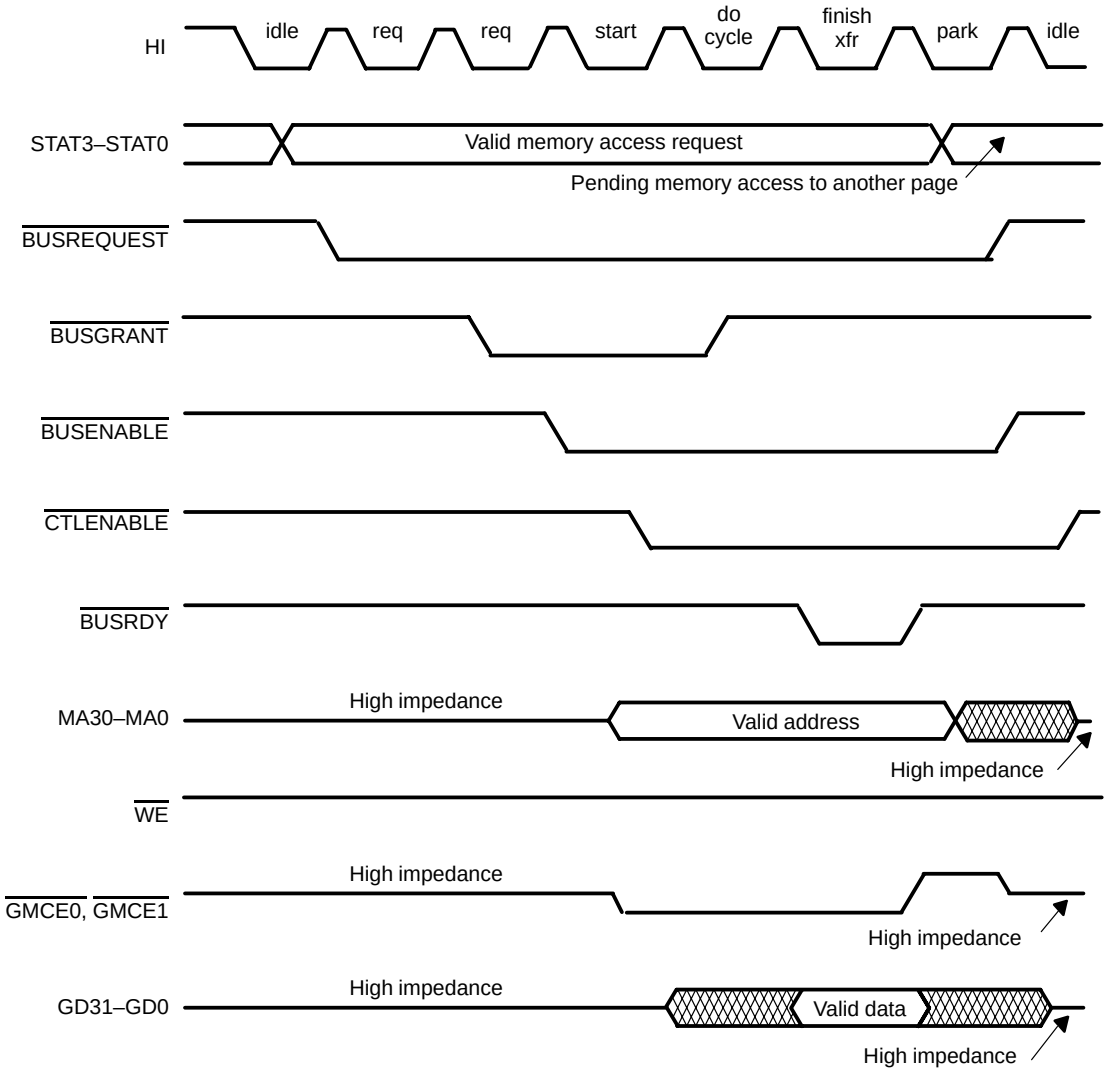
In Figure 2–9, a TMS320C40 wins an arbitration contest immediately and completes one read cycle. It loses arbitration for the next transfer on the shared bus because $\overline{\text{BUSGRANT}}$ and $\overline{\text{STRB0}}$ go inactive high.

When its associated TMS320C40 loses an arbitration contest, the first-level PLD:

- Signals the GBC that it has given up the bus by bringing its $\overline{\text{BUSREQUEST}}$ signal inactive high.
- Sends bus disable signals $\overline{\text{BUSENABLE}}$ and $\overline{\text{CTLENABLE}}$ high to the TMS320C40's $\overline{\text{AE}}$, $\overline{\text{DE}}$, and $\overline{\text{CE}}$ pins, bringing the bus data and control signals to a high-impedance state.

The bus is brought immediately to a high-impedance state because the GBC will grant another CPU access to the shared bus as soon as the GBC sees the $\overline{\text{BUSENABLE}}$ and $\overline{\text{TIMEOUT}}$ signals go inactive.

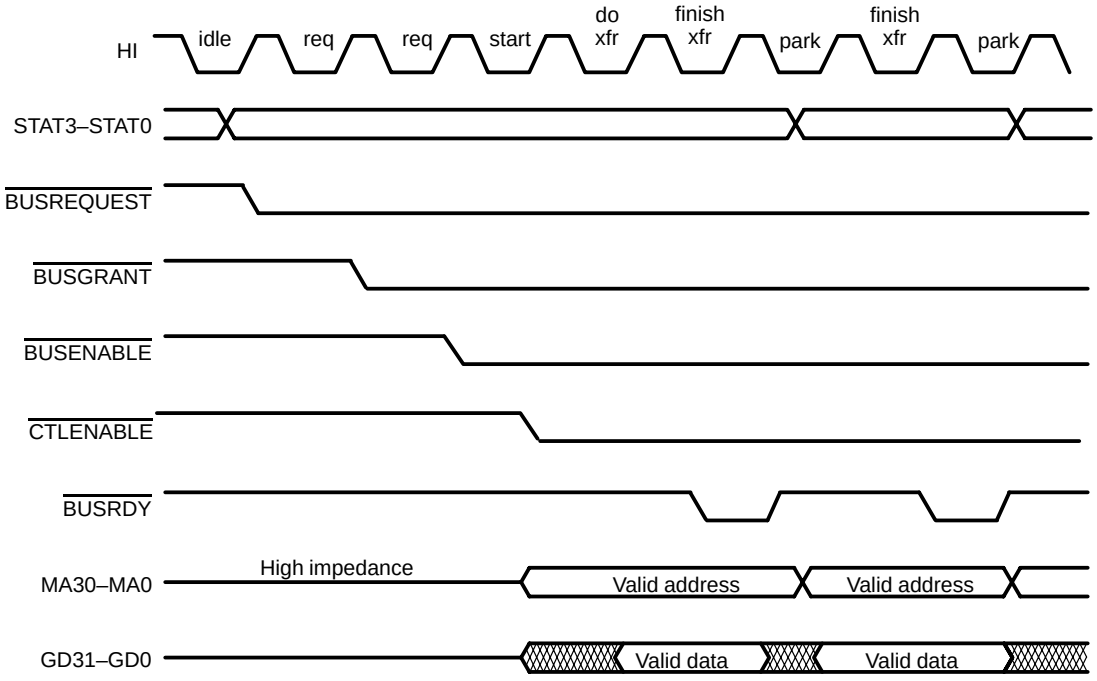
Figure 2–9. Successful TMS320C40 Arbitration and Data Read From Shared Bus Memory, Followed by an Unsuccessful Arbitration Contest



Note: In Chapter 3, the PAL signals corresponding to the signals in this timing diagram indicate that the signals are active low (for example, the BUSREQUEST signal is expressed as `busrequest_`).

Figure 2–10 shows a successful arbitration contest, followed by successive reads. The TMS320C40 can do successive reads on the shared bus because no other CPU has requested access (BUSGRANT stays active).

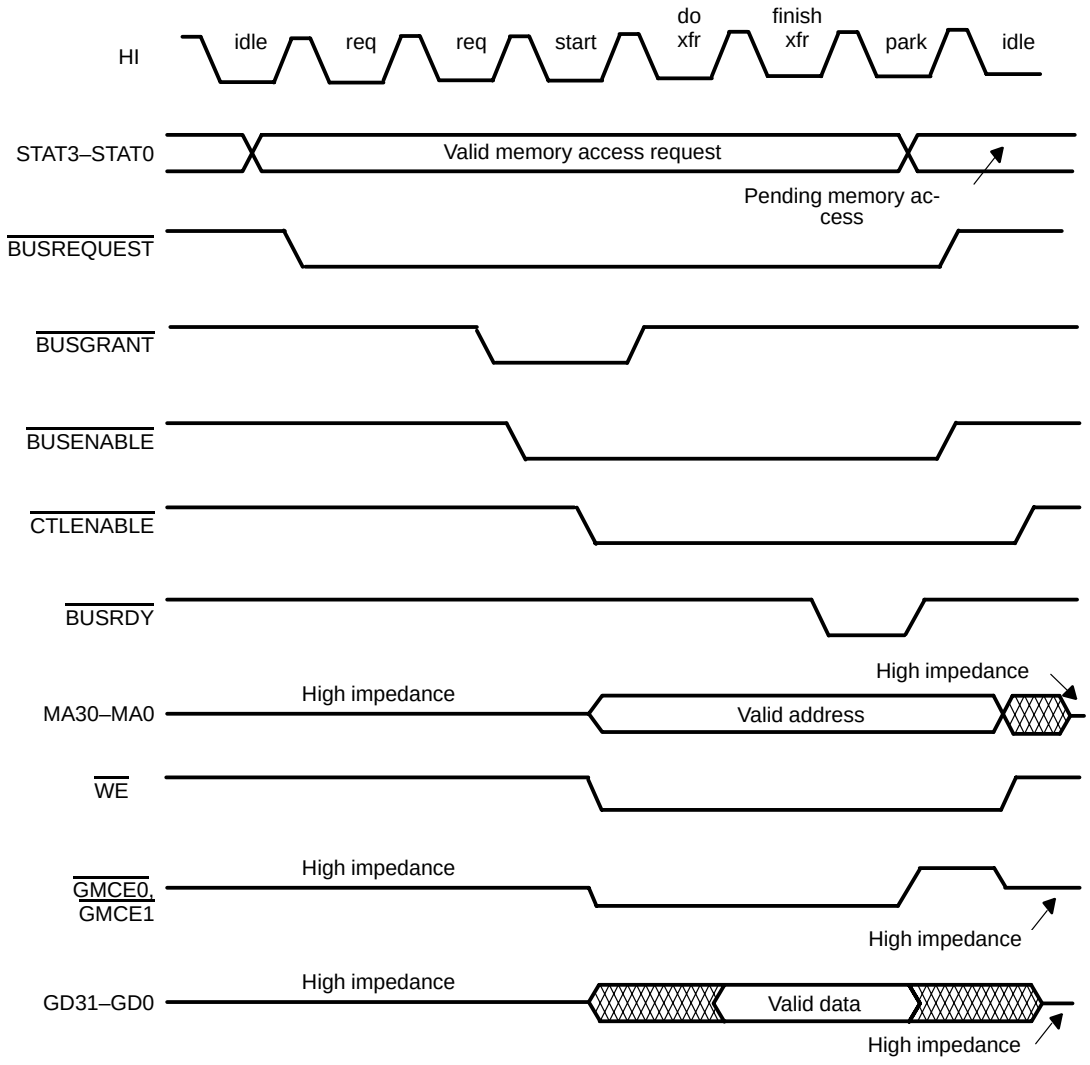
Figure 2–10. Successful TMS320C40 Arbitration; Data Read; Data Read



Note: In Chapter 3, the PAL signals corresponding to the signals in this timing diagram indicate that the signals are active low (for example, the BUSREQUEST signal is expressed as `busrequest_`).

Figure 2–11 shows an arbitration win, followed by a single data write from the shared global bus memory, which is followed by an unsuccessful arbitration contest.

Figure 2–11. Successful TMS320C40 Arbitration and Data Write From Shared Bus Memory, Followed by an Unsuccessful Arbitration Contest

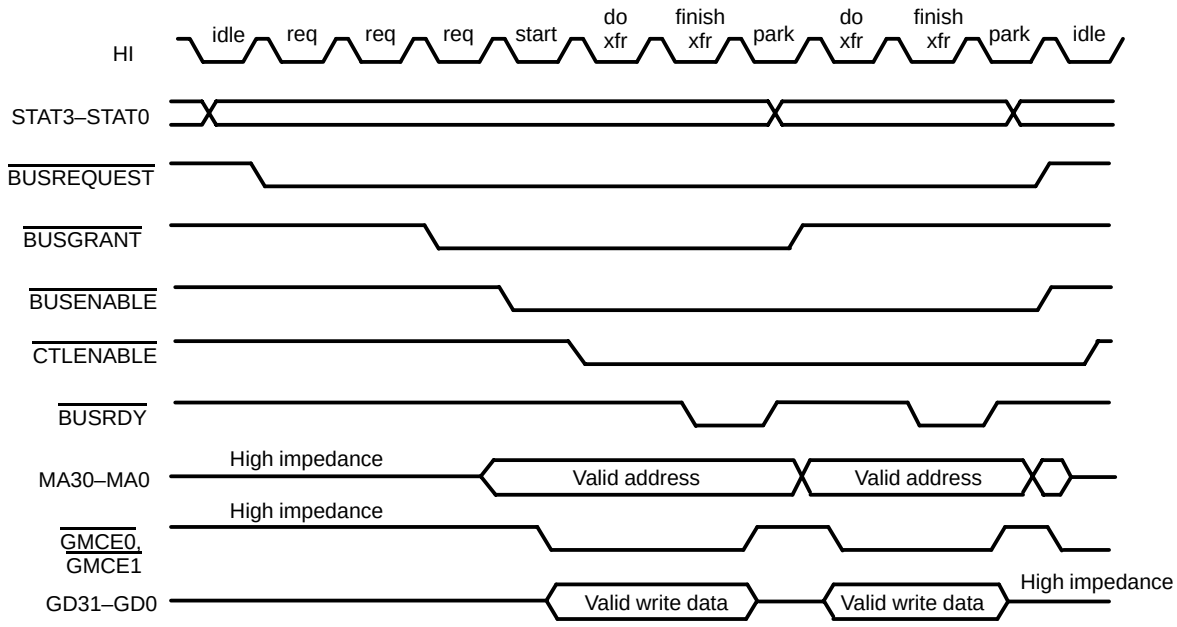


Note: In Chapter 3, the PAL signals corresponding to the signals in this timing diagram indicate that the signals are active low (for example, the BUSREQUEST signal is expressed as `busrequest_`).

Figure 2–12 shows an arbitration win, followed by successive writes and an arbitration loss. The second write occurs because the $\overline{\text{BUSGRANT}}$ signal going inactive is missed by the first-level PLDs on the following H1 rising edge; the PLDs synchronize on H1 rising.

