TEXAS INSTRUMENTS

TMS320C32

Addendum to the TMS320C3x User's Guide

User's Guide

1995

Digital Signal Processing Products

TMS320C32 User's Guide

Addendum to the TMS320C3x User's Guide

March 1995

TEXAS INSTRUMENTS

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.

Printed in U.S.A. by
Champion Press
Houston, Texas

Preface

Read This First

About This Document

This is an addendum to the *TMS320C3x User's Guide* (literature number SPRU031) that describes the architecture and features of the TMS320C32 digital signal processor. Features and topics not covered are identical to those of the TMS320C30 and TMS320C31. The chapter and section numbers supplement the same chapter and section numbers of the *TMS320C3x User's Guide*.

How to Use This Document

This document is intended to be used in conjunction with the *TMS320C3x User's Guide* (literature number SPRU031D) and with the *Interfacing Memory to the TMS320C32 DSP Application Report* (literature number SPRA040).

Notational Conventions

This document uses the following conventions.

- Shading is used in tables to indicate features that are new in the 'C32 and not available in the 'C30 and 'C31 devices. Shading is also used to indicate bit values at reset.
- An overbar over a signal name indicates that the signal is active low. The same applies to pin names.
- Program listings, program examples are shown in a `special typeface` similar to a typewriter's.

  Here is a sample section of a program listing:

```
strobes   CALLU  AR0
          STI    R1,*+AR7(4)      ; IOSTRB  —>(DMA src)
          CALLU  AR0
          STI    R1,*+AR7(6)      ; STRB0   —>(DMA dst)
          CALLU  AR0
          STI    R1,*+AR7(8)      ; STRB1   —>(DMA cnt)
```

If You Need Assistance. . .

If you want to. . .	Do this. . .
Order Texas Instruments documentation	Call the Literature Response Center at **(800) 477–8924**
Obtain technical support, report suspected problems	Call the DSP hotline at **(713) 274–2320**, send a FAX to **(713) 274–2324** or to **+33–1–3070–1032** in Europe. You can also send email to **4389750@mcimail.com.**
Obtain TI product updates, application software	Dial the TMS320 Bulletin Board Service (BBS) at **(713) 274–2323** (24 hrs.). Set your modem to 8 bits,1 stop bit, no parity. Supported speeds are from 300 to 14400 bps. In Europe, dial **+44–2–3422–3248**
Access the TMS320 BBS from Internet	Connect via anonymous ftp to **ftp.ti.com** (192.94.94.5), subdirectory /pub/mirrors
Report mistakes or offer suggestions regarding 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** or send email to: **comments@books.sc.ti.com**

Contents

Figures

Tables

Chapter 1

Introduction

The TMS320C32 is the newest product in the TMS320C3x family of DSPs. The 'C32 not only offers the ease of use and performance advantages of 32-bit DSPs, but also offers the device and system cost advantages of 16-bit DSPs. It is also object-code compatible with the 'C3x family and source-code compatible with the 'C4x family, providing a lower-cost device road map for Texas Instruments generations of 32-bit, floating-point DSPs.

1.4 Key Features

Key features of the 'C32:

- ❑ TMS320C3x CPU
- ❑ Instruction cycle time of 33/40/50 ns
- ❑ Two 256 x 32 words of dual-access on-chip RAM blocks
- ❑ Boot loader
- ❑ Serial port
- ❑ Two timers
- ❑ Two channel DMA controller
- ❑ Enhanced memory interface

Chapter 2

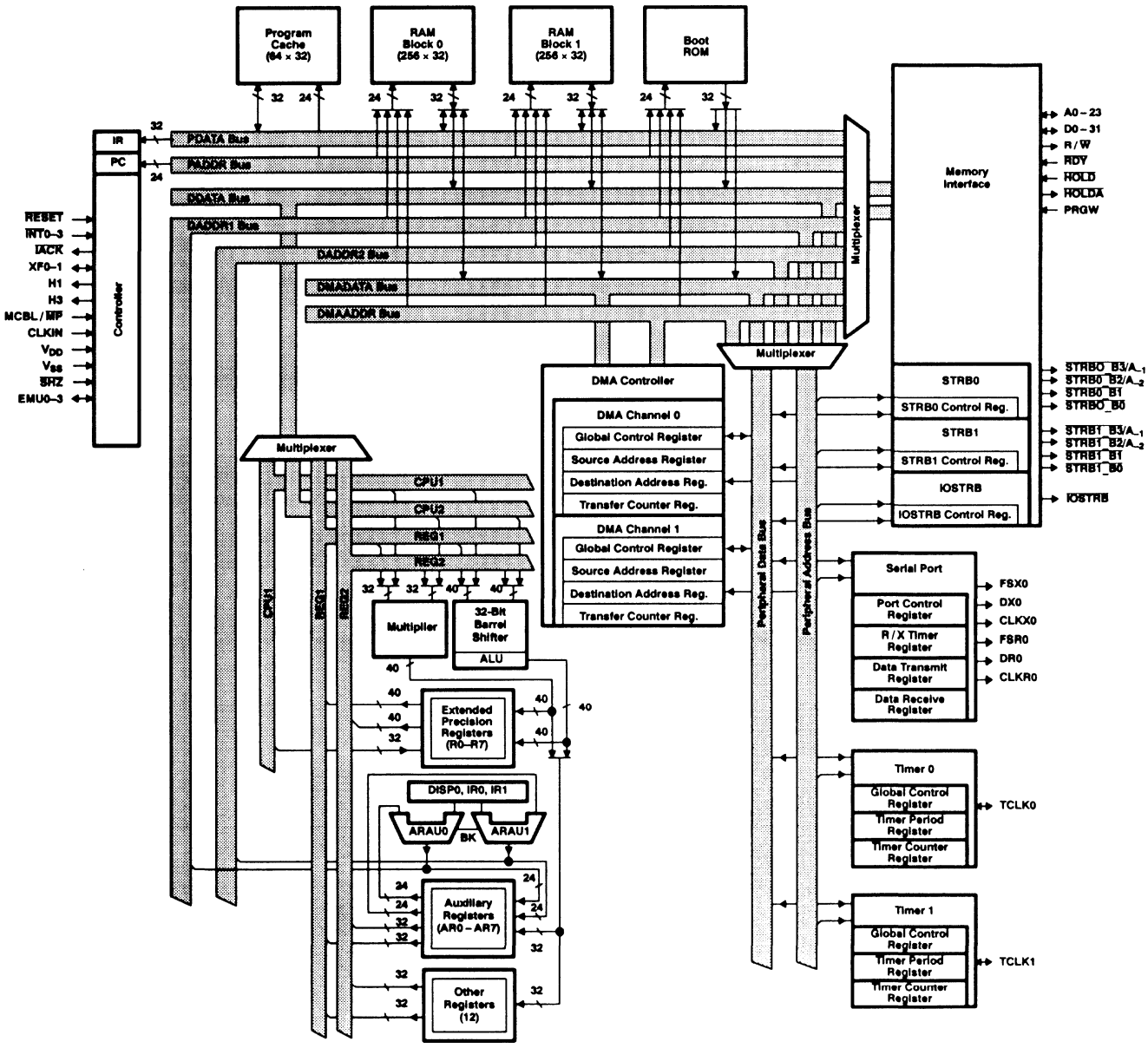
Architecture

This chapter describes the CPU, memory interface, boot loader, peripherals, and DMA of the 'C32. Chapters 3, 4, 6, 7, and 8 describe in more detail the functionality of these components.

2.1 Architectural Overview

Figure 2–1 shows a functional block diagram with the key components of the 'C32.

Figure 2–1. TMS320C32 Functional Block Diagram



2.2 Central Processing Unit

The 'C32 central processing unit (CPU) is an enhanced version of the 'C3x CPU. The enhancements to the CPU include variable-width memory interface, faster instruction cycle time, power-down modes, relocatable interrupt vector table, and edge- or level-triggered interrupts.

2.2.1 Instruction Cycle Time

The fast instruction cycle time of the 'C32 allows it to operate at 33, 40, and 50 ns. This corresponds to external clock rates of 60, 50, and 40 MHz, respectively.

2.2.2 Power Management Modes

Two power management modes, IDLE2 and LOPOWER, have been added to the 'C32 CPU. In IDLE2 mode, no instructions are executed and the CPU, peripherals, and memory retain their previous state while the external bus output pins are idle. During IDLE2 mode, the H1 clock signal is held high while the H3 clock signal is held low until one of the four external interrupts is asserted. In LOPOWER (low power) mode, the CPU continues executing instructions and the DMA continues performing transfers, but at a reduced clock rate. The CLKIN frequency is divided by 16, which makes a 'C32 with a CLKIN frequency of 32 MHz perform like a 2-MHz 'C32, with an instruction cycle time of 1000 ns (or 1 MHz). Refer to Section 6.7 for complete details.

2.2.3 Edge- or Level-Triggered Interrupts

To reduce external logic and simplify the interface, the external interrupts are edge- or level-triggered. The triggering is user-selectable through a bit in the status register. See subsection 3.1.7 for detailed information.

2.2.4 Relocatable Interrupt Vector Table

Unlike the fixed interrupt-trap vector table location of the 'C30 and 'C31 devices, the 'C32 has a user-relocatable interrupt-trap vector table. The interrupt-trap vector table must start on a 256-word boundary. The starting location is programmable through a bit field in the CPU interrupt flag (IF) register: the interrupt-trap table pointer (ITTP). Refer to subsection 3.1.9 for more information.

2.3 Enhanced Memory Interface

The 'C3x family was designed for 32-bit instructions and 32-bit data operations. This architecture has many advantages including a high degree of parallelism and provisions for a C compiler. However, the 'C30 and 'C31 require a 32-bit wide external memory even when the data requires only 8-or 16-bit wide memory. The 'C32 enhanced external memory interface overcomes this limitation by providing the flexibility to address 8-, 16-, or 32-bit data independently of the external memory width. In this way, the chip count and size of external memory is reduced. The number of memory chips can be further reduced by the 'C32 ability to allow code execution from 16- or 32-bit wide memories. The 'C32 memory interface also reduces the total amount of RAM by allowing the physical data memory to be 8-, 16-, or 32-bit wide. Note that internally the 'C32 has a 32-bit architecture. Therefore, you can treat the 'C32 as a 32-bit device regardless of the physical external memory width. The external memory interface handles the conversion between external memory width and 'C32 internal 32-bit architecture. Refer to Chapter 7 for detailed description of the external memory interface.

2.3.1 16- and 32-Bit Program Memory

The 'C32 executes code from either 16- or 32-bit wide memories. When connected to 32-bit memories, 'C32 program execution is identical to that of the 'C31. When connected to 16-bit zero wait-state memory, the 'C32 takes two instruction cycles to fetch a single 32-bit instruction. During the first cycle, the 'C32 fetches the lower 16 bits. During the second cycle, the 'C32 fetches the upper 16 bits and concatenates them with the previously fetched lower 16 bits. This process occurs entirely within the memory interface and is transparent to you. An external pin, PRGW, dictates the external program memory width. Refer to Section 13.2 for signal descriptions.

2.3.2 8-, 16-, and 32-Bit Data Memory

'C32 external memory interface can load and store 8-, 16-, or 32-bit quantities into external memory and convert them into an internally equivalent 32-bit representation. The external memory interface accomplishes this added functionality without changing the CPU instruction set. Figure 2–2 shows the supported external memory widths and data types sizes.

Figure 2–2. 'C32 Supported Data Types Sizes and External Memory Widths

		Memory Width		
		8	16	32
Data	8	♦	♦	♦
Type	16	⊕	♦	♦
Size	32	♣	⊕	♦

♦ Single-cycle read
⊕ Two-cycle read
♣ Four-cycle read

To access 8-/16-/32-bit data quantities (types) from 8-/16-/32-bit wide memory, the memory interface utilizes either strobe $\overline{\text{STRB0}}$ or $\overline{\text{STRB1}}$ depending on the address location within the memory map. Each strobe consists of four pins for byte enables and/or additional address. For 32-bit memory interface, all four pins are used as strobe-byte enable pins. These strobe-byte enable pins select one or more bytes of the external memory. For 16-bit memory interface, the 'C32 uses one of these pins as an additional address pin while using two pins as strobe-byte enable pins. For 8-bit memory interface, the 'C32 uses two of these pins as additional address pins while using one pin as strobe pin. The 'C32 manipulates the behavior of these pins according to the contents of the bus control registers (one control register per strobe). By setting a few bit fields in this register, you indicate the data type size and external memory width. Refer to Chapter 7 for detailed information.

2.4 On-Chip RAM and Boot Loader

The 'C32 has two 256 x 32-bit dual-access on-chip RAM blocks. Each RAM block allows two accesses per instruction cycle by the CPU and/or DMA.

The 'C32 boot loader functionality is equivalent to that of the 'C31, but with additional modes to handle the data type sizes and memory widths supported by the external memory interface (8-, 16-, or 32-bit). 'C32 boot loader loads programs from the serial port, EPROM, or other standard memory devices. The memory boot load supports data transfers with and without handshaking. The handshake mode allow synchronous transfer of program by utilizing two pins as data acknowledge and data ready signals. See Section 3.4 for a detailed description of the boot loader functions.

2.5 Peripherals

The 'C32 peripherals are one serial port, two timers, and two DMA channels. The serial port and timers are functionally identical to those in the 'C31 peripherals. Refer to the *TMS320C3x User's Guide* Chapter 8 for a detailed description. This section covers the difference between the 'C32 DMA channels and the 'C30 or 'C31 DMA channel.

2.5.1 Two-Channel Direct Memory Access (DMA)

The 'C32 has a two-channel DMA controller (one more than the 'C30 or 'C31). Each channel is equivalent to the 'C30/31 DMA with the addition of user-configurable priorities. Because the DMA and CPU have distinct buses on the 'C3x devices, they can operate independently of each other. However, when the CPU and DMA access the same on-chip or external resources, the bandwidth can be exceeded and priorities must be established. The 'C30 and 'C31 assign highest priority to the CPU. The 'C32 DMA controller provides more flexibility by allowing you to choose one of the following priorities:

CPU: For all resource conflicts the CPU has priority over the DMA.

DMA: For all resource conflicts the DMA has priority over the CPU.

Rotating: When the CPU and DMA have a resource conflict during consecutive instruction cycles, the CPU is granted priority. On the following cycle, the DMA is granted priority. Alternate access continues as long as the CPU and DMA requests conflict in consecutive instruction cycles.

The DMA/CPU priority is configured by the DMA PRI bit fields of the corresponding DMA global control register. Refer to Section 8.3 of the *TMS320C3x User's Guide* for a complete description.

Chapter 3

CPU Registers, Memory, and Cache

The new features of the 'C32 required changes to the CPU register file, memory map, and boot loader. The following sections discuss these changes.

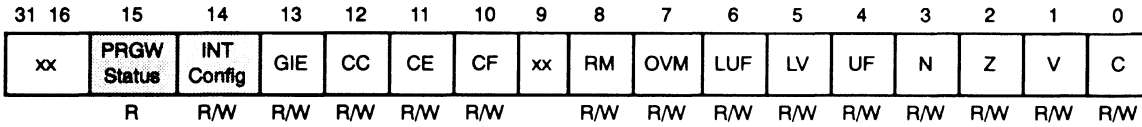
3.1 CPU Register File

Three registers in the CPU register file have been modified to support the new features of the 'C32 (such as: two channel DMAs, program execution from 16-bit memory width, etc.). The modified registers are: the status (ST) register, interrupt enable (IE) register, and interrupt flag (IF) register.

3.1.7 Status Register (ST)

The 'C32's status register (ST) has two new bit fields: INT config and PRGW status. Figure 3–3 shows the bit fields of the status register. At system reset, 0 is written to the ST register. The following paragraphs describe the functionality of these new bit fields.

Figure 3–3. Status Register



The INT Config field sets the external interrupt signals $\overline{INT3}$–$\overline{INT0}$ for level-triggered or edge-triggered interrupts. This field can have the following values (the shaded entry highlights the reset value):

Bit 14	INT Config Function Description
0	All the external interrupts, $\overline{INT3}$–$\overline{INT0}$, are configured as level-triggered interrupts. Multiple interrupts may be triggered when the signal is active for a long period of time.
1	All the external interrupts, $\overline{INT3}$–$\overline{INT0}$, are configured as edge-triggered interrupts. Edge and duration are required for the interrupt to be recognized.

The PRGW Status field indicates the status of the external input PRGW pin. When the signal of the PRGW pin is high, the PRGW status bit is set to 1 indicating a 16-bit program memory width. The 'C32 performs two fetches to retrieve a single 32-bit instruction word. In the first fetch, the 'C32 retrieves the lower 16 bits. In the second fetch, the 'C32 retrieves the upper 16 bits and concatenates them with the lower 16 bits. When the signal of the PRGW pin is low, the PRGW status bit is cleared to 0 indicating a 32-bit program memory width. The 'C32 performs a single fetch to retrieve a single 32-bit instruction word. The PRGW bit is a read-only bit. This field can have the following values:

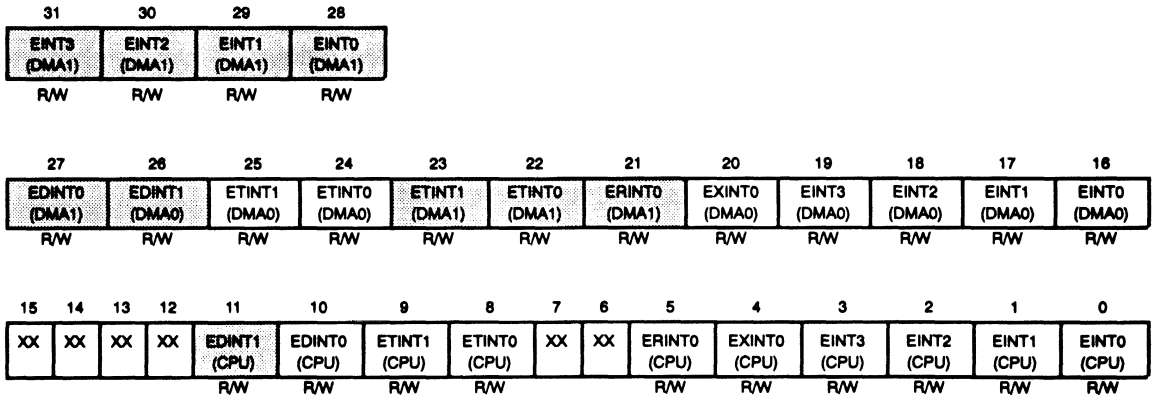
Bit 15	PRGW Status Function Description
0	Instruction fetches utilize a 32-bit external program memory read.
1	Instruction fetches utilize two consecutive 16-bit external program memory reads.

The value of the PRGW bit affects the reset value of the STRB0 and STRB1 control register. See subsections 7.3.1 and 7.3.2 for detailed information.

3.1.8 CPU/DMA Interrupt Enable Register (IE)

The 'C32's CPU/DMA interrupt enable register (IE) has ten new bit fields to support the additional DMA channel interrupts. These are EINT3 (DMA1), EINT2 (DMA1), EINT1 (DMA1), EINT0 (DMA1), EDINT0 (DMA1), EDINT1 (DMA0), ETINIT1 (DMA1), ETINIT0 (DMA1), ERINT0 (DMA1), and EDINT1 (CPU). Figure 3–4 shows the CPU/DMA interrupt enable register. In this figure, the lower 16 bits are used for CPU interrupt enable and the upper 16 bits are used for DMA channels interrupt enable. The corresponding DMA channel (DMA0 or DMA1) are accentuated in parentheses. Note that the serial port receive DMA interrupt is hardwired to DMA1, while the serial port transmit DMA interrupt is hard-wired to DMA0. At system reset, 0 is written to this register.

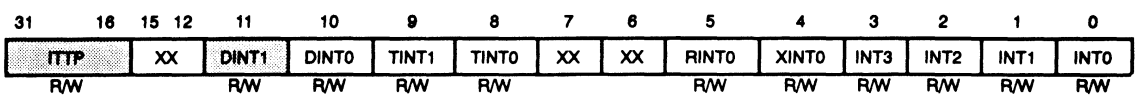
Figure 3–4. CPU/DMA Interrupt Enable Register



3.1.9 CPU Interrupt Flag Register (IF)

The 'C32's CPU interrupt flag register (IF) has two new bit fields: DINT1 and ITTP. Figure 3–5 depicts the CPU interrupt flag register. In this figure, the DINT0 bit refers to DMA channel 0 interrupt flag and DINT1 bit refers to the DMA channel 1 interrupt flag. At system reset, 0 is written to this register.

Figure 3–5. CPU Interrupt Flag Register



3.1.10 Interrupt-Trap Table Pointer (ITTP)

Similarly to the rest of the 'C3x device family, the 'C32's reset vector location remains at address 0. On the other hand, the interrupt and trap vectors are relocatable. This is achieved by a new bit field in the CPU interrupt flag register called the interrupt-trap table pointer (ITTP), shown in Figure 3–5. The ITTP bit field dictates the starting location (base) of the interrupt-trap vector table. This base address is formed by left-shifting by eight bits the value of the ITTP bit field. This shifted value is called the effective base address and is referenced as EA[ITTP], as shown in Figure 3–6. Therefore, the location of an interrupt or trap vector is given by the addition of the effective base address formed by the ITTP bit field (EA[ITTP]) and the offset of the interrupt or trap vector in the interrupt-trap vector table, as shown in Figure 3–7. For example, if the ITTP contains the value 100h, the serial port transmit interrupt vector will be located at 10005h. Note that the vectors stored in the interrupt-trap vector table are the addresses of the start of the respective interrupt and trap routines. Furthermore, the interrupt-trap vector table must lie on a 256-word boundary since the eighth least significant bits of the effective base address of the interrupt-trap vector table are 0.

Figure 3–6. Effective Base Address of the Interrupt-Trap Vector Table

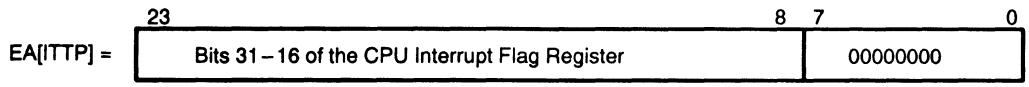


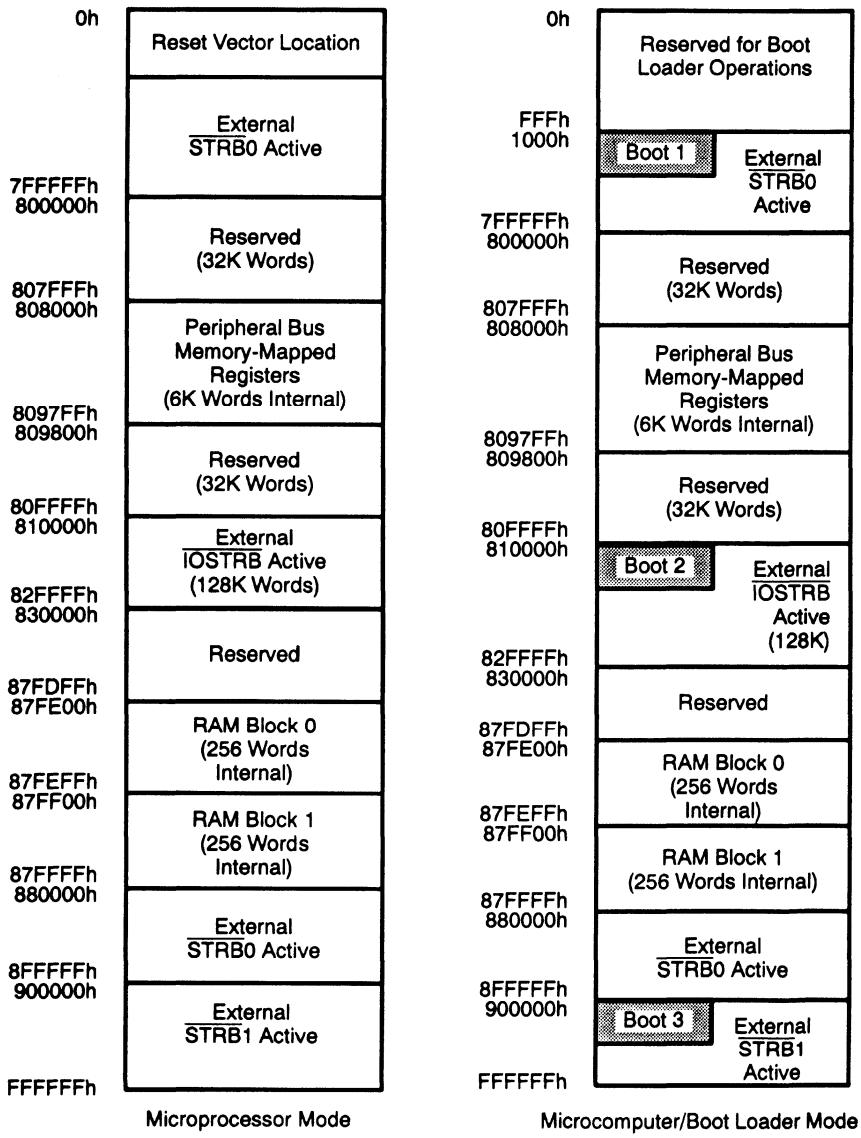
Figure 3–7. Interrupt and Trap Vector Locations

EA[ITTP] + 00h	Reserved
EA[ITTP] + 01h	INT0
EA[ITTP] + 02h	INT1
EA[ITTP] + 03h	INT2
EA[ITTP] + 04h	INT3
EA[ITTP] + 05h	XINT0
EA[ITTP] + 06h	RINT0
EA[ITTP] + 07h	Reserved
EA[ITTP] + 08h	Reserved
EA[ITTP] + 09h	TINT0
EA[ITTP] + 0Ah	TINT1
EA[ITTP] + 0Bh	DINT0
EA[ITTP] + 0Ch	DINT1
EA[ITTP] + 0Dh EA[ITTP] + 1Fh	Reserved
EA[ITTP] + 20h	TRAP 0
	. . .
EA[ITTP] + 3Bh	TRAP 27
EA[ITTP] + 3Ch	TRAP 28 (Reserved)
EA[ITTP] + 3Dh	TRAP 29 (Reserved)
EA[ITTP] + 3Eh	TRAP 30 (Reserved)
EA[ITTP] + 3Fh	TRAP 31 (Reserved)

3.2 Memory Map

The 'C32's memory space of 16 MB has program, data, and I/O spaces. Program and data spaces are controlled by the $\overline{\text{STRB0}}$ and $\overline{\text{STRB1}}$ signals, while the I/O space is controlled by the $\overline{\text{IOSTRB}}$ signal. $\overline{\text{IOSTRB}}$ reads and writes take two cycles to accommodate slow peripheral devices. The MCBL/$\overline{\text{MP}}$ pin determines the operating mode (microprocessor or microcomputer/boot loader) and the memory map configuration. Figure 3–8 shows the 'C32's memory map.

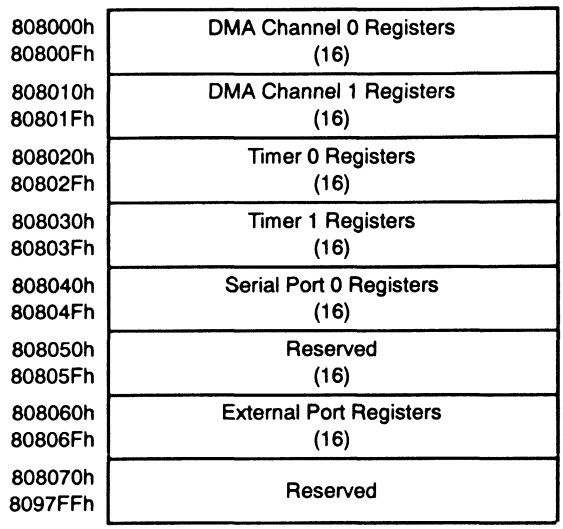
Figure 3–8. TMS320C32 Memory Map



3.2.4 Peripheral-Bus Memory Map

'C32's memory-mapped peripheral and external bus control registers are located starting at address 808000h, as showed in Figure 3–8. Figure 3–11 shows the peripheral bus memory map. Note that each peripheral occupies a 16-word region of the peripheral bus memory map. Also, note that locations 808050h through 80805Fh and 808070h through 8097FFh are reserved.

Figure 3–11. Peripheral-Bus Memory Map



3.4 Boot Loader

The 'C32's boot loader is an enhanced version of that found in the 'C31. The boot loader can load and execute programs received from a host processor via standard memory devices (including EPROM), with and without handshake, or via the serial port. 'C32's boot loader supports 16- and 32-bit program external memory widths, as well as 8-, 16-, and 32-bit data type sizes and external memory widths.

3.4.1 Boot Loader Mode Selection

The 'C32 boot loader functions as a memory boot loader, memory boot loader with handshake, or a serial-port boot loader. The boot loader mode selection is determined by the status of the $\overline{\text{INT}}3$–$\overline{\text{INT}}0$ pins immediately following reset. Table 3–7 lists the boot loader modes. The memory boot loader supports user-definable byte, half-word, and full-word data formats, allowing the flexibility to load a source program from memories having widths of 8-, 16-, and 32-bits with or without handshaking. The source programs to be loaded reside in one of three memory locations: 1000h, 81 0000h, and 90 0000h. The handshaking mode utilizes XF0 and XF1 as data acknowledge and data ready signals, respectively. On the other hand, the serial port boot loader supports 32-bit fixed burst loads from the 'C32's serial port with an externally-generated serial port clock and FSR.

Table 3–7. Boot Loader Mode Selection

$\overline{INT0}$	$\overline{INT1}$	$\overline{INT2}$	$\overline{INT3}$	Boot Loader Mode	Source Program Location
0	1	1	1	External Memory	Boot 1 address 1000h
1	0	1	1	External Memory	Boot 2 address 81 0000h
1	1	0	1	External Memory	Boot 3 address 90 0000h
1	1	1	0	32-bit fixed burst serial	Serial Port
0	1	1	0	External Memory with Handshake	Boot 1 address 1000h, XF0 and XF1 used in handshaking
1	0	1	0	External Memory with Handshake	Boot 2 address 81 0000h, XF0 and XF1 used in handshaking
1	1	0	0	External Memory with Handshake	Boot 3 address 90 0000h, XF0 and XF1 used in handshaking

3.4.2 Boot Loading Sequence

The following is the sequence of events that occur during the boot load of a source program. Table 3–8 shows the structure of the source program.

