TMS320C62x DSP Expansion Bus (XBUS) Reference Guide

Literature Number: SPRU579A
December 2003

Texas Instruments

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

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

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265

Preface

Read This First

About This Manual

This document describes the expansion bus (XBUS) used by the CPU to access off-chip peripherals, FIFOs, and peripheral component interconnect (PCI) interface devices in the TMS320C62x™ digital signal processors (DSPs) of the TMS320C6000™ DSP family.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.
- Registers in this document are shown in figures and described in tables.
  - Each register figure shows a rectangle divided into fields that represent the fields of the register. Each field is labeled with its bit name, its beginning and ending bit numbers above, and its read/write properties below. A legend explains the notation used for the properties.
  - Reserved bits in a register figure designate a bit that is used for future device expansion.

Related Documentation From Texas Instruments

The following documents describe the C6000™ devices and related support tools. Copies of these documents are available on the Internet at www.ti.com. *Tip:* Enter the literature number in the search box provided at www.ti.com.

TMS320C6000 CPU and Instruction Set Reference Guide (literature number SPRU189) describes the TMS320C6000™ CPU architecture, instruction set, pipeline, and interrupts for these digital signal processors.

TMS320C6000 DSP Peripherals Overview Reference Guide (literature number SPRU190) describes the peripherals available on the TMS320C6000™ DSPs.

TMS320C6000 Technical Brief (literature number SPRU197) gives an introduction to the TMS320C62x™ and TMS320C67x™ DSPs, development tools, and third-party support.

TMS320C6000 Programmer's Guide (literature number SPRU198) describes ways to optimize C and assembly code for the TMS320C6000™ DSPs and includes application program examples.

TMS320C6000 Code Composer Studio Tutorial (literature number SPRU301) introduces the Code Composer Studio™ integrated development environment and software tools.

Code Composer Studio Application Programming Interface Reference Guide (literature number SPRU321) describes the Code Composer Studio™ application programming interface (API), which allows you to program custom plug-ins for Code Composer.

TMS320C6x Peripheral Support Library Programmer's Reference (literature number SPRU273) describes the contents of the TMS320C6000™ peripheral support library of functions and macros. It lists functions and macros both by header file and alphabetically, provides a complete description of each, and gives code examples to show how they are used.

TMS320C6000 Chip Support Library API Reference Guide (literature number SPRU401) describes a set of application programming interfaces (APIs) used to configure and control the on-chip peripherals.

Trademarks

Code Composer Studio, C6000, C62x, C64x, C67x, TMS320C6000, TMS320C62x, TMS320C64x, TMS320C67x, and VelociTI are trademarks of Texas Instruments.

Contents

Figures

Tables

This page is intentionally left blank.

Expansion Bus (XBUS)

This document describes the expansion bus (XBUS) used by the CPU to access off-chip peripherals, FIFOs, and peripheral component interconnect (PCI) interface devices in the TMS320C62x™ digital signal processors (DSPs) of the TMS320C6000™ DSP family.