The first-level PLD transcends to the do_cycle state (defined in the PAL equations in Section 3.1, *Global Bus Interface Logic — UE10, UE19, US4, and US13*) because the $\overline{\text{GMCE(0,1)}}$ signal is high and the PLD has not seen the $\overline{\text{BUSGRANT}}$ signal go inactive from the synchronizer output. Even though the first-level PLD sees that the $\overline{\text{BUSGRANT}}$ signal is taken away during the next H1/H3 cycle, it does not take away its $\overline{\text{BUSREQUEST}}$ signal until the end of the second write cycle. Then, the $\overline{\text{BUSREQUEST}}$ signal is made inactive, and the bus is disabled.

Figure 2–12. Successful TMS320C40 Arbitration Win, Followed by Successive Writes and an Arbitration Loss



Note: In Chapter 3, the PAL signals corresponding to the signals in this timing diagram indicate that the signals are active low (for example, the $\overline{\text{BUSREQUEST}}$ signal is expressed as `busrequest_`).

2.5.8 Global Expansion Bus Connector (P3)

The global expansion bus connector (P3) supports system expansion by making the shared global bus available to external memory or peripherals. For more information about pinouts and signal availability, see Section A.1, *The Expansion Bus Connector, P3.*

2.5.8.1 External Bus Connector Accesses

The GBC and its associated logic control external global expansion accesses through the connector (P3) by means of the TMS320C40's STRB1 control signal.

Access timing is essentially the same as the TMS320C4x PPDS shared-memory timing that is shown in Figure 2–9 through Figure 2–12. However, the control and data signals are buffered by devices that have propagation delays of less than 10 ns.

2.5.8.2 Expansion Bus Connector and Shared Memory Timing Differences

P3 timing and shared memory timing differ in the length of the shared bus access and the way valid external accessing is ended.

- For shared memory accesses, the first access after bus arbitration requires three H1 cycles. Subsequent reads require two H1 cycles; subsequent writes require three H1 cycles.
- The first access to P3 after bus arbitration requires a minimum of three H1 cycles. Subsequent reads require a minimum of two H1 cycles. Subsequent writes require a minimum of three H1 cycles.

Accesses from external expansion memory or external peripherals through P3 are terminated by the $\overline{\text{XRDY1}}$ signal, which is an external input to P3.

The $\overline{\text{XRDY1}}$ signal is latched by an SN74F175 D-edge flip-flop (UA2) on the TMS320C4x PPDS. This flip-flop generates the $\overline{\text{XRDY}}$ signal, which is routed to the local control PLDs (UE10, UE19, US4, and US13) to end valid external accesses.

Do not assert the $\overline{\text{XRDY1}}$ signal until the fin_cycle. The fin_cycle is guaranteed to be achieved two cycles after $\overline{\text{XSTRB}}$ goes active. The $\overline{\text{XRDY1}}$ signal needs to have a pulse duration of only one cycle.

The expanding shared-memory interface requires some board modification. Contact the DSP hotline at (713) 274-2320 for details.

Figure 2–13 and Figure 2–14 show the timing relationships for the $\overline{\text{XRDY1}}$ signal during write and read cycles, respectively, and Table 2–4 shows the $\overline{\text{XRDY1}}$ timing constraints.

Figure 2–13. Expansion Bus Connector $\overline{\text{XRDY1}}$ Write Timing

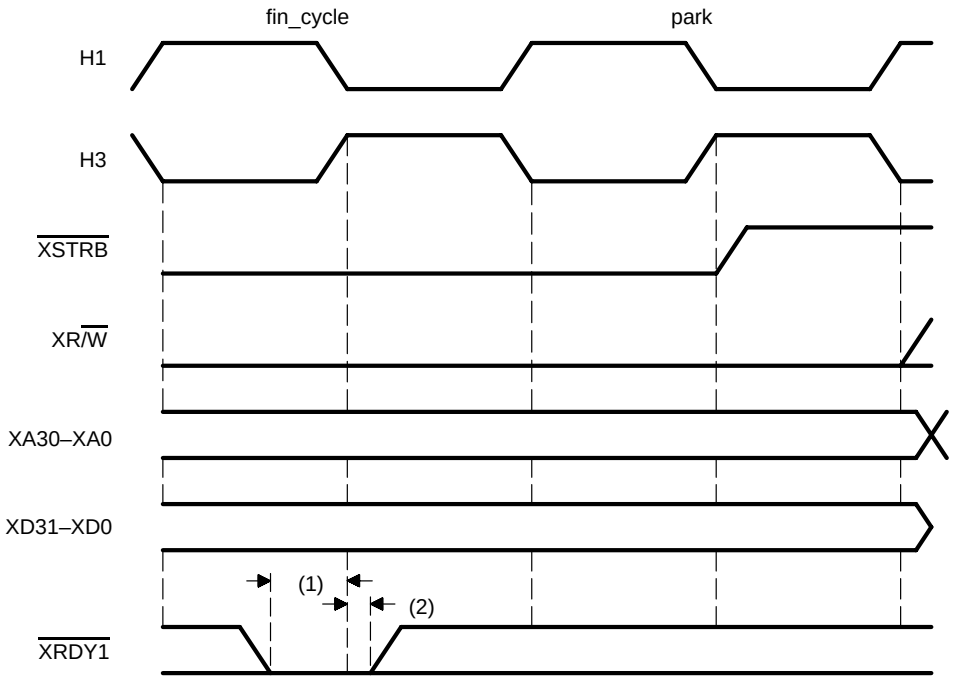


Figure 2–14. Expansion Bus Connector $\overline{XRDY1}$ Read Timing

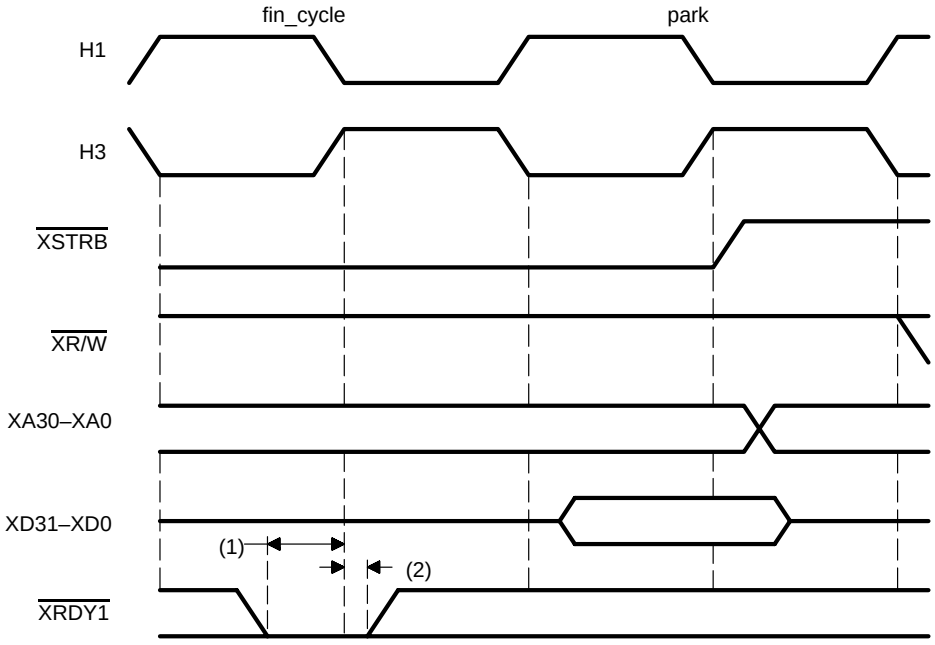
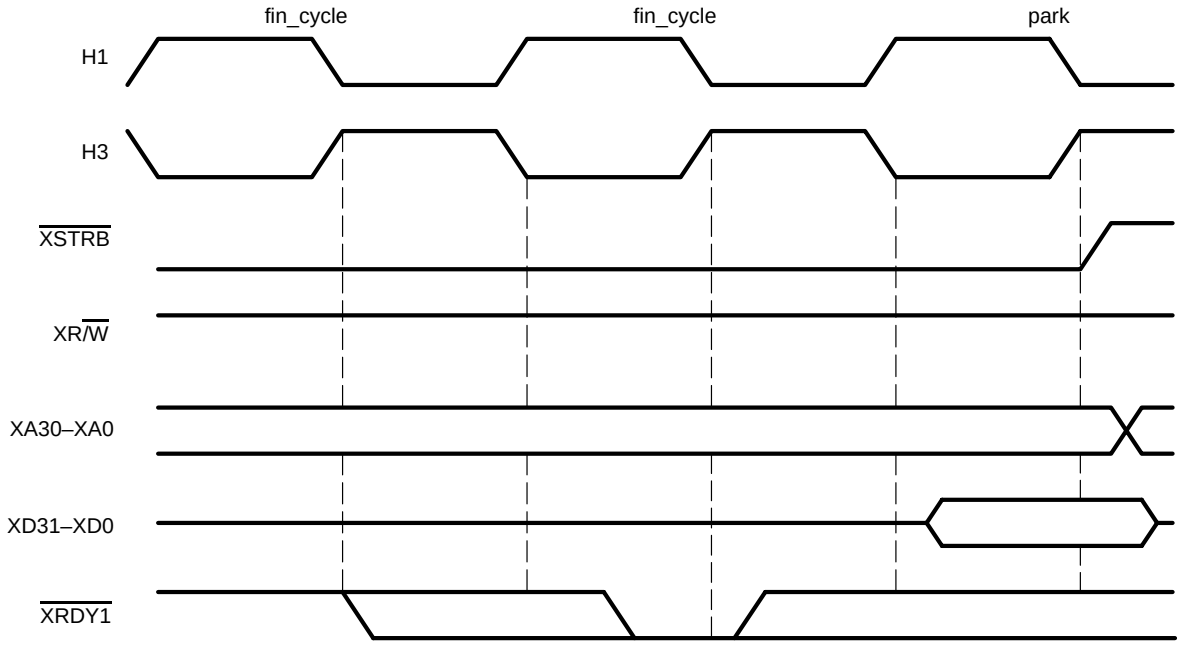


Table 2–4. Timing Constraints for $\overline{XRDY1}$

No.	Name	Description	Min	Max	Unit
1	$t_{su(H3)}$	$\overline{XRDY1}$ valid before H3 high	3		ns
2	$t_{h(H3)}$	$\overline{XRDY1}$ hold time after H3 high	1		ns

Figure 2–15. Timing for $\overline{XRDY1}$ With One Wait State



2.6 Interrupts and the IIOF Flag Register (IIF)

Each TMS320C40 has a memory-mapped read/write flag register (IIF) that is used to:

- Control the on-chip boot loader,
- Control external interrupt pins IIOF[0:3], and
- Receive interprocessor software synchronization interrupts.

2.6.1 The Configuration of the IIOF Register

Table 2–5 summarizes how the IIOF flag register is used to control software synchronization interrupts.

Table 2–5. IIOF Flag Register Pin Usage

IIOF Pin	How Used	Description
IIOF0	Input from the global connector (P3).	
IIOF1	Global interrupt synchronization input signal.	The PPDS's LCSR register issues this flag when its bit 0 is set. It is input simultaneously to each TMS320C40's IIOF1 pin.
IIOF2	Point-to-point interprocessor synchronization signal.	The PPDS's LCSR register issues this flag when the appropriate bit in the register is set. It is input to the IIOF2 pin of one of the TMS320C40s as follows: a. TMS320C40 A is interrupted when bit 1 is set, b. TMS320C40 B is interrupted when bit 2 is set, c. TMS320C40 C is interrupted when bit 3 is set, and d. TMS320C40 D is interrupted when bit 4 is set.
IIOF3	Boot load selection during system initialization.	This signal is not accessible to the user.

Each TMS320C40 expects edge-triggered interrupts to appear at its IIOF pins. Table 2–6 summarizes how the IIOF register of each TMS320C40 must be configured to support proper point-to-point, global interrupt, and expansion connector interrupt operations.

Table 2–6. How the IIOF Register Can Be Used

Bit Number	Name	Function
0	FUNC0	When FUNC = 0, IIOF pin 0 is a general-purpose I/O (R/W) pin. When FUNC = 1, IIOF pin 0 is an interrupt from the connector.
1	TYPE0	If FUNC = 0, then TYPE = 0 makes IIOF0 an input pin. If FUNC = 0, then TYPE = 1 makes IIOF0 an output pin. If FUNC = 1, then TYPE = 0 makes IIOF0 an edge-triggered interrupt pin. If FUNC = 1, then TYPE = 1 makes IIOF0 a level-triggered interrupt pin.
3	EIIOF0	If FUNC = 1, then EIIOF must equal 1 to enable the interrupt.
4	FUNC1	If FUNC = 0, IIOF1 is an input. If FUNC = 1, IIOF1 is a global interrupt,
5	TYPE1	If FUNC = 0, then TYPE must equal 0 to make IIOF1 an input pin. If FUNC = 1, then TYPE must equal 0 to make IIOF1 an edge-triggered interrupt.
7	EIIOF1	If FUNC = 1, then EIIOF must equal 1 to enable the interrupt.
8	FUNC2	If FUNC = 0, IIOF2 is an input; if FUNC = 1, IIOF2 is an interrupt from the remaining CPUs.
9	TYPE2	If FUNC = 0, then TYPE must equal 0 to make IIOF2 an input pin. If FUNC = 1, then TYPE must equal 0 to make IIOF2 an edge-triggered interrupt.
11	EIIOF2	If FUNC = 1, then EIIOF must equal 1 to enable the interrupt.
12	FUNC3	FUNC must equal 0, and TYPE must equal 0. IIOF3 is used for boot source memory selection.

Note: Bits 2, 6, and 10 are interrupt flags if FUNC = 1. These bits are status bits if FUNC = 0.

Remember that IIOF pins 1 and 2 can be used as edge-triggered interrupt or general-purpose input pins. They **cannot** be used as general-purpose outputs on the TMS320C4x PPDS.

For more information about the IIOF register, see Chapter 3, *CPU Registers, Memory, and Cache,* of the *TMS320C4x User's Guide*.

2.6.2 Boot Loader Considerations

The TMS320C40 on-chip boot loader uses the values on the IIOF pins to determine the location of the source program so that the source can be loaded into SRAM.

Note:

Do not modify any TMS320C40 IIOF registers until boot loader operations are complete; if you do, the TMS30C4x PPDS will not initialize properly.

Chapter 3

TMS320C4x PPDS PLD Equations

This chapter contains the programmable logic source for the PLDs used on the TMS320C4x PPDS with TMS320C40 devices installed. These PLD equations were reduced with DATA I/O ABEL version 3.1 at a reduction level of 3.

Table 3–1 describes the symbols used in these equations. Note that an underbar at the end of a signal name in the PAL source files indicates that the signal is active low.

Table 3–1. Programming Symbols

Symbol	Description
!	Inversion
#	OR function
&	AND function
:=	D-edge flip-flop

Topic **Page**

3.1 Global Bus Interface Logic—UE10, UE19, US4, and US13

These components are TIBPAL16R6 devices that compose the shared memory interface control circuitry for each TMS320C40 CPU.

- UE10 controls the EPROM (UB10) for CPU A.
- UE19 controls the EPROM (UB19) for CPU B.
- US4 controls the EPROM (UW3) for CPU D.
- US13 controls the EPROM (UW12) for CPU C.

Figure 3–1 shows the ABEL source file, and Figure 3–2 shows the reduced equations required for programming these devices.