1) The boot loader mode is invoked by resetting the 'C32 while driving the MCBL/$\overline{MP}$ pin high and the corresponding $\overline{INT3}$–$\overline{INT0}$ pins low. The MCBL/$\overline{MP}$ must stay high during boot loading, but can be changed anytime after boot loading has terminated. No reset is necessary when changing the $\overline{INT3}$–$\overline{INT0}$ pin, as long as the 'C32 is not accessing the overlapping memory (0h–FFFh) during this transition. In nonhandshake mode, one of the $\overline{INT3}$–$\overline{INT0}$ pins can be driven any time after deasserting the $\overline{RESET}$ pin (driven low and then high). While in handshake mode, two interrupt pins have to be asserted before deasserting the $\overline{RESET}$ pin.
2) The status of the interrupt flag (IF) register's INT3–INT0 bit fields dictate the boot loading mode. The bits are polled in the order described in the flow chart in Figure 3–13.
   a) If only the interrupt flag (IF) register's INT3 bit field is set, the boot loader configures the serial port for 32-bit fixed burst mode reads with an externally generated serial port clock and FSR. Then, it proceeds to boot load the source program from the serial port. A header indicating the STRB0, STRB1, and IOSTRB control registers precedes the actual program, refer to Table 3–8. These header values are loaded into the corresponding locations at the completion of the boot load operation. The transferred data-bit order supplied to the serial port must begin with the most significant bit (MSB) and end with the least significant bit (LSB). Figure 3–14 depicts the boot loader serial port flow.
   b) Otherwise, the boot loader attempts a memory boot load. Figure 3–15 shows the boot loader memory flow. If the Interrupt Flag (IF) register's

INT0 bit field is set, the source program is loaded from memory location 1000h. If the Interrupt Flag (IF) register's INT1 bit field is set, the source program is loaded from memory location 810000h. If the Interrupt Flag (IF) register's INT2 bit field is set, the source program is loaded from memory location 900000h. After determining the memory location of the source program, the boot loader checks INT3 bit field in the Interrupt Flag (IF) register. If this bit is set, all data transfers are performed with synchronous handshake. The handshake protocol utilizes XF0 and XF1 as data acknowledge and data ready signals, respectively. 'C32's XF0 is an output pin while the XF1 is an input pin. Figure 3–16 shows the handshake data transfer operation. The data transfer operation occurs as follows:

i) The 'C32's boot loader waits until the host sets XF1 low to read in the data. While the 'C32 waits for XF1 to drop low, the $\overline{\text{IACK}}$ pin pulses. Setting XF1 low communicates to the 'C32 that the data is valid. The IACK pulses indicate that the 'C32 is waiting for data.

ii) The boot loader sets XF0 low after reading the data value. Dropping XF0 acknowledges to the host that the data was read.

iii) The host sets XF1 high to inform the 'C32 that the data is no longer valid.

iv) Finally, the 'C32 terminates the transfer by setting XF0 high.

Note that the memory boot load source program has a header indicating the boot memory width, $\overline{\text{STRB0}}$, $\overline{\text{STRB1}}$, and $\overline{\text{IOSTRB}}$ control registers, refer to Table 3–8.

3) After reading the header, the boot loader copies the source program blocks. The source program blocks have three entries preceding the source program block data. The first entry in the source program block indicates the size of the block, the second entry indicates the address where the block is to be loaded, while the third entry contains the destination memory strobe including a pointer that identifies the destination memory strobe ($\overline{\text{STRB0}}$, $\overline{\text{STRB1}}$, or $\overline{\text{IOSTRB}}$) and a value that describes the strobe configuration for the memory width and data type size. If the destination memory is internal, the third entry should contain a 0. Note that the boot loader cannot load the source program to any memory address below 1000h, unless the address decode logic is remapped.

4) Once all the program blocks are loaded into their respective address locations with the given data type sizes, the boot loader resets the IOSTRB, STRB0, and STRB1 control registers to the values read at the beginning of the boot load process.

5) Finally, the boot loader branches to the destination address of the first source block loaded and begins program execution.

Figure 3–13. Boot Loader Mode Selection Flowchart

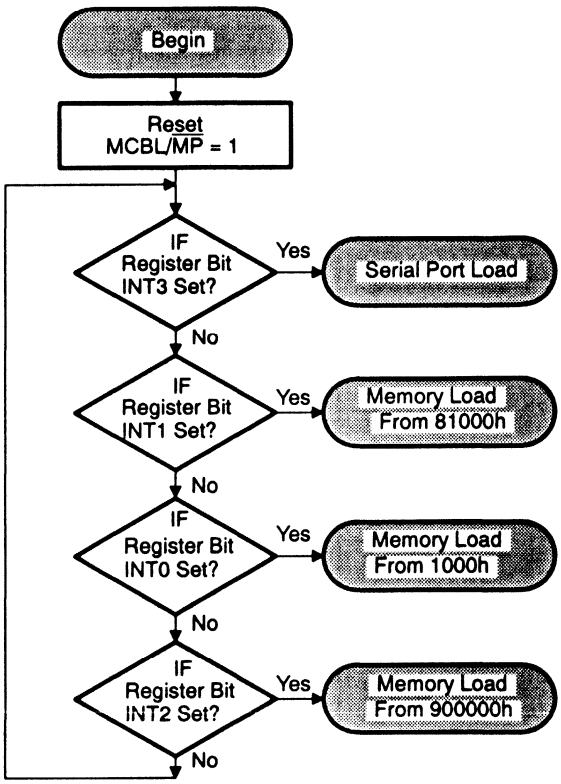


Figure 3–14. Boot Loader Serial Port Load Flowchart

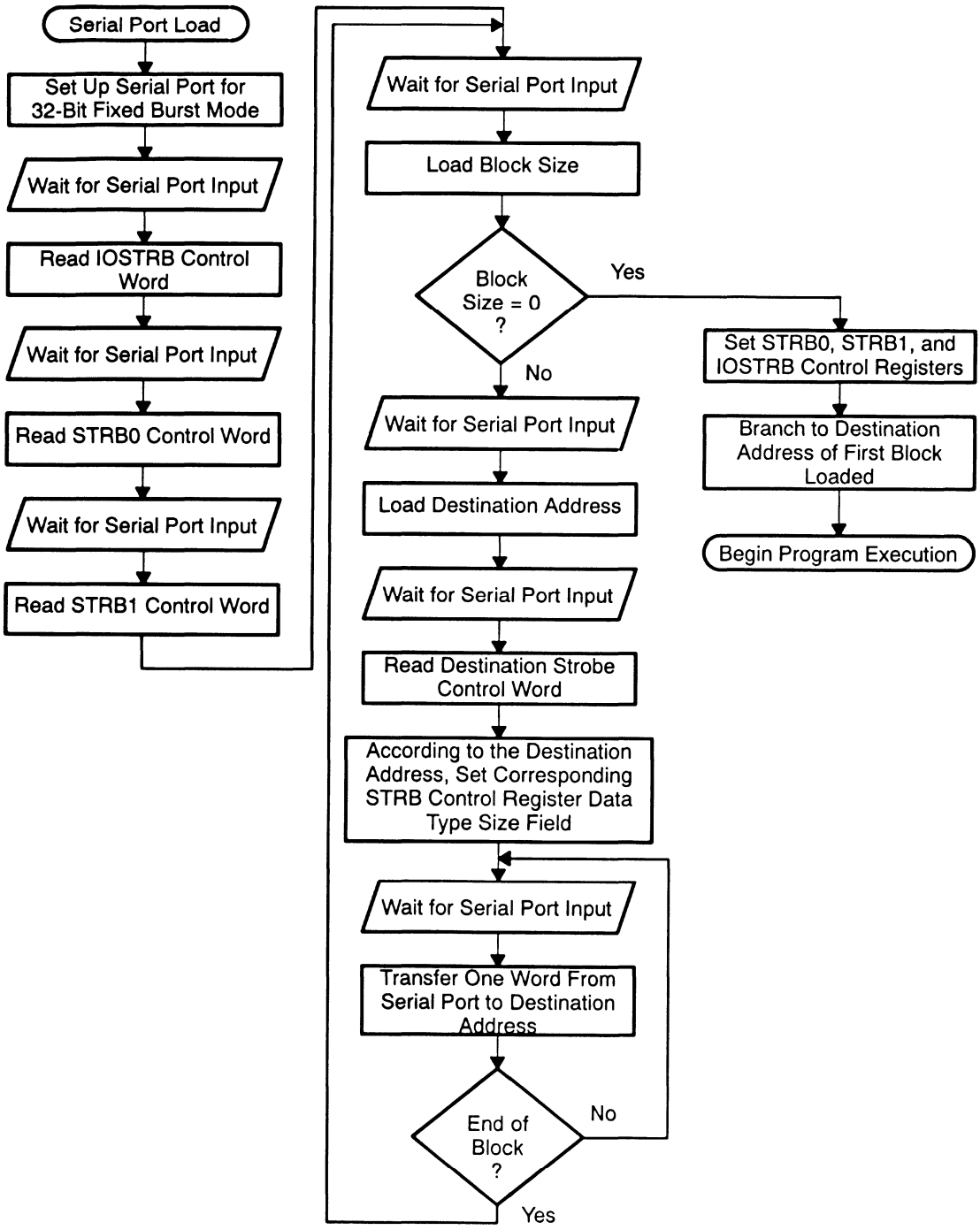


Figure 3–15. Boot Loader Memory Load Flowchart

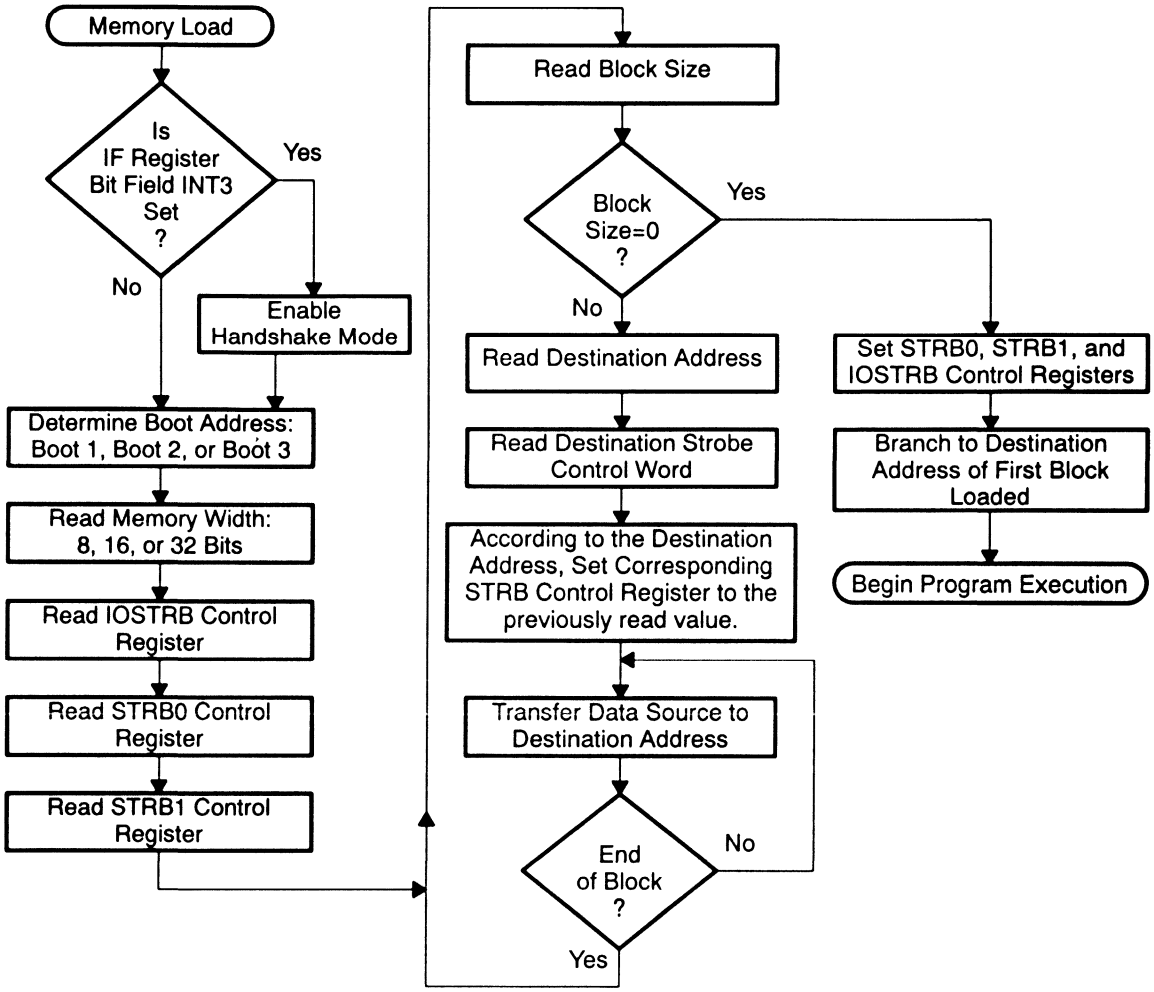
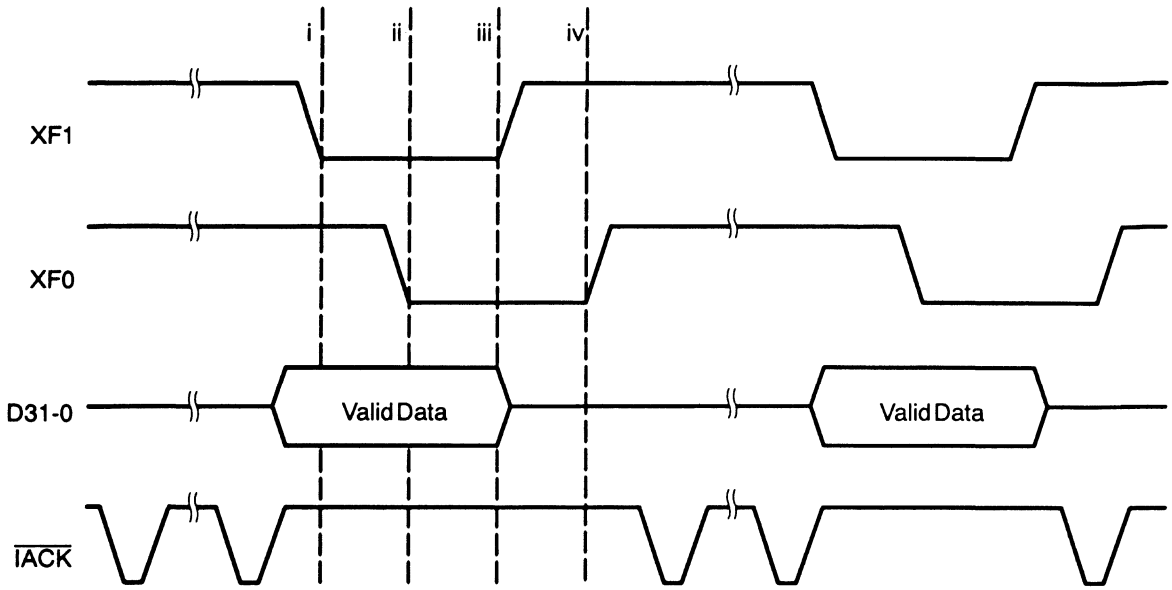


Figure 3–16. Handshake Data Transfer Operation



3.4.3 Boot Data Stream Structure

Table 3–8 shows the data stream structure. The data stream is composed of a header of 3 (serial port load) or 4 (memory load) words and one or more blocks of source data. The boot loader utilizes this header to determine the physical memory width where the source program resides (memory load) and to configure the STRBs after completion of source program boot load. The blocks of source data have three entries in addition to the raw data. The first entry in this block indicates the size of the block. The second entry in this block indicates the memory address where the boot loader copies this source block. The third entry contains the destination memory strobe configuration including memory width and data type size. This allows the boot loader to copy and store 8-, 16-, or 32-bit data values into 8-, 16-, or 32-bit wide memory. Words 8 through n, of the shaded entries in Table 3–8, contain the source data for the first block.

Table 3–8. Source Data Stream Structure

Word†	Content	Valid Data Entries
1	Memory width (8, 16, or 32 bits) where source program resides	8h, 10h, or 20h
2	Value to set the IOSTRB control register	See subsection 7.3.3
3	Value to set the STRB0 control register	See subsection 7.3.1
4	Value to set the STRB1 control register	See subsection 7.3.2
5	First source block size in terms of the data type size. A zero in this entry signifies the end of the source data stream.	$0 \leq size \leq 2^{24}$
6	Destination address to load the first source block	A valid 'C32 24-bit address
7	First source block memory width and data type size in the format used in the STRB control register data type size field.	SSSSSS6*x*h‡
8	First word of first source block.	A 'C32 valid instruction or any 8-, 16-, or 32-bit wide data value
. . .	. . .	. . .
n	Last word of first source block	A 'C32 valid instruction or any 8-, 16-, or 32-bit wide data value
. . .	. . .	. . .
m	Last source block size in terms of the data type size. If the next word following this block is not zero, another block is loaded.	$0 \leq size \leq 2^{24}$
m + 1	Destination address to load the last source block	A valid 'C32 24-bit address
m + 2	Last source block memory width and data type size in the format used in the STRB control register data type size field.	*SSSSSS6x*h‡
m + 3	First word of last source block.	A 'C32 valid instruction or any 8-, 16-, or 32-bit wide data value
. . .	. . .	. . .
j	Last word of last source block	
j+1	Zero word. Note that if more than one source block was read, word *j* shown above would be the last word of the last source block. Each block would have the format shown in the shaded entries.	0h

† Word 1 does not exist in serial port boot load since the source program does not reside in memory.

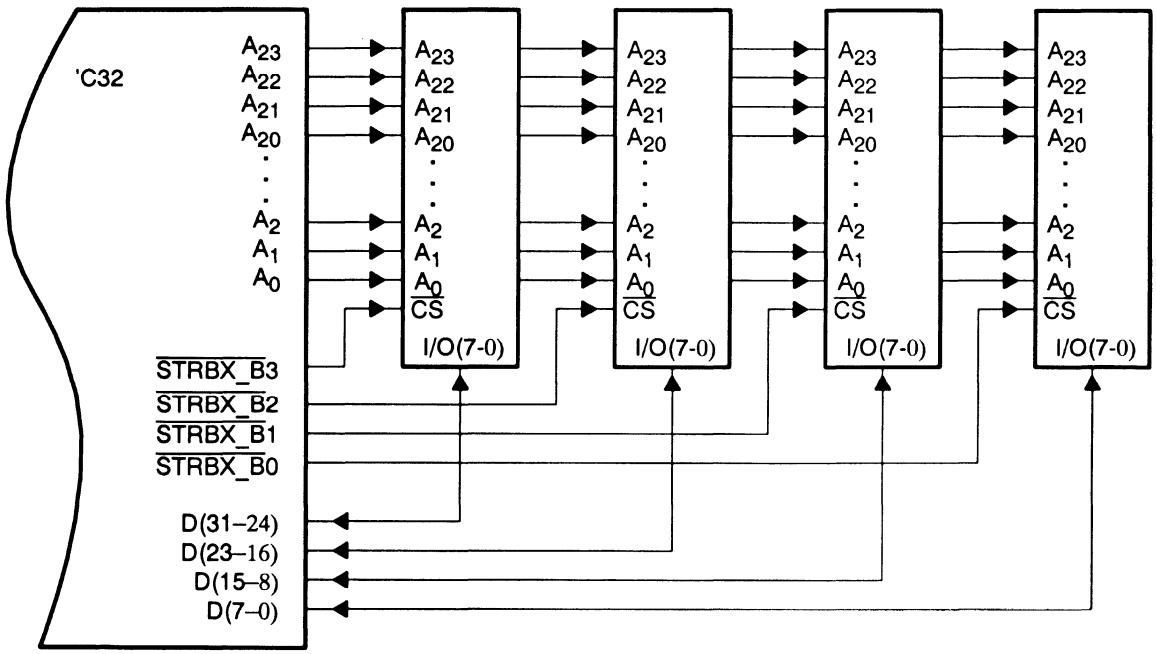
‡ The *SSSSSS* hexadecimal digits refer to the lower 24 bits of the strobe control register. The *x* hexadecimal digit identifies the strobe as follows: 0 for IOSTRB, 4 for STRB0, and 8 for STRB1. Note that when loading into internal memory the entire field, *SSSSSS6x*h, should be cleared to 0.

Each source block of data can be loaded to different memory locations. Each block specifies its own size and destination address. The last source block of the data stream is appended with a zero word. Because the 'C32's STRBs can be configured to support different external memory widths and data type sizes, each source block specifies its data type size. The external memory width was set when the boot loader read the STRBs control register values in the source data stream header.

3.4.4 Boot Loader Hardware Interface

The hardware interface for the memory boot load utilizes $\overline{\text{STRBX\_B3}}$ through $\overline{\text{STRBX\_B0}}$ pins as strobe byte enable pins as shown in Figure 3–17. The hardware interface is **independent of the boot source memory width**. This interface is identical to the 32-bit wide memory interface described in Case 2, in subsection 7.3.2. For 16-bit memory widths, remove the left-most two memory devices of Figure 3–17. For 8-bit memory widths, remove all but the right-most one of the memory devices of Figure 3–17.

Figure 3–17. External Memory Interface for Source Data Stream Memory Boot Load



Chapter 4

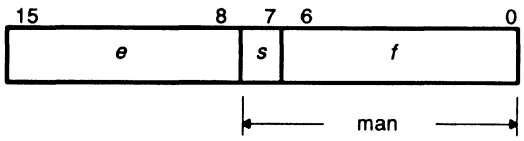
Data Formats and Floating-Point Operation

To facilitate the handling of 16-bit floating point data types, the 'C32 adds a new short floating point format for external 16-bit data types. Note that the following short floating-point format is used only in external 16-bit floating point data access. This format is different than the 16-bit immediate short floating point data format used in the 'C32's instruction set. See subsection 4.3.1 of the *TMS320C3x User's Guide* (literature number SPRU031) for detailed information of the 16-bit immediate short floating-point data format.

4.3.1 Short Floating-Point Format for External 16-Bit Data

In the short floating point format for external 16-bit data type size, floating point numbers are represented by a 2s-complement 8-bit exponent field (*e*), a sign bit (*s*), and an 8-bit mantissa field (*man*) with an implied most-significant non-sign bit.

Figure 4–6. Short Floating-Point Format



Operations are performed with an implied binary point between bits 7 and 6. When the implied most significant nonsign bit is made explicit, it is located to the immediate left of the binary point. The floating-point 2s-complement number *x* in the short floating-point format is given by:

$$x = \begin{cases} 01.f \times 2^e & \text{if } s = 0 \\ 10.f \times 2^e & \text{if } s = 1 \\ 0 & \text{if } e = -128 \end{cases}$$

Note that the floating-point instructions such as LDF, MPYF, ADDF, etc., and the integer instructions such as LDI, MPYI, ADDI, etc., produce different results when accessing the same memory location. The ***integer*** load instructions store the value in the ***least*** significant bits of the 'C32's registers. A bit field in the strobe control register controls sign extension or zero-fill of the most significant bits of the integer value. On the other hand, the ***floating-point*** load instructions store the value in the ***most*** significant bits of the 'C32's registers. For example:

If AR1= 4000h, R1 = 00 00000000h, the value stored at memory location 4000h is 0180h, and STRB0 is configured for a physical memory size and data type size of 16 bits. Then,

the result of: ADDI *AR1,R1 is R1 = 00 00000180h, while

the result of: ADDF *AR1,R1 is R1 = 01 C0000000h (= – 3.0), since – 4.0 + 1.0 = – 3.0

Chapter 5

Addressing

The addressing modes in the 'C32 are identical to those in the 'C30 and 'C31 and are discussed in the *TMS320C3x User's Guide* (literature number SPRU031).

Chapter 6

CPU Program Flow Control

This chapter discusses the operations that occur during reset. It also discusses the IDLE2 and LOPOWER power management modes available in the 'C32.

6.5 Reset Operation

At reset, the 'C32 performs the following operations:

- The peripherals are reset
- The CPU/DMA interrupt enable (IE), CPU interrupt flag (IF), and I/O flag (IOF) registers are loaded with 0s.
- All the bit fields in the status register (ST) are loaded with zero, except the PRGW status bit field that is loaded with the status of the PRGW pin.
- The external bus control registers are reset, see Section 7.3 for a description of the reset value.
- The 'C32 performs a 32-bit read to fetch the reset vector from memory location 0h. Once read, this value is loaded into the program counter.
- The 'C32 starts executing code from the memory location dictated by the program counter.

Table 6–3 shows the state of the 'C32's pins after reset is pulled low.

Table 6–3. Pin Operation at Reset

Signal	# Pins	Operation at Reset
		External Bus Interface (70 pins)
D31–0	32	Synchronous reset. Placed in high-impedance state.
A23–0	24	Synchronous reset. Placed in high-impedance state.
R/$\overline{W}$	1	Synchronous reset. Deasserted by going to a high level.
$\overline{IOSTRB}$	1	Synchronous reset. Deasserted by going to a high level.
$\overline{STRB0}$_B3/A_1	1	Synchronous reset. Deasserted by going to a high level.
$\overline{STRB0}$_B2/A_2	1	Synchronous reset. Deasserted by going to a high level.
$\overline{STRB0}$_B1	1	Synchronous reset. Deasserted by going to a high level.
$\overline{STRB0}$_B0	1	Synchronous reset. Deasserted by going to a high level.
$\overline{STRB1}$_B3/A_1	1	Synchronous reset. Deasserted by going to a high level.
$\overline{STRB1}$_B2/A_2	1	Synchronous reset. Deasserted by going to a high level.
$\overline{STRB1}$_B1	1	Synchronous reset. Deasserted by going to a high level.
$\overline{STRB1}$_B0	1	Synchronous reset. Deasserted by going to a high level.
$\overline{RDY}$	1	Reset has no effect.
$\overline{HOLD}$	1	Reset has no effect.
$\overline{HOLDA}$	1	Reset has no effect.
PRGW	1	Reset has no effect.
		Control Signals (9 Pins)
$\overline{RESET}$	1	Reset input pin.
$\overline{INT3}$–$\overline{INT0}$	4	Reset has no effect.
$\overline{IACK}$	1	Synchronous reset. Deasserted by going to a high level.
MCBL/$\overline{MP}$	1	Reset has no effect.
$\overline{XF1}$–$\overline{XF0}$	2	Asynchronous reset. Placed in high-impedance state.
		Serial Port Signals (6 Pins)
CLKX0	1	Asynchronous reset. Placed in high-impedance state.
DX0	1	Asynchronous reset. Placed in high-impedance state.
FSX0	1	Asynchronous reset. Placed in high-impedance state.
CLKR0	1	Asynchronous reset. Placed in high-impedance state.
DR0	1	Asynchronous reset. Placed in high-impedance state.
FSR0	1	Asynchronous reset. Placed in high-impedance state.

Note: Shaded entries corresponds to the (new) signals that are available **only** in the 'C32.

Table 6–3. Pin Operation at Reset (Continued)

Signal	# Pins	Operation at Reset
		Timer Signals (2 Pins)
TCLK0	1	Asynchronous reset. Placed in high-impedance state.
TCLK1	1	Asynchronous reset. Placed in high-impedance state.
		Clock Signals (3 Pins)
H1	1	Synchronous reset. Goes to its initial state when $\overline{\text{RESET}}$ makes a 1-to-0 transition. See Chapter 13 of the *TMS320C3x User's Guide.*
H3	1	Synchronous reset. Goes to its initial state when $\overline{\text{RESET}}$ makes a 1-to-0 transition. See Chapter 13 of the *TMS320C3x User's Guide.*
CLKIN	1	Reset has no effect.
		Emulation and Test Signals (5 Pins)
EMU3–EMU0	4	Reset has no effect.
$\overline{\text{SHZ}}$	1	Reset has no effect.
		Supply Signals (45 pins)
CV_{SS}	7	Reset has no effect.
DV_{SS}	7	Reset has no effect.
IV_{SS}	4	Reset has no effect.
DV_{DD}	12	Reset has no effect.
V_{DDL}	8	Reset has no effect.
V_{SSL}	6	Reset has no effect.
V_{SUBS}	1	Reset has no effect.

6.7 Power Management Modes

The 'C32 CPU has been enhanced by the addition of two power management modes, IDLE2 and LOPOWER. Either mode is invoked by executing the corresponding power-down instruction.

6.7.1 IDLE2 Power-Down Mode

In IDLE2 mode (opcode = 06000001h), the 'C32 behaves as follows:

- No instructions are executed.
- The CPU, peripherals, and internal memory retain their previous state.
- The external bus output pins are idle (the address lines remain in their previous state, the data lines are in the high-impedance state, and the output control signals are inactive).
- When the device is in the functional (nonemulation) mode, the clocks stop with H1 high and H3 low (see Figure 6–1).
- The 'C3x remains in IDLE2 until one of the four external interrupts ($\overline{INT3}$–$\overline{INT0}$) is asserted for at least one H1 cycle. When one of the four interrupts is asserted, the clocks start after a delay of one H1 cycle. The clocks can start up in the phase opposite to that in which they were stopped (that is, H1 may start low when H3 was low before stopping the clocks and H3 may start low when H1 was previously low). However, the H1 and H3 clocks remain 180 degrees out of phase with each other (see Figure 6–2).
- During IDLE2 operation, the CPU recognizes one of the four external interrupts if it is asserted for at least two H1 cycles. To avoid generating multiple false interrupts in level-triggered mode, the interrupt must be asserted for fewer than three H1 cycles.
- The interrupt service routine (ISR) must have been setup before placing device in IDLE2 mode since the instruction following the IDLE2 instruction is not executed until the RETI (return from interrupt) instruction is executed.
- When the device is in emulation mode and the IDLE2 instruction is executed, the H1 and H3 clocks continue to operate normally and the CPU behaves as if an IDLE instruction had been executed. The clocks continue to run for correct operation of the emulator.