1 Overview

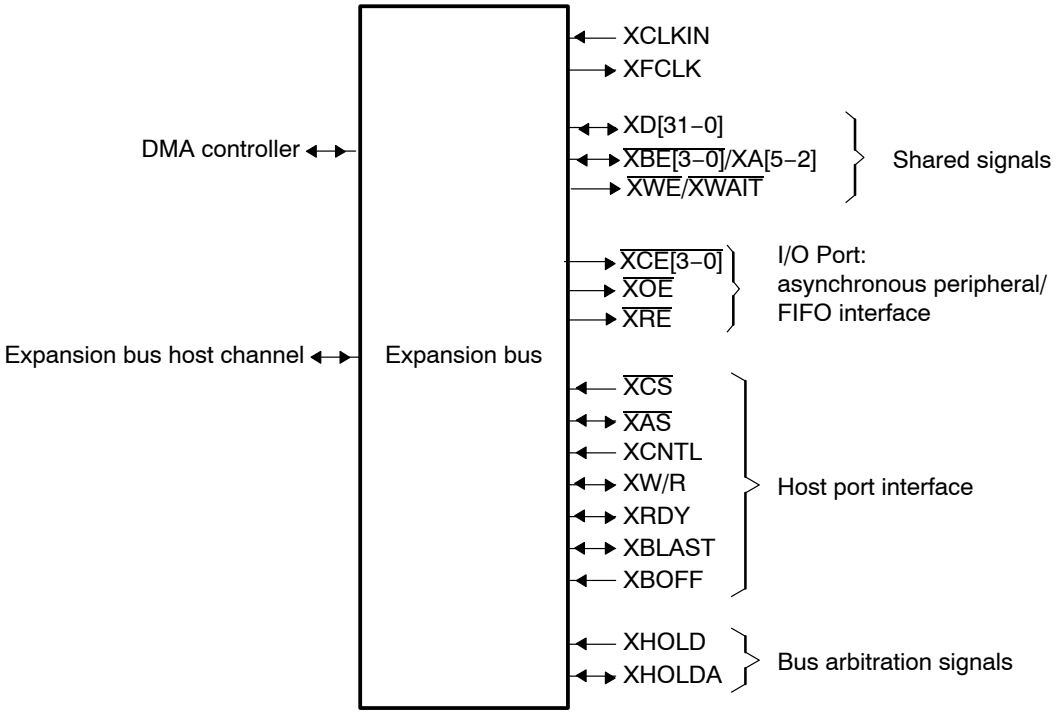
The expansion bus (XBUS) is a 32-bit wide bus that supports interfaces to a variety of asynchronous peripherals, asynchronous or synchronous FIFOs, PCI bridge chips, and other external masters.

The XBUS offers a flexible bus arbitration scheme, implemented with two signals, XHOLD and XHOLDA. The XBUS can operate with the internal arbiter enabled, in which case any external hosts must request the bus from the DSP. For increased flexibility, the internal arbiter can be disabled and the DSP requests the bus from an external arbiter.

The XBUS has two major subblocks, the I/O port and host port interface. A block diagram of the XBUS is shown in Figure 1. The I/O port has two modes of operation that can coexist in a single system: asynchronous I/O mode and synchronous FIFO mode. These modes are selectable for each of the four XCE spaces in the XBUS. The asynchronous I/O mode provides output strobes that are highly programmable, like the asynchronous signals of the external memory interface (EMIF). The XBUS interface provides four output address signals in this mode, and with external decode this provides for up to 16 devices per XCE space. The FIFO mode provides a glueless interface to a single synchronous read FIFO or up to four synchronous write FIFOs. With a minimal amount of glue, this can be extended to up to 16 read and 16 write FIFOs per XCE space. Connectivity of the XBUS I/O port and DSP memory is provided through the direct-memory access (DMA) controller.

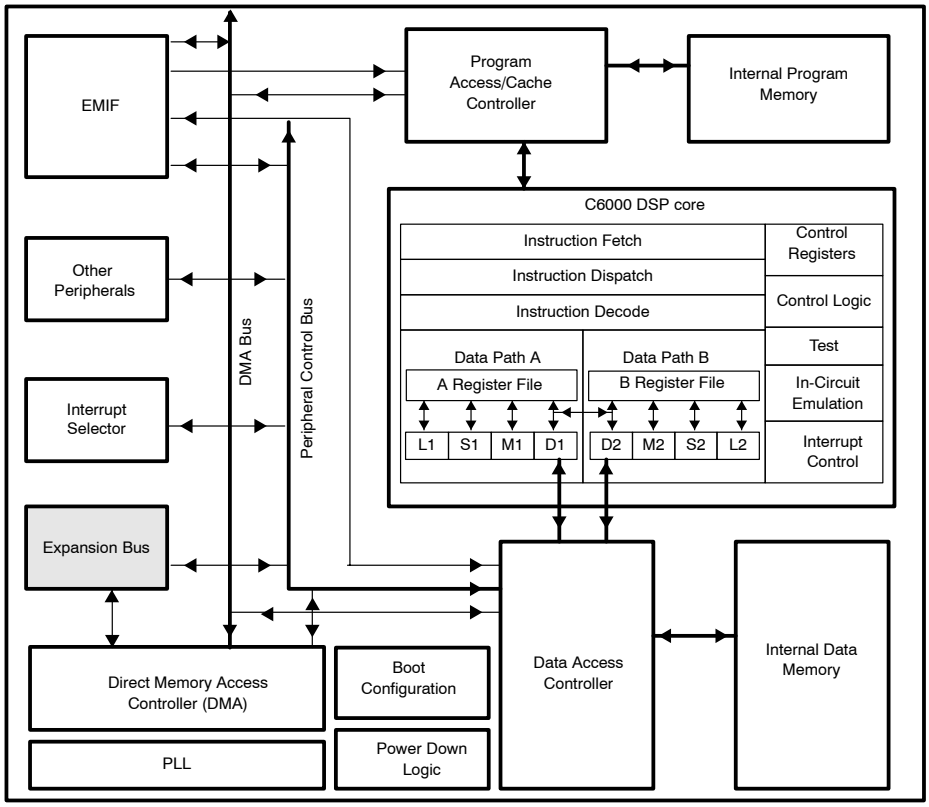
The host port interface can operate in one of two modes: synchronous and asynchronous. The synchronous mode offers master and slave functionality, and has multiplexed address and data signals. The asynchronous mode is slave only and is similar to the host-port interface (HPI) on the C6201/C6211/C6701/C6711 DSP, but is extended to a 32-bit data path. The asynchronous mode is used to interface to microprocessors that utilize an asynchronous bus.