Figure 3–1. ABEL Source File for UE10, UE19, US4, and US13

```
module    c40_local_glob_bus_interf
title'
DWG NAME  EEPROM and local control (of shared bus arbitration logic)
DWG #
COMPANY   TEXAS INSTRUMENTS INCROPORATED
DATE      4/10/91'

ue10      device          'P16R6';

"inputs for global interface logic
h1              Pin 1;  "clock input
priDMA_         Pin 2;  "flag reg output (unused input)
stat3           Pin 3;  "stat3=0 STRB0 access, stat3=1 STRB1 access
stat2           Pin 4;
stat1           Pin 5;
stat0           Pin 6;
strb0_          Pin 7;
bg_             Pin 8;  "busgrant (from bus arbiter)
xrdy_           Pin 9;  "rdy signal from the external expansion connector
lock_           Pin 12;

"outputs for global interface logic
start_state     Pin 18;  "low if the output state is the strt_cycl state
busreq_         Pin 17;
busenable_      Pin 16;
busrdy_         Pin 15;
park_state      Pin 14;  "low if the output state is the park state

gwe_            Pin 13;  "write enable signal for shared memory
rdy1_           Pin 19;  "rdy signal to the C40 for expansion memory

"define machine state bits
"[start,park,busreq_,busenable_,busrdy_];

iddle           = ^b11111;     "31
req_cycle       = ^b11011;     "27
strt_cycl       = ^b01001;     "09
do_cycle        = ^b11001;     "25
fin_cycle       = ^b11000;     "24
park            = ^b10001;     "17
```

Figure 3–1. ABEL Source File for UE10, UE19, US4, and US13 (Continued)

```
     "convert to positive logic to make the test vectors
     "easier to understand

     lock      = !lock_;
     bg        = !bg_;
     priDMA    = !priDMA_;
     idle_stat = (stat2 & stat1 & stat0);

     outst = [start_state,park_state,busreq_,busenable_,busrdy_];

     c,H,L,X  = .C.,1,0,.X.;

@page
state_diagram  outst
state idle:
    case     ( idle_stat) :idle;
             (!idle_stat) :req_cycle;
    endcase;
state req_cycle:
    case         idle_stat                    :idle;
              ( bg_ & !idle_stat)             :req_cycle;
              (!bg_ & !idle_stat)             :strt_cycl;
    endcase;
state strt_cycl:
         GOTO do_cycle;
state do_cycle:
         GOTO fin_cycle;
state fin_cycle:
    if       (!stat3 # !xrdy_ # idle_stat)       then park
    else                       fin_cycle;
state park:
    if       (  bg_ & lock_ & stat2)             then idle
    else if   (!stat2 & (stat3 # !strb0_))       then fin_cycle
    else if   (!idle_stat & (strb0_ # (stat3 & stat2)) &
              (!bg_ # !lock_))                   then do_cycle
    else                                         park;
equations
       !rdy1_ = !xrdy_ & stat3 & ( (outst == fin_cycle) # (outst == park) );
       !gwe_ := !stat3 & stat2 & !idle_stat & ((!busreq_ & !bg_) # !busenable_);
@page

     "Test 1st level global arbitration logic
     test_vectors

     "test reset condition
     ([h1,stat3,stat2,stat1,stat0,lock_,strb0_,bg,xrdy_]->
            [start_state,park_state,busreq_,busenable_,busrdy_,rdy1_,gwe_])
     [ c,   H,    H,  H,    H,   H,         H,     H,   H  ] -> [X,X,X,X,X,X,X];
     [ c,   H,    H,  H,    H,   H,         H,     H,   H  ] -> [X,X,X,X,X,X,X];
     [ c,   H,    H,  H,    H,   H,         H,     H,   H  ] -> [X,X,X,X,X,X,X];
     [ c,   H,    H,  H,    H,   H,         H,     H,   H  ] -> [X,X,X,X,X,X,X];
     [ c,   H,    H,  H,    H,   H,         H,     H,   H  ] -> [X,X,X,X,X,X,X];
     [ c,   H,    H,  H,    H,   H,         H,     H,   H  ] -> [X,X,X,X,X,X,X];

     test_vectors
     "test to make sure you do not get stuck in the fin_cycle state
     ([h1,stat3,stat2,stat1,stat0,lock_,strb0_,bg,xrdy_]->[outst,rdy1_,gwe_])
     [ c,   H,    H,  H,    H,   H,         H,     H,   H  ] -> [ idle,H,H];

     [ c,   H,    H,  H,    L,   H,         H,     L,   H  ] -> [req_cycle,H,H];
     [ c,   H,    H,  H,    L,   H,         H,     H,   H  ] -> [strt_cycl,H,H];
     [ c,   H,    H,  H,    L,   H,         H,     H,   H  ] -> [do_cycle,H,H];
     [ c,   H,    H,  H,    L,   H,         H,     H,   H  ] -> [fin_cycle,H,H];
     [ c,   H,    H,  H,    H,   H,         H,     H,   H  ] -> [park,H,H];
```

Figure 3–1. ABEL Source File for UE10, UE19, US4, and US13 (Continued)

```
test_vectors
([h1,stat3,stat2,stat1,stat0,lock_,strb0_,bg,xrdy_]->[outst,rdy1_,gwe_])
"normal operation
[ c,   H,    H,  H,    H,   H,         H,     L,   H  ] -> [idle,H,H];

[ c,   H,    H,  L,    H,   H,         H,     L,   H  ] -> [req_cycle,H,H];
[ c,   H,    H,  H,    H,   H,         H,     H,   H  ] -> [ idle,H,H];

[ c,   H,    H,  H,    L,   H,         H,     L,   H  ] -> [req_cycle,H,H];
[ c,   L,    H,  H,    L,   H,         H,     H,   H  ] -> [strt_cycl,H,L];
[ c,   L,    H,  H,    L,   H,         H,     H,   H  ] -> [do_cycle,H,L];
[ c,   L,    H,  H,    L,   H,         H,     H,   H  ] -> [fin_cycle,H,L];
[ c,   L,    H,  H,    L,   H,         H,     H,   H  ] -> [park,H,L];

"vector 09
[ c,   L,    H,  H,    H,   H,         H,     L,   H  ] -> [ idle,H,H];
[ c,   H,    H,  H,    H,   H,         H,     H,   H  ] -> [ idle,H,H];
[ c,   H,    L,  H,    L,   H,         H,     L,   H  ] -> [req_cycle,H,H];
[ c,   H,    L,  H,    L,   H,         H,     L,   H  ] -> [req_cycle,H,H];
[ c,   H,    L,  H,    L,   H,         H,     H,   H  ] -> [strt_cycl,H,H];
[ c,   H,    L,  H,    L,   H,         H,     L,   H  ] -> [do_cycle,H,H];
[ c,   H,    L,  H,    L,   H,         H,     L,   H  ] -> [fin_cycle,H,H];
[ c,   L,    L,  H,    L,   H,         H,     L,   H  ] -> [park,H,H];
[ c,   H,    H,  H,    H,   H,         H,     L,   H  ] -> [idle,H,H];

[ c,   L,    L,  H,    L,   H,         H,     L,   H  ] -> [req_cycle,H,H];
[ c,   L,    L,  H,    L,   H,         L,     H,   H  ] -> [strt_cycl,H,H];
[ c,   L,    L,  H,    L,   H,         L,     L,   H  ] -> [do_cycle,H,H];
[ c,   L,    L,  H,    L,   H,         L,     L,   H  ] -> [fin_cycle,H,H];
[ c,   L,    L,  H,    L,   H,         L,     L,   H  ] -> [park,H,H];
[ c,   L,    L,  H,    L,   H,         L,     L,   H  ] -> [fin_cycle,H,H];
[ c,   L,    L,  H,    L,   H,         L,     L,   H  ] -> [park,H,H];
[ c,   L,    L,  H,    L,   H,         L,     L,   H  ] -> [fin_cycle,H,H];
[ c,   L,    L,  H,    L,   H,         L,     L,   H  ] -> [park,H,H];
[ c,   H,    L,  H,    L,   H,         H,     L,   H  ] -> [fin_cycle,H,H];
[ c,   H,    L,  H,    L,   H,         H,     L,   L  ] -> [park,L,H];
[ c,   H,    L,  H,    L,   H,         H,     L,   L  ] -> [fin_cycle,L,H];
[ c,   H,    L,  H,    L,   H,         H,     L,   L  ] -> [park,L,H];

"vector 18
[ c,   H,    H,  L,    L,   H,         H,     L,   H  ] -> [idle,H,H];
[ c,   H,    L,  H,    H,   L,         H,     L,   H  ] -> [req_cycle,H,H];
[ c,   H,    L,  H,    H,   L,         H,     L,   H  ] -> [req_cycle,H,H];
[ c,   H,    L,  H,    H,   H,         H,     H,   H  ] -> [strt_cycl,H,H];
[ c,   H,    L,  H,    H,   L,         H,     H,   H  ] -> [do_cycle,H,H];
[ c,   H,    L,  H,    H,   H,         H,     L,   H  ] -> [fin_cycle,H,H];
[ L,   H,    L,  H,    H,   H,         H,     L,   L  ] -> [fin_cycle,L,H];
[ c,   H,    L,  H,    H,   H,         H,     H,   L  ] -> [park,L,H];
[ L,   H,    L,  H,    H,   H,         H,     H,   H  ] -> [park,H,H];
[ c,   H,    L,  H,    H,   L,         H,     L,   H  ] -> [fin_cycle,H,H];
[ L,   H,    L,  H,    H,   L,         L,     L,   L  ] -> [fin_cycle,L,H];
[ c,   H,    L,  H,    H,   L,         H,     H,   L  ] -> [park,L,H];
[ L,   H,    L,  H,    H,   L,         H,     H,   H  ] -> [park,H,H];
[ c,   L,    L,  L,    H,   L,         H,     H,   H  ] -> [do_cycle,H,H];
[ c,   L,    L,  L,    H,   H,         H,     L,   H  ] -> [fin_cycle,H,H];
[ c,   L,    L,  L,    H,   H,         H,     H,   L  ] -> [park,H,H];
[ c,   H,    H,  H,    H,   H,         H,     H,   H  ] -> [park,H,H];
[ c,   H,    H,  H,    H,   H,         H,     H,   H  ] -> [park,H,H];
[ c,   H,    H,  H,    H,   H,         H,     L,   H  ] -> [idle,H,H];
@page
```

Figure 3–1. ABEL Source File for UE10, UE19, US4, and US13 (Concluded)

```
"vector 37
[ c,   H,    H,  H,    H,   L,        H,     L,   H  ] -> [idle,H,H];
[ c,   H,    L,  L,    L,   L,        H,     L,   H  ] -> [req_cycle,H,H];
[ c,   H,    L,  L,    L,   L,        H,     L,   H  ] -> [req_cycle,H,H];
[ c,   H,    L,  L,    L,   H,        H,     H,   H  ] -> [strt_cycl,H,H];
[ c,   H,    L,  L,    L,   L,        H,     H,   H  ] -> [do_cycle,H,H];
[ c,   H,    L,  L,    L,   H,        H,     L,   H  ] -> [fin_cycle,H,H];
[ L,   H,    L,  L,    L,   H,        H,     L,   L  ] -> [fin_cycle,L,H];
[ c,   H,    L,  L,    L,   H,        H,     H,   L  ] -> [park,L,H];
[ L,   H,    L,  L,    L,   H,        H,     H,   H  ] -> [park,H,H];
[ c,   H,    L,  L,    L,   L,        H,     L,   H  ] -> [fin_cycle,H,H];
[ L,   H,    L,  L,    L,   L,        L,     L,   L  ] -> [fin_cycle,L,H];
[ c,   H,    L,  L,    L,   L,        H,     H,   L  ] -> [park,L,H];
[ L,   H,    L,  L,    L,   L,        H,     H,   H  ] -> [park,H,H];
[ c,   L,    L,  L,    L,   L,        H,     H,   H  ] -> [do_cycle,H,H];
[ c,   H,    L,  L,    L,   L,        H,     L,   H  ] -> [fin_cycle,H,H];
[ c,   H,    L,  L,    L,   L,        H,     L,   H  ] -> [fin_cycle,H,H];
[ L,   H,    L,  L,    L,   L,        H,     L,   L  ] -> [fin_cycle,L,H];
[ c,   H,    L,  L,    L,   L,        H,     L,   H  ] -> [fin_cycle,H,H];
[ c,   H,    L,  L,    L,   H,        H,     H,   L  ] -> [park,L,H];
[ L,   H,    L,  L,    L,   H,        H,     H,   H  ] -> [park,H,H];
[ c,   H,    H,  H,    H,   L,        H,     H,   H  ] -> [park,H,H];
[ c,   H,    H,  H,    H,   L,        H,     H,   H  ] -> [park,H,H];
[ c,   L,    L,  H,    H,   H,        L,     H,   H  ] -> [fin_cycle,H,H];
[ c,   L,    L,  H,    H,   H,        H,     H,   L  ] -> [park,H,H];
[ c,   H,    H,  H,    H,   H,        H,     L,   H  ] -> [idle,H,H];
"vector 62
[ c,   H,    H,  H,    H,   L,        H,     L,   H  ] -> [idle,H,H];
[ c,   H,    L,  H,    H,   L,        H,     L,   H  ] -> [req_cycle,H,H];
[ c,   H,    L,  H,    H,   H,        H,     H,   H  ] -> [strt_cycl,H,H];
[ c,   H,    L,  H,    H,   L,        H,     H,   H  ] -> [do_cycle,H,H];
[ c,   H,    L,  H,    H,   H,        H,     L,   H  ] -> [fin_cycle,H,H];
[ L,   H,    L,  H,    H,   H,        H,     L,   L  ] -> [fin_cycle,L,H];
[ c,   H,    L,  H,    H,   H,        H,     H,   L  ] -> [park,L,H];
[ L,   H,    L,  H,    H,   H,        H,     H,   H  ] -> [park,H,H];
[ c,   H,    L,  H,    H,   L,        H,     L,   H  ] -> [fin_cycle,H,H];
[ c,   H,    L,  H,    H,   L,        L,     L,   H  ] -> [fin_cycle,H,H];
[ L,   H,    L,  H,    H,   L,        L,     L,   L  ] -> [fin_cycle,L,H];
[ c,   H,    L,  H,    H,   L,        H,     H,   L  ] -> [park,L,H];
[ L,   H,    L,  H,    H,   L,        H,     H,   H  ] -> [park,H,H];
[ c,   L,    H,  L,    H,   L,        H,     H,   H  ] -> [do_cycle,H,L];
[ c,   L,    H,  L,    H,   H,        H,     L,   H  ] -> [fin_cycle,H,L];
[ c,   L,    H,  L,    H,   H,        H,     H,   L  ] -> [park,H,L];
[ c,   H,    L,  L,    L,   L,        H,     L,   H  ] -> [fin_cycle,H,H];
[ L,   H,    L,  L,    L,   L,        H,     L,   L  ] -> [fin_cycle,L,H];
[ c,   H,    L,  L,    L,   H,        H,     H,   L  ] -> [park,L,H];
[ L,   H,    L,  L,    L,   H,        H,     H,   H  ] -> [park,H,H];
[ c,   H,    L,  H,    H,   H,        L,     H,   H  ] -> [fin_cycle,H,H];
[ c,   L,    L,  H,    H,   H,        H,     H,   H  ] -> [park,H,H];
[ c,   H,    H,  H,    H,   H,        H,     H,   H  ] -> [park,H,H];
[ c,   H,    H,  H,    H,   H,        H,     H,   H  ] -> [park,H,H];
[ c,   H,    H,  H,    H,   H,        H,     L,   H  ] -> [idle,H,H];
@page
[ c,   H,    H,  H,    L,   H,        H,     L,   H  ] -> [req_cycle,H,H];
[ c,   L,    H,  H,    L,   H,        H,     H,   H  ] -> [strt_cycl,H,L];
[ c,   L,    H,  H,    L,   H,        H,     H,   H  ] -> [do_cycle,H,L];
[ c,   L,    H,  H,    L,   H,        H,     H,   H  ] -> [fin_cycle,H,L];
[ c,   L,    H,  H,    L,   H,        H,     H,   H  ] -> [park,H,L];
[ c,   L,    H,  H,    H,   H,        H,     L,   H  ] -> [ idle,H,H];
[ c,   H,    H,  L,    L,   H,        H,     L,   H  ] -> [req_cycle,H,H];
"([h1,stat3,stat2,stat1,stat0,lock_,strb0_,bg, xrdy_] -> [outst,rdy1_])

end    c40_local_glob_bus_interf
```