Note:

For correct device operation, the three instructions following a delayed branch should not include either IDLE or IDLE2 instructions.

Figure 6–1. IDLE2 Timing

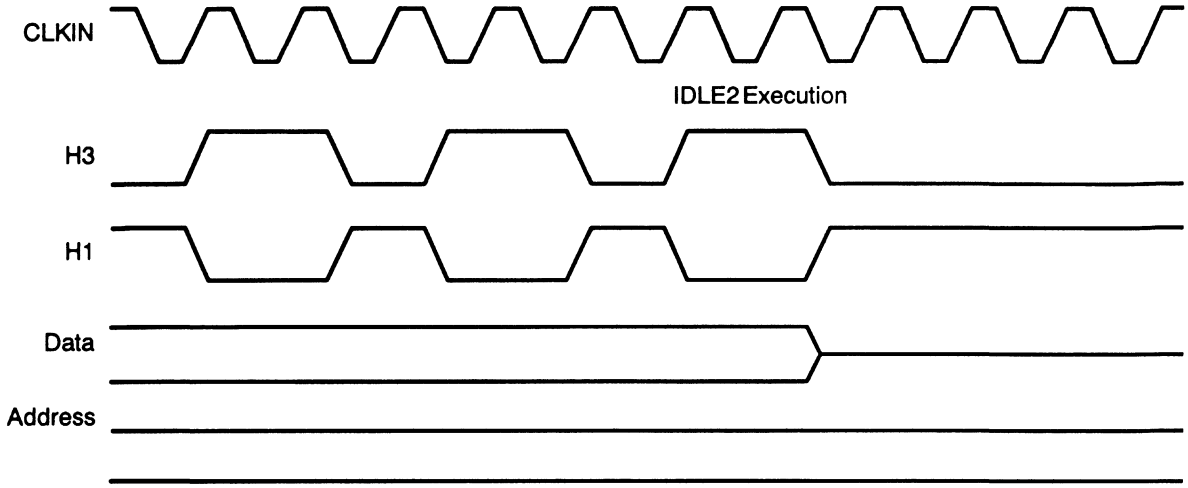
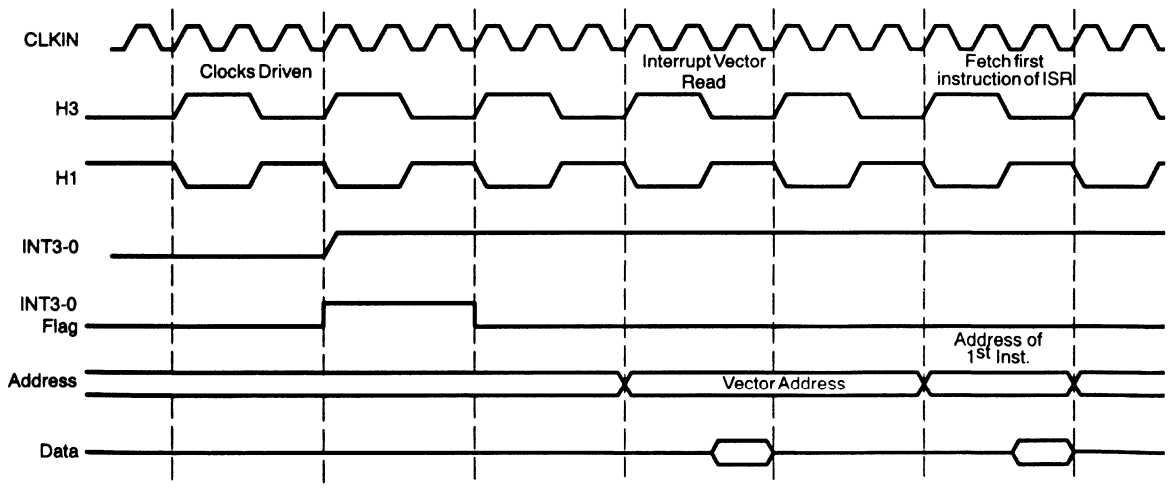


Figure 6–2. Interrupt Response Timing After IDLE2 Operation



6.7.2 LOPOWER Mode

In the LOPOWER (low-power) mode, the CPU continues to execute instructions and the DMA can continue to perform transfers, but at a reduced clock rate of the CLKIN frequency divided by 16, that is, a 'C3x with a CLKIN frequency of 32 MHz performs the same as a 2-MHz 'C3x that has an instruction cycle time of 1000 ns or 1 MHz).

The 'C3x slows down to 1/16 of full speed operation during the read phase of the LOPOWER instruction. The 'C3x resumes full speed operation during the read phase of the MAXSPEED instruction. The LOPOWER instruction encoding (opcode) is 1080 0001h and the MAXSPEED instruction encoding is 1080 0000h.

Figure 6–9. LOPOWER Timing

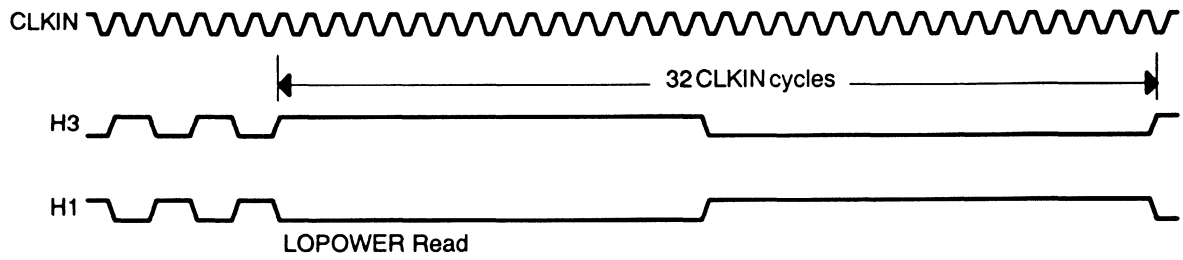
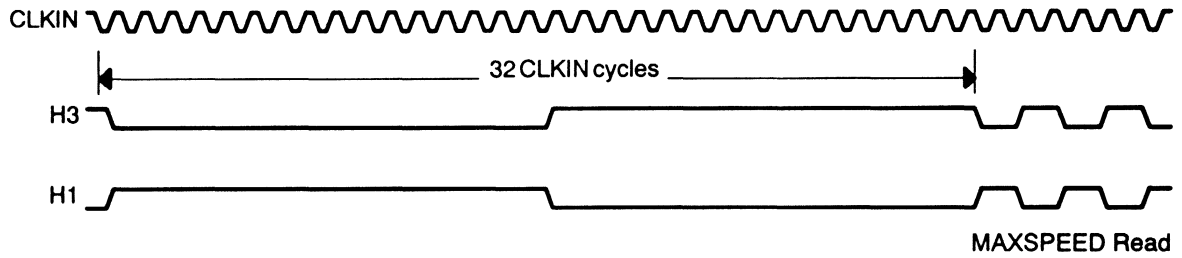


Figure 6–10. MAXSPEED Timing



Chapter 7

Enhanced External Memory Interface

The 'C32 external memory interface provides greater flexibility by improving the 'C3x core with several new features. This chapter describes these features and enhancements in detail.

7.1 Features

The C32's external memory interface includes the following features:

- One external pin, PRGW, configures the external program memory width to 16 or 32 bits.
- Two sets of memory STRBs ($\overline{\text{STRB0}}$ and $\overline{\text{STRB1}}$) and one $\overline{\text{IOSTRB}}$ allow zero glue-logic interface to two banks of memory and one bank of external peripherals.
- Separate bus control registers for each STRB control wait state generation, external memory width, and data type size.
- Each memory STRB handles 8-, 16- or 32-bit external data accesses (reads and writes).
- Multiprocessor support through the HOLD and HOLDA signals, is valid for all the STRBs.

7.2 Overview

The following sections describe examples, control register setups, and restrictions necessary to fully understand the operation and functionality of the external memory interface.

7.2.1 External Memory Interface Overview

The 'C32 memory interface accesses external memory through one 24-bit address and one 32-bit data bus that is shared by three mutually-exclusive strobes ($\overline{\text{STRB0}}$, $\overline{\text{STRB1}}$, and $\overline{\text{IOSTRB}}$). Depending on the address accessed, the 'C32 activates one of these strobes according to the memory map shown in Figure 3–8.

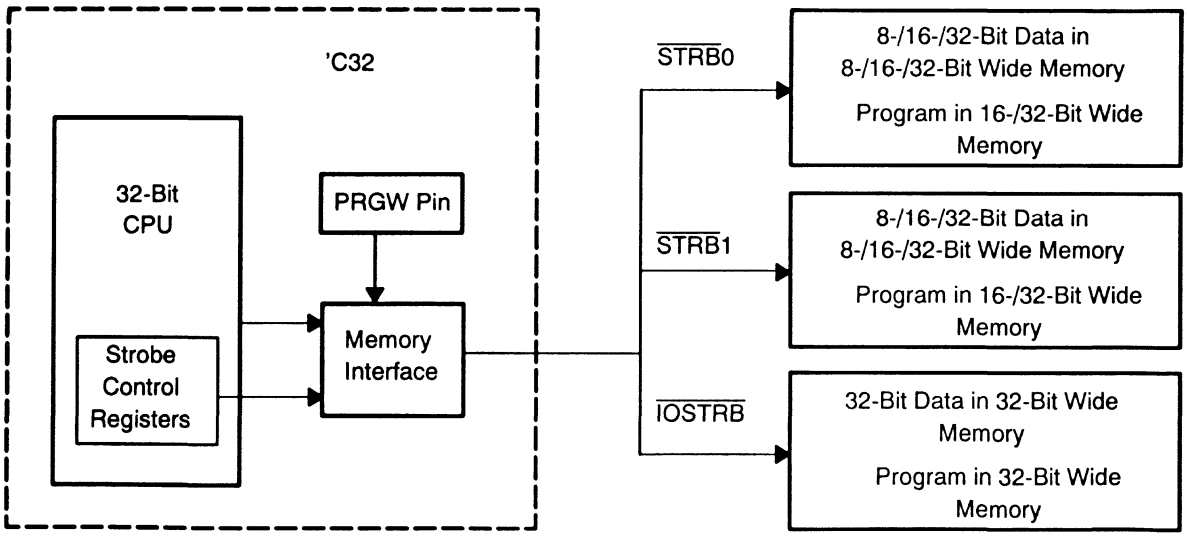
$\overline{\text{STRB0}}$ and $\overline{\text{STRB1}}$ can access 8-, 16-, or 32-bit data from 8-, 16-, or 32-bit wide memory. This is accomplished by four signals in each strobe: $\overline{\text{STRBx\_B3}}/A_{-1}$, $\overline{\text{STRBx\_B2}}/A_{-2}$, $\overline{\text{STRBxB1}}$, and $\overline{\text{STRBx\_B0}}$. These signals serve as byte enable pins to access one byte, half-word, or a full-word from the external memory. The first two signals also serve as additional address pins to perform two or four consecutive accesses in 8-bit or 16-bit wide external memory. The 'C32 controls the behavior of these pins through the data size and memory width bit fields in the corresponding strobe control register, as follows:

- Memory width (default value dependent on PRGW pin level)
  - 8-bit wide memory
    - $\overline{\text{STRBx\_B3}}/A_{-1}$ and $\overline{\text{STRBx\_B2}}/A_{-2}$ as address pins
    - $\overline{\text{STRBx\_B0}}$ as byte enable/chip select signal
    - $\overline{\text{STRBx\_B1}}$ unused
  - 16-bit wide memory
    - $\overline{\text{STRBx\_B3}}/A_{-1}$ as address pin
    - $\overline{\text{STRBx\_B1}}$ and $\overline{\text{STRBx\_B0}}$ as byte enable signal
    - $\overline{\text{STRBx\_B2}}$ unused
  - 32-bit memory
    - $\overline{\text{STRBx\_B3}}$, $\overline{\text{STRBx\_B2}}$, $\overline{\text{STRBx\_B1}}$, and $\overline{\text{STRBx\_B0}}$ as byte enable signals
- Data size
  - 8-bit data, physical address = logical address shift right by 2
  - 16-bit data, physical address = logical address shift right by 1
  - 32-bit data, physical address = logical address

IOSTRB can access 32-bit data from 32-bit wide memory. It does not have the flexibility of STRB0 and STRB1 since it is composed of a single signal: IOSTRB. IOSTRB bus cycles are different from those of STRB0 and STRB1 and are discussed in Section 7.4. This timing difference accomodates slower I/O peripherals.

Summarizing, the 'C32 memory interface parallel bus implements three mutually-exclusive address spaces distinguished via three separate control signals as shown in Figure 7–1. STRB0 and STRB1 support 8-/16-, or 32-bit data access in 8-/16-/32-bit wide external memory and 16-/32-bit program access in 16-/32-bit wide external memory. IOSTRB address space supports 32-bit data/program access in 32-bit wide external memory. Internally, the 'C32 has a 32-bit architecture, hence, the memory interface packs and unpacks the data accessed accordingly.

Figure 7–1. Memory Address Spaces

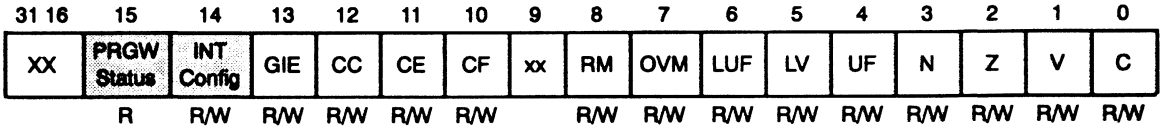


7.2.2 Program Memory Access

The 'C32 supports program execution from 16- or 32-bit external memory width. The PRGW pin configures the width of the external program memory. When this pin is pulled high, the 'C32 executes from 16-bit wide memory. When this pin is pulled low, the 'C32 executes from 32-bit wide memory. For 16-bit wide zero wait-state memory, the 'C32 takes two instruction cycles to fetch a single 32-bit instruction. During the first cycle the lower 16 bits of the instruction are fetched. During the second cycle, the upper 16 bits are fetched and concatenated with the lower 16 bits. 32-bit memory fetches are identical to those of the 'C30 and 'C31.

The PRGW status bit field of the CPU status (ST) register reflects the setting of the PRGW pin. Figure 7–2 depicts all the bit fields of the CPU status (ST) register.

Figure 7–2. Status Register



The status of the PRGW pin also affects the reset value of the physical memory width bit fields of the STRB0 and STRB1 bus control registers. The physical memory width is set to 32-bit memory width if the PRGW pin is logic low after the device reset. The physical memory width is set to 16-bit memory width if the PRGW pin is logic high after the device reset (see Section 7.3 for more information).

The cycle before and the cycle after changing the PRGW should not perform a program fetch over the external memory interface.

CAUTION

7.2.3 Data Memory Access

The 'C32 can load and store 8-, 16-, or 32-bit data quantities from and into memory. Because the CPU has a 32-bit architecture, the device internally handles all 8-, 16-, or 32-bit data quantities as a 32-bit value. Hence, the external memory interface handles the conversion between 8- and 16-bit data quantities to the internal 32-bit representation. The external memory interface also handles the storage of 32-, 16-, or 8-bit data quantities into 32-, 16-, or 8-bit wide memories.

7.2.4.1 8-, 16-, or 32-Bit Integers Data Types

The 'C32 supports 8-, 16- or 32-bit integer data quantities. When 8- or 16-bit integers are read from external memory, the value is loaded into the least significant bits of the register with the most significant bits sign-extended or zero-filled. The polarity of the Sign Ext/Zero Fill bit field of the corresponding STRB control register controls the sign extension or zero fill (see paragraphs 7.3.1.1 and 7.3.1.2). 32-bit integer data access is identical to that of the 'C30 and 'C31.

7.2.4.2 16- or 32-Bit Floating-Point Data Types

The 'C32 supports 16- or 32-bit floating point data. For 16-bit floating-point reads, the eight MSBs are the signed exponent and the eight LSBs are the signed mantissa (See subsection 4.3.1). When a 16-bit floating-point value is loaded into a 40-bit register, the external memory interface zero-fills the least significant 24 bits of the register. When a 16-bit floating-point value is used as a 32-bit on-chip input operand, the external memory interface zero-fills the 16 least significant bits of the 32-bit input operand. 32-bit floating-point data access is identical to those of 'C30 and 'C31.

7.3 Configuration

To access 8-, 16-, or 32-bit data (types) from 8-, 16-, or 32-bit wide memory, the memory interface of the 'C32 device uses either strobe $\overline{\text{STRB0}}$ or $\overline{\text{STRB1}}$ with four pins each. These pins serve as byte enable and/or additional address pins. In conjunction with a shifted version of the internal address presented to the external address, the 'C32 can select a single byte from one external memory location or combine up to four bytes from contiguous memory locations. The behavior of these pins is controlled by the external memory width and the data type size. The selected data size also determines the amount of internal to physical address shift. You communicate these values to the 'C32 memory interface through bit fields in the bus control registers.

7.3.1 External Interface Control Registers

The following sections describe the bus control registers used to manipulate the byte addressability features of the 'C32. Figure 7–3 shows the external interface control memory map.

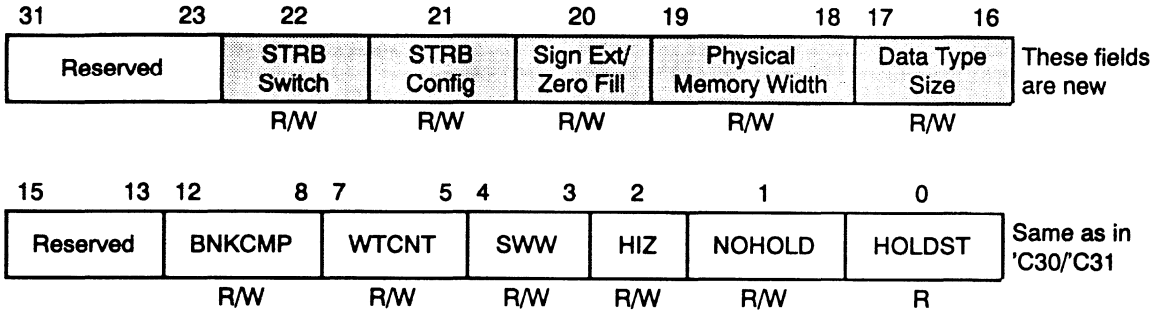
Figure 7–3. Memory-Mapped External Interface Control Registers

Address	Register
808060h	IOSTRB Control
808061h	Reserved
808062h	Reserved
808063h	Reserved
808064h	STRB0 Control
808065h	Reserved
808066h	Reserved
808067h	Reserved
808068h	STRB1 Control
808069h	Reserved
	. . .
80806Fh	Reserved

7.3.1.1 STRB0 Control Register

The STRB0 control register (Figure 7–4) is a 32-bit register that contains the control bits for the portion of the external bus memory space that is mapped to STRB0. The following table lists the register bits with the bit names and functions. At the system reset, 0F10F8h is written to the STRB0 control register if PRGW pin is logic low and 0710F8h is written to the STRB0 control register if the PRGW pin is logic high.

Figure 7–4. STRB0 Control Register

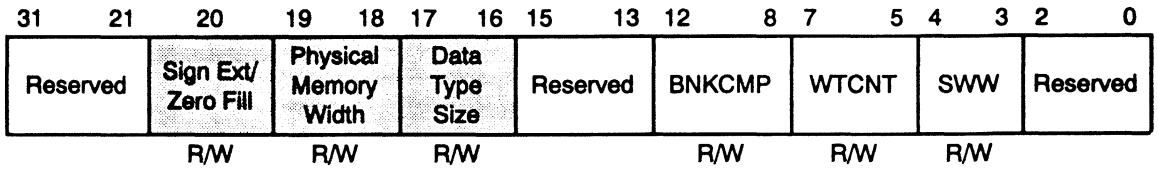


The instruction immediately preceding a change in the data size or memory width bit fields should not perform a multicycle store. Do not follow a change in the data size or memory width bit fields with a store instruction. Also, do not perform a load in the next two instructions following a change in the data size or memory width bit fields

7.3.1.2 STRB1 Control Register

The STRB1 control register (Figure 7–5) is a 32-bit register that contains the control bits for the portion of the external bus memory space that is mapped to STRB1. Figure 7–5 shows the register bits with their names and functions. At system reset, 0F10F8h is written to the STRB1 control register if PRGW pin is logic low and 0710F8h is written to the STRB1 control register if PRGW pin is logic high.

Figure 7–5. STRB1 Control Register

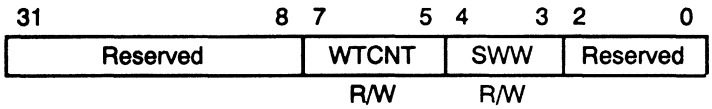


The instruction immediately preceding a change in the data size or memory width bit fields should not perform a multicycle store. Do not follow a change in the data size or memory width bit fields with a store instruction. Also, do not perform a load in the next two instructions following a change in the data size or memory width bit fields

7.3.1.3 IOSTRB Control Register

The $\overline{\text{IOSTRB}}$ control register (Figure 7–6) is a 32-bit register that contains the control bits for the portion of the external bus memory space that is mapped to $\overline{\text{IOSTRB}}$. Unlike the $\overline{\text{STRB0}}$ and $\overline{\text{STRB1}}$, there is no byte enable signal for the $\overline{\text{IOSTRB}}$. The data access through the $\overline{\text{IOSTRB}}$ is always 32-bit. The following table lists the register bits with the bit names and functions. At the system reset, 0F8h is written to the $\overline{\text{IOSTRB}}$ control register. The $\overline{\text{IOSTRB}}$ timing is identical to the 'C30's $\overline{\text{IOSTRB}}$ timing.

Figure 7–6. $\overline{\text{IOSTRB}}$ Control Register



7.3.1.4 Data Type Size Field

The Data Type Size field indicates the size of the data type written in memory. This field can have the following values (the reset value is shown by the shaded entry):

Bit 17	Bit 16	Data Type Size
0	0	8-bit
0	1	16-bit
1	0	Reserved
1	1	32-bit

7.3.1.5 Physical Memory Width Field

The Physical Memory Width fields indicates the size of the physical memory connected to the device. The reset value depends on the status of the PRGW

pin. If PRGW pin is logic low, the physical memory width is configured to 32 bits (= 11_2). If PRGW pin is logic high, the physical memory width is configured to 16 bits (= 01_2). This field can have the following values:

Bit 19	Bit 18	Physical Memory Width
0	0	8-bit
0	1	16-bit
1	0	Reserved
1	1	32-bit

Setting the Physical Memory Width field of the STRB0 or STRB1 control registers changes the functionality of the $\overline{\text{STRB0}}$ or $\overline{\text{STRB1}}$ signals. When the Physical Memory Width field is configured to 32 bits, the corresponding $\overline{\text{STRBx\_B0}}$–$\overline{\text{STRBx\_B3}}$ signals are configured as byte enable pins (refer to Figure 7–7). When the Physical Memory Width field is configured to 16 bits, the corresponding $\overline{\text{STRBx\_B3}}$ signal is configured as an address pin while $\overline{\text{STRBx\_B0}}$ and $\overline{\text{STRBx\_B1}}$ signals are configured as byte enable pins (refer to Figure 7–11). When the Physical Memory Width field is configured to 8 bits, the $\overline{\text{STRBx\_B3}}$ and $\overline{\text{STRBx\_B2}}$ signals are configured as address while $\overline{\text{STRBx\_Bx}}$ is configured as byte enable pin (refer to Figure 7–15). Note that once a $\overline{\text{STRBx\_Bx}}$ signal is configured as an address pin it will be active for any external memory access ($\overline{\text{STRB0}}$, $\overline{\text{STRB1}}$, $\overline{\text{IOSTRB}}$, or external program fetch).

7.3.1.6 Sign Ext/Zero Fill Field

The Sign Ext/Zero Fill field selects the method of converting 8- and 16-bit integer data to 32-bit integer data when transferring data from external memory to an internal register or memory location. This field can have the following values (the shaded entry is the reset value):

Bit 20	Sign Ext/Zero-Fill Function Description
0	8- or 16-bit integer reads are sign-extended to 32-bits
1	The most significant bits of an 8- or 16 bit integer reads are zero-filled to make the number 32-bits

Note that 8- and 16-bit integer loads are stored in the least significant bits of the 'C32 registers/memory with the most significant bits sign-extended or zero-filled according to the setting of this bit field.

7.3.1.7 STRB Config Field

The STRB Config field indicates if the $\overline{\text{STRB0\_Bx}}$ signals are active when accessing data from either STRB0 or STRB1 memory spaces. This mode is

useful when accessing a single external memory bank that stores two different data types, each mapped to a different STRB (refer to Chapter 12 for examples). This field can have the following values (the shaded entry depicts the reset value):

Bit 21 (STRB0 only)	STRB Config Function Description
0	$\overline{\text{STRB0}}$_Bx signals are active for address locations 0h–7FFFFFh and 880000h–8FFFFFh. $\overline{\text{STRB1}}$_Bx signals are active for address locations 900000h–FFFFFFh
1	$\overline{\text{STRB0}}$_Bx signals are active for address locations 0h–7FFFFFh, 880000h–8FFFFFh, and 900000h–FFFFFh. $\overline{\text{STRB1}}$_Bx signals are active for address locations 900000h–FFFFFFh

7.3.1.8 STRB Switch Field

The STRB Switch field defines whether a single cycle is inserted between back-to-back reads when crossing STRB0 to STRB1 or STRB1 to STRB0 boundaries (switching STRBs). The extra cycle toggles the strobe signal during back-to-back reads. Otherwise, the strobe will remain low during back-to-back reads. This field can have the following values (the shaded entry highlights the reset value):

Bit 22 (STRB0 only)	STRB Switch Function Description
0	Does not insert a single cycle between back-to-back reads that switch from STRB0 to STRB1 or vice versa.
1	Inserts a single cycle between back-to-back reads when switching from STRB0 to STRB1 or vice versa.

7.3.1.9 Example

For example, consider a 'C32 connected to two banks of external memory. In this configuration, one bank is mapped to STRB0 while the other bank is mapped to STRB1. The STRB0 bank of memory is 32 bits wide and stores 32-bit data types. The STRB1 bank of memory is 16 bits wide and stores 16-bit data types. You transfer these configurations to the TMS320C32 by setting the Physical Memory Width and Data Type Size fields of the respective STRB0 and STRB1 control registers. Also, you must clear the STRB Config bit field to 0 since the banks are separate memories. Note that 'C32's address pins $A_{23}A_{22}A_{21}...A_1A_0$ are connected to the STRB0 memory bank address pins $A_{23}A_{22}A_{21}...A_1A_0$. But, 'C32's address pins $A_{22}A_{21}...A_1A_0A_{-1}$ are connected to the STRB1 memory bank address pins $A_{23}A_{22}A_{21}...A_1A_0$.

Executing the following code on this device results in the data access sequence shown in the Table 7–1:

```
1) LDI    4000h, AR1   ; AR1 = 4000h
2) LDI    *AR1++, R2   ; R2 = *4000h and AR1 = AR1 + 1
3) ADDI   *AR1++, R2   ; R2 = R2 + *4001h and AR1 = AR1 + 1
4) ADDI   *AR1++, R2   ; R2 = R2 + *4002h and AR1 = AR1 + 1
5) ADDI   *AR1++, R2   ; R2 = R2 + *4003h and AR1 = AR1 + 1
6) LDI    900h, AR2    ; AR2 = 900h
7) LSH    12, AR2      ; AR2 = 900000h
8) LDI    *AR2++, R3   ; R3 = *900000h and AR2 = AR2 + 1
9) ADDI   *AR2, R3     ; R3 = R3 + 900001h
```

By setting the bit fields of the STRB0 bus control register with a Physical Memory Width of 32 bits and a Data Type Size of 32-bit, the external address referring to STRB0 location is identical to the internal address used by the 'C32 CPU. On the other hand, setting the bit fields of the STRB1 Bus Control register with a Physical Memory Width of 16-bit and a Data Type Size of 16-bit, the address presented by the 'C32's external pins is the internal address shifted right by one bit with A_{23} driving A_{23} and A_{22}. Since STRB1 memory bank address pins $A_{23}A_{22}A_{21}...A_1A_0$ are connected to the 'C32's address pins $A_{22}A_{21}...A_1A_0A_{-1}$, the address seen by the STRB1 memory bank is identical to the 'C32 CPU internal address.

Table 7–1. Data Access Sequence for a Memory Configuration with Two Banks

Instruction #	Internal Address	External Address	Active Strobe	Data Accessed
(2)	4000h	4000h	$\overline{STRB0}\_\overline{B0/B1/B2/B3}$	Data 0
(3)	4001h	4001h	$\overline{STRB0}\_\overline{B0/B1/B2/B3}$	Data 1
(4)	4002h	4002h	$\overline{STRB0}\_\overline{B0/B1/B2/B3}$	Data 2
(5)	4003h	4003h	$\overline{STRB0}\_\overline{B0/B1/B2/B3}$	Data 3
(8)	900000h	C80000h	$\overline{STRB1}\_\overline{B0/B1}$ and $\overline{STRB1}\_\overline{B3}/A_{-1} = 0$	Data 4
(9)	900001h	C80001h	$\overline{STRB1}\_\overline{B0/B1}$ and $\overline{STRB1}\_\overline{B3}/A_{-1} = 1$	Data 5

External Memory

	31 — 0
4000h	Data 0
4001h	Data 1
4002h	Data 2
4003h	Data 3
	15 — 0
900000h	Data 4
	Data 5

'C32 ability to select a single byte from a single external memory location or combinations of bytes from several contiguous memory locations, dictates that the internal address seen by the CPU corresponds to a shifted version of the address presented to the external pins. The C32's external memory interface handles this conversion automatically as long as you configure the Bus

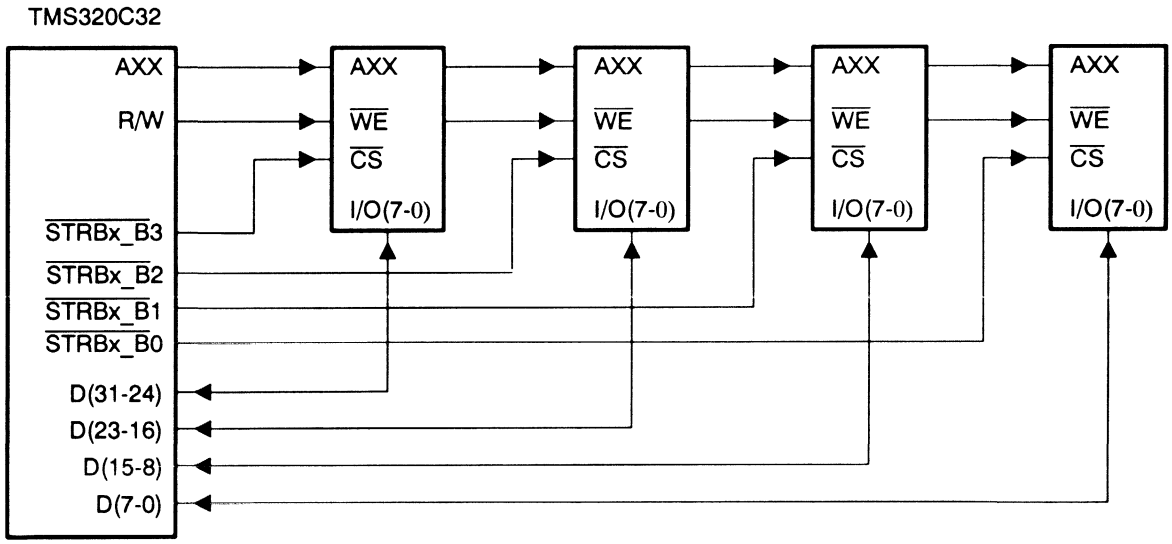
Control register to match the external memory configuration present in your hardware implementation.