Figure 1. Expansion Bus Block Diagram



Connectivity of the XBUS host port interface and the DSP memory space is provided by the DMA auxiliary port. Dedicated address and data registers connect the host port interface to the XBUS host channel. An external master accesses these registers using external data and interface control signals. Through a dedicated port, the DMA provides connectivity between the processor and the XBUS I/O port. To initiate transfers via the synchronous host port interface, the CPU has to configure a set of registers. Figure 2 shows the XBUS interface on the C62x™ DSP.

Figure 2. TMS320C62x DSP Block Diagram



Note: Refer to the specific device datasheet for its peripheral set.

2 Expansion Bus Signals

Table 1 lists the XBUS signals and their functionality in each mode. If only the I/O port of the XBUS is used (or if the XBUS is not used at all), the XBUS signals should be pulled inactive according to Table 2.

Table 1. Expansion Bus Signals

	I/O Port Mode (Non-Exclusive)				Mutually Exclusive Host Port Modes			
XBUS Signal	**I/O/Z**	**Async Signal**	**I/O/Z**	**Sync FIFO Signal**	**I/O/Z**	**Sync Mode**	**I/O/Z**	**Async Mode**
XD[31–0]	I/O/Z	D[31–0]	I/O/Z	D[31–0]	I/O/Z	D[31–0]	I/O/Z	D[31–0]
XFCLK			O	XFCLK				
XCLKIN					I	CLK		
$\overline{\text{XCE}}$[3–0]	O	$\overline{\text{CS}}$	O	$\overline{\text{RE}}$/$\overline{\text{WE}}$/$\overline{\text{CS}}$				
$\overline{\text{XBE0}}$/XA2	O/Z	XA2	O/Z	XA2	I/O/Z	BE0	I	BE0
$\overline{\text{XBE1}}$/XA3	O/Z	XA3	O/Z	XA3	I/O/Z	BE1	I	BE1
$\overline{\text{XBE2}}$/XA4	O/Z	XA4	O/Z	XA4	I/O/Z	BE2	I	BE2
$\overline{\text{XBE3}}$/XA5	O/Z	XA5	O/Z	XA5	I/O/Z	BE3	I	BE3
$\overline{\text{XOE}}$	O	$\overline{\text{OE}}$	O	$\overline{\text{OE}}$				
$\overline{\text{XRE}}$	O	$\overline{\text{RE}}$	O	$\overline{\text{RE}}$				
$\overline{\text{XWE/XWAIT}}$	O	$\overline{\text{WE}}$	O	$\overline{\text{WE}}$	O	$\overline{\text{WAIT}}$		
$\overline{\text{XAS}}$					I/O/Z	AS		
XRDY	I	XRDY			I/O/Z	READY	O/Z	READY
XW/R					I/O/Z	W/$\overline{\text{R}}$	I	W/$\overline{\text{R}}$
XBLAST					I/O/Z	BLAST		
XHOLD	I/O/Z	HOLD	I/O/Z	HOLD	I/O/Z	HOLD	I/O/Z	HOLD
XHOLDA	I/O/Z	HOLDA	I/O/Z	HOLDA	I/O/Z	HOLDA	I/O/Z	HOLDA
XCNTL					I	CNTL	I	CNTL
XBOFF					I	BOFF		
$\overline{\text{XCS}}$					I	CS	I	CS