Figure 3–2. Reduced Equations for UE10, UE19, US4, and US13

```
– Reduced Equations:

!start_state := (!bg_ & busenable_ & busrdy_ & !busreq_ & park_state &
                 start_state & !stat0
                    # !bg_ & busenable_ & busrdy_ & !busreq_ & park_state &
                 start_state & !stat1
                    # !bg_ & busenable_ & busrdy_ & !busreq_ & park_state &
                 start_state & !stat2);

!park_state := (bg_ & !busenable_ & busrdy_ & !busreq_ & lock_ &
                !park_state & start_state & !stat2 & !stat3 & strb0_
                  # !busenable_ & busrdy_ & !busreq_ & !lock_ &
                !park_state & start_state & stat2 & !stat3 & !strb0_
                  # !bg_ & !busenable_ & busrdy_ & !busreq_ & !park_state
                & start_state & stat2 & !stat3 & !strb0_
                  # !busenable_ & busrdy_ & !busreq_ & !lock_ &
                !park_state & start_state & stat0 & stat1 & stat2
                  # !bg_ & !busenable_ & busrdy_ & !busreq_ & !park_state
                & start_state & stat0 & stat1 & stat2
                  # !busenable_ & !busrdy_ & !busreq_ & park_state &
                start_state & stat0 & stat1 & stat2
                  # !busenable_ & !busrdy_ & !busreq_ & park_state &
                start_state & !xrdy_
                  # !busenable_ & !busrdy_ & !busreq_ & park_state &
                start_state & !stat3);

!busreq_ := (!busenable_ & busrdy_ & !busreq_ & !lock_ & start_state
             # !bg_ & !busenable_ & busrdy_ & !busreq_ & start_state
             # !busenable_ & busrdy_ & !busreq_ & start_state & !stat2
             # !busenable_ & !busreq_ & park_state & start_state
             # !busenable_ & busrdy_ & !busreq_ & park_state
             # busenable_ & busrdy_ & park_state & start_state & !stat0

             # busenable_ & busrdy_ & park_state & start_state & !stat1

             # busenable_ & busrdy_ & park_state & start_state & !stat2);

!busenable_ := (!busenable_ & busrdy_ & !busreq_ & !lock_ & start_state

                # !bg_ & !busenable_ & busrdy_ & !busreq_ & start_state

                # !busenable_ & busrdy_ & !busreq_ & start_state &
               !stat2
                # !busenable_ & !busreq_ & park_state & start_state
                # !busenable_ & busrdy_ & !busreq_ & park_state


                # !bg_ & busrdy_ & !busreq_ & park_state & start_state &
               !stat0
                # !bg_ & busrdy_ & !busreq_ & park_state & start_state &
               !stat1
                # !bg_ & busrdy_ & !busreq_ & park_state & start_state &
               !stat2);

!busrdy_ := (!busenable_ & busrdy_ & !busreq_ & start_state & !stat2 &
             !strb0_
               # !busenable_ & busrdy_ & !busreq_ & start_state & !stat2 &
             stat3
               # !busenable_ & !busreq_ & park_state & start_state &
             !stat0 & stat3 & xrdy_
               # !busenable_ & !busreq_ & park_state & start_state &
             !stat1 & stat3 & xrdy_
               # !busenable_ & !busreq_ & park_state & start_state &
             !stat2 & stat3 & xrdy_
               # !busenable_ & busrdy_ & !busreq_ & park_state &
             start_state);
```

Figure 3–2. Reduced Equations for UE10, UE19, US4, and US13 (Continued)

```
!rdy1_ = (!busenable_ & busrdy_ & !busreq_ & !park_state & start_state
        & stat3 & !xrdy_
          # !busenable_ & !busrdy_ & !busreq_ & park_state & start_state
        & stat3 & !xrdy_);

!gwe_ := (!busenable_ & !stat0 & stat2 & !stat3
          # !busenable_ & !stat1 & stat2 & !stat3
          # !bg_ & !busreq_ & !stat0 & stat2 & !stat3
          # !bg_ & !busreq_ & !stat1 & stat2 & !stat3);



Reduced Equations:

    start_state := !(!bg_ & busenable_ & busrdy_ & !busreq_
                          & park_state & start_state & !stat0
                      # !bg_ & busenable_ & busrdy_ & !busreq_
                          & park_state & start_state & !stat1
                      # !bg_ & busenable_ & busrdy_ & !busreq_
                          & park_state & start_state & !stat2);

    park_state  := !(!busenable_ & !busrdy_ & !busreq_ & park_state
                   & start_state & !stat3
                      # !busenable_ & !busrdy_ & !busreq_ & park_state
                     & start_state & !xrdy_
                      # !busenable_ & !busrdy_ & !busreq_ & park_state
                     & start_state & stat0 & stat1 & stat2
                      # !bg_ & !busenable_ & busrdy_ & !busreq_
                          & !park_state & start_state & stat0
                          & stat1 & stat2
                      # !busenable_ & busrdy_ & !busreq_ & !park_state
                     & !priDMA_ & start_state & stat0 & stat1
         & stat2);

    busreq_     := !(!busenable_ & busrdy_ & !busreq_ & park_state
                      # busenable_ & busrdy_ & park_state
                          & start_state & !stat0
                      # busenable_ & busrdy_ & park_state
                          & start_state & !stat1
                      # busenable_ & busrdy_ & park_state
                          & start_state & !stat2
                      # !busenable_ & !busreq_ & park_state
                          & start_state
                      # !bg_ & !busenable_ & busrdy_ & !busreq_
                          & start_state
                      # !busenable_ & busrdy_ & !busreq_ & !priDMA_
                          & start_state);


    busenable_  := !(!busenable_ & busrdy_ & !busreq_ & park_state
                      # !bg_ & busrdy_ & !busreq_ & park_state
                          & start_state & !stat0
                      # !bg_ & busrdy_ & !busreq_ & park_state
                          & start_state & !stat1
                      # !bg_ & busrdy_ & !busreq_ & park_state
                          & start_state & !stat2
                      # !busenable_ & !busreq_ & park_state
                          & start_state
                      # !bg_ & !busenable_ & busrdy_ & !busreq_
                          & start_state
                      # !busenable_ & busrdy_ & !busreq_ & !priDMA_
                          & start_state);
```

Figure 3–2. Reduced Equations for UE10, UE19, US4, and US13 (Concluded)

```
busrdy_     := !(!busenable_ & busrdy_ & !busreq_ & park_state
                      & start_state
                 # !bg_ & !busenable_ & busrdy_ & !busreq_
                      & start_state & !strb0_
                 # !busenable_ & busrdy_ & !busreq_ & !priDMA_
                      & start_state & !strb0_
                 # !bg_ & !busenable_ & busrdy_ & !busreq_
                      & start_state & !stat2 & stat3
                 # !busenable_ & !busreq_ & park_state
                      & start_state & !stat0 & stat3 & xrdy_
                 # !busenable_ & !busreq_ & park_state
                      & start_state & !stat1 & stat3 & xrdy_
                 # !busenable_ & !busreq_ & park_state
                      & start_state & !stat2 & stat3 & xrdy_
                 # !busenable_ & busrdy_ & !busreq_ & !priDMA_
                      & start_state & !stat2 & stat3);
```

3.2 Global Bus and EPROM Interface Logic—UC10, UC19, UV4, and UV13

UC10, UC19, UV4, and UV13 are TIBPAL16R4 devices that compose the EPROM control and the arbitration control signals logic for the global bus interface.

Figure 3–3 shows the ABEL source file, and Figure 3–4 shows the reduced equation set required to program them.

Figure 3–3. ABEL Source File for UC10, UC19, UV4, and UV13

```
ABEL(tm) 3.10  –  Document Generator
DWG NAME  EPROM and local control
DWG #
COMPANY   TEXAS INSTRUMENTS INCROPORATED
DATE      10/05/90
Reduced equations for Module c40_global_bus_interface

  uc10, uc19, uv4, and uv13    devices       'P16R4'


"inputs
        h3              Pin 1;
        lrw0            Pin 2;
        lstrb0_         Pin 3;
        la22            Pin 4;   "select bit: la22=0 for EPROM,
                                 "la22=1 for flag reg.
        bg_             Pin 7;
        busrdy_         Pin 8;   "busrdy from global interface PAL
        busenable_      Pin 9;   "busenable from global interface PAL

outputs
        epromce_        Pin 19;  "chip enable for the EPROM
        ctrl_enable_    Pin 18;  "enable signal for control lines
        rdy0_           Pin 17;  "rdy signal for shared SRAM
        sync_ae_        Pin 15;  "synchronized busenable signal
        bg_sync_        Pin 14;
        aiord_          Pin 13;  "read signal for the flag register
                                 "(read back reg)
        aioclk          Pin 12;  "write signal for the flag register

name substitutions
        CE_     = ctrl_enable_;
        bry_     = rdy0_;

        "substitutions for test vectors
        c,H,L,X  = .C.,1,0,.X.;

equations
         sync_ae_       :=  busenable_;
        !ctrl_enable_    = !sync_ae_ & !busenable_;
         rdy0_          :=  busrdy_;
        !epromce_        = !la22 &  lrw0 & !lstrb0_;
        !aiord_          =  la22 &  lrw0 & !lstrb0_;
        !aioclk          =  la22 & !lrw0 & !lstrb0_;
        bg_sync_        :=  bg_;

@page
```

Figure 3–3. ABEL Source File for UC10, UC19, UV4, and UV13 (Concluded)

```
"Test 1st level global arbitration logic
test_vectors
([h3,busenable_,busrdy_,la22,lrw0,lstrb0_,bg_]->[CE_,bry_,aiord_,aioclk,epromce_
                                                              ,bg_sync_])
[ c,     H,        H,      H,   H,   H,  H] -> [ H,  H   ,  H   ,   H,    H,   H];
[ c,     L,        H,      H,   H,   H,  H] -> [ L,  H   ,  H   ,   H,    H,   H];
[ c,     L,        H,      H,   H,   H,  L] -> [ L,  H   ,  H   ,   H,    H,   L];
[ L,     L,        L,      H,   H,   H,  H] -> [ L,  H   ,  H   ,   H,    H,   L];
[ c,     L,        L,      H,   H,   H,  H] -> [ L,  L   ,  H   ,   H,    H,   H];
[ c,     H,        H,      H,   H,   H,  H] -> [ H,  H   ,  H   ,   H,    H,   H];
[ c,     H,        H,      H,   H,   H,  H] -> [ H,  H   ,  H   ,   H,    H,   H];
[ c,     H,        H,      H,   H,   H,  H] -> [ H,  H   ,  H   ,   H,    H,   H];
[ c,     H,        H,      H,   H,   H,  H] -> [ H,  H   ,  H   ,   H,    H,   H];
[ L,     L,        H,      H,   H,   H,  L] -> [ H,  H   ,  H   ,   H,    H,   H];
[ c,     L,        H,      H,   H,   H,  H] -> [ L,  H   ,  H   ,   H,    H,   H];
[ c,     L,        H,      H,   H,   H,  H] -> [ L,  H   ,  H   ,   H,    H,   H];
[ c,     L,        L,      H,   H,   H,  H] -> [ L,  L   ,  H   ,   H,    H,   H];
[ c,     L,        H,      H,   H,   H,  L] -> [ L,  H   ,  H   ,   H,    H,   L];
[ c,     L,        L,      H,   H,   H,  L] -> [ L,  L   ,  H   ,   H,    H,   L];
[ L,     H,        H,      H,   H,   H,  H] -> [ H,  L   ,  H   ,   H,    H,   L];
[ c,     H,        H,      X,   X,   H,  H] -> [ H,  H   ,  H   ,   H,    H,   H];
[ c,     H,        H,      H,   H,   L,  H] -> [ H,  H   ,  L   ,   H,    H,   H];
[ c,     H,        H,      L,   H,   L,  L] -> [ H,  H   ,  H   ,   H,    H,   L];

@page

[ c,     H,        H,      H,   L,   L,  L] -> [ H,  H   ,  H   ,   L,    H,   L];
[ c,     H,        H,      L,   L,   L,  L] -> [ H,  H   ,  H   ,   H,    H,   L];
[ c,     H,        H,      H,   H,   H,  L] -> [ H,  H   ,  H   ,   H,    H,   L];
[ c,     H,        H,      X,   X,   H,  L] -> [ H,  H   ,  H   ,   H,    H,   L];
[ c,     L,        H,      X,   X,   H,  H] -> [ L,  H   ,  H   ,   H,    H,   H];
[ c,     L,        H,      X,   X,   H,  H] -> [ L,  H   ,  H   ,   H,    H,   H];
[ L,     L,        L,      X,   X,   H,  H] -> [ L,  H   ,  H   ,   H,    H,   H];
[ c,     L,        L,      H,   H,   L,  H] -> [ L,  L   ,  L   ,   H,    H,   H];
[ c,     H,        H,      L,   H,   L,  H] -> [ H,  H   ,  H   ,   H,    L,   H];
[ c,     H,        H,      H,   L,   L,  H] -> [ H,  H   ,  H   ,   L,    H,   H];
[ c,     H,        H,      L,   L,   L,  H] -> [ H,  H   ,  H   ,   H,    H,   H];
[ c,     H,        H,      X,   X,   H,  H] -> [ H,  H   ,  H   ,   H,    H,   H];
[ L,     L,        H,      X,   X,   H,  H] -> [ H,  H   ,  H   ,   H,    H,   H];
[ c,     L,        H,      X,   X,   H,  H] -> [ L,  H   ,  H   ,   H,    H,   H];
[ c,     L,        H,      H,   H,   L,  H] -> [ L,  H   ,  L   ,   H,    H,   H];
[ c,     L,        L,      H,   L,   L,  H] -> [ L,  L   ,  H   ,   L,    H,   H];
[ c,     L,        H,      L,   L,   L,  H] -> [ L,  H   ,  H   ,   H,    H,   H];
[ c,     L,        L,      H,   L,   L,  H] -> [ L,  L   ,  H   ,   L,    H,   H];

end    c40_global_bus_interface
```

Figure 3–4. Reduced Equations for UC10, UC19, UV4, and UV13

```
– Reduced Equations:

    !sync_ae_ := (!busenable_);

    !ctrl_enable_ = (!busenable_ & !sync_ae_);

    !rdy0_ := (!busrdy_);

    !epromce_ = (!la22 & lrw0 & !lstrb0_);

    !aiord_ = (la22 & lrw0 & !lstrb0_);

    !aioclk = (la22 & !lrw0 & !lstrb0_);

    !bg_sync_ := (!bg_);
```

3.3 Global Bus Controller Arbitrator—UL14

UL14 is a TIBPAL16R8 device that is used as the global bus arbitrator for each TMS320C40 CPU that requests access to the global bus.