As seen in Figure 2–2, 'C32 handles nine different memory access cases. The following sections discusses these cases in detail.

7.3.2 32-Bit Wide Memory Interface

'C32 memory interface to 32-bit wide external memory utilizes $\overline{\text{STRBx\_B3}}$ through $\overline{\text{STRBx\_B0}}$ pins as strobe-byte enable pins as shown in Figure 7–7. In this manner, the 'C32 can read/write a single 32-, 16-, or 8-bit value from the external 32-bit wide memory.

Figure 7–7. 'C32 External Memory Interface for 32-Bit SRAMs



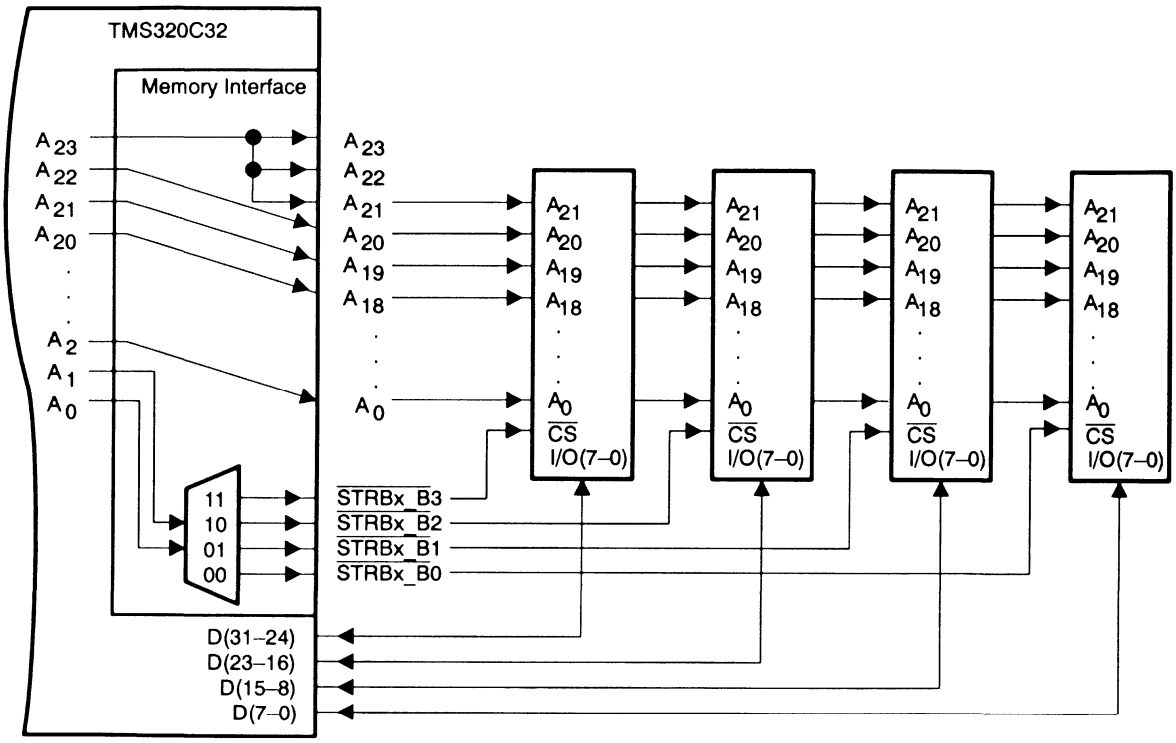
Case 1: 32-Bit Wide Memory With 8-Bit Data Type Size

When the data type size is 8-bit, the 'C32 shifts the internal address two bits to the right before presenting it to the external address pins. In this shift, the memory interface copies the value of the internal address A_{23} to the external address pins A_{23}, A_{22}, and A_{21}. Also, the memory interface activates the $\overline{\text{STRBx\_B3}}$ through $\overline{\text{STRBx\_B0}}$ pins according to the value of the internal address bits A_1 and A_0 as shown in Table 7–2. Figure 7–8 shows a functional diagram of the memory interface for 32-bit wide memory with 8-bit data type size.

Table 7–2. Strobe-Byte Enable for 32-Bit Wide Memory With 8-Bit Data Type Size

Internal A_1	Internal A_0	Active Strobe-Byte Enable
0	0	$\overline{\text{STRBx\_B0}}$
0	1	$\overline{\text{STRBx\_B1}}$
1	0	$\overline{\text{STRBx\_B2}}$
1	1	$\overline{\text{STRBx\_B3}}$

Figure 7–8. Functional Diagram for 8-Bit Data Type Size and 32-Bit External Memory Width



For example, reading or writing to memory locations 90 4000h to 90 4004h involves the following:

Internal Address Bus	External Address Pins	Active Strobe-Byte Enable	Accessed Data Pins
904000h	E41000h	$\overline{\text{STRB1\_B0}}$	D_{7-0}
904001h	E41000h	$\overline{\text{STRB1\_B1}}$	D_{15-8}
904002h	E41000h	$\overline{\text{STRB1\_B2}}$	D_{23-16}
904003h	E41000h	$\overline{\text{STRB1\_B3}}$	D_{31-24}
904004h	E41001h	$\overline{\text{STRB1\_B0}}$	D_{7-0}

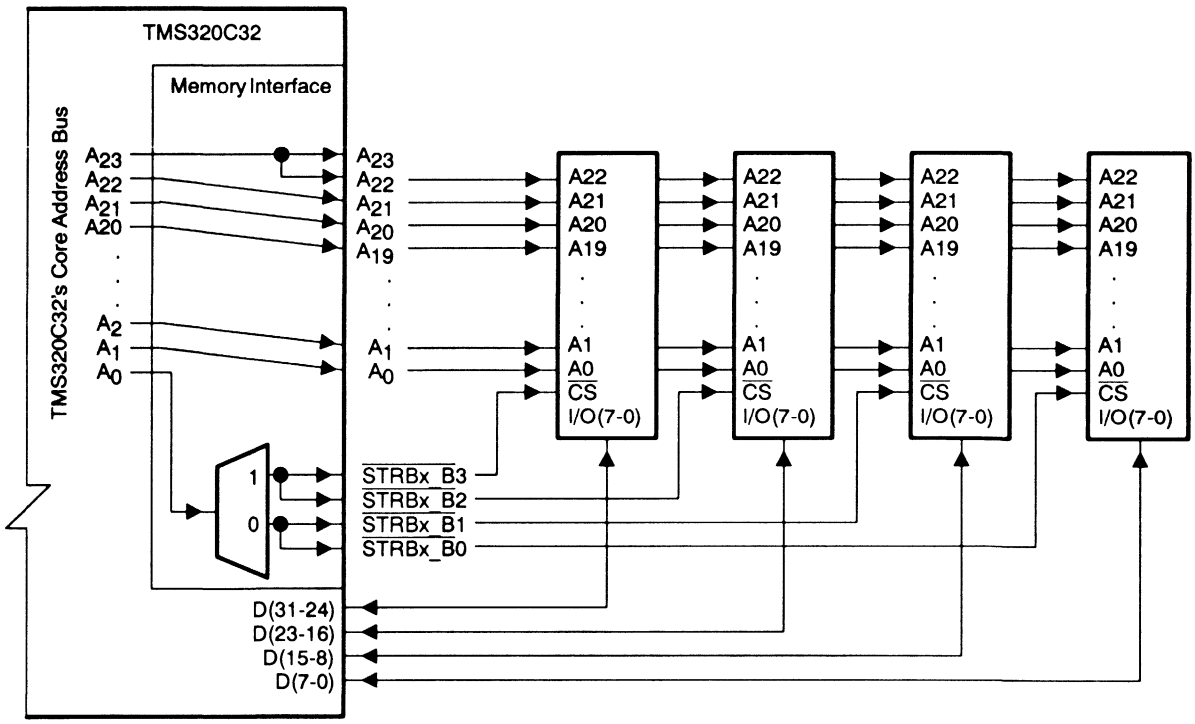
Case 2: 32-Bit Wide Memory With 16-Bit Data Type Size

When the data type size is 16-bit, the 'C32 shifts the internal address one bit to the right before presenting it to the external address pins. In this shift, the memory interface copies the value of the internal address A_{23} to the external address pins A_{23} and A_{22}. Also, the memory interface activates the $\overline{\text{STRBX-B3}}$ through $\overline{\text{STRBx\_B0}}$ pins according to the value of the internal address bit A_0 as shown in Table 7–3. Figure 7–9 shows a functional diagram of the memory interface for 32-bit wide memory with 16-bit data type size.

Table 7–3. Strobe-Byte Enable for 32-Bit Wide Memory With 16-Bit Data Type Size

Internal A_0	Active Strobe-Byte Enable
0	$\overline{\text{STRBx\_B1}}$ and $\overline{\text{STRBx\_B0}}$
1	$\overline{\text{STRBx\_B3}}$ and $\overline{\text{STRBx\_B2}}$

Figure 7–9. Functional Diagram for 16-Bit Data Type Size and 32-Bit External Memory Width



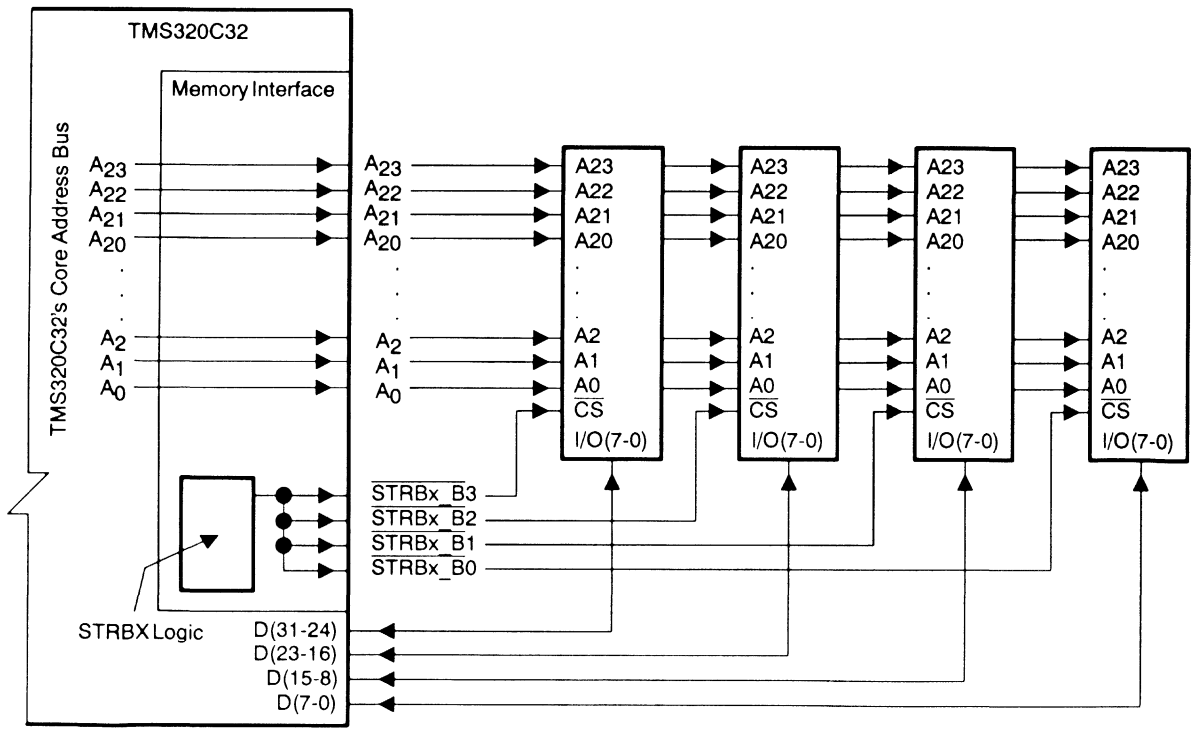
For example, reading or writing to memory locations 904000h to 904004h involves the following:

Internal Address Bus	External Address Pins	Active Strobe-Byte Enable	Accessed Data Pins
904000h	C82000h	$\overline{\text{STRB1\_B1}}$ and $\overline{\text{STRB1\_B0}}$	D_{15-0}
904001h	C82000h	$\overline{\text{STRB1\_B3}}$ and $\overline{\text{STRB1\_B2}}$	D_{31-16}
904002h	C82001h	$\overline{\text{STRB1\_B1}}$ and $\overline{\text{STRB1\_B0}}$	D_{15-0}
904003h	C82001h	$\overline{\text{STRB1\_B3}}$ and $\overline{\text{STRB1\_B2}}$	D_{31-16}
904004h	C82002h	$\overline{\text{STRB1\_B1}}$ and $\overline{\text{STRB1\_B0}}$	D_{15-0}

Case 3: 32-Bit Wide Memory With 32-Bit Data Type Size

When the data size is 32-bit, the 'C32 does not shift the internal address before presenting it to the external address pins. In this case, the memory interface copies the value of the internal address bus to the respective external address pins. Also, the memory interface activates $\overline{\text{STRBx\_B3}}$ through $\overline{\text{STRBx\_B0}}$ pins during accesses. Figure 7–10 shows a functional diagram of the memory interface for 32-bit wide memory with 32-bit data size.

Figure 7–10. Functional Diagram for 32-Bit Data Size and 32-Bit External Memory Width



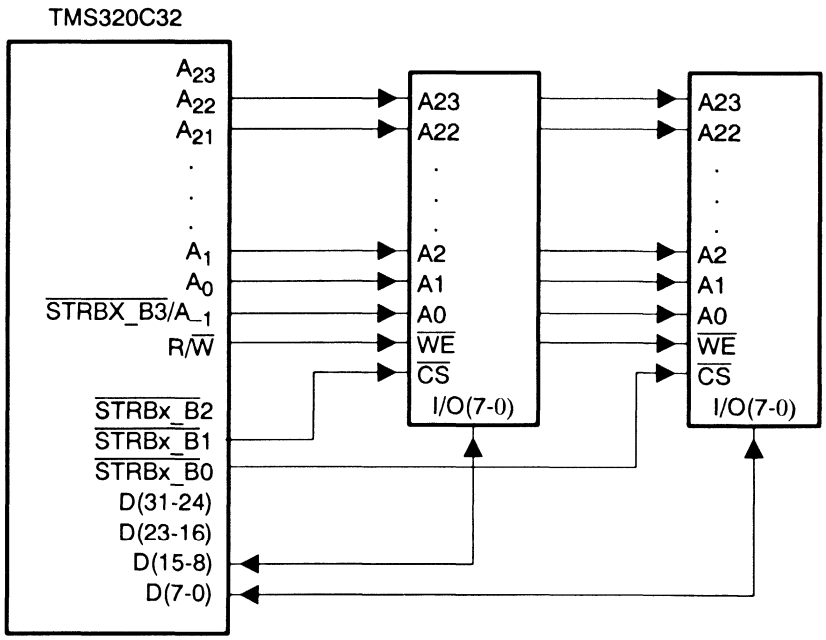
For example, reading or writing to memory locations 904000h to 904004h involves the following:

Internal Address Bus	External Address Pins	Active Strobe-Byte Enable	Accessed Data Pins
904000h	904000h	$\overline{\text{STRB1\_B0}}$, $\overline{\text{STRB1\_B1}}$, $\overline{\text{STRB1\_B2}}$, and $\overline{\text{STRB1\_B3}}$	D_{31-0}
904001h	904001h	$\overline{\text{STRB1\_B0}}$, $\overline{\text{STRB1\_B1}}$, $\overline{\text{STRB1\_B2}}$, and $\overline{\text{STRB1\_B3}}$	D_{31-0}
904002h	904002h	$\overline{\text{STRB1\_B0}}$, $\overline{\text{STRB1\_B1}}$, $\overline{\text{STRB1\_B2}}$, and $\overline{\text{STRB1\_B3}}$	D_{31-0}
904003h	904003h	$\overline{\text{STRB1\_B0}}$, $\overline{\text{STRB1\_B1}}$, $\overline{\text{STRB1\_B2}}$, and $\overline{\text{STRB1\_B3}}$	D_{31-0}
904004h	904004h	$\overline{\text{STRB1\_B0}}$, $\overline{\text{STRB1\_B1}}$, $\overline{\text{STRB1\_B2}}$, and $\overline{\text{STRB1\_B3}}$	D_{31-0}

7.3.3 16-Bit Wide Memory Interface

'C32 memory interface to 16-bit wide external memory utilizes $\overline{\text{STRBx\_B3}}$ pin as an additional address pin, A_{-1}, while using $\overline{\text{STRBx\_B0}}$ and $\overline{\text{STRBx\_B1}}$ as strobe-byte enable pins as shown in Figure 7–11. Note that the external memory address pins are connected to the 'C32's address pins $A_{22}A_{21}...A_1A_0A_{-1}$. In this manner, the 'C32 can read/write a single 32-, 16-, or 8-bit value from the external 16-bit wide memory.

Figure 7–11. External Memory Interface for 16-Bit SRAMs



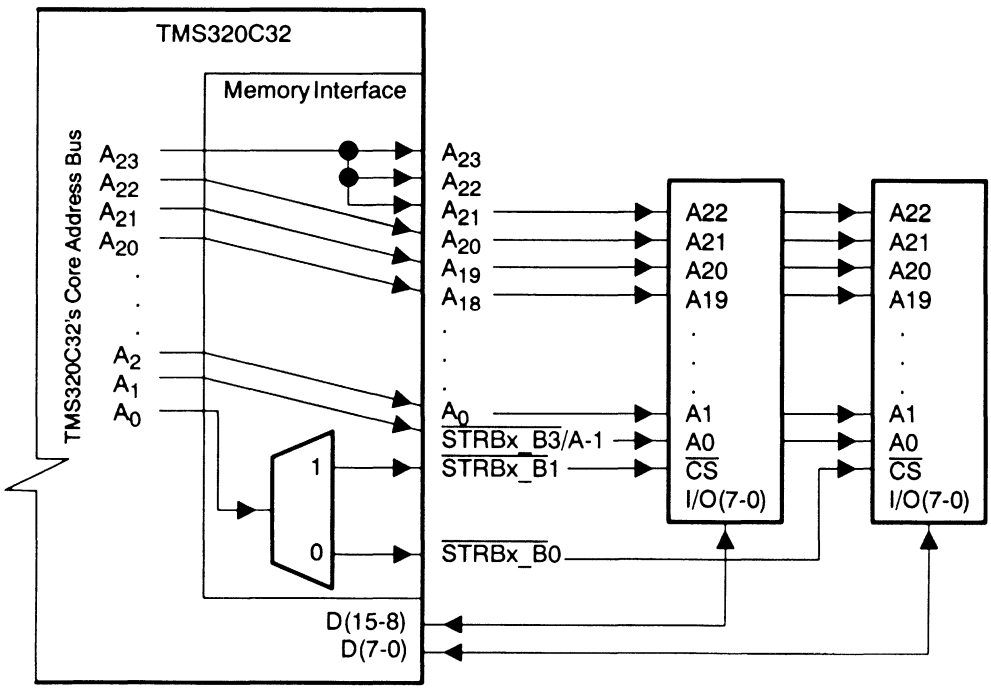
Case 4: 16-Bit Wide Memory With 8-Bit Data Type Size

When the data type size is 8-bit, the 'C32 shifts the internal address two bits to the right before presenting it to the external address pins. In this shift, the memory interface copies the value of the internal address A_{23} to the external address pins A_{23}, A_{22}, and A_{21}. The memory interface also copies the value of the internal address A_1 to the external $\overline{\text{STRBx\_B3}}/A_{-1}$ pin. Furthermore, the memory interface activates the $\overline{\text{STRBx\_B1}}$ and $\overline{\text{STRBx\_B0}}$ pins according to the value of the internal address bit A_0 as shown in Table 7–4. Figure 7–12 shows a functional diagram of the memory interface for 16-bit wide memory with 8-bit data type size.

Table 7–4. Strobe-Byte Enable Behavior for 16-Bit Wide Memory with 8-Bit Data Type Size

Internal A_0	Active Strobe-Byte Enable
0	$\overline{\text{STRBx\_B0}}$
1	$\overline{\text{STRBx\_B1}}$

Figure 7–12. Functional Diagram for 8-Bit Data Type Size and 16-Bit External Memory Width



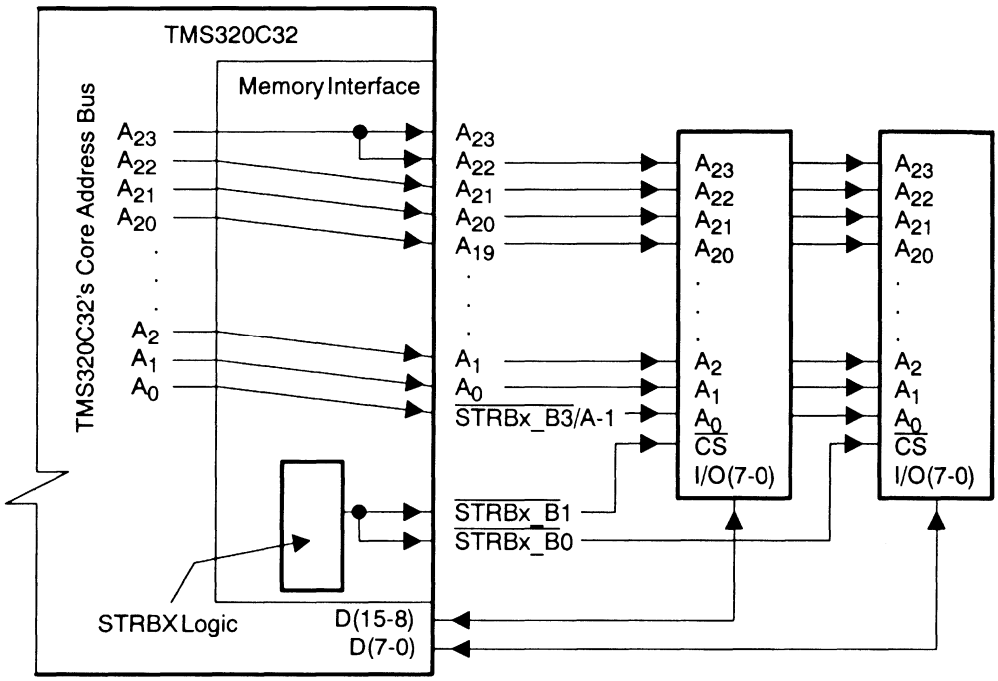
For example, reading or writing to memory locations 4000h to 4004h involves the following:

Internal Address Bus	External Address Pins	$\overline{STRB0\_B3}/A_{-1}$	Active Strobe-Byte Enable	Accessed Data Pins
4000h	1000h	0	$\overline{STRB0\_B0}$	D_{7-0}
4001h	1000h	0	$\overline{STRB0\_B1}$	D_{15-8}
4002h	1000h	1	$\overline{STRB0\_B0}$	D_{7-0}
4003h	1000h	1	$\overline{STRB0\_B1}$	D_{15-8}
4004h	1001h	0	$\overline{STRB0\_B0}$	D_{7-0}

Case 5: 16-Bit Wide Memory With 16-Bit Data Type Size

When the data type size is 16-bit, the 'C32 shifts the internal address one bit to the right before presenting it to the external address pins. In this shift, the memory interface copies the value of the internal address A_{23} to the external address pins A_{23} and A_{22}. Also, the memory interface copies the value of the internal address A_1 to the external $\overline{STRBx\_B3}/A_{-1}$ pin. Moreover, the memory interface activates the $\overline{STRBx\_B1}$ and $\overline{STRBx\_B0}$ during accesses. Figure 7–13 depicts a functional diagram of the memory interface for 16-bit wide memory with 16-bit data type size.

Figure 7–13. Functional Diagram for 16-Bit Data Type Size and 16-Bit External Memory Width



For example, reading or writing to memory locations 4000h to 4004h involves the following:

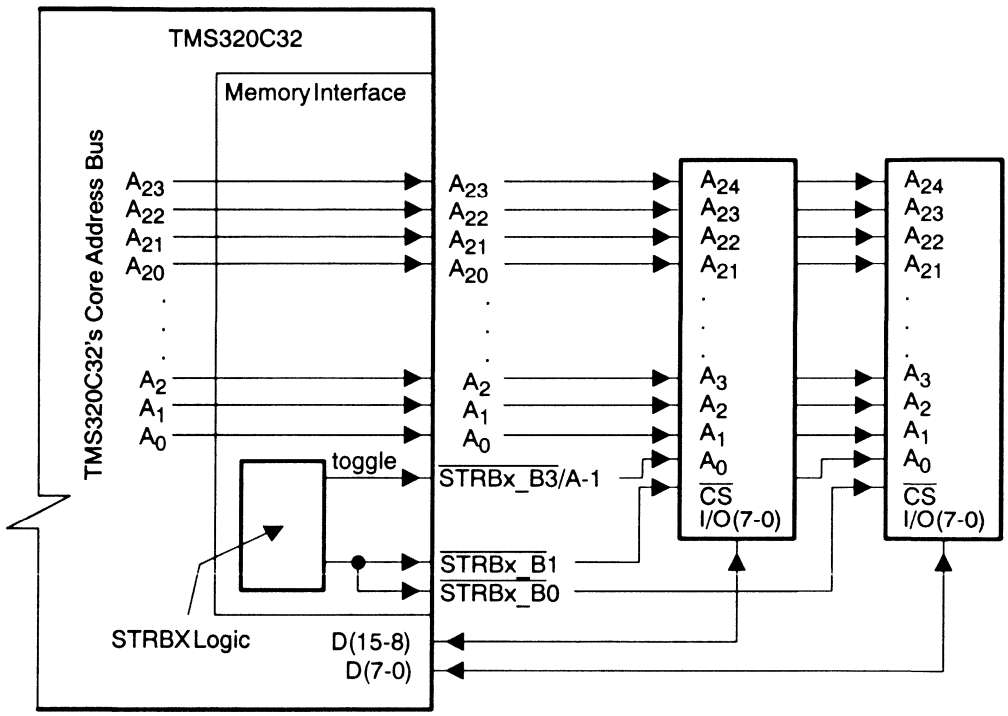
Internal Address Bus	External Address Pins	$\overline{STRB0\_B3}/A_{-1}$	Active Strobe-Byte Enable	Accessed Data Pins
4000h	2000h	0	$\overline{STRB0\_B0}$ and $\overline{STRB0\_B1}$	D_{15-0}
4001h	2000h	1	$\overline{STRB0\_B0}$ and $\overline{STRB0\_B1}$	D_{15-0}
4002h	2001h	0	$\overline{STRB0\_B0}$ and $\overline{STRB0\_B1}$	D_{15-0}
4003h	2001h	1	$\overline{STRB0\_B0}$ and $\overline{STRB0\_B1}$	D_{15-0}
4004h	2002h	0	$\overline{STRB0\_B0}$ and $\overline{STRB0\_B1}$	D_{15-0}

Case 6: 16-Bit Wide Memory with 32-Bit Data Type Size

When the data type size is 32-bit, the 'C32 does not shift the internal address before presenting it to the external address pins. In this case, the memory interface copies the value of the internal address bus to the respective external address pins. The memory interface also toggles $\overline{STRBx\_B3}/A_{-1}$ twice to perform two 16-bit memory accesses. In the consecutive memory accesses, the memory interface activates $\overline{STRBx\_B1}$ and $\overline{STRBx\_B0}$. In summary, the

memory interface adds one wait state to the 32-bit data access. Figure 7–14 depicts a functional diagram of the memory interface for 16-bit wide memory with 32-bit data type size.