Table 2. Expansion Bus Signal State for Disabled Host Port

XBUS Signal	I/O Port Mode (I/O/Z)	External Connection
XD[31−0]	I/O/Z	According to system (See section 6)
XFCLK	O	N/C
XCLKIN	I	Pull up
$\overline{\text{XCE}}$[3−0]	O	N/C
$\overline{\text{XBE}}$[3−0]/XA[5−2]	O/Z	Pull down
$\overline{\text{XOE}}$	O	N/C
$\overline{\text{XRE}}$	O	N/C
$\overline{\text{XWE}}$	O	N/C
$\overline{\text{XAS}}$	I/O/Z	Pull up
XRDY	I/O/Z	Pull up
XW/R	I/O/Z	Pull up
XBLAST	I/O/Z	Pull up, if BLPOL = 0; Pull down, if BLPOL = 1
XHOLD†	I/O/Z	Pull down
XHOLDA†	I/O/Z	Pull down
XCNTL	I	Pull up
XBOFF	I	Pull down
$\overline{\text{XCS}}$	I	Pull up

† Internal arbitration should be enabled, such that the DSP is the master of the bus when not using the host port. See section 5 for more details.

3 Expansion Bus I/O Port Operation

For external I/O port accesses on the XBUS, the $\overline{\text{XBE}}$ signals act as address signals XA[5−2]. You can use the address signals to address as many as 16 different read/write peripherals or 32 FIFOs in each XCE space. For the FIFO interface, 32 devices are possible since a separate read and write FIFO can be located at each address.

Access to the XBUS I/O port can only be done through DMA channels 0−3. The DMEMC does not have direct access to the XBUS. Therefore, load and store (LD/ST) commands to the memory spaces of the XBUS I/O port via the CPU are not allowed, and result in undefined operation. A DMA transfer cannot occur from one XCE space to another XCE space. Also, a host port transaction cannot access any of the XCE spaces.

For reads, care must be taken to ensure that contention on the data bus does not occur when switching from one peripheral to the next in the same XCE space. The DMA can accomplish this since inactive cycles occur when the DMA switches from one frame to the next. The DMA can be set up to read (or write) a frame from each of the peripherals or FIFOs in turn. For example, the element index can be cleared to 0 and the frame index can be set to a multiple of 4 (ensure word strides), thus incrementing to a different location after each frame has completed.

Although the XBUS does not explicitly support memory widths of less than 32 bits, the DMA can be used to read/write to 8-bit or 16-bit peripherals or FIFOs by controlling the byte/halfword logical addressing. For example, if an 8-bit-wide FIFO is in XCE2, then the DMA ESIZE bits can specify 8-bit transfers. The lower two address bits in the DMA source or destination address register determines the byte lane used for accessing the I/O port. If the bottom two bits are 00b (word aligned), then only XD[7−0] is used for valid data; if 01b, then XD[15−8] is used (see Figure 3 and Table 3).

Figure 3 shows how to interface four 8-bit FIFOs to the I/O port (memory map for this case is described in Table 4). Figure 4 is an example of an interface between two 16-bit FIFOs and the I/O port.

The $\overline{\text{XOE}}$, $\overline{\text{XRE}}$, $\overline{\text{XWE}}$, and $\overline{\text{XCEn}}$ signals are not tri-stated while the DSP releases control of the XBUS.