Figure 3–5 shows the ABEL source file, and Figure 3–6 shows the reduced equation set required to program UL14.

Figure 3–5. ABEL Source File for UL14

```
ABEL(tm) 3.10  –  Document Generator
DWG NAME  global arbitration
DWG #
COMPANY   TEXAS INSTRUMENTS INCROPORATED
DATE      10/05/90

equations for module global_bus_cntrl

Device ul14

        h50       Pin 1;    ″50 MHz clock
        br1_      Pin 2;    ″bus request 1
        br2_      Pin 4;    ″bus request 2
        br3_      Pin 6;    ″bus request 3
        br4_      Pin 8;    ″bus request 4
        reset_    Pin 5;
        fix_rot   Pin 9;    ″fix_rot = 1 fixed priority
                            ″for shared bus fix_rot = 0 rotating
                            ″priority for shared bus
        timeout_  Pin 7;

        bg1_      Pin 17;   ″grant 1
        bg2_      Pin 16;   ″grant 2
        bg3_      Pin 15;   ″grant 3
        bg4_      Pin 14;   ″grant 4
        s3        Pin 19;   ″state 3
        s2        Pin 18;   ″state 2
        s1        Pin 13;   ″state 1
        s0        Pin 12;   state 0

        c,H,L,X = .C.,1,0,.X.;

        ″define state machine bits
        bus_state = [s3,s2,s1,s0,bg4_,bg3_,bg2_,bg1_];

        ″states

        bry1  = ^b01111111;       ″ready 1
        bry2  = ^b10111111;       ″ready 2
        bry3  = ^b11011111;       ″ready 3
        bry4  = ^b11101111;       ″ready 4
        idle  = ^b11111111;       ″idle state

        gr1   = ^b11111110;       ″grant 1
        gr2   = ^b11111101;       ″grant 2
        gr3   = ^b11111011;       ″grant 3
        gr4   = ^b11110111;       ″grant 4

         ″convert inputs to positive logic
         br1   = !br1_;
         br2   = !br2_;
         br3   = !br3_;
         br4   = !br4_;
         reset = !reset_;
@page
```

Figure 3–5. ABEL Source File for UL14 (Continued)

```
state_diagram  bus_state
       state   idle:
              if (!reset &   !br1 & !br2 & !br3 &  br4)  then gr4
         else if (!reset &   !br1 & !br2 &  br3       )  then gr3
         else if (!reset &   !br1 &  br2              )  then gr2
         else if (!reset &    br1                     )  then gr1
         else     idle;

       state   bry4:
              if (!reset &   br4                       )  then bry4
         else if ( reset # (!br4 & fix_rot)            )  then idle
         else if (!br1 & !br2 & !br3 & !br4            )  then idle
         else if (!reset &  !br1 & !br2 &  br3 & !br4  )  then gr3
         else if (!reset &  !br1 &  br2        & !br4  )  then gr2
         else if (!reset &   br1               & !br4  )  then gr1;

       state   bry3:
              if (!reset &   br3                       ) then bry3
         else if ( reset # (!br3 & fix_rot)            ) then idle
         else if (!br4 & !br1 & !br2 & !br3            ) then idle
         else if (!reset &  !br4 & !br1 &  br2 & !br3  ) then gr2
         else if (!reset &  !br4 & br1         & !br3  ) then gr1
         else if (!reset &   br4               & !br3  ) then gr4;

       state   bry2:
              if (!reset &   br2                       )  then bry2
         else if ( reset # (!br2 & fix_rot)            )  then idle
         else if (!br3 & !br4 & !br1 & !br2            )  then idle
         else if (!reset &  !br3 & !br4 &  br1 & !br2  )  then gr1
         else if (!reset &  !br3 &  br4        & !br2  )  then gr4
         else if (!reset &   br3               & !br2  )  then gr3;

       state   bry1:
              if (!reset &   br1                       )  then bry1
         else if ( reset # (!br1 & fix_rot)            )  then idle
         else if (!br2 & !br3 & !br4 & !br1            )  then idle
         else if (!reset &  !br2 & !br3 &  br4 & !br1  )  then gr4
         else if (!reset &  !br2 &  br3        & !br1  )  then gr3
         else if (!reset &   br2               & !br1  )  then gr2;

       state    gr4:
              if (!reset & (timeout_ # (!br1 & !br2
                                        & !br3  )))  then gr4
         else if ( reset # !br4                      )  then idle
         else if (!reset & (!timeout_ & (br1 # br2
                                        # br3   )))  then bry4;

       state    gr3:
              if (!reset & (timeout_ # (!br4 & !br1
                                        & !br2  )))  then gr3
         else if ( reset # !br3                      )  then idle
         else if (!reset & (!timeout_ & (br1 # br2
                                        # br4   )))  then bry3;

       state    gr2:
              if (!reset & (timeout_ # (!br3 & !br4
                                        & !br1  )))  then gr2
         else if (reset # !br2                       )  then idle
         else if (!reset & (!timeout_ & (br1 # br3
                                        # br4   )))  then bry2;

       state    gr1:
              if (!reset & (timeout_ # (!br2 & !br3
                                      & !br4   )))  then gr1
         else if (reset # !br1                        )  then idle
         else if (!reset & (!timeout_ & (br2 # br3
                                        # br4    )))  then bry1;
@page
```

Figure 3–5. ABEL Source File for UL14 (Continued)

```
test_vectors

"rotating priority vectors
([ h50 ,br1,br2,br3,br4,fix_rot,timeout_,reset_]  -> [bus_state])
"check for go to IDLE
 [  c  , X , X , X ,  X ,   L,  X, L    ] -> [idle];
 [  c  , L , L , L ,  H ,   L,  X, H    ] -> [gr4];
 [  c  , X , X , X ,  X ,   L,  X, L    ] -> [idle];
 [  c  , L , L , H ,  X ,   L,  X, H    ] -> [gr3];
 [  c  , X , X , X ,  X ,   L,  X, L    ] -> [idle];
 [  c  , L , H , X ,  X ,   L,  X, H    ] -> [gr2];
 [  c  , X , X , X ,  X ,   L,  X, L    ] -> [idle];
 [  c  , H , X , X ,  X ,   L,  X, H    ] -> [gr1];
 [  c  , X , X , X ,  X ,   L,  X, L    ] -> [idle];
 [  c  , L , L , L ,  H ,   L,  X, H    ] -> [gr4];
 [  c  , H , L , L ,  H ,   L,  L, H    ] -> [bry4];
 [  c  , X , X , X ,  X ,   L,  X, L    ] -> [idle];
 [  c  , L , L , H ,  X ,   L,  X, H    ] -> [gr3];
 [  c  , L , H , H ,  L ,   L,  L, H    ] -> [bry3];
 [  c  , X , X , X ,  X ,   L,  X, L    ] -> [idle];
 [  c  , L , H , X ,  X ,   L,  X, H    ] -> [gr2];
 [  c  , L , H , H ,  H ,   L,  L, H    ] -> [bry2];
 [  c  , X , X , X ,  X ,   L,  X, L    ] -> [idle];
 [  c  , H , X , X ,  X ,   L,  X, H    ] -> [gr1];
 [  c  , H , H , H ,  H ,   L,  L, H    ] -> [bry1];
 [  c  , X , X , X ,  X ,   L,  X, L    ] -> [idle];

 [  c  , L , L , L ,  H ,   L,  X, H    ] -> [gr4];
 [  c  , X , X , X ,  L ,   L,  X, H    ] -> [idle];
 [  c  , L , L , H ,  X ,   L,  X, H    ] -> [gr3];
 [  c  , X , X , L ,  X ,   L,  X, H    ] -> [idle];
 [  c  , L , H , X ,  X ,   L,  X, H    ] -> [gr2];
 [  c  , X , L , X ,  X ,   L,  X, H    ] -> [idle];
 [  c  , H , X , X ,  X ,   L,  X, H    ] -> [gr1];
 [  c  , L , X , X ,  X ,   L,  X, H    ] -> [idle];
 [  c  , L , L , L ,  H ,   L,  X, H    ] -> [gr4];
 [  c  , H , L , L ,  H ,   L,  L, H    ] -> [bry4];
 [  c  , L , L , L ,  L ,   L,  X, H    ] -> [idle];
 [  c  , L , L , H ,  X ,   L,  X, H    ] -> [gr3];
 [  c  , L , H , H ,  L ,   L,  L, H    ] -> [bry3];
 [  c  , L , L , L ,  L ,   L,  X, H    ] -> [idle];
 [  c  , L , H , X ,  X ,   L,  X, H    ] -> [gr2];
 [  c  , L , H , H ,  H ,   L,  L, H    ] -> [bry2];
 [  c  , L , L , L ,  L ,   L,  X, H    ] -> [idle];
 [  c  , H , X , X ,  X ,   L,  X, H    ] -> [gr1];
 [  c  , H , H , H ,  H ,   L,  L, H    ] -> [bry1];
 [  c  , L , L , L ,  L ,   L,  X, H    ] -> [idle];
"vector 7
 [  c  , H , X , X ,  X ,   L,  X, H    ] -> [gr1];
 [  c  , H , X , X ,  X,    L,  H, H    ] -> [gr1];
 [  c  , H , L , L ,  L ,   L,  L, H    ] -> [gr1];
 [  c  , H , X , X ,  H ,   L,  L, H    ] -> [bry1];
 [  c  , H , X , X ,  X ,   L,  X, H    ] -> [bry1];
 [  c  , H , X , X ,  X ,   L,  X, H    ] -> [bry1];
 @page
```

Figure 3–5. ABEL Source File for UL14 (Continued)

```
"vector 15
 [  c  , L , L , L ,  H ,   L,  X, H    ] -> [gr4];
 [  c  , L , L , L ,  H ,   L,  X, H    ] -> [gr4];
 [  c  , L , L , L ,  H ,   L,  X, H    ] -> [gr4];
 [  c  , X , X , X ,  H ,   L,  H, H    ] -> [gr4];
 [  c  , X , X , H ,  X ,   L,  L, H    ] -> [bry4];
 [  c  , X , X , X ,  H ,   L,  X, H    ] -> [bry4];
 [  c  , X , X , X ,  H ,   L,  X, H    ] -> [bry4];
"vector 21
 [  c  , L , L , H ,  L ,   L,  X, H    ] -> [gr3];
 [  c  , L , L , H ,  L ,   L,  X, H    ] -> [gr3];
 [  c  , L , L , H ,  L ,   L,  H, H    ] -> [gr3];
 [  c  , X , X , H ,  X ,   L,  H, H    ] -> [gr3];
 [  c  , X , H , H ,  X ,   L,  L, H    ] -> [bry3];
 [  c  , X , X , H ,  X ,   L,  X, H    ] -> [bry3];
 [  c  , X , X , H ,  X ,   L,  X, H    ] -> [bry3];
"vector 27
 [  c  , L , H , L ,  L ,   L,  X, H    ] -> [gr2];
 [  c  , L , H , L ,  L ,   L,  X, H    ] -> [gr2];
 [  c  , X , H , X ,  X ,   L,  H, H    ] -> [gr2];
 [  c  , L , H , L ,  L ,   L,  L, H    ] -> [gr2];
 [  c  , H , H , X ,  X ,   L,  L, H    ] -> [bry2];
 [  c  , X , H , X ,  X ,   L,  X, H    ] -> [bry2];
 [  c  , X , H , X ,  X ,   L,  X, H    ] -> [bry2];
"vector 33
 [  c  , H , L , L ,  L ,   L,  X, H    ] -> [gr1];
 [  c  , H , X , H ,  X ,   L,  L, H    ] -> [bry1];
 [  c  , L , L , H ,  X ,   L,  X, H    ] -> [gr3];
 [  c  , H , X , H ,  X ,   L,  L, H    ] -> [bry3];
 [  c  , H , X , L ,  L ,   L,  X, H    ] -> [gr1];
 [  c  , H , H , X ,  X ,   L,  L, H    ] -> [bry1];
 [  c  , L , H , X ,  X ,   L,  X, H    ] -> [gr2];
 [  c  , X , H , X ,  H ,   L,  L, H    ] -> [bry2];
 [  c  , X , L , L ,  H ,   L,  X, H    ] -> [gr4];
 [  c  , X , H , X ,  H ,   L,  L, H    ] -> [bry4];
 [  c  , L , H , X ,  L ,   L,  X, H    ] -> [gr2];
 [  c  , X , H , H ,  X ,   L,  L, H    ] -> [bry2];
"vector 45
 [  c  , X , L , H ,  X ,   L,  X, H    ] -> [gr3];
 [  c  , X , X , H ,  H ,   L,  L, H    ] -> [bry3];
 [  c  , X , X , L ,  H ,   L,  X, H    ] -> [gr4];
 [  c  , H , X , X ,  H ,   L,  L, H    ] -> [bry4];
 [  c  , H , X , X ,  L ,   L,  X, H    ] -> [gr1];
 [  c  , H , H , X ,  X ,   L,  L, H    ] -> [bry1];

@page
```