Figure 7–14. Functional Diagram for 32-Bit Data Type Size and 16-Bit External Memory Width



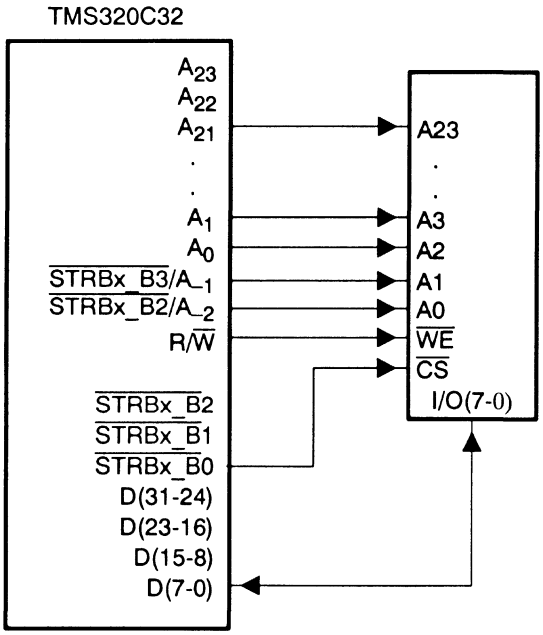
For example, reading or writing to memory locations 4000h to 4004h involves the following:

Internal Address Bus	External Address Pins	$\overline{STRB0\_B3}/A_{-1}$	Active Strobe-Byte Enable	Accessed Data Pins
4000h	4000h	0	$\overline{STRB0\_B0}$ and $\overline{STRB0\_B1}$	D_{15-0}
	4000h	1	$\overline{STRB0\_B0}$ and $\overline{STRB0\_B1}$	D_{15-0}
4001h	4001h	0	$\overline{STRB0\_B0}$ and $\overline{STRB0\_B1}$	D_{15-0}
	4001h	1	$\overline{STRB0\_B0}$ and $\overline{STRB0\_B1}$	D_{15-0}
4002h	4002h	0	$\overline{STRB0\_B0}$ and $\overline{STRB0\_B1}$	D_{15-0}
	4002h	1	$\overline{STRB0\_B0}$ and $\overline{STRB0\_B1}$	D_{15-0}
4003h	4003h	0	$\overline{STRB0\_B0}$ and $\overline{STRB0\_B1}$	D_{15-0}
	4003h	1	$\overline{STRB0\_B0}$ and $\overline{STRB0\_B1}$	D_{15-0}
4004h	4004h	0	$\overline{STRB0\_B0}$ and $\overline{STRB0\_B1}$	D_{15-0}
	4004h	1	$\overline{STRB0\_B0}$ and $\overline{STRB0\_B1}$	D_{15-0}

7.3.4 8-Bit Wide Memory Interface

'C32 memory interface to 8-bit wide external memory utilizes $\overline{\text{STRBx\_B3}}$ and $\overline{\text{STRBx\_B2}}$ pins as an additional address pins, A_{-1} and A_{-2}, respectively, while using $\overline{\text{STRBx\_B0}}$ as strobe-byte enable pin as shown in Figure 7–15. Note that the external memory address pins are connected to the 'C32's address pins $A_{21}A_{20}...A_1A_0A_{-1}A_{-2}$. In this manner, the 'C32 can read/write a single 32-, 16-, or 8-bit value from the external 8-bit wide memory.

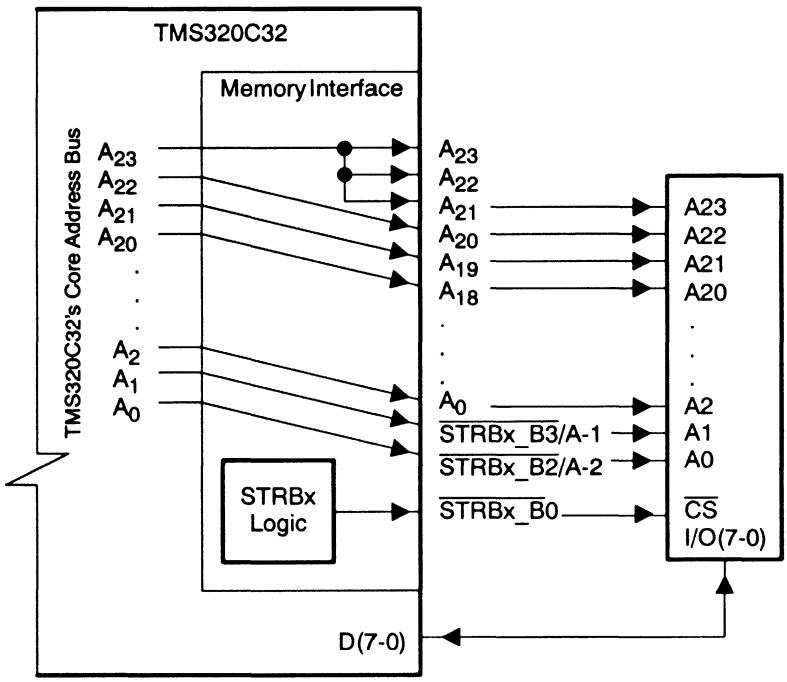
Figure 7–15. External Memory Interface for 8-Bit SRAMs



Case 7: 8-Bit Wide Memory With 8-Bit Data Type Size

Similarly to case 4, the 'C32 shifts the internal address two bits to the right before presenting it to the external address pins when the data type is 8-bit. As in case 4, the memory interface copies the value of the internal address A_{23} to the external address pins A_{23}, A_{22}, and A_{21}. But in case 7, the memory interface also copies the value of the internal address A_1 to the external $\overline{\text{STRBx\_B3}}/A_{-1}$ pin and the value of A_0 to the external $\overline{\text{STRBx\_B2}}/A_{-2}$. Moreover, the memory interface only actives the $\overline{\text{STRBx\_B0}}$ pin during the external memory access. Figure 7–16 depicts a functional diagram of the memory interface for 8-bit wide memory with 8-bit data type size.

Figure 7–16. Functional Diagram for 8-Bit Data Type Size and 8-Bit External Memory Width



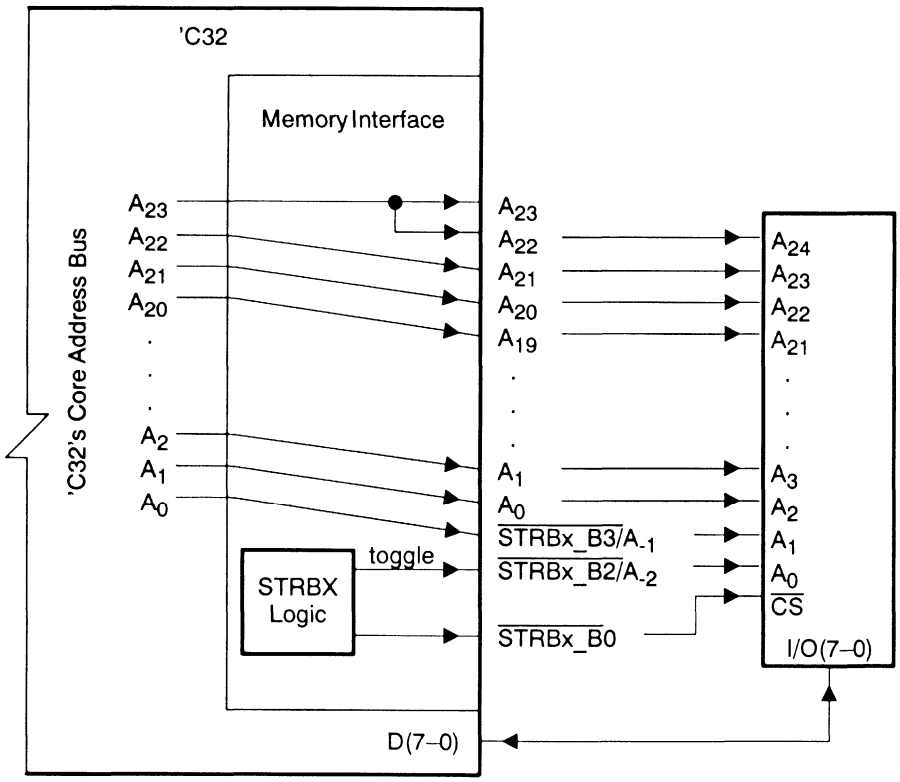
For example, reading or writing to memory locations A04000h to A04004h involves the following:

Internal Address Bus	External Address Pins	$\overline{STRB0\_B3}/A_{-1}$	$\overline{STRB0\_B3}/A_{-2}$	Active Strobe-Byte Enable	Accessed Data Pins
A04000h	E81000h	0	0	$\overline{STRB1\_B0}$	D_{7-0}
A04001h	E81000h	0	1	$\overline{STRB1\_B0}$	D_{7-0}
A04002h	E81000h	1	0	$\overline{STRB1\_B0}$	D_{7-0}
A04003h	E81000h	1	1	$\overline{STRB1\_B0}$	D_{7-0}
A04004h	E81001h	0	0	$\overline{STRB1\_B0}$	D_{7-0}

Case 8: 8-Bit Wide Memory With 16-Bit Data Type Size

When the data type size is 16-bit, the 'C32 shifts the internal address one bit to the right before presenting it to the external address pins. In this shift, the memory interface copies the value of the internal address A_{23} to the external address pins A_{23} and A_{22}. Also, the memory interface copies the value of the internal address A_0 to the external $\overline{\text{STRBx\_B3}}/A_{-1}$ pin. Furthermore, the memory interface toggles $\overline{\text{STRBx\_B2}}/A_{-2}$ twice to perform two 8-bit memory accesses. Moreover, the memory interface activates the $\overline{\text{STRBx\_B0}}$ during accesses. In summary, the memory interface adds one wait state to the 16-bit data access. Figure 7–17 depicts a functional diagram of the memory interface for 8-bit wide memory with 16-bit data type size.

Figure 7–17. Functional Diagram for 16-Bit Data Type Size and 8-Bit External Memory Width



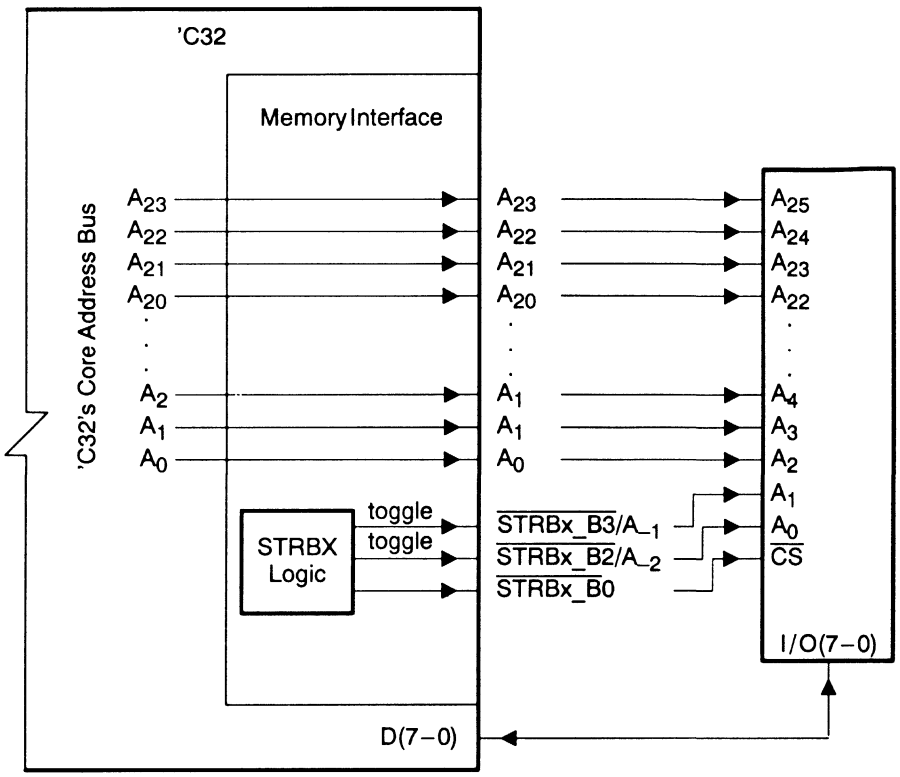
For example, reading or writing to memory locations A04000h to A04002h involves the following:

Internal Address Bus	External Address Pins	$\overline{STRB0\_B3}/A_{-1}$	$\overline{STRB0\_B3}/A_{-2}$	Active Strobe-Byte Enable	Accessed Data Pins
A04000h	D02000h	0	0	$\overline{STRB1\_B0}$	D_{7-0}
	D02000h	0	1	$\overline{STRB1\_B0}$	D_{7-0}
A04001h	D02001h	1	0	$\overline{STRB1\_B0}$	D_{7-0}
	D02001h	1	1	$\overline{STRB1\_B0}$	D_{7-0}
A04002h	D02002h	0	0	$\overline{STRB1\_B0}$	D_{7-0}
	D02002h	0	1	$\overline{STRB1\_B0}$	D_{7-0}

Case 9: 8-Bit Wide Memory With 32-Bit Data Type Size

When the data type size is 32-bit, the 'C32 does not shift the internal address before presenting it to the external address pins. In this case, the memory interface copies the value of the internal address bus to the respective external address pins. The memory interface also toggles $\overline{STRBx\_B3}/A_{-1}$ and $\overline{STRBx\_B2}/A_{-2}$ to perform four 8-bit memory accesses. In the consecutive memory accesses, the memory interface activates $\overline{STRBx\_B0}$. In summary, the memory interface adds three wait states to the 32-bit data access. Figure 7–18 depicts a functional diagram of the memory interface for 8-bit wide memory with 32-bit data type size.

Figure 7–18. Functional Diagram for 32-Bit Data Type Size and 8-Bit External Memory Width



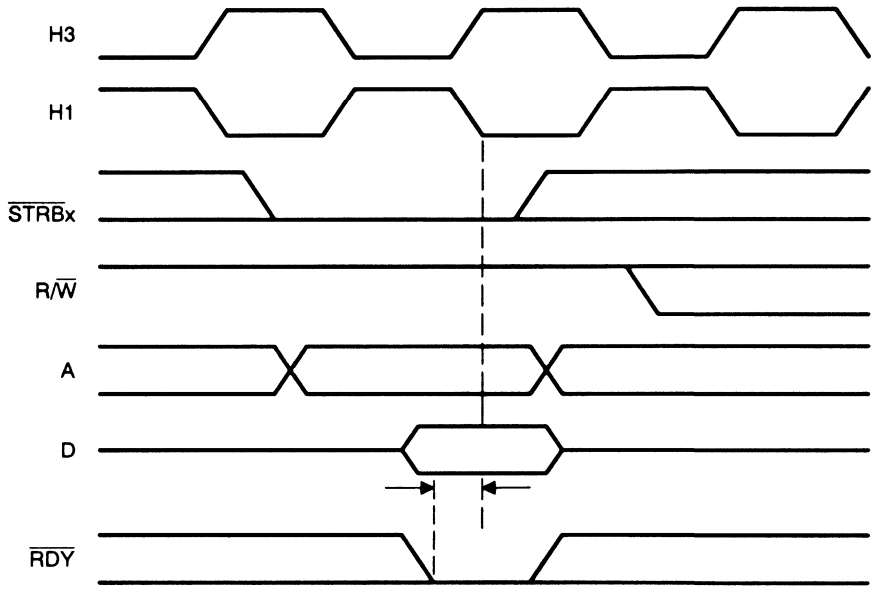
For example, reading or writing to memory locations A04000h to A04001h involves the following:

Internal Address Bus	External Address Pins	$\overline{STRB0\_B3}/A_{-1}$	$\overline{STRB0\_B3}/A_{-2}$	Active Strobe-Byte Enable	Accessed Data Pins
A04000h	A04000h	0	0	$\overline{STRB1\_B0}$	D_{7-0}
	A04000h	0	1	$\overline{STRB1\_B0}$	D_{7-0}
	A04000h	1	0	$\overline{STRB1\_B0}$	D_{7-0}
	A04000h	1	1	$\overline{STRB1\_B0}$	D_{7-0}
A04001h	A04001h	0	0	$\overline{STRB1\_B0}$	D_{7-0}
	A04001h	0	1	$\overline{STRB1\_B0}$	D_{7-0}
	A04001h	1	0	$\overline{STRB1\_B0}$	D_{7-0}
	A04001h	1	1	$\overline{STRB1\_B0}$	D_{7-0}

7.3.5 External Ready Timing Improvement

The $\overline{RDY}$ timing should reference to the H1 low signal as shown in Figure 7–19. This is equivalent to the 'C4x's ready timing, which increases the time between valid address and the sampling of $\overline{RDY}$. This facilitates the memory hardware interface by increasing the address decode circuit response time to generate a ready signal.

Figure 7–19. $\overline{RDY}$ Timing for Memory Read



7.4 Bus Timing

This section discusses functional timing of operations on the external memory bus. Detailed timing specifications are contained in the *TMS320C32 Data Sheet*. The timing of $\overline{\text{STRB}}$0 and $\overline{\text{STRB}}$1 bus cycles is identical and discussed in subsection 7.4.1. The acronym $\overline{\text{STRB}}$x is used in references that pertain equally to $\overline{\text{STRB}}$0 and $\overline{\text{STRB}}$1. The $\overline{\text{IOSTRB}}$ bus cycles are timed differently and are discussed in subsection 7.4.2.

7.4.1 $\overline{\text{STRB}}$0 and $\overline{\text{STRB}}$1 Bus Cycles

All bus cycles comprise integral numbers of H1 clock cycles. One H1 cycle is defined from one falling edge of H1 to the next falling edge of H1. For full speed (zero wait-state) accesses on $\overline{\text{STRB}}$0 and $\overline{\text{STRB}}$1, writes take two H1 cycles and reads take one cycle. However, if the read immediately follows a write, the read takes two cycles. Note that writes to internal memory take one cycle if no other accesses to that interface are in progress. The following discussion pertains to zero wait-state accesses, unless otherwise specified.

The $\overline{\text{STRB}}$x signal is low for the active portion of both reads and writes (one H1 cycle). Additionally, before and after the active portions of writes only ($\overline{\text{STRB}}$x low), there is a transition of one H1 cycle. During this transition cycle the following might occur:

- $\overline{\text{STRB}}$x is high.
- If required, R/$\overline{\text{W}}$ changes state on the rising edge of H1.
- If required, address changes on the rising edge of H1 if the previous H1 cycle performed a write. If the previous H1 cycle performed a read, address changes on the falling edge of H1.

Figure 7–20 illustrates a zero wait-state read-read-write sequence for $\overline{\text{STRB}}$x active. The data is read as late in the cycle as possible to allow for the maximum access time from address valid. Note that although external writes take two cycles, writes to internal memory take one cycle if no other accesses to that interface are in progress. Similarly to typical external interfaces, the R/$\overline{\text{W}}$ signal does not change until $\overline{\text{STRB}}$0 and $\overline{\text{STRB}}$1 are deactivated.

Figure 7–20. Read-Read-Write Sequence for $\overline{\text{STRB}}$x active

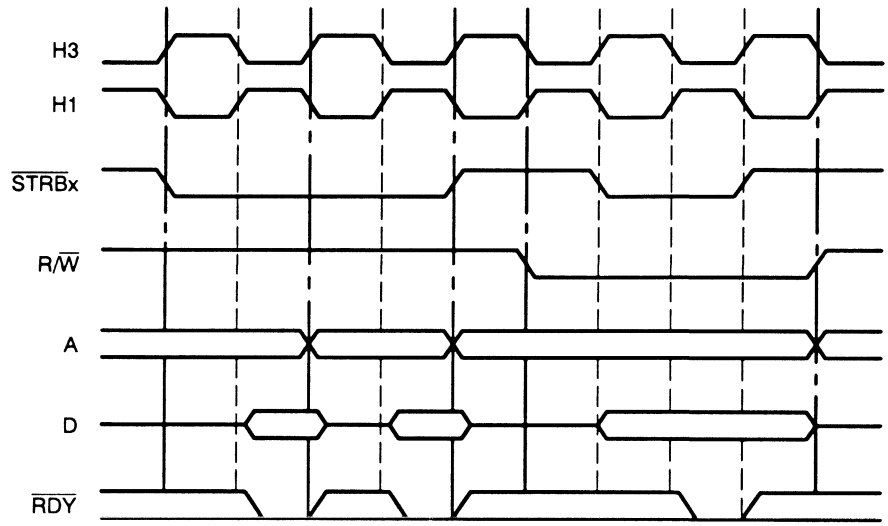


Figure 7–21 illustrates a zero wait-state write-write-read sequence for $\overline{\text{STRB}}$x active. During back-to-back writes, the data is valid approximately one-half cycle after $\overline{\text{STRB}}$x changes for the first write, but for subsequent writes the data is valid when $\overline{\text{STRB}}$x changes.

Figure 7–21. Write-Write-Read Sequence for $\overline{\text{STRB}}$x active

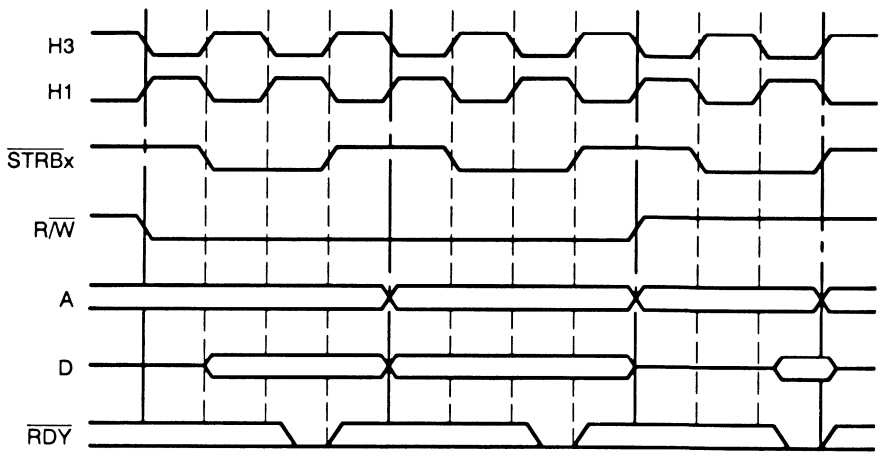


Figure 7–22 illustrates a one wait-state read sequence for $\overline{\text{STRB}}$x active. On the first H1 cycle $\overline{\text{RDY}}$ is high therefore, the read sequence is extended for one extra cycle. On the second H1 cycle $\overline{\text{RDY}}$ is low and the read sequence is terminated.

Figure 7–22. One Wait-State Read Sequence for $\overline{\text{STRB}}$x active

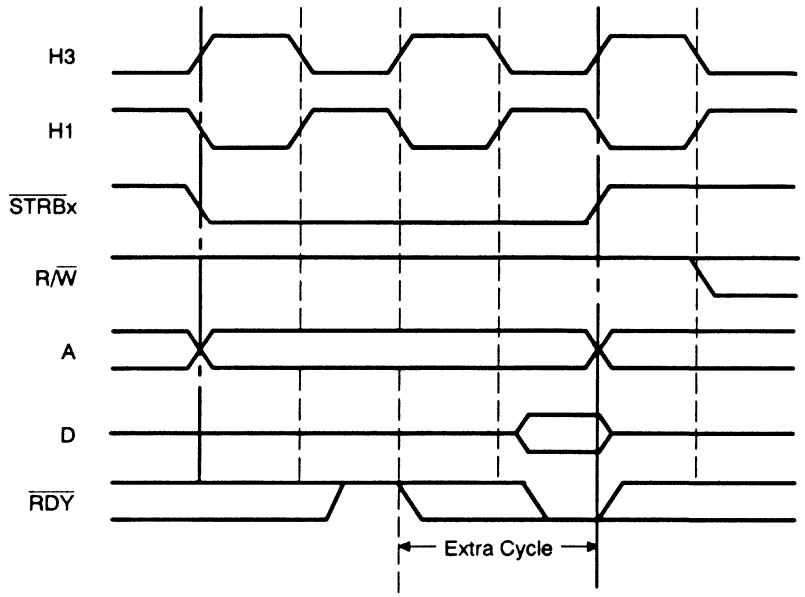
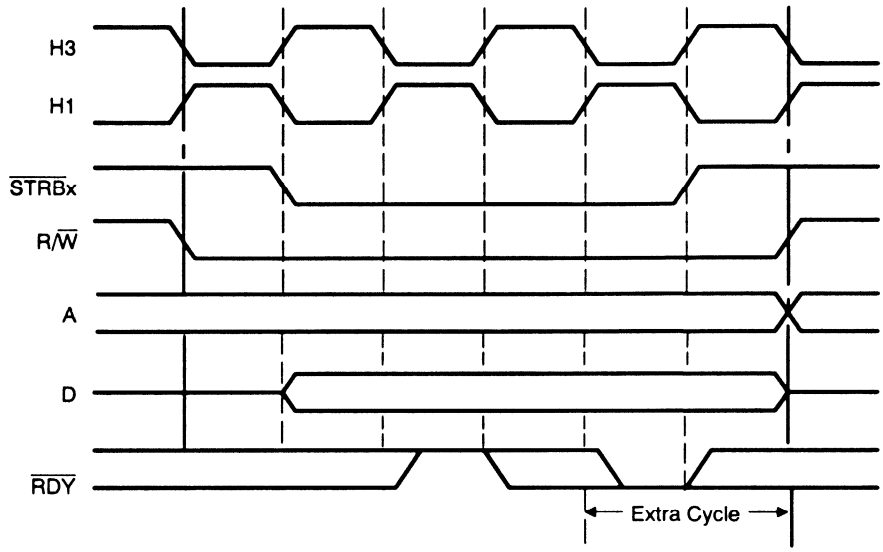


Figure 7–23 illustrates a one wait-state write sequence for $\overline{\text{STRB}}$x active. On the first H1 cycle $\overline{\text{RDY}}$ is high therefore, the write sequence is extended for one extra cycle. On the second H1 cycle $\overline{\text{RDY}}$ is low and the write sequence is terminated.

Figure 7–23. One Wait-State Write Sequence for $\overline{\text{STRB}}$x Active



7.4.2 $\overline{\text{IOSTRB}}$ Bus Cycles

In contrast to $\overline{\text{STRB}}$0 and $\overline{\text{STRB}}$1 bus cycles, $\overline{\text{IOSTRB}}$ full speed (zero wait-state) reads and writes consume two H1 cycles. During these cycles, the $\overline{\text{IOSTRB}}$ signal is low from the rising edge of the first H1 cycle to the rising edge of the second H1 cycle. Also, the address changes on the falling edge of the first H1 cycle and R/$\overline{\text{W}}$ changes state on the falling edge of H1. This provides a valid address to peripherals that may change their status bits when read or written while $\overline{\text{IOSTRB}}$ is active. Moreover, the $\overline{\text{IOSTRB}}$ signal is high between $\overline{\text{IOSTRB}}$ read and write cycles.

Figure 7–24 illustrates a zero wait-state read and write sequence for $\overline{\text{IOSTRB}}$ active. During writes, the data is valid when $\overline{\text{IOSTRB}}$ changes.

Figure 7–24. Zero Wait-State Read and Write Sequence for $\overline{\text{IOSTRB}}$ Active

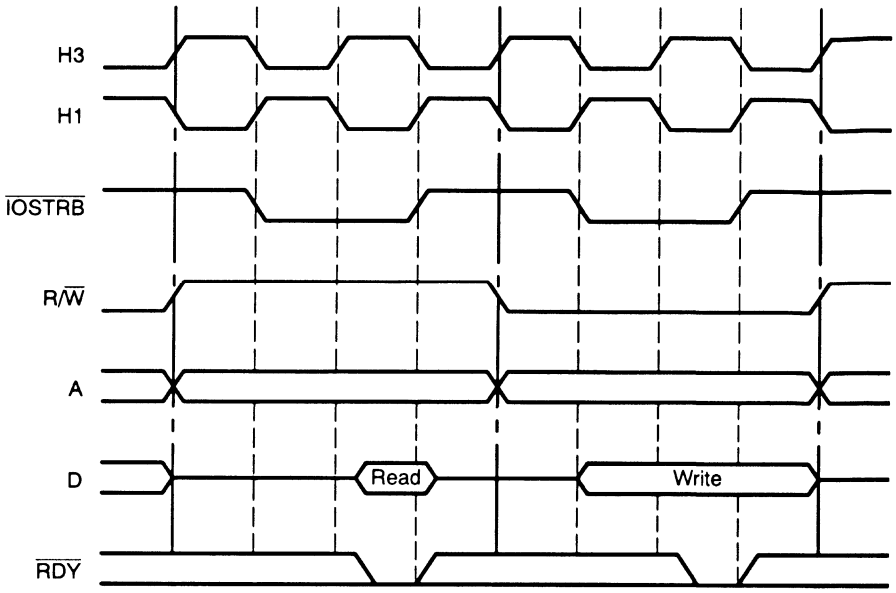


Figure 7–25 depicts a one wait-state read sequence for $\overline{\text{IOSTRB}}$ active. Figure 7–26 shows a one wait-state write sequence for $\overline{\text{IOSTRB}}$ active. For each wait-state added, $\overline{\text{IOSTRB}}$, R/$\overline{\text{W}}$, and A are extended for one extra clock cycle. Writes hold the data on the bus for one extra clock cycle. $\overline{\text{RDY}}$ is sampled on each extra cycle and the sequenced is terminated when $\overline{\text{RDY}}$ is low.