Figure 3–5. ABEL Source File for UL14 (Continued)

```
" fixed priority vectors
"check for go to IDLE
 [  c  , X , X , X ,  X ,    H,  X, L    ] -> [idle];
 [  c  , L , L , L ,  H ,    H,  X, H    ] -> [gr4];
 [  c  , X , X , X ,  X ,    H,  X, L    ] -> [idle];
 [  c  , L , L , H ,  X ,    H,  X, H    ] -> [gr3];
 [  c  , X , X , X ,  X ,    H,  X, L    ] -> [idle];
 [  c  , L , H , X ,  X ,    H,  X, H    ] -> [gr2];
 [  c  , X , X , X ,  X ,    H,  X, L    ] -> [idle];
 [  c  , H , X , X ,  X ,    H,  X, H    ] -> [gr1];
 [  c  , X , X , X ,  X ,    H,  X, L    ] -> [idle];
 [  c  , L , L , L ,  H ,    H,  X, H    ] -> [gr4];
 [  c  , H , L , L ,  H ,    H,  L, H    ] -> [bry4];
 [  c  , L , L , L ,  L ,    H,  X, L    ] -> [idle];
 [  c  , L , L , H ,  X ,    H,  X, H    ] -> [gr3];
 [  c  , L , H , H ,  L ,    H,  L, H    ] -> [bry3];
 [  c  , L , L , L ,  L ,    H,  X, L    ] -> [idle];

 [  c  , L , L , L ,  H ,    H,  X, H    ] -> [gr4];
 [  c  , X , X , X ,  L ,    H,  X, H    ] -> [idle];
 [  c  , L , L , H ,  X ,    H,  X, H    ] -> [gr3];
 [  c  , X , X , L ,  X ,    H,  X, H    ] -> [idle];
 [  c  , L , H , X ,  X ,    H,  X, H    ] -> [gr2];
 [  c  , X , L , X ,  X ,    H,  X, H    ] -> [idle];
 [  c  , H , X , X ,  X ,    H,  X, H    ] -> [gr1];
 [  c  , L , X , X ,  X ,    H,  X, H    ] -> [idle];
 [  c  , L , L , L ,  H ,    H,  X, H    ] -> [gr4];
 [  c  , H , L , L ,  H ,    H,  L, H    ] -> [bry4];
 [  c  , L , L , L ,  L ,    H,  X, H    ] -> [idle];
 [  c  , L , L , H ,  X ,    H,  X, H    ] -> [gr3];
 [  c  , L , H , H ,  L ,    H,  L, H    ] -> [bry3];
 [  c  , L , L , L ,  L ,    H,  X, H    ] -> [idle];
 [  c  , L , H , X ,  X ,    H,  X, H    ] -> [gr2];
 [  c  , L , H , H ,  H ,    H,  L, H    ] -> [bry2];
 [  c  , L , L , L ,  L ,    H,  X, H    ] -> [idle];
 [  c  , H , X , X ,  X ,    H,  X, H    ] -> [gr1];
 [  c  , H , H , H ,  H ,    H,  L, H    ] -> [bry1];
 [  c  , L , L , L ,  L ,    H,  X, H    ] -> [idle];
"vector 51
 [  c  , H , X , X ,  X ,    H,  X, H    ] -> [gr1];
 [  c  , H , X , X ,  X,     H,  H, H    ] -> [gr1];
 [  c  , H , L , L ,  L ,    H,  L, H    ] -> [gr1];
 [  c  , H , X , X ,  H ,    H,  L, H    ] -> [bry1];
 [  c  , H , X , X ,  X ,    H,  X, H    ] -> [bry1];
 [  c  , H , X , X ,  X ,    H,  X, H    ] -> [bry1];
 [  c  , H , X , X ,  X ,    H,  X, H    ] -> [bry1];
"vector 64
 [  c  , L , X , X ,  X ,    H,  X, H    ] -> [idle];
 [  c  , L , L , L ,  H ,    H,  X, H    ] -> [gr4];
 [  c  , L , L , L ,  H ,    H,  L, H    ] -> [gr4];
 [  c  , X , X , X ,  H ,    H,  H, H    ] -> [gr4];
 [  c  , L , L , L ,  H ,    H,  L, H    ] -> [gr4];
 [  c  , X , X , H ,  H ,    H,  L, H    ] -> [bry4];
 [  c  , X , X , X ,  H ,    H,  X, H    ] -> [bry4];
 [  c  , X , X , X ,  H ,    H,  X, H    ] -> [bry4];
 @page
```

Figure 3–5. ABEL Source File for UL14 (Concluded)

```
"vector 70
 [  c  , X , X , X ,  L ,   H,  X, H    ] -> [idle];
 [  c  , L , L , H ,  L ,   H,  X, H    ] -> [gr3];
 [  c  , L , L , H ,  L ,   H,  L, H    ] -> [gr3];
 [  c  , X , X , H ,  X ,   H,  H, H    ] -> [gr3];
 [  c  , L , L , H ,  L ,   H,  L, H    ] -> [gr3];
 [  c  , X , H , H ,  X ,   H,  L, H    ] -> [bry3];
 [  c  , X , X , H ,  X ,   H,  X, H    ] -> [bry3];
 [  c  , X , X , H ,  X ,   H,  X, H    ] -> [bry3];
"vector 76
 [  c  , X , X , L ,  X ,   H,  X, H    ] -> [idle];
 [  c  , L , H , L ,  X ,   H,  X, H    ] -> [gr2];
 [  c  , L , H , L ,  L ,   H,  L, H    ] -> [gr2];
 [  c  , X , H , X ,  X ,   H,  H, H    ] -> [gr2];
 [  c  , L , H , L ,  L ,   H,  H, H    ] -> [gr2];
 [  c  , H , H , X ,  X ,   H,  L, H    ] -> [bry2];
 [  c  , X , H , X ,  X ,   H,  X, H    ] -> [bry2];
 [  c  , X , H , X ,  X ,   H,  X, H    ] -> [bry2];
"vector 82
 [  c  , X , L , X ,  X ,   H,  X, H    ] -> [idle];
 [  c  , H , L , X ,  X ,   H,  X, H    ] -> [gr1];
 [  c  , H , X , H ,  X ,   H,  L, H    ] -> [bry1];
 [  c  , L , X , X ,  X ,   H,  X, H    ] -> [idle];
 [  c  , L , L , H ,  X ,   H,  X, H    ] -> [gr3];
 [  c  , H , X , H ,  X ,   H,  L, H    ] -> [bry3];
 [  c  , L , X , L ,  X ,   H,  X, H    ] -> [idle];
 [  c  , H , X , L ,  X ,   H,  L, H    ] -> [gr1];
 [  c  , H , H , X ,  X ,   H,  L, H    ] -> [bry1];
 [  c  , L , X , X ,  L ,   H,  X, H    ] -> [idle];
 [  c  , L , H , X ,  X ,   H,  L, H    ] -> [gr2];
 [  c  , X , H , X ,  H ,   H,  L, H    ] -> [bry2];
 [  c  , L , L , X ,  X ,   H,  X, H    ] -> [idle];
 [  c  , L , L , L ,  H ,   H,  L, H    ] -> [gr4];
 [  c  , X , H , X ,  H ,   H,  L, H    ] -> [bry4];
 [  c  , L , L , X ,  L ,   H,  X, H    ] -> [idle];
 [  c  , L , H , X ,  L ,   H,  L, H    ] -> [gr2];
 [  c  , X , H , H ,  X ,   H,  L, H    ] -> [bry2];
 [  c  , H , L , X ,  X ,   H,  X, H    ] -> [idle];
"vector 94
 [  c  , L , L , H ,  X ,   H,  L, H    ] -> [gr3];
 [  c  , X , X , H ,  H ,   H,  L, H    ] -> [bry3];
 [  c  , L , H , L ,  X ,   H,  X, H    ] -> [idle];
 [  c  , L , L , L ,  H ,   H,  L, H    ] -> [gr4];
 [  c  , H , X , X ,  H ,   H,  L, H    ] -> [bry4];
 [  c  , L , X , X ,  L ,   H,  X, H    ] -> [idle];
 [  c  , H , X , X ,  L ,   H,  L, H    ] -> [gr1];
 [  c  , H , H , X ,  X ,   H,  L, H    ] -> [bry1];
end global_bus_cntrl
```

Figure 3–6. Reduced Equations for UL14

```
– Reduced Equations:

    !s3   := (!bg1_ & bg2_ & bg3_ & bg4_ & !br1_ & !br4_ & reset_
                   & s0 & s1 & s2 & s3 & !timeout_
              # !bg1_ & bg2_ & bg3_ & bg4_ & !br1_ & !br3_ & reset_
                   & s0 & s1 & s2 & s3 & !timeout_
              # !bg1_ & bg2_ & bg3_ & bg4_ & !br1_ & !br2_ & reset_
                   & s0 & s1 & s2 & s3 & !timeout_
              #  bg1_ & bg2_ & bg3_ & bg4_ & !br1_ & reset_ & s0
                   & s1 & s2 & !s3);

    !s2   := (bg1_ & !bg2_ & bg3_ & bg4_ & !br1_ & !br2_ & reset_
                   & s0 & s1 & s2 & s3 & !timeout_
              #  bg1_ & !bg2_ & bg3_ & bg4_ & !br2_ & !br4_ & reset_
                   & s0 & s1 & s2 & s3 & !timeout_
              #  bg1_ & !bg2_ & bg3_ & bg4_ & !br2_ & !br3_ & reset_
                   & s0 & s1 & s2 & s3 & !timeout_
              #  bg1_ & bg2_ & bg3_ & bg4_ & !br2_ & reset_ & s0 & s1
                   & !s2 & s3);

    !s1   := (bg1_ & bg2_ & !bg3_ & bg4_ & !br2_ & !br3_ & reset_
                   & s0 & s1 & s2 & s3 & !timeout_
              #  bg1_ & bg2_ & !bg3_ & bg4_ & !br1_ & !br3_ & reset_
                   & s0 & s1 & s2 & s3 & !timeout_
              #  bg1_ & bg2_ & !bg3_ & bg4_ & !br3_ & !br4_ & reset_
                   & s0 & s1 & s2 & s3 & !timeout_
              #  bg1_ & bg2_ & bg3_ & bg4_ & !br3_ & reset_ & s0
                   & !s1 & s2 & s3);

    !s0   := (bg1_ & bg2_ & bg3_ & !bg4_ & !br3_ & !br4_ & reset_
                  & s0 & s1 & s2 & s3 & !timeout_
              #  bg1_ & bg2_ & bg3_ & !bg4_ & !br2_ & !br4_ & reset_
                   & s0 & s1 & s2 & s3 & !timeout_
              #  bg1_ & bg2_ & bg3_ & !bg4_ & !br1_ & !br4_ & reset_
                   & s0 & s1 & s2 & s3 & !timeout_
              #  bg1_ & bg2_ & bg3_ & bg4_ & !br4_ & reset_ & !s0
                   & s1 & s2 & s3);

    !bg4_ := (bg1_ & bg2_ & bg3_ & !bg4_ & br1_ & br2_ & br3_
                   & reset_ & s0 & s1 & s2 & s3
              #  bg1_ & bg2_ & bg3_ & !bg4_ & reset_ & s0 & s1
                   & s2 & s3 & timeout_
              #  bg1_ & bg2_ & bg3_ & bg4_ & br1_ & br2_ & br3_
                   & !br4_ & !fix_rot & reset_ & s0 & s1 & s2
              #  bg1_ & bg2_ & bg3_ & bg4_ & br2_ & br3_ & !br4_
                   & !fix_rot & reset_ & s0 & s1 & !s2 & s3
              #  bg1_ & bg2_ & bg3_ & bg4_ & br3_ & !br4_ & !fix_rot
                   & reset_ & s0 & !s1 & s2 & s3
              #  bg1_ & bg2_ & bg3_ & br1_ & br2_ & br3_ & !br4_
                   & reset_ & s0 & s1 & s2 & s3);

    !bg3_ := (bg1_ & bg2_ & !bg3_ & bg4_ & br1_ & br2_ & br4_
                   & reset_ & s0 & s1 & s2 & s3
              #  bg1_ & bg2_ & !bg3_ & bg4_ & reset_ & s0 & s1
                   & s2 & s3 & timeout_
              #  bg1_ & bg2_ & bg3_ & bg4_ & br1_ & br2_ & !br3_
                   & !fix_rot & reset_& s0 & s1 & s2
              #  bg1_ & bg2_ & bg3_ & bg4_ & br2_ & !br3_ & !fix_rot
                   & reset_ & s0 & s1 & !s2 & s3
              #  bg1_ & bg2_ & bg3_ & bg4_ & br1_ & br2_ & !br3_
                   & br4_ & !fix_rot & reset_ & s1 & s2 & s3
              #  bg1_ & bg2_ & bg3_ & bg4_ & br1_ & br2_ & !br3_
                   & reset_& s0 & s1 & s2 & s3);
```

Figure 3–6. Reduced Equations for UL14 (Concluded)

```
!bg2_ := (bg1_ & !bg2_ & bg3_ & bg4_ & br1_ & br3_ & br4_
          & reset_ & s0 & s1 & s2 & s3
        #  bg1_ & !bg2_ & bg3_ & bg4_ & reset_ & s0 & s1
          & s2 & s3 & timeout_
        #  bg1_ & bg2_ & bg3_ & bg4_ & br1_ & !br2_
          & !fix_rot & reset_ & s0 & s1 & s2
        #  bg1_ & bg2_ & bg3_ & bg4_ & br1_ & !br2_ & br3_
          & br4_ & !fix_rot & reset_ & s0 & s2 & s3
        #  bg1_ & bg2_ & bg3_ & bg4_ & br1_ & !br2_ & br4_
          & !fix_rot & reset_& s1 & s2 & s3
        #  bg1_ & bg2_ & bg3_ & bg4_ & br1_ & !br2_ & reset_
          & s0 & s1 & s2 & s3);

!bg1_ := (!bg1_ & bg2_ & bg3_ & bg4_ & br2_ & br3_ & br4_
          & reset_ & s0 & s1 & s2 & s3
        # !bg1_ & bg2_ & bg3_ & bg4_ & reset_ & s0 & s1 & s2
          & s3 & timeout_
        #  bg1_ & bg2_ & bg3_ & bg4_ & !br1_ & br2_ & br3_
          & br4_ & !fix_rot & reset_& s0 & s1 & s3
        #  bg1_ & bg2_ & bg3_ & bg4_ & !br1_ & br3_ & br4_
          & !fix_rot & reset_ & s0 & s2 & s3
        #  bg1_ & bg2_ & bg3_ & bg4_ & !br1_ & br4_
          & !fix_rot & reset_ & s1 & s2 & s3
        #  bg1_ & bg2_ & bg3_ & bg4_ & !br1_ & reset_ & s0
          & s1 & s2 & s3);
```

3.4 Global Bus Timeout Controller—UK14

UK14 is a TIBPAL16R6 device that is part of the global bus controller circuitry that controls the timeout function for the shared GBC.

Figure 3–7 shows the ABEL source file, and Figure 3–8 shows the reduced equation set required for programming UK14.