Figure 7–25. One Wait-State Read Sequence for $\overline{IOSTRB}$ Active

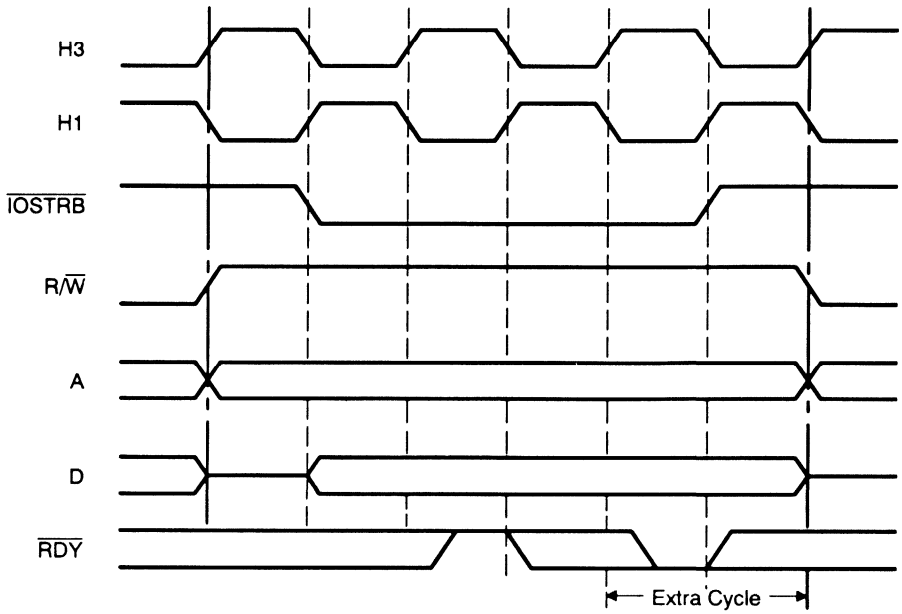


Figure 7–26. One Wait-State Write Sequence for $\overline{IOSTRB}$ Active

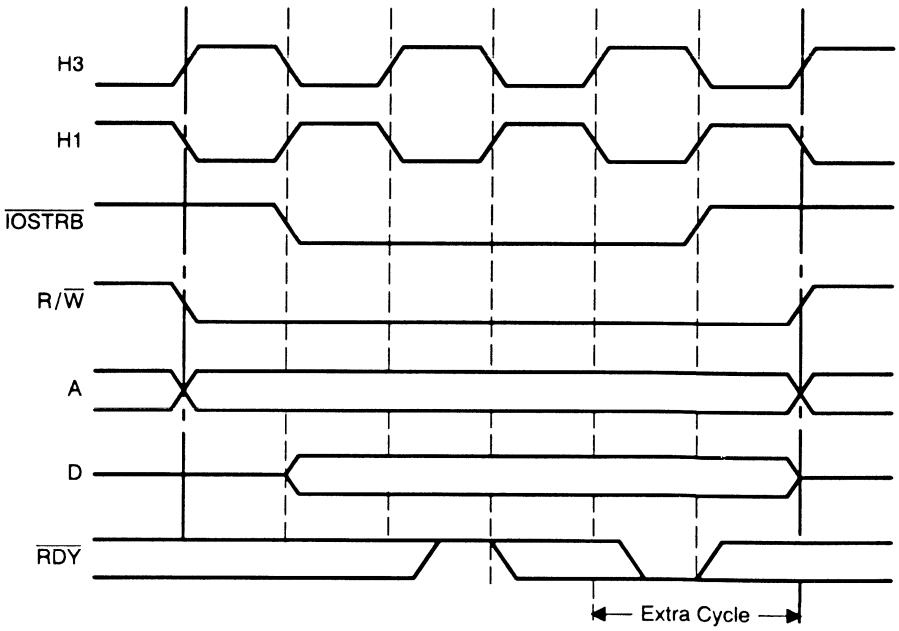


Figure 7–27 and Figure 7–28 illustrate the transitions between $\overline{STRB}$x reads and $\overline{IOSTRB}$ writes and reads, respectively. In these transitions, the address changes on the falling edge of the H1 cycle.

Figure 7–27. $\overline{STRB}$x Read and $\overline{IOSTRB}$ Write

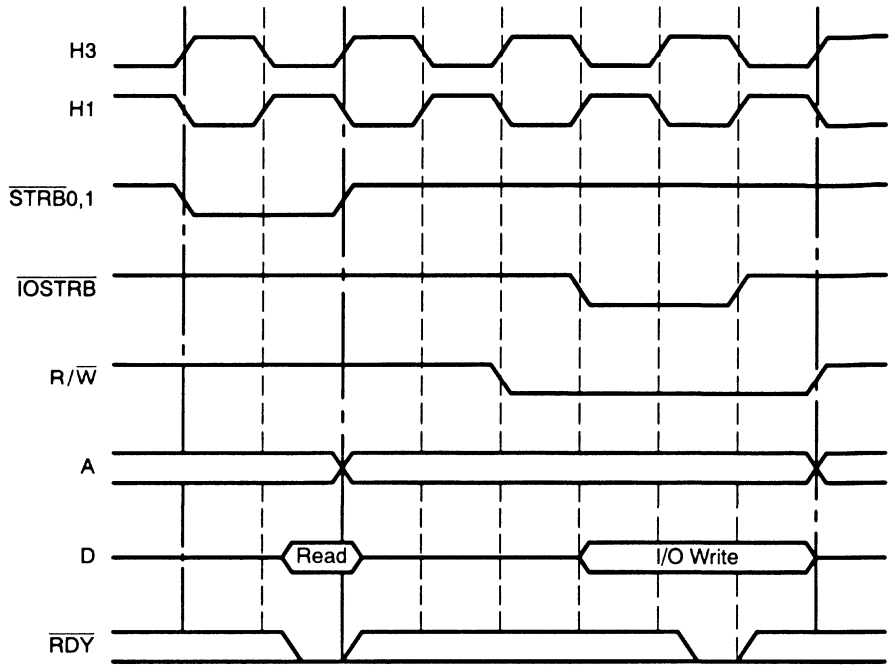


Figure 7–28. $\overline{STRB}$x Read and $\overline{IOSTRB}$ Read

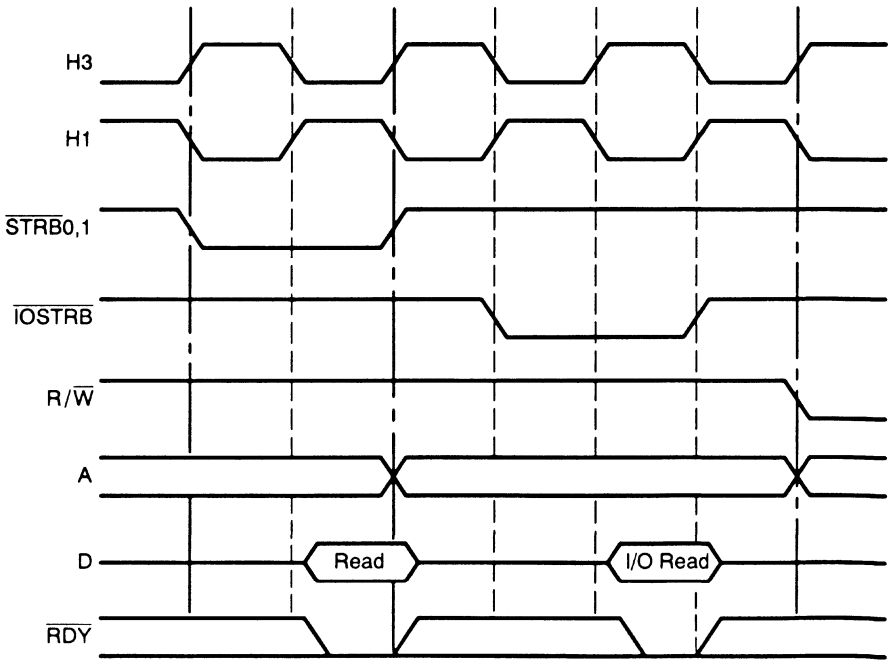


Figure 7–29 and Figure 7–30 illustrate the transitions between $\overline{\text{STRBx}}$ writes and $\overline{\text{IOSTRB}}$ writes and reads, respectively. In these transitions, the address changes on the falling edge of the H3 cycle.

Figure 7–29. $\overline{\text{STRBx}}$ Write and $\overline{\text{IOSTRB}}$ Write

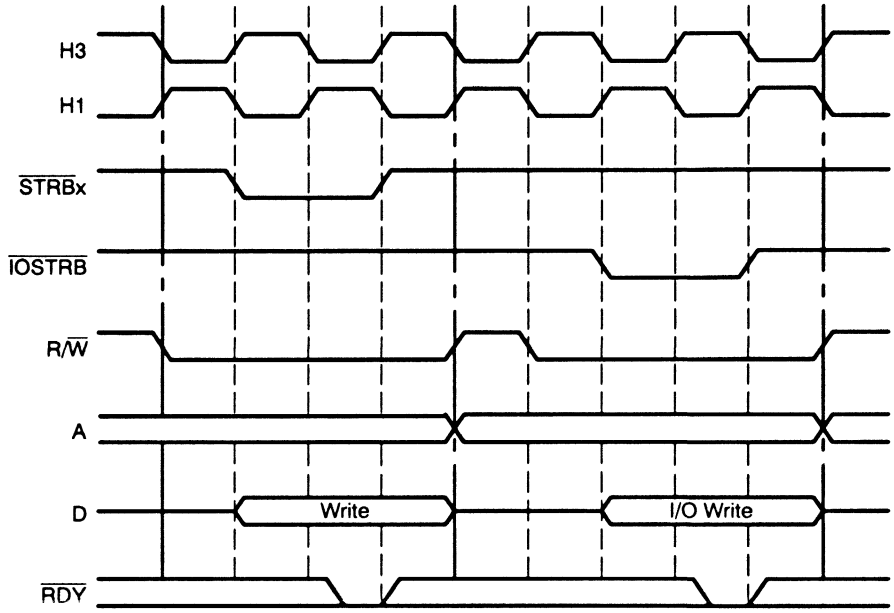


Figure 7–30. $\overline{\text{STRBx}}$ Write and $\overline{\text{IOSTRB}}$ Read

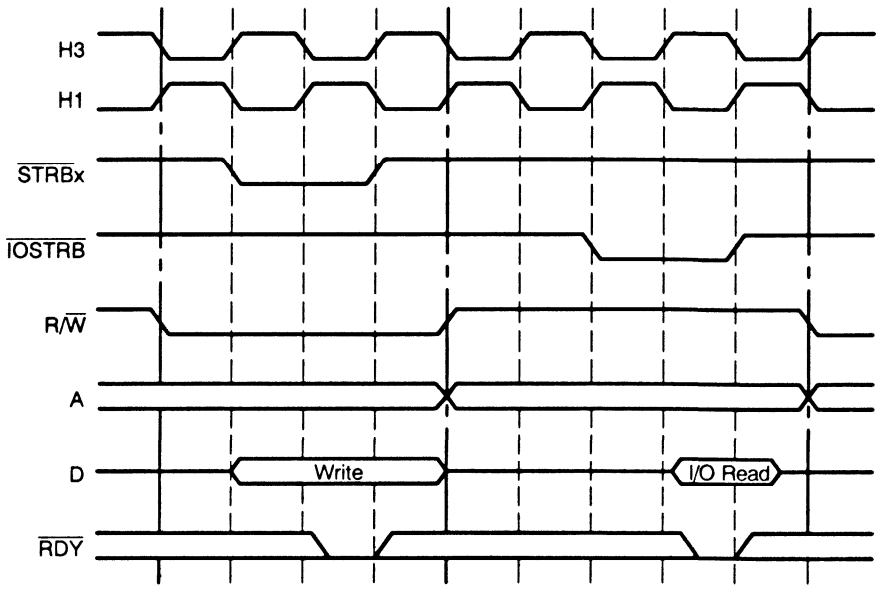


Figure 7–31 through Figure 7–34 show the transitions between $\overline{\text{IOSTRB}}$ writes/reads and $\overline{\text{STRB}}$x writes/reads. In these transitions, the address changes on the rising edge of the H3 cycle.

Figure 7–31. $\overline{\text{IOSTRB}}$ Write and $\overline{\text{STRB}}$x Write

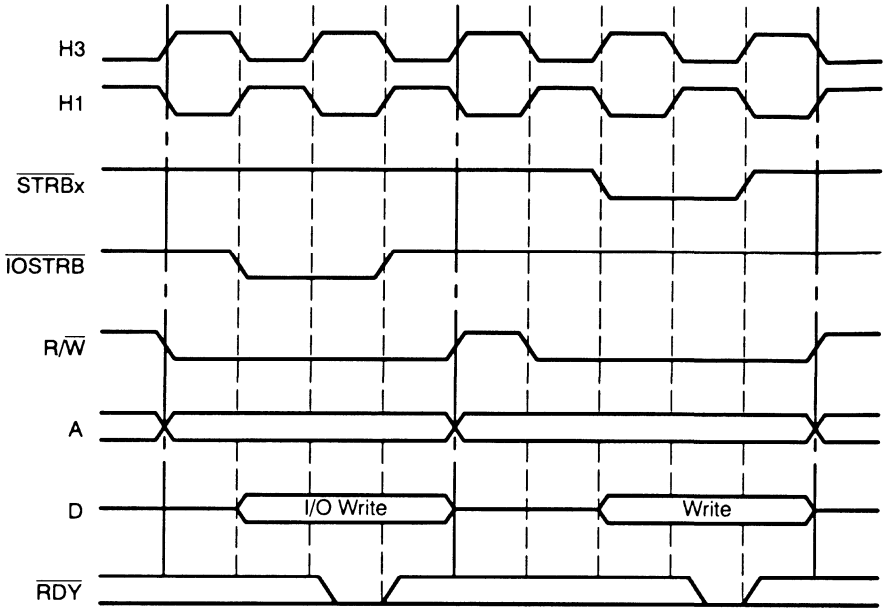


Figure 7–32. $\overline{IOSTRB}$ Write and $\overline{STRB}$x Read

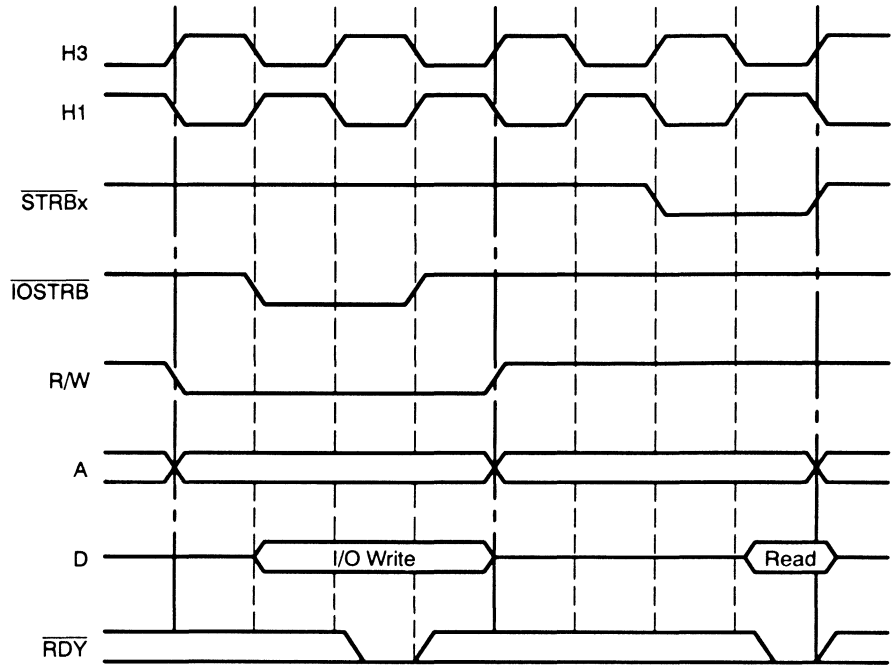


Figure 7–33. $\overline{IOSTRB}$ Read and $\overline{STRB}$x Write

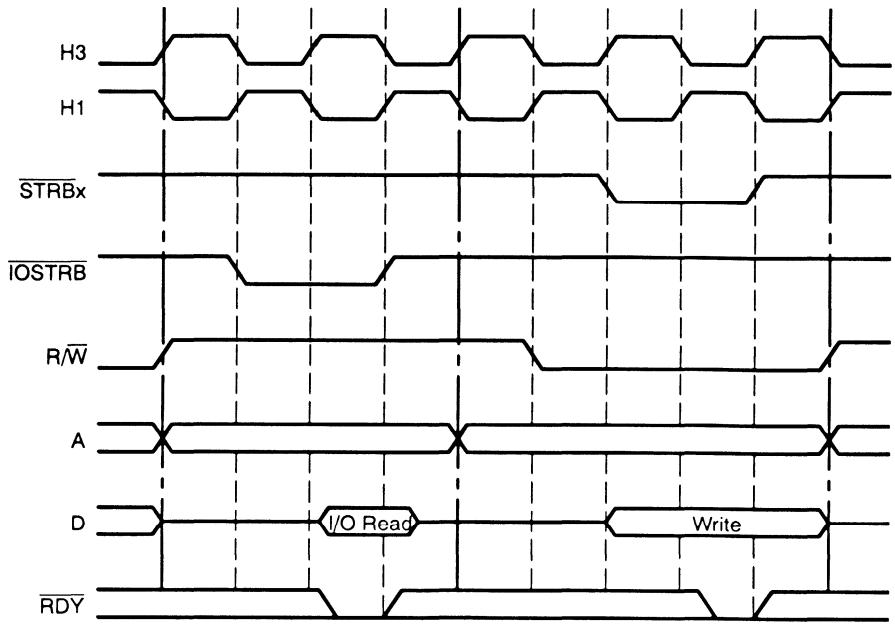


Figure 7–34. $\overline{IOSTRB}$ Read and $\overline{STRB}$x Read

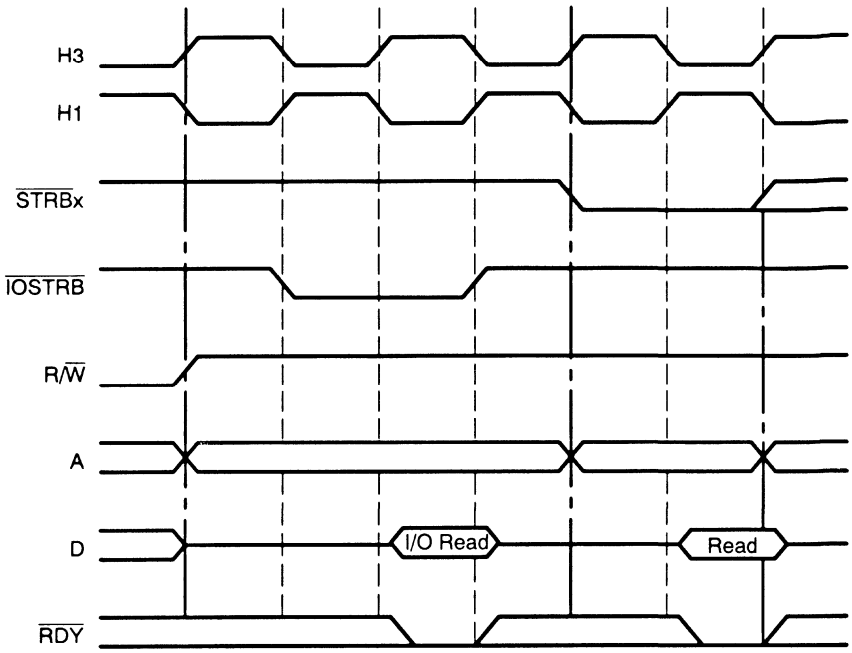


Figure 7–35 through Figure 7–37 illustrate the transitions between reads and writes.

Figure 7–35. $\overline{IOSTRB}$ Write and Read

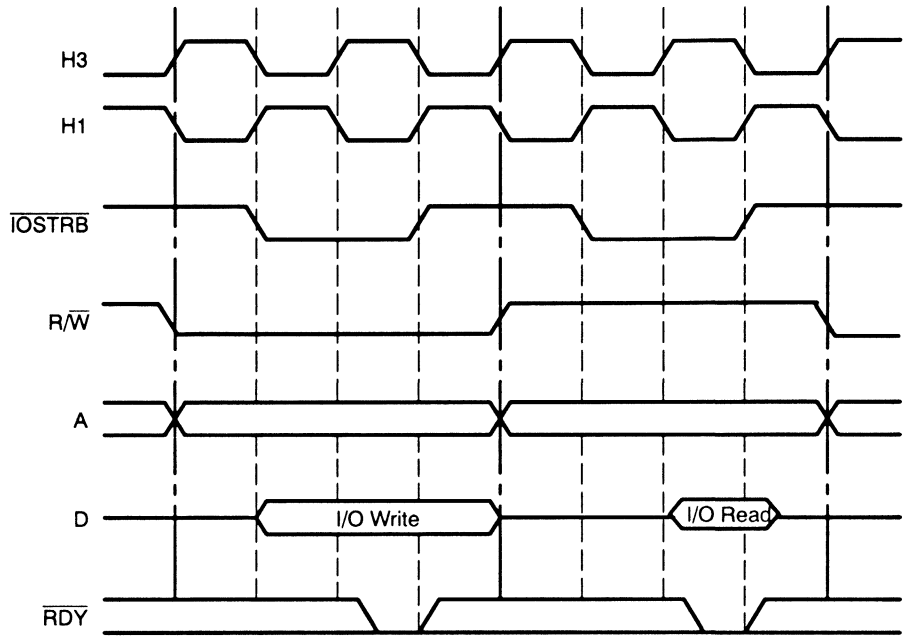


Figure 7–36. $\overline{IOSTRB}$ Write and Write

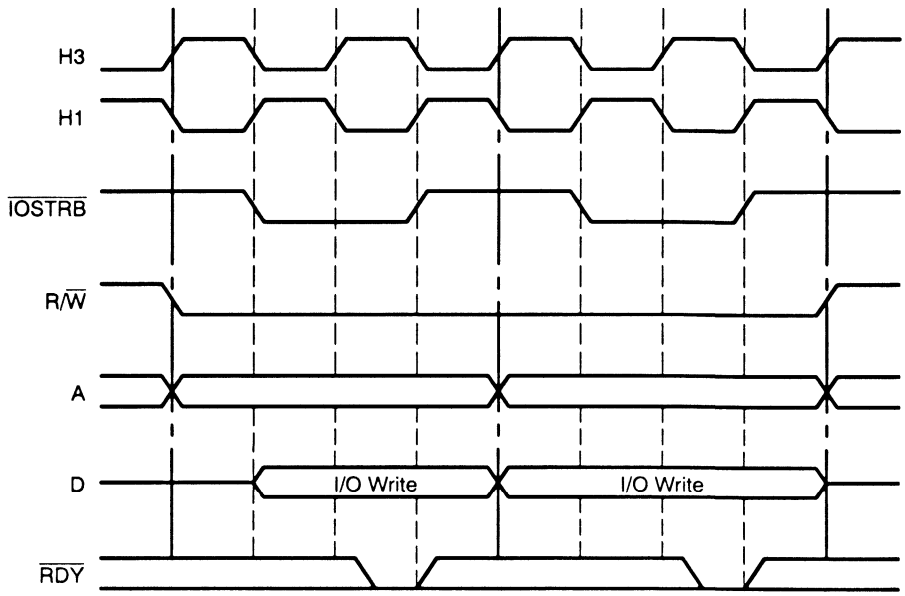
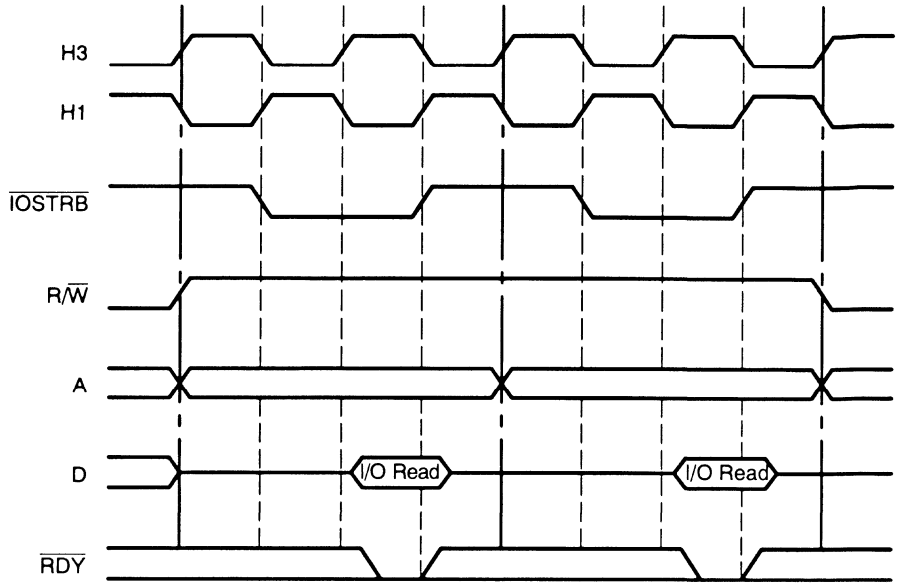


Figure 7–37. $\overline{IOSTRB}$ Read and Read



7.4.3 Inactive Bus States

Figure 7–38 and Figure 7–39 show the signal states when a bus becomes inactive after an $\overline{IOSTRB}$ or $\overline{STRB}$x, respectively. The strobes ($\overline{STRB0}$, $\overline{STRB1}$, $\overline{IOSTRB}$, and R/$\overline{W}$) are deasserted going to a high level. The address bus preserves the last value and the ready signal ($\overline{RDY}$) is ignored.

Figure 7–38. Inactive Bus States Following $\overline{IOSTRB}$ Bus Cycle

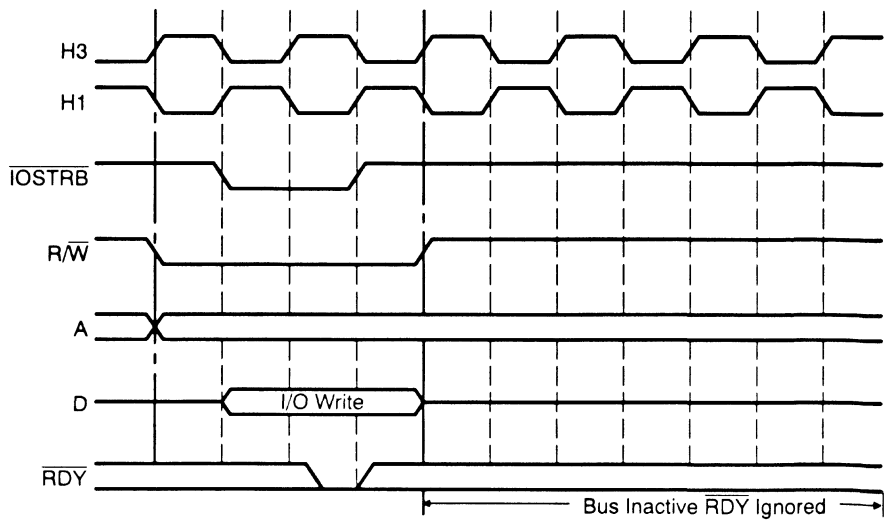
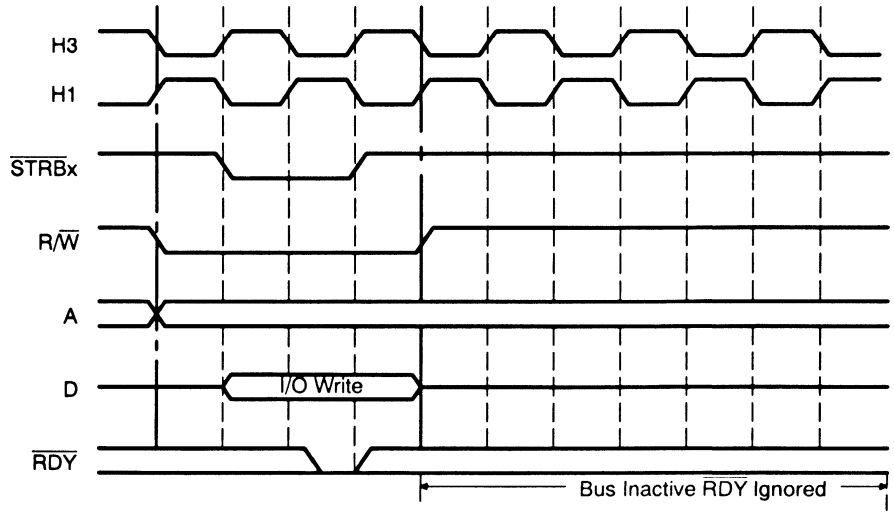


Figure 7–39. Inactive Bus States Following $\overline{\text{STRB}}$x Bus Cycle



Chapter 8

Peripherals

The 'C32 has an improved DMA that supports two channels and configurable priorities. The next sections discuss the new features.

8.1 Two-Channel DMA Features

'C32 has a two-channel (channel 0 and channel 1) DMA instead of a one-channel DMA as in the 'C30/'C31 device. The 'C32's DMA functions similarly to that of the 'C30/'C31 DMA but with the addition of DMA/CPU priority scheme and inter-DMA priority mode. Although the 'C32 CPU supports both floating point and integer data access with different data size from the external memory, the 'C32's DMA transfer is strictly an integer data transfer. The integer data access of the 'C32 DMA is the same as the CPU integer data access – 32-bit internal and data size conversion at the external memory interface port.

8.1.1 DMA Global Control Registers

Each of the two channels has its own control, source and destination address, and transfer counter registers (Figure 8–1).

Figure 8–1. Memory-Mapped Locations for a DMA Channels

Address	Register	Address	Register
808000h	DMA0 Global Control	808010h	DMA1 Global Control
808001h	Reserved	808011h	Reserved
808002h	Reserved	808012h	Reserved
808003h	Reserved	808013h	Reserved
808004h	DMA0 Source Address	808014h	DMA1 Source Address
808005h	Reserved	808015h	Reserved
808006h	DMA0 Destination Address	808016h	DMA1 Destination Address
808007h	Reserved	808017h	Reserved
808008h	DMA0 Transfer Counter	808018h	DMA1 Transfer Counter
808009h	Reserved	808019h	Reserved
	. . .		. . .
80800Fh	Reserved	80801Fh	Reserved

8.1.4 CPU/DMA Interrupts

Channel 0 transfers can be synchronized through the use of INT0, INT1, INT2, INT3, XINT0, TINT0, TINT1, and DINT1. Channel 1 transfers can be synchronized through the use of INT0, INT1, INT2, INT3, RINT0, TINT0, TINT1, and DINT0. The Interrupt Enable Register is shown in Figure 8–2.

Figure 8–2. CPU/DMA Interrupt Enable Register

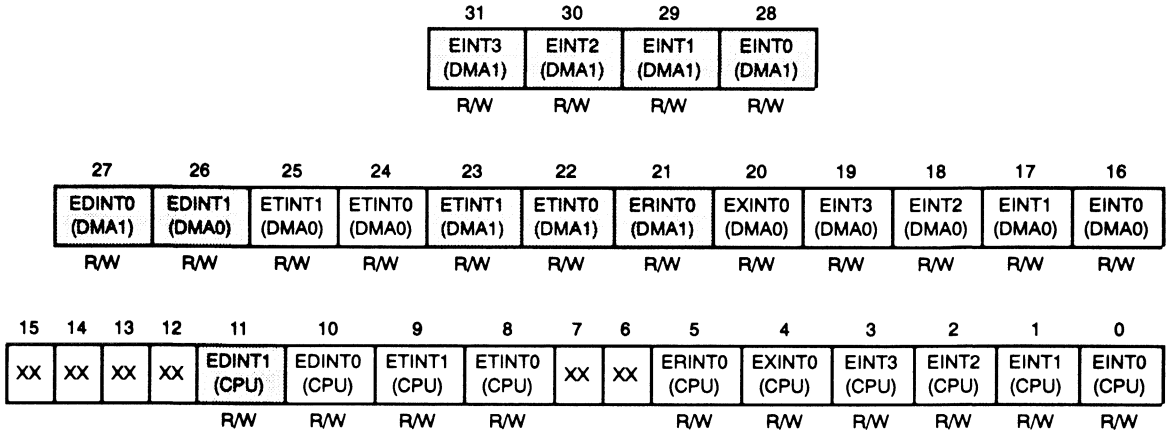
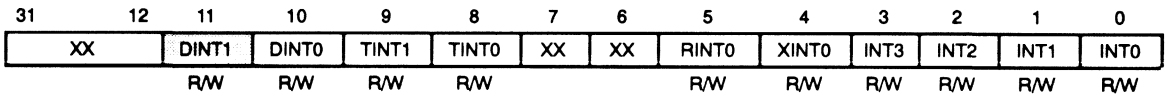


Figure 8–3 depicts the Interrupt Flag Register. In this figure, the DINT0 bit refers to DMA channel 0 interrupt flag while DINT1 bit refers to the DMA channel 1 interrupt flag.