Figure 3–7. ABEL Source File for UK14

```
ABEL(tm) 3.10 – Document Generator

DWG NAME  global arbitration
DWG #
COMPANY   TEXAS INSTRUMENTS INCROPORATED
DATE      10/05/90
Equations for Module c40_global_timeout

       uk14          device        'P16R6'

        "inputs
        h50           Pin 1;
        bg1_          Pin 2;
        bg2_          Pin 3;
        bg3_          Pin 4;
        bg4_          Pin 5;
        timeout_      Pin 13; "output
        s1            Pin 16;
        s0            Pin 15;

        "name substitution to increase readability
        bus_active = (!bg1_ # !bg2_ # !bg3_ # !bg4_);

        "define machine state bits
        "[timeout_,s1,s0];

        "states

        idle     = ^b111;
        count1   = ^b110;
        count2   = ^b101;
        count3   = ^b100;
        time     = ^b011;

        outstate = [timeout_,s1,s0];

        c,H,L,X  = .C.,1,0,.X.;
```

Figure 3–7. ABEL Source File for UK14 (Concluded)

```
state_diagram outstate

state idle:
      if (!bus_active) then idle
      else                  count1;

state count1:
      if (!bus_active) then idle
      else                  count2;

state count2:
      if (!bus_active) then idle
      else                  count3;

state count3:
      if (!bus_active) then idle
      else                  time;

state time:    GOTO idle;

@page

"Test counter
test_vectors

([h50, bg1_, bg2_, bg3_, bg4_] -> [outstate])
 [ c,   H,     H,     H,     H  ] -> [ idle   ];
 [ c,   L,     H,     H,     H  ] -> [count1  ];
 [ c,   H,     H,     H,     H  ] -> [ idle   ];
 [ c,   L,     H,     H,     H  ] -> [count1  ];
 [ c,   X,     X,     X,     X  ] -> [count2  ];
 [ c,   H,     H,     H,     H  ] -> [ idle   ];
 [ c,   L,     H,     H,     H  ] -> [count1  ];
 [ c,   X,     X,     X,     X  ] -> [count2  ];
 [ c,   X,     X,     X,     X  ] -> [count3  ];
 [ c,   H,     H,     H,     H  ] -> [ idle   ];
 [ c,   L,     H,     H,     H  ] -> [count1  ];
 [ c,   X,     X,     X,     X  ] -> [count2  ];
 [ c,   X,     X,     X,     X  ] -> [count3  ];
 [ c,   X,     X,     X,     X  ] -> [time    ];
 [ c,   X,     X,     X,     X  ] -> [ idle   ];
 [ c,   H,     L,     H,     H  ] -> [count1  ];
 [ c,   X,     X,     X,     X  ] -> [count2  ];
 [ c,   X,     X,     X,     X  ] -> [count3  ];
 [ c,   X,     X,     X,     X  ] -> [time    ];
 [ c,   X,     X,     X,     X  ] -> [ idle   ];
 [ c,   H,     H,     L,     H  ] -> [count1  ];
 [ c,   X,     X,     X,     X  ] -> [count2  ];
 [ c,   X,     X,     X,     X  ] -> [count3  ];
 [ c,   X,     X,     X,     X  ] -> [time    ];
 [ c,   X,     X,     X,     X  ] -> [ idle   ];
 [ c,   H,     H,     H,     L  ] -> [count1  ];
 [ c,   X,     X,     X,     X  ] -> [count2  ];
 [ c,   X,     X,     X,     X  ] -> [count3  ];
 [ c,   X,     X,     X,     X  ] -> [time    ];
 [ c,   X,     X,     X,     X  ] -> [ idle   ];

end    c40_global_timeout
```

Figure 3–8. Reduced Equations for UK14

```
– Reduced Equations:

    !timeout_ := (!bg4_ & !s0 & !s1 & timeout_
                  # !bg3_ & !s0 & !s1 & timeout_
                  # !bg2_ & !s0 & !s1 & timeout_
                  # !bg1_ & !s0 & !s1 & timeout_);

    !s1       := (!bg4_ & s0 & !s1 & timeout_
                  # !bg3_ & s0 & !s1 & timeout_
                  # !bg2_ & s0 & !s1 & timeout_
                  # !bg1_ & s0 & !s1 & timeout_
                  # !bg4_ & !s0 & s1 & timeout_
                  # !bg3_ & !s0 & s1 & timeout_
                  # !bg2_ & !s0 & s1 & timeout_
                  # !bg1_ & !s0 & s1 & timeout_);

    !s0       := (!bg4_ & s0 & timeout_
                  # !bg3_ & s0 & timeout_
                  # !bg2_ & s0 & timeout_
                  # !bg1_ & s0 & timeout_
```

3.5 Global Memory Control Logic—UL6

Device UL6 is a TIBPAL16L8 programmable logic device that is used to issue control signals to the shared SRAM buffers and memory devices.

Figure 3–9 shows the ABEL source file, and Figure 3–10 shows the reduced equation set required to program UL6.

Figure 3–9. ABEL Source File for UL6

```
module   c40_global_sig_logic
title'
DWG NAME  global memory buffers/control
DWG #
COMPANY   TEXAS INSTRUMENTS INCROPORATED
DATE      2/10/91'

       ul6          device         'P16L8';

        "inputs

        ga16         Pin 1;  "select bit: ga16=0 for PAGE 0, ga16=1 "
                             "for PAGE 1                            "
        gstrb0_      Pin 2;
        gstrb1_      Pin 4;
        grw1         Pin 5;
        agwe_        Pin 6;  "a–d, write enable signals from C40s   "
        bgwe_        Pin 7;
        cgwe_        Pin 8;
        dgwe_        Pin 9;

        "outputs
        gmce0_       Pin 19; "chip enable for PAGE 0 of shared SRAM "
        gmrw0        Pin 18; "read/write signal for PAGE 0 of       "
                             "shared  SRAM                          "
        gmce1_       Pin 16; "chip enable for PAGE 1 of shared SRAM "
        gmrw1        Pin 15; "read/write signal for PAGE 1 of       "
                             "shared SRAM                           "
        gden_        Pin 13; "enable signal for connector buffer    "
        gddir        Pin 12; "direction signal for connector buffer "

        "name substitutions

        gs0_ = gstrb0_;
        gs1_ = gstrb1_;
        a_   = agwe_;
        b_   = bgwe_;
        c_   = cgwe_;
        d_   = dgwe_;

        H,L,X  = 1,0,.X.;

equations

        !gmce0_ = !ga16 & !gstrb0_;
        !gmce1_ =  ga16 & !gstrb0_;
        !gmrw0  = !ga16 & (!agwe_ # !bgwe_ # !cgwe_ # !dgwe_);
        !gmrw1  =  ga16 & (!agwe_ # !bgwe_ # !cgwe_ # !dgwe_);

        !gden_  = !gstrb1_;
         gddir  =  !grw1;

@page
```

Figure 3–9. ABEL Source File for UL6 (Concluded)

```
"Test 1st level global arbitration logic
test_vectors
([ga16,gs0_,gs1_,grw1,a_,b_,c_,d_] -> [gmce0_,gmce1_,gmrw0,gmrw1,gden_,gddir])
"vectors to check shared memory chip enables
 [  H,  H,   H,    H,  H, H, H, H] -> [  H,     H,     H,    H,     H,     L  ];
 [  H,  L,   H,    H,  H, H, H, H] -> [  H,     L,     H,    H,     H,     L  ];
 [  L,  H,   H,    H,  H, H, H, H] -> [  H,     H,     H,    H,     H,     L  ];
 [  L,  L,   H,    H,  H, H, H, H] -> [  L,     H,     H,    H,     H,     L  ];
 [  H,  H,   H,    H,  H, H, H, H] -> [  H,     H,     H,    H,     H,     L  ];
 [  H,  L,   H,    H,  L, H, H, H] -> [  H,     L,     H,    L,     H,     L  ];
 [  L,  H,   H,    H,  L, H, H, H] -> [  H,     H,     L,    H,     H,     L  ];
 [  L,  L,   H,    H,  L, H, H, H] -> [  L,     H,     L,    H,     H,     L  ];
 [  H,  H,   H,    H,  L, H, H, H] -> [  H,     H,     H,    L,     H,     L  ];
 [  H,  L,   H,    H,  H, L, H, H] -> [  H,     L,     H,    L,     H,     L  ];
 [  L,  H,   H,    H,  H, L, H, H] -> [  H,     H,     L,    H,     H,     L  ];
 [  L,  L,   H,    H,  H, L, H, H] -> [  L,     H,     L,    H,     H,     L  ];
 [  H,  H,   H,    H,  H, L, H, H] -> [  H,     H,     H,    L,     H,     L  ];
 [  H,  L,   H,    H,  H, H, L, H] -> [  H,     L,     H,    L,     H,     L  ];
 [  L,  H,   H,    H,  H, H, L, H] -> [  H,     H,     L,    H,     H,     L  ];
 [  L,  L,   H,    H,  H, H, L, H] -> [  L,     H,     L,    H,     H,     L  ];
 [  H,  H,   H,    H,  H, H, L, H] -> [  H,     H,     H,    L,     H,     L  ];
 [  H,  L,   H,    H,  H, H, H, L] -> [  H,     L,     H,    L,     H,     L  ];
 [  L,  H,   H,    H,  H, H, H, L] -> [  H,     H,     L,    H,     H,     L  ];
 [  L,  L,   H,    H,  H, H, H, L] -> [  L,     H,     L,    H,     H,     L  ];
"[ga16,gs0_,gs1_,grw1,a_,b_,c_,d_] -> [gmce0_,gmce1_,gmrw0_,gmrw1_,gden_,gddir])

@page

"vectors to check generation of expansion connector buffers
 [  H,  H,   H,    H,  H, H, H, H] -> [  H,     H,     H,    H,     H,     L  ];
 [  H,  H,   H,    L,  H, H, H, H] -> [  H,     H,     H,    H,     H,     H  ];
 [  H,  H,   L,    H,  H, H, H, H] -> [  H,     H,     H,    H,     L,     L  ];
 [  H,  H,   L,    L,  H, H, H, H] -> [  H,     H,     H,    H,     L,     H  ];


end    c40_global_sig_logic
```

Figure 3–10. Reduced Equations for UL6

```
– Reduced Equations:

    !gmce0_ = (!ga16 & !gstrb0_);

    !gmce1_ = (ga16 & !gstrb0_);

    !gmrw0 = (!dgwe_ & !ga16
              # !cgwe_ & !ga16
              # !bgwe_ & !ga16
              # !agwe_ & !ga16);


    !gmrw1 = (!dgwe_ & ga16
              # !cgwe_ & ga16
              # !bgwe_ & ga16
              # !agwe_ & ga16);

    !gden_ = (!gstrb1_);

    !gddir = (grw1);
```

Appendix A

TMS320C4x PPDS Connectors

This appendix gives the pin assignments and explains the major signals available for the expansion bus connector (P3) and the communication port connectors (P5–P12) with TMS320C40 devices installed on the TMS320C4x PPDS.

Topic **Page**

A.1 The Expansion Bus Connector, P3

Table A–1 shows P3 pin assignments in pin order, Table A–2 shows pin assignments alphabetically, and Table A–3 gives a brief description of each signal.

Table A–1. Pin Assignments for the Expansion Bus Connector, P3

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
P3A-1	V_{CC}	P3A-32	$\overline{XRDY}$	P3B-26	XD12	P3C-20	GND	P3D-14	XA23
P3A-2	V_{CC}	P3A-33	XPAGE	P3B-27	XD14	P3C-21	GND	P3D-15	XA25
P3A-3	XINTA	P3A-34	NC	P3B-28	XD16	P3C-22	GND	P3D-16	XA27
P3A-4	XINTB	P3A-35	XH1	P3B-29	XD18	P3C-23	GND	P3D-17	XA29
P3A-5	XINTC	P3A-36	NC	P3B-30	XD20	P3C-24	GND	P3D-18	XA31
P3A-6	XINTD	P3A-37	V_{CC}	P3B-31	XD22	P3C-25	GND	P3D-19	V_{CC}
P3A-7	V_{CC}	P3B-1	V_{CC}	P3B-32	XD24	P3C-26	GND	P3D-20	XD1
P3A-8	$\overline{XIACKA}$	P3B-2	V_{CC}	P3B-33	XD26	P3C-27	GND	P3D-21	XD3
P3A-9	$\overline{XIACKB}$	P3B-3	XA0	P3B-34	XD28	P3C-28	GND	P3D-22	XD5
P3A-10	$\overline{XIACKC}$	P3B-4	XA2	P3B-35	XD30	P3C-29	GND	P3D-23	XD7
P3A-11	$\overline{XIACKD}$	P3B-5	XA4	P3B-36	V_{CC}	P3C-30	GND	P3D-24	XD9
P3A-12	V_{CC}	P3B-6	XA6	P3B-37	V_{CC}	P3C-31	GND	P3D-25	XD11
P3A-13	XTCLK1A	P3B-7	XA8	P3C-1	V_{CC}	P3C-32	GND	P3D-26	XD13
P3A-14	XTCLK1B	P3B-8	XA10	P3C-2	V_{CC}	P3C-33	GND	P3D-27	XD15
P3A-15	XTCLK1C	P3B-9	XA12	P3C-3	GND	P3C-34	GND	P3D-28	XD17
P3A-16	XTCLK1D	P3B-10	XA14	P3C-4	GND	P3C-35	GND	P3D-29	XD19
P3A-17	V_{CC}	P3B-11	XA16	P3C-5	GND	P3C-36	V_{CC}	P3D-30	XD21
P3A-18	NC	P3B-12	XA18	P3C-6	GND	P3C-37	V_{CC}	P3D-31	XD23
P3A-19	NC	P3B-13	XA20	P3C-7	GND	P3D-1	V_{CC}	P3D-32	XD25
P3A-20	NC	P3B-14	XA22	P3C-8	GND	P3D-2	V_{CC}	P3D-33	XD27
P3A-21	NC	P3B-15	XA24	P3C-9	GND	P3D-3	XA1	P3D-34	XD29
P3A-22	NC	P3B-16	XA26	P3C-10	GND	P3D-4	XA3	P3D-35	XD31
P3A-23	NC	P3B-17	XA28	P3C-11	GND	P3D-5	XA5	P3D-36	V_{CC}
P3A-24	NC	P3B-18	XA30	P3C-12	GND	P3D-6	XA7	P3D-37	V_{CC}
P3A-25	NC	P3B-19	V_{CC}	P3C-13	GND	P3D-7	XA9		
P3A-26	NC	P3B-20	XD0	P3C-14	GND	P3D-8	XA11		
P3A-27	XNMI	P3B-21	XD2	P3C-15	GND	P3D-9	XA13		
P3A-28	XRESET	P3B-22	XD4	P3C-16	GND	P3D-10	XA15		
P3A-29	V_{CC}	P3B-23	XD6	P3C-17	GND	P3D-11	XA17		
P3A-30	$\overline{XSTRB}$	P3B-24	XD8	P3C-18	GND	P3D-12	XA19		
P3A-31	XR/$\overline{W}$	P3B-25	XD10	P3C-19	V_{CC}	P3D-13	XA21		

Note: NC means that the pin is not connected.

Table A–2. Pin Assignments for the Expansion Bus Connector, P3 (Alphabetically)

Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin
GND	P3C-3	GND	P3C-35	V_{CC}	P3D-19	XA28	P3B-17	XD27	P3D-33
GND	P3C-4	NC	P3A-18	V_{CC}	P3D-36	XA29	P3D-17	XD28	P3B-34
GND	P3C-5	NC	P3A-19	V_{CC}	P3D-37	XA30	P3B-18	XD29	P3D-34
GND	P3C-6	NC	P3A-20	XA0	P3B-3	XA31	P3D-18	XD30	P3B-35
GND	P3C-7	NC	P3A-21	XA1	P3D-3	XD0	P3B-20	XD31	P3D-35
GND	P3C-8	NC	P3A-22	XA2	P3B-4	XD1	P3D-20	XH1	P3A-35
GND	P3C-9	NC	P3A-23	XA3	P3D-4	XD2	P3B-21	$\overline{\text{XIACKA}}$	P3A-8
GND	P3C-10	NC	P3A-24	XA4	P3B-5	XD3	P3D-21	$\overline{\text{XIACKB}}$	P3A-9
GND	P3C-11	NC	P3A-25	XA5	P3D-5	XD4	P3B-22	$\overline{\text{XIACKC}}$	P3A-10
GND	P3C-12	NC	P3A-26	XA6	P3B-6	XD5	P3D-22	$\overline{\text{XIACKD}}$	P3A-11
GND	P3C-13	NC	P3A-34	XA7	P3D-6	XD6	P3B-23	XINTA	P3A-3
GND	P3C-14	NC	P3A-36	XA8	P3B-7	XD7	P3D-23	XINTB	P3A-4
GND	P3C-15	V_{CC}	P3A-1	XA9	P3D-7	XD8	P3B-24	XINTC	P3A-5
GND	P3C-16	V_{CC}	P3A-2	XA10	P3B-8	XD9	P3D-24	XINTD	P3A-6
GND	P3C-17	V_{CC}	P3A-7	XA11	P3D-8	XD10	P3B-25	$\overline{\text{XNMI}}$	P3A-27
GND	P3C-18	V_{CC}	P3A-12	XA12	P3B-9	XD11	P3D-25	XPAGE	P3A-33
GND	P3C-20	V_{CC}	P3A-17	XA13	P3D-9	XD12	P3B-26	XR/$\overline{\text{W}}$	P3A-31
GND	P3C-21	V_{CC}	P3A-29	XA14	P3B-10	XD13	P3D-26	$\overline{\text{XRDY}}$	P3A-32
GND	P3C-22	V_{CC}	P3A-37	XA15	P3D-10	XD14	P3B-27	$\overline{\text{XRESET}}$	P3A-28
GND	P3C-23	V_{CC}	P3B-1	XA16	P3B-11	XD15	P3D-27	$\overline{\text{XSTRB}}$	P3A-30
GND	P3C-24	V_{CC}	P3B-2	XA17	P3D-11	XD16	P3B-28	XTCLK1A	P3A-13
GND	P3C-25	V_{CC}	P3B-19	XA18	P3B-12	XD17	P3D-28	XTCLK1B	P3A-14
GND	P3C-26	V_{CC}	P3B-36	XA19	P3D-12	XD18	P3B-29	XTCLK1C	P3A-15
GND	P3C-27	V_{CC}	P3B-37	XA20	P3B-13	XD19	P3D-29	XTCLK1D	P3A-16
GND	P3C-28	V_{CC}	P3C-1	XA21	P3D-13	XD20	P3B-30		
GND	P3C-29	V_{CC}	P3C-2	XA22	P3B-14	XD21	P3D-30		
GND	P3C-30	V_{CC}	P3C-19	XA23	P3D-14	XD22	P3B-31		
GND	P3C-31	V_{CC}	P3C-36	XA24	P3B-15	XD23	P3D-31		
GND	P3C-32	V_{CC}	P3C-37	XA25	P3D-15	XD24	P3B-32		
GND	P3C-33	V_{CC}	P3D-1	XA26	P3B-16	XD25	P3D-32		
GND	P3C-34	V_{CC}	P3D-2	XA27	P3D-16	XD26	P3B-33		

Note: NC means that the pin is not connected.

Table A–3. Major Signals for the Expansion Bus Connector

Signal	Number of Pins	Type	Description
XINTA	1	I/O	IIOF0 signal of CPU A
XINTB	1	I/O	IIOF0 signal of CPU B
XINTC	1	I/O	IIOF0 signal of CPU C
XINTD	1	I/O	IIOF0 signal of CPU D
$\overline{\text{IACKA}}$	1	O/B	Interrupt acknowledge signal from CPU A
$\overline{\text{IACKB}}$	1	O/B	Interrupt acknowledge signal from CPU B
$\overline{\text{IACKC}}$	1	O/B	Interrupt acknowledge signal from CPU C
$\overline{\text{IACKD}}$	1	O/B	Interrupt acknowledge signal from CPU D
XTLCKA	1	I/O	Timer clock signal of CPU A
XTLCKB	1	I/O	Timer clock signal of CPU B
XTLCKC	1	I/O	Timer clock signal of CPU C
XTLCKD	1	I/O	Timer clock signal of CPU D
$\overline{\text{XNMI}}$	1	I/B	Nonmaskable interrupt that is shared by all of the TMS320C40s
$\overline{\text{XRDY}}$	1	I	Ready signal
$\overline{\text{XRESET}}$	1	O/B	System reset
$\overline{\text{XSTRB}}$	1	O/B	Global bus access STROBE signal 1 ($\overline{\text{STRB1}}$)
XR/$\overline{\text{W}}$	1	O/B	Global R/W signal 1 (R/$\overline{\text{W1}}$)
XPAGE	1	O/B	Page transistion signal 1 (PAGE1)
XH1	1	O/B	Output clock (H1) from CPU A
XA(30–0)	31	O/B	31-bit address bus that is shared by all of the TMS320C40s
XD(31–0)	32	I/O/B	32-bit data bus that is shared by all of the TMS320C40s
V_{CC}	22	I	
GND	32	I	
NC	11		

Note: I = input signal
O = output signal
B = buffered signals that are driven by devices that have a propagation delay of less than 10 ns
NC = not connected

A.2 External Communication Port Connectors, P5–P12

Connectors P5–P12 provide a means to connect off-board TMS320C40s, A/D and D/A converters, frame grabbers, SCSI controllers, and other peripherals for accessing the TMS320C40s. Communications ports 2 and 5 (COM2,COM5) of each TMS320C40 are connected directly to a pair of connectors:

- TMS320C40 CPU A is connected to connectors P5 and P6.
- TMS320C40 CPU B is connected to connectors P7 and P8.
- TMS320C40 CPU C is connected to connectors P9 and P10.
- TMS320C40 CPU D is connected to connectors P11 and P12.

Table A–4 shows pin assignments for connectors P5–P12.

Table A–4. Pin Assignments for External Communication Connectors, P5–P12

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
P5-1	GND	P7-1	GND	P9-1	GND	P11-1	GND
P5-2	$\overline{\text{XACREQ5}}$	P7-2	$\overline{\text{XBCREQ5}}$	P9-2	$\overline{\text{XCCREQ5}}$	P11-2	$\overline{\text{XDCREQ5}}$
P5-3	$\overline{\text{XACACK5}}$	P7-3	$\overline{\text{XBCACK5}}$	P9-3	$\overline{\text{XCCACK5}}$	P11-3	$\overline{\text{XDCACK5}}$
P5-4	$\overline{\text{XACSTRB5}}$	P7-4	$\overline{\text{XBCSTRB5}}$	P9-4	$\overline{\text{XCCSTRB5}}$	P11-4	$\overline{\text{XDCSTRB5}}$
P5-5	$\overline{\text{XACRDY5}}$	P7-5	$\overline{\text{XBCRDY5}}$	P9-5	$\overline{\text{XCCRDY5}}$	P11-5	$\overline{\text{XDCRDY5}}$
P5-6	GND	P7-6	GND	P9-6	GND	P11-6	GND
P5-7	XAC5D0	P7-7	XBC5D0	P9-7	XCC5D0	P11-7	XDC5D0
P5-8	XAC5D1	P7–8	XBC5D1	P9-8	XCC5D1	P11-8	XDC5D1
P5-9	XAC5D2	P7-9	XBC5D2	P9-9	XCC5D2	P11-9	XDC5D2
P5-10	XAC5D3	P7-10	XBC5D3	P9-10	XCC5D3	P11-10	XDC5D3
P5-11	XAC5D4	P7-11	XBC5D4	P9-11	XCC5D4	P11-11	XDC5D4
P5-12	XAC5D5	P7-12	XBC5D5	P9–12	XCC5D5	P11-12	XDC5D5
P5-13	XAC5D6	P7-13	XBC5D6	P9-13	XCC5D6	P11-13	XDC5D6
P5-14	XAC5D7	P7-14	XBC5D7	P9-14	XCC5D7	P11-14	XDC5D7
P5-15	GND	P7-15	GND	P9-15	GND	P11-15	GND
P5-16	NC	P7-16	NC	P9-16	NC	P11-16	NC
P6-1	GND	P8-1	GND	P10-1	GND	P12-1	GND
P6-2	$\overline{\text{XACREQ2}}$	P8-2	$\overline{\text{XBCREQ2}}$	P10-2	$\overline{\text{XCCREQ2}}$	P12-2	$\overline{\text{XDCREQ2}}$
P6-3	$\overline{\text{XACACK2}}$	P8-3	$\overline{\text{XBCACK2}}$	P10-3	$\overline{\text{XCCACK2}}$	P12-3	$\overline{\text{XDCACK2}}$
P6-4	$\overline{\text{XACSTRB2}}$	P8-4	$\overline{\text{XBCSTRB2}}$	P10-4	$\overline{\text{XCCSTRB2}}$	P12-4	$\overline{\text{XDCSTRB2}}$
P6-5	$\overline{\text{XACRDY2}}$	P8-5	$\overline{\text{XBCRDY2}}$	P10-5	$\overline{\text{XCCRDY2}}$	P12-5	$\overline{\text{XDCRDY2}}$
P6-6	GND	P8-6	GND	P10-6	GND	P12-6	GND
P6-7	XAC2D0	P8-7	XBC2D0	P10-7	XCC2D0	P12-7	XDC2D0
P6-8	XAC2D1	P8-8	XBC2D1	P10-8	XCC2D1	P12-8	XDC2D1
P6-9	XAC2D2	P8-9	XBC2D2	P10-9	XCC2D2	P12-9	XDC2D2
P6-10	XAC2D3	P8-10	XBC2D3	P10-10	XCC2D3	P12-10	XDC2D3
P6-11	XAC2D4	P8-11	XBC2D4	P10-11	XCC2D4	P12-11	XDC2D4
P6-12	XAC2D5	P8-12	XBC2D5	P10-12	XCC2D5	P12-12	XDC2D5
P6-13	XAC2D6	P8-13	XBC2D6	P10-13	XCC2D6	P12-13	XDC2D6
P6-14	XAC2D7	P8-14	XBC2D7	P10-14	XCC2D7	P12-14	XDC2D7
P6-15	GND	P8-15	GND	P10-15	GND	P12-15	GND
P6-16	NC	P8-16	NC	P10-16	NC	P12-16	NC

Note: NC means that the pin is not connected.

Appendix B

TMS320C4x PPDS Schematics

This appendix contains the schematics for the PPDS with TMS320C40 devices installed.

Appendix C

Glossary

A

address: A location in an array of bits, bytes, or words of information.

arithmetic logic unit (ALU): The section of the computer that carries out all arithmetic operations (addition, subtraction, multiplication, division, or comparison) and logic functions.

archiver: A software program that allows the collection of several individual files into a single file called an archive library.

ASCII: American Standard Code for Information Interchange, 1968. The standard set of 7-bit coded characters (8-bit including parity check) used for information interchange among data processing systems, communications systems, and associated equipment. The ASCII set consists of control characters and graphics characters.

assemble: To prepare a machine-language program from a symbolic language program by substituting absolute operation codes for symbolic operation codes and absolute or relocatable addresses for symbolic addresses.

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.

assembly language: A low-level symbolic programming language, closely resembling machine code language and composed of groups of letters — each group representing a single instruction; allows a computer user to write a program using mnemonics instead of numeric instructions.

attribute: A parameter specifying some characteristic or feature to be applied to subsequent pictorial information.

B

base: 1. A reference value. 2. A number that is multiplied by itself as many times as indicated by an exponent. 3. Same as radix.

breakpoint: A place in a routine specified by an instruction, instruction digit, or other condition, where the routine may be interrupted by external intervention or by a monitor routine.

BTT: Breakpoint/trace/timing.

C

central processing unit (CPU): Part of a computer system that contains the main storage, arithmetic unit, and special register groups. It performs arithmetic operations, controls instruction processing, and provides timing signals and other housekeeping operations.

compiler: A translation program that converts a high-level language set of instructions into a target machine's assembly language.

configured memory: Memory that is allocated.

D

DMA: Direct memory access.

download: To call for and receive a file from another computer storage medium.

dump: To copy the contents of all or part of a storage, usually internal storage.

E

emulator: A hardware development system that emulates a device-specific operation.

erasable programmable read-only memory (EPROM): A read-only memory in which stored data can be erased by ultraviolet light or other means and reprogrammed bit by bit with appropriate voltage pulses.

F

fetch: That portion of a computer cycle during which the next instruction is retrieved from memory.

flag: A binary status indicator whose state indicates whether a particular condition has occurred or is in effect.

G

GBC: Global bus controller.

GMICR: Global memory interface control register.

H

housekeeping: Those operations or routines that do not contribute directly to the solution of a computer program, but rather to the organization of the program.

I

icon: A graphic symbol representing a menu item.

interrupt: To stop a process in such a way that it can be resumed.

L

LCSR: Local control synchronization register. The LCSR controls software synchronization between the four TMS320C40s and their dedicated EPROM and SRAM.

linker: A software tool that combines object files to form an object module, which can be loaded into memory and executed.

LMICR: Local memory interface control register.

load: To enter data into storage or working registers.

loop: A sequence of instructions executed repeatedly until a terminal condition prevails.

LSB: Least significant bit.

M

macro: A program made up of one or more sequences of statements or instructions, each sequence represented by a symbolic name and grouped into single instructions.

map file: An output file created by the linker that shows the memory configuration, section composition and allocation, and symbols with the addresses where they are defined.

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

microcomputer: An integrated circuit that consists of a microprocessor, controller, storage registers, some sort of ALU, and memory.

microprocessor: An integrated circuit that can be programmed with stored instructions to perform a wide variety of functions.

mnemonic: An instruction name that the assembler translates into machine code.

MSB: Most significant bit.

O

object file: A file that has been assembled or linked and contains machine-language object code.

operand: Any one of the quantities entering into or arising from an operation, such as the arguments or parameters of an assembly language instruction, assembler directive, or macro directive.

operation: 1. A defined action; namely, the act of obtaining a result from one or more operands in accordance with a rule that completely specifies the result of any permitted combination of operands. 2.The set of such acts specified by a rule, or the rule itself. 3. The act specified by a single computer instruction. 4. A program step undertaken or executed by a computer, e.g., addition, multiplication, extraction, comparison, shift, transfer, etc. 5. The specific action performed by a logic element.

P

PGA: Pin-grid array.

PLD: Programmable logic devices.

programmable read-only memory (PROM): A large-scale integrated circuit chip for storing digital data. It can be erased with ultraviolet light and reprogrammed, or it can be programmed only once, either at the factory or in the field.

R

random-access memory (RAM): A memory element that can be written to, as well as read.

read-only memory (ROM): A semiconductor storage element containing permanent data that cannot be changed.

real time: The actual time during which the physical process of a computation transpires in order that results of the computation interact with the physical process.

register: Temporary storage area for digital data.

S

scrolling: Moving through text strings or graphic strings vertically or horizontally.

SRAM: Static read-only memory.

symbol: A programmer-defined letter, numeral, sign, or other mark that represents the location of a particular datum item, instruction, routine, value, or address.

syntax: The grammatical and structural rules of a language. All higher-level programming languages possess a formal syntax.

T

target memory: Physical memory in a device into which executable object code is loaded.

TAZ: Tool-actuated ZIF.

U

unconfigured memory: Memory that is not defined as part of the device's memory map.

W

window: A specified rectangular area of virtual space shown on the display screen.

Z

ZIF: Zero insertion force.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

O

P

R

S

T

U

W

X

Z

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software 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, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

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

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI 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.