Figure 8–3. CPU Interrupt Flag Register



8.3.5 DMA Channel Arbitration

'C32's DMA controller priority is configurable through the DMA PRI and PRIORITY MODE bits of the DMA global control register as shown in Figure 8–4, Table 8–2, and Table 8–3. The PRIORITY MODE bit is only available on DMA0 control register. The shaded entries in Table 8–2, and Table 8–3 indicate reset values.

Figure 8–4. DMA0 Global Control Register

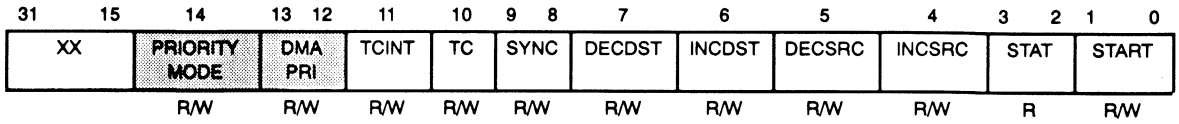


Table 8–2. CPU/DMA Priority

Bit 13	Bit 12	DMA PRI Function Description
0	0	DMA has lower priority than the CPU access. If the DMA channel and the CPU are requesting the same resource, then the CPU will proceed. This is the reset condition.
0	1	Rotating arbitration, which sets priorities between the CPU and DMA channel by alternating their accesses, but not exactly equally. Priority rotates between the CPU and DMA accesses when they conflict during consecutive instruction cycles.
1	0	Reserved.
1	1	DMA access has higher priority than the CPU accesses. If the DMA channel and the CPU are requesting the same resource, then the DMA proceeds.

Table 8–3. DMA Priority Mode of DMA0 Control Register

Bit 14	PRIORITY MODE Description
0	Rotating priority for the two DMA channels.
1	Fixed priority for the two DMA channels.

For fixed DMA priority mode, DMA channel 0 always has priority over DMA channel 1. If both DMA channels request the service, DMA channel 0 will transfer first. For rotating DMA priority mode, DMA channel 0 has priority after the device is reset. After reset, the last channel serviced has the lowest priority. The arbitration is performed at DMA service boundaries, that is, after either a DMA read or a DMA write.

8.1.6 CPU Changes To Support DMA

CPU conflicts do not prevent both DMA channels from servicing interrupts.

Chapter 9

Pipeline Operation

The pipeline operation in the 'C32 is identical to that in the 'C30 and 'C31 and is discussed in the *TMS320C3x User's Guide* (literature number SPRU031).

Chapter 10

Assembly Language Instructions

The instruction set for the 'C32 is identical to the 'C30 and 'C31 instruction set and is discussed in the *TMS320C3x User's Guide* (literature number SPRU031).

Chapter 11

Software Applications

The software applications for the 'C32 are the same as those for the 'C30 and 'C31 and are discussed in the *TMS320C3x User's Guide* (literature number SPRU031).

Chapter 12

Hardware Applications

The 'C32 enhanced memory interface design can be used to implement a wide variety of system configurations without additional logic. Its external bus provides a parallel 8-, 16- or 32-bit interface to external memories and peripherals. By grouping data type sizes of equal length into a particular memory strobe section, the 'C32 can mix two data type sizes with zero wait-state accesses. This chapter describes examples that exploit these techniques to achieve maximum performance and to minimize memory storage. Refer to the *Interfacing Memory to the TMS320C32 DSP Application Report* (literature number SPRA040) for more information.

12.1 Maximum Performance

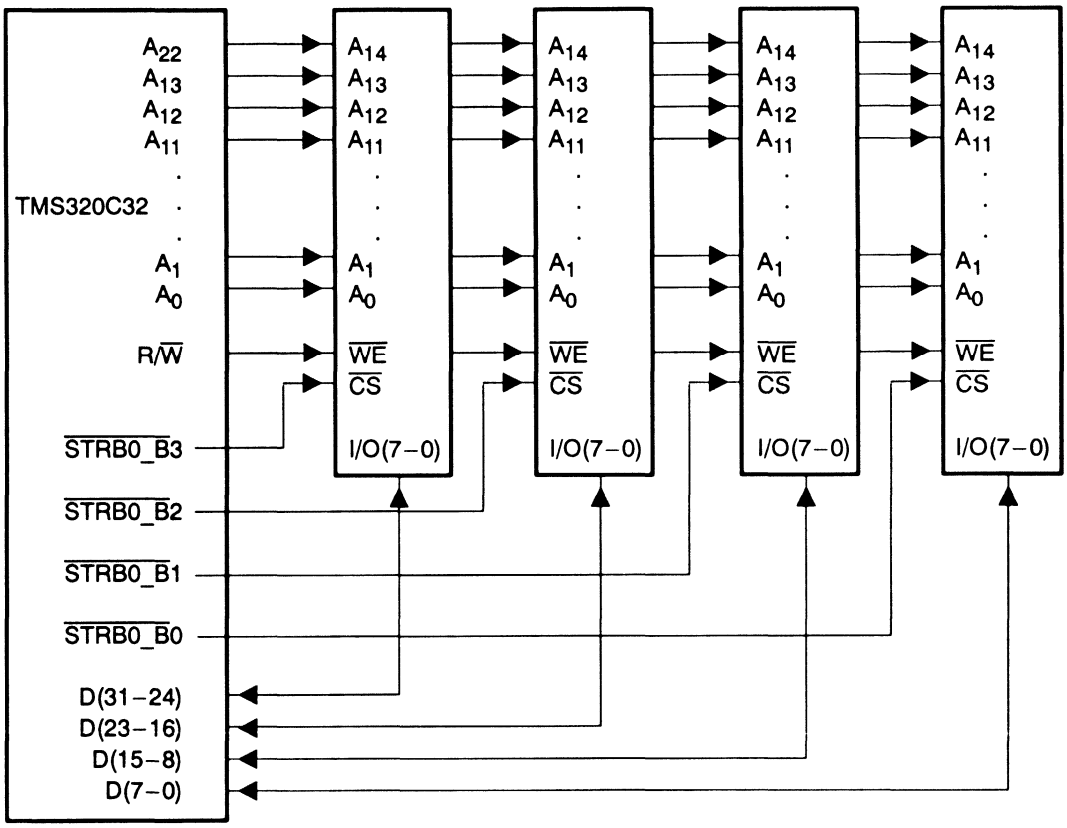
The 'C32 operates at its maximum performance when executing code from 32-bit wide memory. The rest of the memory can be used to store two different data type sizes.

For example, a typical audio compression application written in C language requires 32-bit data for system stack and 16-bit data for the audio buffers. In this case, you must interface the 'C32 as shown in Figure 12–1. This example assumes an external memory of 32K of 32-bit words with 8K of 32-bit words of stack, 8K of 32-bit words of program, and 32K of 16-bit words data buffers.

This interface requires you to set the STRB0 control register Physical Memory Width to 32 bits, Data Type Size to 32 bits, and set the STRB Config bit field to 1 (STBR0 Control Register = 002F0000h). It also requires you to set the STRB1 control register Physical Memory Width to 32 bits and the Data Type Size to 16 bits (STBR1 control register = 000D0000h). Moreover, the PRGW pin must be pulled low to indicate 32-bit program memory width.

In essence, this example combines *Case 3: 32-bit Wide Memory With 32-Bit Data Type Size* and *Case 2: 32-Bit Wide Memory with 16-Bit Data Type Size* discussed in subsection 7.3.2.

Figure 12–1. Zero Wait-State Interface for 32-Bit SRAMs With 16- and 32-Bit Data Accesses



Note that the external memory address pins, $A_{14}A_{13}...A_1A_0$, are mapped to the 'C32's $A_{22}A_{13}A_{12}...A_1A_0$. This mapping was chosen to place the system stack following the 'C32 internal RAM, thus improving performance by placing the top of the stack in internal RAM and allowing it to grow into external RAM. With this mapping, external memory accesses in the range 4000h through 7FFFh read or write 16-bit data while memory accesses in the range 0h through 3FFFh read or write 32-bit data. The PRGW pin controls the program fetches.

Figure 12–2 shows the contents of the external memory. Due to the address shift of the 'C32 external memory interface, the memory map seen by the 'C32 CPU is slightly different. Figure 12–3 shows this memory map. Note that since the STRB1 is configured for 16-bit data type size, the external address presented on 'C32' s pins is shifted right by one bit.

Figure 12–2. External Memory Map

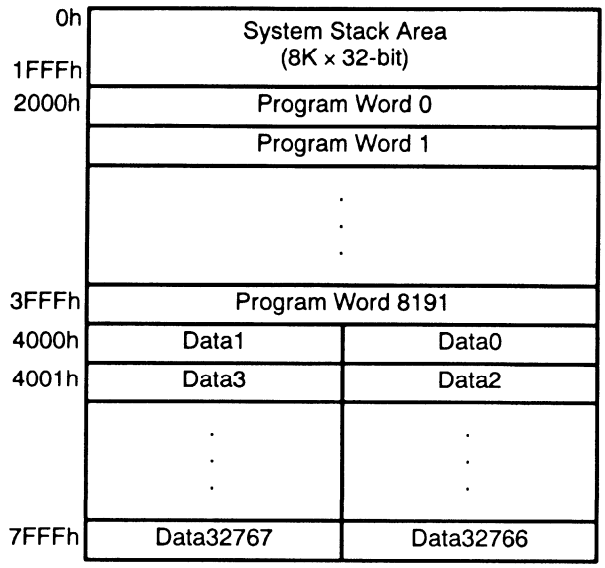
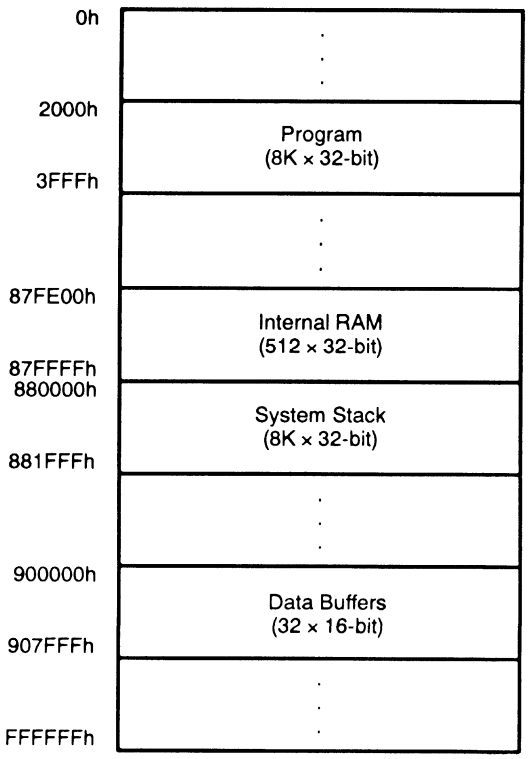


Figure 12–3. 'C32 Memory Map



12.2 Minimum Memory

To minimize system cost, the 'C32 can trade the number of external memory chips with lower performance by utilizing a zero wait-state 16-bit wide external memory. In this configuration, external program accesses and 32-bit data type accesses have an additional wait-state, while memory chip count is halved. Figure 12–4 shows this configuration.

Figure 12–4. Zero Wait-State Interface for 16-Bit SRAMs With 16- and 32-Bit Data Accesses

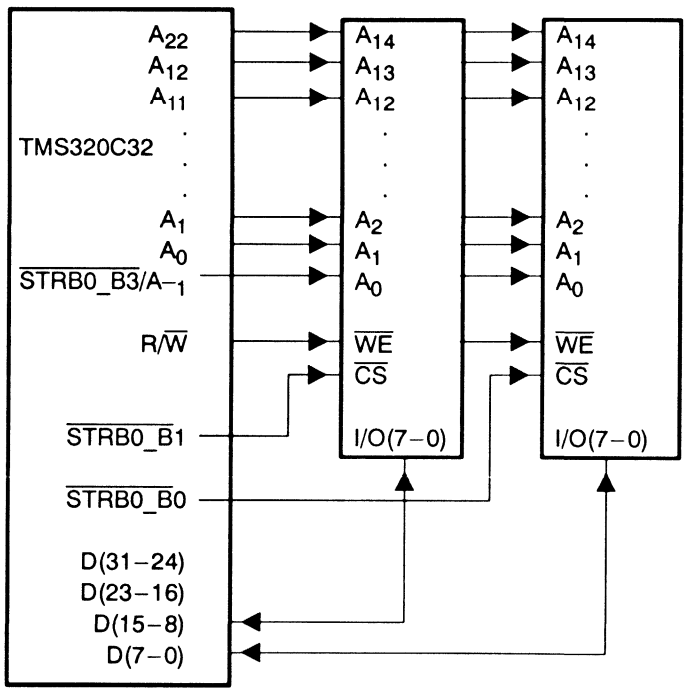


Figure 12–4 shows a 32K of 16-bit words external memory that contains 4.5K of 32-bit words of stack, 4K of 32-bit words of program, and 16K of 16-bit words data buffers and tables.

For this example, you must set the STRB0 control register Physical Memory Width to 16 bits, Data Type Size to 32 bits, and set the STRB Config bit field to 1 (STBR0 control register = 00270000h). It also requires you to set the STRB1 control register Physical Memory Width to 16 bits and the Data Type Size to 16 bits (STBR1 Control Register = 00050000h). Furthermore, the PRGW pin must be pulled high to indicate 16-bit program memory width.

In essence, this example combines *Case 5: 16-bit Wide Memory with 16-bit Data Type Size* and *Case 6: 16-bit Wide Memory with 32-bit Data Type Size* discussed in subsection 7.3.3.

As described in Section 12.1, this example maps the system stack contiguous with the 'C32 internal RAM. To achieve this, the external memory address pins, $A_{14}A_{13}...A_1A_0$, are mapped to the 'C32's $A_{22}A_{12}...A_1A_0A_{-1}$. Figure 12–5 shows the contents of the external memory. Due to the address shift of the 'C32 external memory interface, the memory map seen by the 'C32 CPU is shown in Figure 12–6.

Note that since STRB1 is configured for 16-bit data, the external address presented on the 'C32' s pins is shifted right by one bit. Since STRB0 is configured for 32-bit data size, the $\overline{\text{STRB0\_B3}}/A_{-1}$ pin is used to decode the low and high bytes of the word. With this mapping, external memory accesses in the range 4000h through 7FFFh read or write 16-bit data, while memory accesses in the range 0h through 3FFFh perform two consecutive reads or writes to retrieve or store 32-bit data. The PRGW pin controls the program fetches.

Figure 12–5. External Memory Map

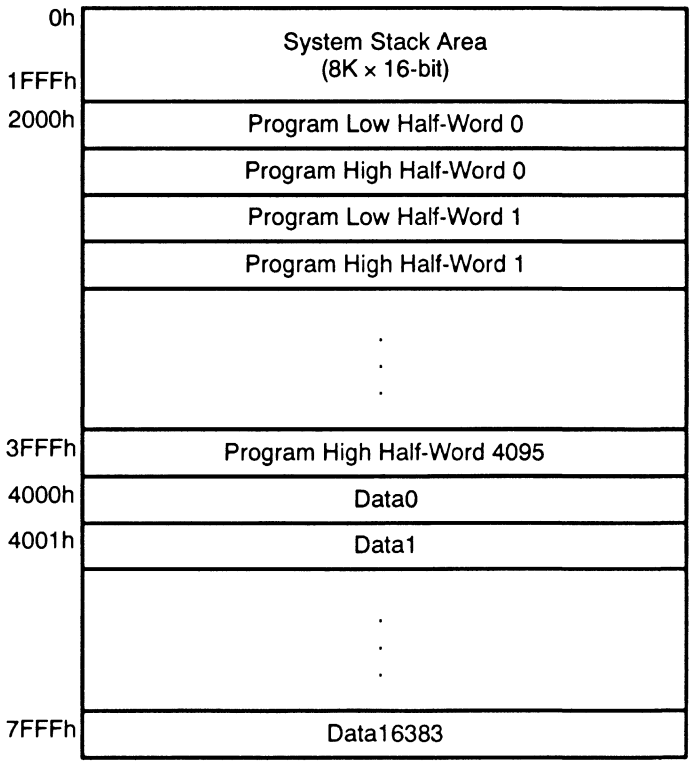
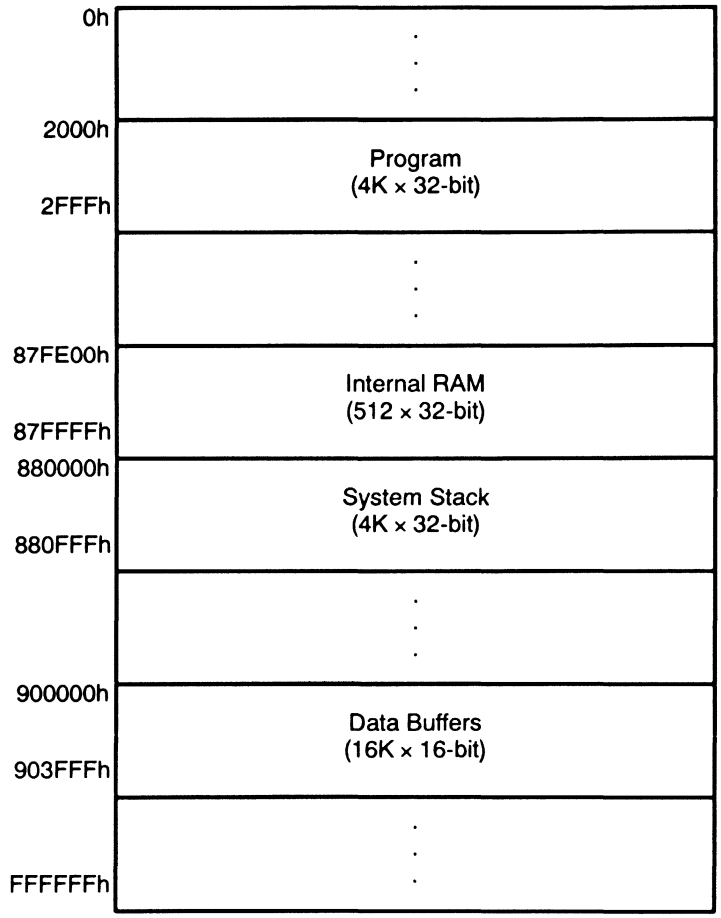


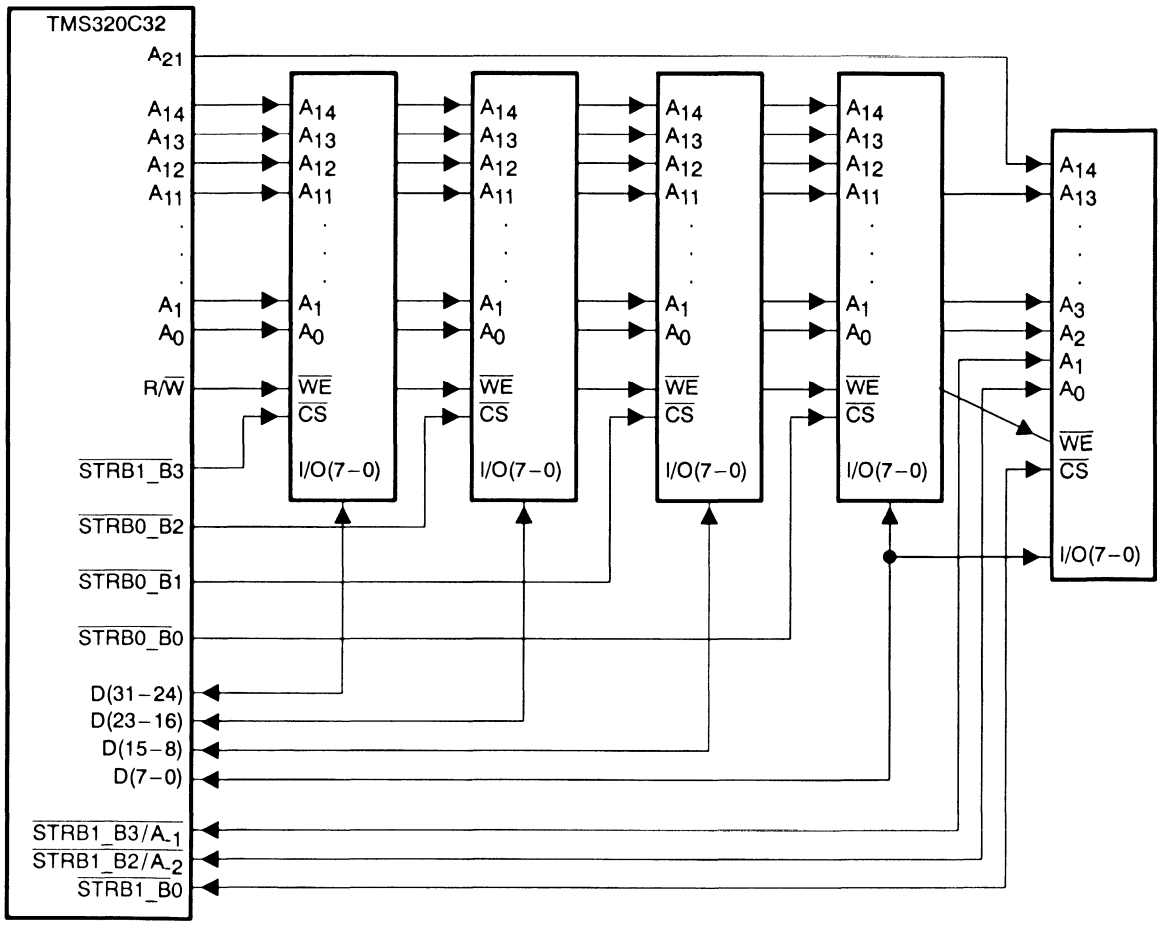
Figure 12–6. 'C32 Memory Map



12.3 Two External Memory Banks

'C32 external memory interface allows the use of two zero wait-state external memory banks with different widths without incurring in any access penalty and additional logic. This gives you the flexibility to trade off performance for system cost (fewer memory chips). For instance, you could execute code from 32-bit wide memory while storing data in 8-bit memory, as shown in Figure 12–7. This would be advantageous to applications with large amounts of 8-bit data that require execution at the fastest speed of the device.

Figure 12–7. Zero Wait-State Interface for 32-Bit and 8-Bit SRAM Banks



In Figure 12–7, a bank of 32K × 32-bit words is mapped to STRB0 while a bank of 32K × 8-bit words is mapped to STRB1.

For this configuration, you must set the STRB0 control register Physical Memory Width to 32 bits, Data Type Size to 32 bits, and the STRB Config bit

field to 0 since the banks are separate memories (STBR0 control register = 000F0000h). Also, you must set the STRB1 control register Physical Memory Width to 8 bits and the Data Type Size to 8 bits (STBR1 control register = 00000000h).

This example maps the external memory address pins of the 32-bit wide bank, $A_{14}A_{13}...A_1A_0$, to the 'C32's $A_{14}A_{13}A_{12}...A_1A_0$. On the other hand, the 8-bit wide bank memory address pins, $A_{14}A_{13}...A_1A_0$, are mapped to the 'C32's $A_{21}A_{13}A_{12}...A_1A_0A_{-1}$. Note that since STRB1 is configured for 8-bit memory width, the external address presented on 'C32s pins is shifted right by two bits. With this mapping, external memory accesses in the range 0h through 7FFFh read/write 32-bit data to the 32-bit wide bank (STRB0) while memory accesses in the range 900000h through 907FFFh read/write 8-bit data to the 8-bit wide bank (STRB1).

Note that two banks of different memory widths should not be connected to the same STRB without external decode logic. Different memory widths require $\overline{\text{STRBX\_Bx}}$ signals to be configured as address pins. These address pins are active for any external memory access ($\overline{\text{STRB0}}$, $\overline{\text{STRB1}}$, $\overline{\text{IOSTRB}}$, and program fetches).

Chapter 13

TMS320C32 Signal Descriptions

This chapter contains descriptions of the signals that are specific to the 'C32. Table 13–7 describes the signals that the 'C32 uses in the microprocessor mode. It lists the signal (or bit) name, the number of pins allocated; the input (I), output (O), or high-impedance (Z) operating modes; a brief description of the signal's function; and the condition that places an output pin in high impedance. The shading indicates new external signals.

13.2 Signal Descriptions

Table 13–7. TMS320C32 Signal Descriptions

Signal	No. of-Pins†	I/O/Z†	Description	Condition In High Z‡		
			External Bus Interface (70 pins)			
D31–0	32	I/O/Z	32-bit data port of the external bus interface	S	H	R
A23–0	24	O/Z	24-bit address port of the external bus interface	S	H	R
R/$\overline{W}$	1	O/Z	Read/write signal for the external bus interface. The pin is high when a read is performed and low when a write is performed over the parallel interface.	S	H	R
$\overline{IOSTRB}$	1	O/Z	External peripheral I/O strobe for the external bus interface	S	H	
$\overline{STRB0\_B3}$/A_{-1}	1	O/Z	External memory access strobe 0, byte enable 3 for 32-bit memory interface and address pin for 8-bit and 16-bit external bus interface	S	H	
$\overline{STRB0\_B2}$/A_{-2}	1	O/Z	External memory access strobe 0, byte enable 2 for 32-bit memory interface and address pin for 8-bit external bus interface	S	H	
$\overline{STRB0\_B1}$	1	O/Z	External memory access strobe 0, byte enable 1 for the external bus interface	S	H	
$\overline{STRB0\_B0}$	1	O/Z	External memory access strobe 0, byte enable 0 for the external bus interface	S	H	
$\overline{STRB1\_B3}$/A_{-1}	1	O/Z	External memory access strobe 1, byte enable 3 for 32-bit external bus interface and address pin for 8-bit and 16-bit external bus interface	S	H	
$\overline{STRB1\_B2}$/A_{-2}	1	O/Z	External memory access strobe 1, byte enable 2 for 32-bit external bus interface and address pin for 8-bit external bus interface	S	H	
$\overline{STRB1\_B1}$	1	O/Z	External memory access strobe 1, byte enable 1 for the external bus interface	S	H	
$\overline{STRB1\_B0}$	1	O/Z	External memory access strobe 1, byte enable 0 for the external bus interface	S	H	
$\overline{RDY}$	1	I	Ready signal. This pin indicates that an external device is prepared for a external bus interface transaction to complete			
$\overline{HOLD}$	1	I	Hold signal. When is a logic low, any ongoing transaction is completed. The A23–0, D31–D0, $\overline{IOSTRB}$, $\overline{STRB0\_Bx}$, $\overline{STRB1\_Bx}$, and R/$\overline{W}$ are placed in the high impedance state, and all transactions over the external bus interface are held until becomes a logic high, or the NOHOLD bit of the STRB0 bus control register is set.			

† Input (I), Output (O), High impedance state (Z)
‡ $\overline{SHZ}$ active (S), Hold active (H), Reset active (R)
Note: Shaded entries indicate new 'C32 external signals

Table 13–7. TMS320C32 Signal Descriptions (Continued)

Signal	No. of-Pins†	I/O/Z†	Description	Condition In High Z‡	
			External Bus Interface (Continued) (70 pins)		
$\overline{\text{HOLDA}}$	1	O/Z	Hold acknowledge signal. This signal is generated in response to a logic low on $\overline{\text{HOLD}}$. It signals that A23–0, D31–0, $\overline{\text{IOSTRB}}$, $\overline{\text{STRB0\_Bx}}$, $\overline{\text{STRB1\_Bx}}$, and R/$\overline{\text{W}}$ are placed in the high-impedance state and that all transactions over the bus are held. $\overline{\text{HOLDA}}$ is high in response to a logic high of $\overline{\text{HOLD}}$, or the NOHOLD bit of the bus control register is set.	S	
PRGW	1	I	Program memory width select. When this pin is logic low, program is fetched as a single 32-bit word. When it is logic high, two 16-bit program fetches are performed for a single 32-bit instruction word. The status of this pin at reset affects the reset value of the STRB0 and STRB1 bus control register (See Section 7.3)		
			Control Signals (9 Pins)		
$\overline{\text{RESET}}$	1	I	Reset. When this pin is a logic low, the device is placed in the reset condition. When reset becomes a logic 1, execution begins from the location specified by the reset vector.		
$\overline{\text{INT3}}$–$\overline{\text{INT0}}$	4	I	External interrupts.		
$\overline{\text{IACK}}$	1	O/Z	Interrupt acknowledge signal. $\overline{\text{IACK}}$ is set to 1 by the IACK instruction. This signal can be used to indicate the beginning or end of an interrupt service routine.	S	
MCBL/$\overline{\text{MP}}$	1	I	Microcomputer boot loader/microprocessor mode pin.		
XF1–XF0	2	I/O/Z	External flag pins. They are used as general purpose I/O pins or to support interlocked processor instructions.	S	R
			Serial Port Signals (6 Pins)		
CLKX0	1	I/O/Z	Serial port 0 transmit clock. This pin serves as the serial shift clock for the serial port 0	S	R
DX0	1	I/O/Z	Data transmit output. Serial port 0 transmits serial data on this pin.	S	R
FSX0	1	I/O/Z	Frame synchronization pulse for transmit. The FSX0 pulse initiates the transmit data processor over DX0.	S	R
CLKR0	1	I/O/Z	Serial port 0 receive clock. This pin serves as the serial shift clock for the serial port 0	S	R
DR0	1	I/O/Z	Data receive. Serial port 0 receives serial data via the DR0 pin.	S	R
FSR0	1	I/O/Z	Frame synchronization pulse for receive. The FSR0 pulse initiates the receive data processor over DR0.	S	R

Table 13–7. TMS320C32 Signal Descriptions (Concluded)

Signal	No. of-Pins†	I/O/Z†	Description	Condition In High Z‡	
			Timer Signals (2 Pin)		
TCLK0	1	I/O/Z	Timer clock 0. As an input, TLCK0 is used by timer 0 to count external pulses. As an output pin, TCLK0 outputs pulses generated by timer 0.	S	R
TCLK1	1	I/O/Z	Timer clock 1. As an input, TLCK1 is used by timer 1 to count external pulses. As an output pin, TCLK1 outputs pulses generated by timer 1.	S	R
			Clock Signals (3 Pins)		
H1	1	O/Z	External H1 clock. This clock has a period equal to twice CLKIN.	S	
H3	1	O/Z	External H3 clock. This clock has a period equal to twice CLKIN	S	
CLKIN	1	I	The input clock pin from an external clock source.		
			Emulation and Test Signals (5 Pins)		
EMU3	1	O/Z	Reserved for emulation.		
EMU2–EMU0	3	I	Reserved for emulation. Tie to +5-V with 20-kΩ pull-up resistors.		
$\overline{SHZ}$	1	I	Shut down high Z. A low logic level shuts down the 'C32 and places all pins in the high-impedance state. This signal is used for board-level testing to ensure that no dual-drive conditions occur. **Caution:** A low logic level on the $\overline{SHZ}$ pin corrupts 'C32 memory and register contents. Reset the device with an SHZ=1 to restore it to a known operation condition.		
			Supply Signals (45 Pins)		
CV_{SS}	7	I	Ground		
DV_{DD}	12	I	+5-Vdc supply		
DV_{SS}	7	I	Ground		
IV_{SS}	4	I	Ground		
V_{DDL}	8	I	+5-Vdc supply		
V_{SSL}	6	I	Ground		
V_{SUBS}	1	I	Substrate. Tie to ground		

† Input (I), Output (O), High-impedance state (Z)
‡ $\overline{SHZ}$ active (S), Hold active (H), Reset active (R)
Note: Shaded entries indicate new 'C32 external signals

Appendix A

Boot Loader Source Code

This appendix includes a description of the boot loader sequence of events and a listing of its source code.

A.1 Boot Loader Source Code Description

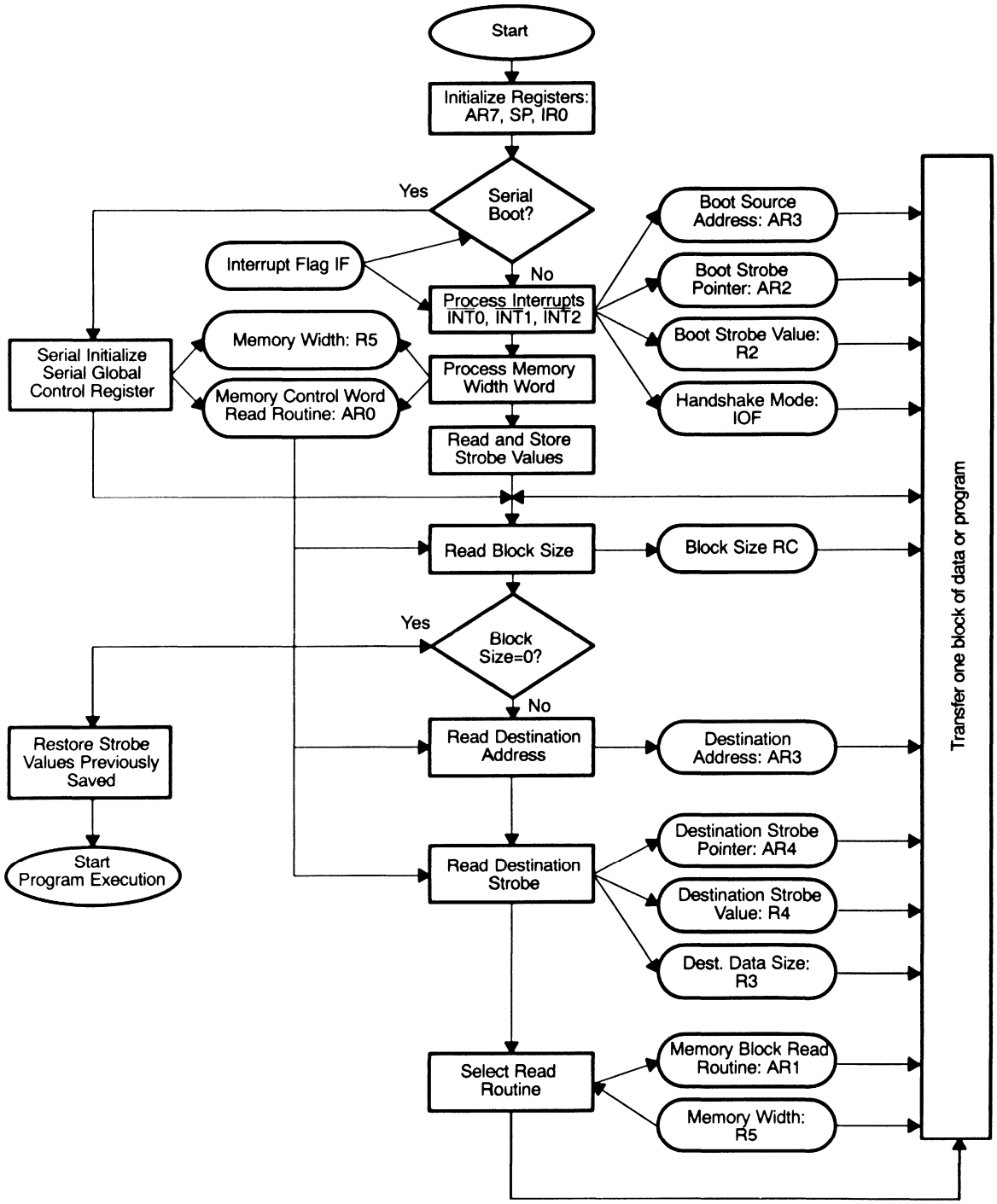
Figure A–1 shows the boot loader program flow chart. The boot loader program starts by initializing three registers: *AR7*, *SP*, and *IR0*. These register hold the Peripheral Bus memory map register base address, the Timer Counter register (used as a stack), and a flag that indicates the first block, respectively. Then, the program checks for serial port boot load or memory boot load mode by processing the bit fields set in the Interrupt Flag Register (*IF*). For a serial port boot load, the program initializes the serial port for 32 bit fixed burst mode reads with an externally generated serial port clock and FSR.

For a memory boot load, *AR3* is set to the boot source address, *AR2* points to the boot source Strobe Control register, and *R2* contains the value that will be stored in this Strobe Control register. The boot loader also sets the bit field $\bar{1}$/OXF0 of the I/O Flag Register (IOF) if the handshake mode was selected. Then the boot loader reads the first word of the boot source program. This 32 bit word indicates the boot memory width and the boot load program stores this value in *R5*. *AR0* points to the *read_mc* routine that performs this read.

After reading the memory width word, the boot loader reads IOSTRB, STRB0, and STRB1 control register values of the source program. These values are temporarily saved in the DMA Source Address register, DMA Destination Address register, and DMA Transfer Counter register, respectively. Then, the program reads the block size with the *read_mc* routine. If the block size is zero, the boot loader restores the values of IOSTRB, STRB0, and STRB1 previously saved and branches to the destination address of the first block loaded and begins program execution. If the block size is not zero, the boot loader stores the block size in the *BK* register. This is used as counter in a repeat block (*RPTB*) to transfer all the data or program in that block.

For each block, the boot loader reads the destination address and the destination strobe control word. The program stores the destination address in the *AR5* register. The destination strobe control word includes the destination strobe identification, the contents of the destination strobe control register (includes memory width and data size). The boot loader extracts this information from the destination control word and stores the destination strobe control register memory mapped address in the *AR4* register, the contents of the destination strobe control register in the *R4* register, and the source data size in the *R3* register. The boot loader sets the *AR1* register to the appropriate read routine *read_s0* for serial port boot load and *read_mb* for memory boot load. The read routine utilizes these registers to control the transfer of a block of data or program.

Figure A–1. Boot Loader Flow Chart



A.2 Boot Loader Source Code Listing

```
*******************************************************************************
* C32BOOT - TMS320C32 BOOT LOADER PROGRAM (142 words) 7-7-94
*   (C) COPYRIGHT TEXAS INSTRUMENTS INC., 1994   v.26
*=============================================================================*
*
* NOTE:
*
*   1. Following device reset, the program waits for an external interrupt.
*      The interrupt type determines the initial address from which the boot
*      loader will start loading the boot table to the destination memory:
```

INTERRUPT PIN	BOOT TABLE START ADDRESS	BOOT SOURCE
INT0	1000h (STRB0)	EPROM
INT1	810000h (IOSTRB)	EPROM
INT2	900000h (STRB1)	EPROM
INT3	80804Ch (sport0 Rx)	SERIAL
INT0 and INT3	1000h (STRB0) ASYNC	EPROM, XFO/XF1
INT1 and INT3	810000h (IOSTRB) ASYNC	EPROM, XFO/XF1
INT2 and INT3	900000h (STRB1) ASYNC	EPROM, XFO/XF1

```
*      If INT3 is asserted together with (INT2 or INT1 or INT0) following reset,
*      that indicates that the boot table is to be read synchronously from EPROM
*      using pins XF0 and XF1 for handshaking. The handshaking protocol assumes
*      that the data ready signal generated by the host arrives through pin XF1.
*      The data acknowledge signal is output from the C32 on pin XF0. Both signals
*      are active low. The C32 will continuously toggle the IACK signal while
*      waiting for the host to assert data ready signal (pin XF1).
*
*   2. The boot operation involves transfer of one or more source blocks from the
*      boot media to the destination memory. The block structure of the boot table
*      serves the purpose of distributing the source data/program among different
*      memory spaces. Each block is preceded by several 32 bit control words
*      describing the block contents to the boot loader program.
*
*   3. When loading from serial port the boot loader reads the source data/program
```

```
*      and writes it to the destination memory. There is only one way to read the
*      serial port. When loading from EPROM, however, there are 4 different ways
*      to read and assemble the source contents, depending on the width of boot
*      memory and the size of the program/data being transferred. Because there is
*      a possibility that reads and writes can span the same STRB space, the boot
*      loader loads the appropriate STRB control registers before each read and
*      write.
*
*  4.  If the boot source is EPROM whose physical width is less then 32 bits, the
*      physical interface of the EPROM device(s) to the processor should be the
*      same as that of the 32 bit interface. (This involves a specific connection
*      to C32's strobe and address signals). The reason for such arrangement is
*      that in order to function properly the boot loader program always expects
*      32 bit data from 32 bit wide memory during the boot load operation. Valid
*      boot EPROM widths are : 8, 16, and 32 bits.
*
*  5.  A single source block cannot cross STRB boundaries. For example, it's
*      destination cannot overlap STRB0 space and IOSTRB space. Additionally, all
*      of the destination addresses of a single source block should reside in
*      physical memory of the same width. It is also not permitted to mix prg and
*      data in the same source block.
*
*  6.  The boot loader stops boot operation when it finds 0 in the block size
*      control word. Therefore each boot table should always end with a 0
*      prompting the boot loader to branch to the first address of the first block
*      and start program execution from that location.
*
*=============================================================================*
* C32 boot loader program register assignments, and altered mem locations
*=============================================================================*
*  AR7 - peripheral memory map              IOF - XF0 (handshake O)
*  AR0 - read cntrl data subr pointer       IOF - XF1 (handshake I)
*  AR1 - read block data/prg subr pointer
*
```

```
*   R2 - read STRB value                R4 - write STRB value
*   AR2 - read STRB pointer             AR4 - write STRB pointer
*   AR3 - read data/prg pointer         AR5 - write data/prg pointer
*
*                   read --> R1 --> write
*
*   IR0 - EXEC start flag               stack -   808024h - TIM0 cnt reg
*   IR1 - EXEC start address                      808028h - TIM0 per reg
*                                       IOSTRB-   808004h - DMA0 dst reg
*   R3 - data SIZE                      STRB0 -   808006h - DMA0 dst reg
*   R5 - mem WIDTH                      STRB1 -   808008h - DMA0 cnt reg
*
*   R6 - memory read value              AR6,R7,R0,BK - scratch registers
*=============================================================================*
reset     .word     start          ; reset vector
          .space 44h               ; program starts @45h
*=============================================================================*
* Init registers : 808000h --> AR7,  808023h --> SP,  -1 --> IR0
*=============================================================================*
start  LDI    4040h,AR7            ; load peripheral memory map
       LSH    9,AR7                ; base address = 808000h
       LDI    23h,SP               ; initialize stack pointer to
       OR     AR7,SP               ; 808023h (timer counter - 1)
       LDI    -1,IR0               ; reset exec start addr flag
*=============================================================================*
*   Test for INT3 and, if set exclusively, proceed with serial boot load. Else,
*   load AR3 with 1000h if INT0, 810000h if INT1, 900000h if INT2. Also load
*   appropriate boot strobe pointer  --> AR2 and force the boot strobe value to
*   reflect 32bit memory width. If (INT0 or INT1 or INT2) and INT3 then turn on
*   the handshake mode.
*=============================================================================*
wait1     LDI    IF,R0
          AND    0Fh,R0            ; clean
          CMPI   8,R0              ; test for INT3
          BEQ    serial    ;***** ; serial boot load mode
          LDI    AR7,AR2
```

```
          ADDI   60h,AR2          ; 808060h (IOSTRB)   —>  AR2
          TSTB   2,R0             ; test for  INT1
          LDINZ  4080h,AR3        ; 810000h / 2**9
          BNZ    exit3     ;***** ;
          ADDI   4,AR2            ; 808064h  (STRB0)   —>  AR2
          TSTB   1,R0             ; test for  INT0
          LDINZ  8,AR3            ; 001000h / 2**9
          BNZ    exit3     ;***** ;
          ADDI   4,AR2            ; 808068h  (STRB1)   —>  AR2
          TSTB   4,R0             ; test for  INT2
          LDINZ  4800h,AR3        ; 900000h / 2**9
          BZ     wait1     ;***** ;
exit3     TSTB   8,R0          ;*; test#1 - INT3 asserted
          BZ     exit2         ;*; test#2 - INXF1 low (not used)
          TSTB   80h,IOF       ;*; enable handshake mode if
          LDI    6,IOF         ;*; test#1 passed
exit2     LDI    0Fh,R2
          LSH    16,R2            ; force boot data size to 32
          OR     *AR2,R2          ; force boot mem width to 32
          STI    R2,*AR2
          LSH    9,AR3            ; boot mem start addr —> AR3
*                                                    xx000001 -  1 bit
*=====================================================  xx000010 -  2 bit
* Process MEMORY WIDTH control word (32 bits long)     xx000100 -  4 bit
*=====================================================  xx001000 -  8 bit
*                                                    xx010000 - 16 bit
*                                                    xx100000 - 32 bit
          LDI    read_mc,AR0      ; use memory to read cntrl words
                                  ;         read_mc —>  AR0

          LDI    1,R5             ; mem width = 1  (init)
          LDI    32,AR6           ; mem reads = 32 (init)
          CALLU  read_m           ; read memory once  (1st read)
loop2     TSTB   1,R6
          BNZ    label4
          LSH    -1,R6            ; look at next bit
```

```
          LSH    -1,AR6            ; decr mem reads
          LSH    1,R5              ; incr mem width ---->  R5
          BU     loop2      ;***** ;
label4    SUBI   2,AR6
          CMPI   0,AR6             ; set flags
          BN     strobes    ;***** ; total # of mem reads = 32/R5
label5    CALLU  read_m            ; read memory once
          DBU    AR6,label5;***** ;
*=============================================================================*
* Read and save IOSTRB, STRB0 & STRB1 (to be loaded at end of boot load)
*=============================================================================*
strobes   CALLU  AR0
          STI    R1,*+AR7(4)       ; IOSTRB  ---> (DMA src)
          CALLU  AR0
          STI    R1,*+AR7(6)       ; STRB0   ---> (DMA dst)
          CALLU  AR0
          STI    R1,*+AR7(8)       ; STRB1   ---> (DMA cnt)

*=============================================================================*
* Process block size (# of bytes, half-words or words after STRB cntrl)
*=============================================================================*
block     CALLU  AR0               ; read boot memory cntrl word
          LDI    R1,R1             ; is this the last block ?
          BNZ    label2     ;***** ; no, go around
          LDI    *+AR7(4),R0       ;          (DMA src)
          STI    R0,*+AR7(60h)     ; restore IOSTRB
          LDI    *+AR7(6),R0       ;          (DMA dst)
          STI    R0,*+AR7(64h)     ; restore STRB0
          LDI    *+AR7(8),R0       ;          (DMA cnt)
          STI    R0,*+AR7(68h)     ; restore STRB1
          BU     IR1        ;***** ; branch to start of program
label2    LDI    R1,RC             ; setup transfer loop
          SUBI   1,RC              ; RC - 1 ---> RC
```

```
*=============================================================================*
* Process block destination address, save start address of first block
*=============================================================================*
          CALLU  AR0              ; read boot memory cntrl word
          LDI    R1,AR5           ; set dest addr  —> AR5
          CMPI   0,IR0            ; look at EXEC start addr flag
          LDINZ  AR5,IR1          ; if -1, EXEC start addr —> IR1
          LDINZ  0,IR0            ; set EXEC start addr flag
*=============================================================================*
* (For internal destination this word should be 0 or 60h. The first case
* will result in 0 —> DMA cntrl reg, in second case 0 —> IOSTRB reg.)
* Process block destination strobe control (sss...sss 0110 xx00)
*======================================== strb value ===  00 - IOSTRB
*                                                        01 -  STRB0
          CALLU  AR0              ;                      10 -  STRB1
          LDI    R1,R4
          AND    6Ch,R1           ; dest mem strb pntr    —> AR4
          OR3    AR7,R1,AR4
          LSH    -8,R4            ; dest memory strobe    —> R4
          LDI    R4,R3
          LSH    -16,R3
          AND    3,R3             ; dest data size     —> R3
          TSTB   0Ch,R1           ; (IOSTRB case)
          LDIZ   3,R3
*=============================================================================*
* Look at R5 and choose serial or memory read for block data/program
*=============================================================================*
          CMPI   0,R5
          LDIEQ  read_s0,AR1      ; read serial port0
          LDINE  read_mb,AR1      ; read memory
*=============================================================================*
* Transfer one block of data or program
*=============================================================*
          RPTB   loop4
          CALLU  AR1              ; read data/prg
          STI    R4,*AR4          ; set write strobe
```

```
          NOP                       ; pipeline
loop4     STI    R1,*AR5++          ; write data/prg
       || STI    R2,*AR2            ; set read strobe
          BU     block     ;*****   ; process next block
*=============================================================================*
* Load R5 with 0, load read_s0 to AR0 and initialize serial port_0
*=============================================================================*
serial    LDI    read_s0,AR0        ; use serial to read cntrl words
          LDI    0,R5               ; memory WIDTH = serial
          LDI    0,R2               ; dummy
          LDI    AR7,AR2            ; dummy
          LDI    111h,R0            ; 0000111h --> R0
          STI    R0,*+AR7(43h)      ; set CLKR,DR,FSR as serial
          LDI    0A30h,R7           ;                 port pins
          LSH    16,R7              ; A300000h --> R7
          STI    R7,*+AR7(40h)      ; set serial global cntrl reg
          BU     strobes    ;*****  ; process first block
*=============================================================================*
* Read a single value from serial or boot memory. The number of memory reads
* depends on mem WIDTH and data SIZE. R1 returns the read value.
* (Serial sim: NOP --> BZ read_s0 & LDI @4000H,R1 --> LDI *+AR7(4Ch),R1)
*=============================================================================*
read_s0   TSTB   20h,IF              ; look at RINT0 flag
          BZ     read_s0             ; wait for receive buffer full
          AND    0FDFh,IF            ; reset interrupt flag
          LDI    *+AR7(4Ch),R1       ; read data      -->  R1
          RETSU
*-----------------------------------------------------------------------------
read_mc   LDI    3,R3                ; data size = 32,    3 -->  R3
read_mb   LDI    1,BK                ; 00000001  (ex: mem width=8)
          LSH    R5,BK               ; 00000100
          SUBI   1,BK                ; 000000FF  =   mask   -->  BK
          LDI    R3,AR6              ;  0 -   1 000  EXPAND
          ADDI   1,AR6               ;  1 -  10 000   DATA  --> AR6
          LSH    3,AR6               ; 11 - 100 000   SIZE
          LDI    R5,R0
```

```
loop3     CMPI   1,R0
          BEQ    exit1              ; DATA SIZE
          LSH    -1,R0              ; ---------- - 1     --> AR6
          LSH    -1,AR6             ; MEM WIDTH
          BU     loop3     ;*****;
exit1     SUBI   1,AR6

          LDI    0,R0               ; init shift value
          LDI    0,R1               ; init accumulator
loop1     ADDI   3,SP               ; 808027h --> SP
          CALLU  read_m             ; read memory once  -->  R6
          SUBI   3,SP               ; 808024h --> SP
          AND3   R6,BK,R7           ; apply mask
          LSH    R0,R7              ; shift
          OR     R7,R1              ; accumulate     -->  R1
          ADDI   R5,R0              ; increment shift value
          DBU    AR6,loop1    ;***** ; decrement # of chunks --> AR6
          RETSU
*===========================================================================*
* Perform a single memory read from the source boot table. Handshake enabled if
* IOXF0 bit of IOF reg is set, disabled when reset. IACK will pulse continuously
* if handshake enabled and data not ready (to achieve zero-glue interface when
* connecting to a C40 comport)
*===========================================================================*
 read_m   TSTB   2,IOF              ; handshake mode enabled ?
          BNZ    loop5              ; yes, jump over
          LDI    *AR3++,R6          ; no,  just read memory & return
          RETSU
*--------------------------------------------------------------------------- (C40)
```

```
loop5   IACK   *AR7              ;*; internal dummy read pulses IACK
        TSTB   80h,IOF           ;*; wait for data ready
        BNZ    loop5             ;*; (XF1 low from host)
        LDI    *AR3++,R6         ;*; read memory once   —>  R6
        LDI    2,IOF             ;*; assert data acknowledge
                                 ;*; (XF0 low to host)
loop6   TSTB   80h,IOF           ;*; wait for data not ready
        BZ     loop6             ;*; (XF1 high from host)
        LDI    6,IOF             ;*; deassert data acknowledge
                                 ;*; (XF0 high to host)
        RETSU
*================================================================================*
```

Index

R

S

T

NOTES

TI Worldwide Sales and Representative Offices

AUSTRALIA / NEW ZEALAND: Texas Instruments Australia Ltd.: Melbourne [61] 3-9696-1211, Fax 3-9696-4446; **Sydney** 2-910-3100, Fax 2-805-1186.
BELGIUM: Texas Instruments Belgium S.A./N.V.: Brussels [32] (02) 726 75 80, Fax (02) 726 72 76.
BRAZIL: Texas Instrumentos Electronicos do Brasil Ltda.: Sao Paulo [55] 11-535-5133.
CANADA: Texas Instruments Canada Ltd.: Montreal (514) 421-2750; **Ottawa** (613) 726-3201; Fax 726-6363; **Toronto** (905) 884-9181; Fax 884-0062.
DENMARK: Texas Instruments A/S: Ballerup [45] (44) 68 74 00.
FRANCE/MIDDLE EAST/AFRICA: Texas Instruments France: Velizy-Villacoublay [33] (1) 30 70 10 01, Fax (1) 30 70 10 54.
GERMANY: Texas Instruments Deutschland GmbH.: Freising [49] (08161) 800, Fax (08161) 80 45 16; **Hannover** (0511) 90 49 60, Fax (0511) 64 90 331; **Ostfildern** (0711) 340 30, Fax (0711) 340 32 57.
HONG KONG: Texas Instruments Hong Kong Ltd.: Kowloon [852] 2956-7288, Fax 2956-2200.
HUNGARY: Texas Instruments Representation: Budapest [36] (1) 319 2748/2767/2768, Fax (1) 319 2814.
IRELAND: Texas Instruments Ireland Ltd.: Dublin [353] (01) 475 52 33, Fax (01) 478 14 63.
ITALY: Texas Instruments Italia S.p.A.: Agrate Brianza [39] (039) 684 21, Fax (039) 684 29 12; **Rome** (06) 657 26 51.
JAPAN: Texas Instruments Japan Ltd.: Kanazawa [81] 0762-23-5471, Fax 0762-23-1583; **Kita Kanto** 0485-22-2440, Fax 0485-23-5787; **Kyoto** 075-341-7713, Fax 075-341-7724; **Kyushu** 0977-73-1557, Fax 0977-73-1583; **Matsumoto** 0263-33-1060, Fax 0263-35-1025; **Nagoya** 052-232-5601, Fax 052-232-7888; **Osaka** 06-204-1881, Fax 06-204-1895; **Tachikawa** 0425-27-6760, Fax 0425-27-6426; **Tokyo** 03-3769-8700, Fax 03-3457-6777; **Yokohama** 045-338-1220, Fax 045-338-1255.
KOREA: Texas Instruments Korea Ltd.: Seoul [82] 2-551-2804, Fax 2-551-2828.
MAINLAND CHINA: Texas Instruments China Inc.: Beijing [86] 10-500-2255, Ext. 3750/3751/3752, Fax 10-500-2705.
MALAYSIA: Texas Instruments Malaysia Sdn Bhd: Kuala Lumpur [60] 3-208-6001, Fax 3-230-6605.
MEXICO: Texas Instruments de Mexico S.A. de C.V.: Colonia del Valle [52] 5-639-9740.
NORWAY: Texas Instruments Norge A/S: Oslo [47] (02) 264 75 70.
PHILIPPINES: Texas Instruments Asia Ltd.: Metro Manila [63] 2-636-0980, Fax 2-631-7702.
SINGAPORE (& INDIA, INDONESIA, THAILAND): Texas Instruments Singapore (PTE) Ltd.: Singapore [65] 390-7100, Fax 390-7062.
SPAIN/PORTUGAL: Texas Instruments España S.A.: Madrid [34] (1) 372 80 51, Fax (1) 307 68 64.
SUOMI/FINLAND: Texas Instruments/OY: Espoo [358] (0) 43 54 20 33, Fax (0) 46 73 23.
SWEDEN: Texas Instruments International Trade Corporation (Sverigefilialen): Kista [46] (08) 752 58 00, Fax (08) 751 97 15.
SWITZERLAND: Texas Instruments Switzerland AG: Dietikon [41] 886-2-3771450.
TAIWAN: Texas Instruments Taiwan Limited: Taipei [886] 2-378-6800, Fax 2-377-2718.
THE NETHERLANDS: Texas Instruments Holland, B.V. Amsterdam [31] (020) 546 98 00, Fax (020) 646 31 36.
UNITED KINGDOM: Texas Instruments Ltd.: Northampton [44] (01604) 66 30 00, Fax (01604) 66 30 01.

UNITED STATES: Texas Instruments Incorporated: ALABAMA: Huntsville (205) 430-0114; **ARIZONA: Phoenix** (602) 224-7800; **CALIFORNIA: Irvine** (714) 660-1200; **Los Angeles** (818) 704-8100; **San Diego** (619) 278-9600; **San Jose** (408) 894-9000; **COLORADO: Denver** (303) 488-9300; **CONNECTICUT: Wallingford** (203) 269-0074; **FLORIDA: Fort Lauderdale** (214) 644-5580 **Orlando** (407) 667-5308; **Tampa** (813) 573-0331; **GEORGIA: Atlanta** (770) 662-7967; **ILLINOIS: Chicago** (708) 517-4500; **INDIANA: Indianapolis** (317) 573-6400; **KANSAS: Kansas City** (913) 451-4511; **MARYLAND: Baltimore** (410) 312-7900; **MASSACHUSETTS: Boston** (617) 895-9100; **MICHIGAN: Detroit** (810) 305-5700; **MINNESOTA: Minneapolis** (612) 828-9300; **NEW JERSEY: Edison** (908) 906-0033; **NEW MEXICO: Albuquerque** (505) 345-2555; **NEW YORK: Long Island** (516) 454-6600; **Poughkeepsie** (914) 897-2900; **Rochester** (716) 385-6770; **NORTH CAROLINA: Charlotte** (704) 522-5487; **Raleigh** (919) 876-2725; **OHIO: Cleveland** (216) 328-2149; **Dayton** (513) 427-6200; **OREGON: Portland** (503) 643-6758; **PENNSYLVANIA: Philadelphia** (610) 825-9500; **PUERTO RICO: Hato Rey** (809) 753-8700; **TEXAS: Austin** (512) 250-6769; **Dallas** (214) 917-1264; **Houston** (713) 778-6592; **Midland** (915)561-6521; **WISCONSIN: Milwaukee** (414) 798-5021.

North American Authorized Distributors

COMMERCIAL	
Almac / Arrow	800-426-1410 / 800-452-9185 Oregon only
Anthem Electronics	800-826-8436
Arrow / Schweber	800-777-2776
Future Electronics (Canada)	800-388-8731
Hamilton Hallmark	800-332-8638
Marshall Industries	800-522-0084 or www.marshall.com
Wyle	800-414-4144
OBSOLETE PRODUCTS	
Rochester Electronics	508-462-9332
MILITARY	
Alliance Electronics Inc	800-608-9494
Future Electronics (Canada)	800-388-8731
Hamilton Hallmark	800-332-8638
Zeus, An Arrow Company	800-524-4735
TI DIE PROCESSORS	
Chip Supply	(407) 298-7100
Elmo Semiconductor	(818) 768-7400
Minco Technology Labs	(512) 834-2022
CATALOG	
Allied Electronics	800-433-5700
Arrow Advantage	800-777-2776
Newark Electonics	800-367-3573

For Distributors outside North America, contact your local Sales Office.

A040896

TEXAS INSTRUMENTS