Figure 3. Expansion Bus Interface to Four 8-Bit FIFOs

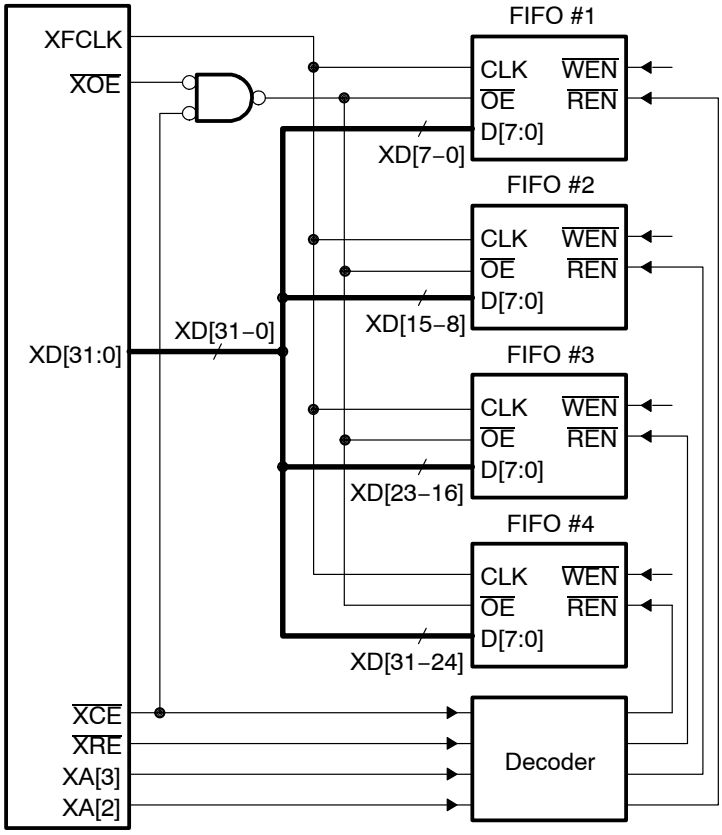


Table 3. Addressing Scheme—Expansion Bus Interfaced to Four 8-Bit FIFOs

Logical Address	A[31–6]	A5	A4	A3	A2	A1	A0
FIFO #1 Address	X	X	X	0	0	0	0
FIFO #2 Address	X	X	X	0	1	0	1
FIFO #3 Address	X	X	X	1	0	1	0
FIFO #4 Address	X	X	X	1	1	1	1
Physical Address		XA5	XA4	XA3	XA2		

Alternatively, if 16-bit (or 8-bit) peripherals are used, the DMA element index can be set up such that the stride value causes a read from alternating byte lanes during each read transfer. For example, the first access can be to address A[5−0] = xxxx00b, causing the lower half of the data bus to be driven by the peripheral. If the next address is A[5−0] = xxxx10b, the top half of the data bus is driven by the other peripheral (or FIFO) and no bus contention occurs. The only address signals that are externally provided are A[5−2]. If address decoding is required to address a specific peripheral or FIFO, these should be modified as necessary by the DMA to ensure that peripherals are only addressed when appropriate (see Figure 4 and Table 4).

Figure 4. Expansion Bus Interface to Two 16-Bit FIFOs

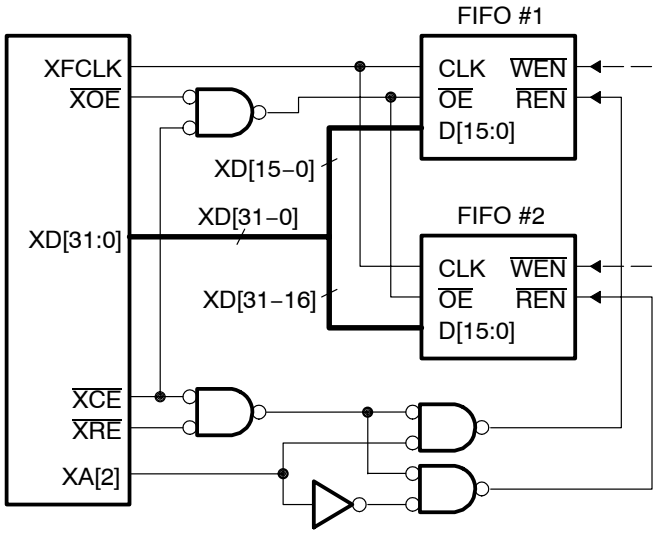


Table 4. Addressing Scheme—Expansion Bus Interfaced to Two 16-Bit FIFOs

Logical Address	A[31−6]	A5	A4	A3	A2	A1	A0
FIFO #1 Address	X	X	X	X	0	0	0
FIFO #2 Address	X	X	X	X	1	1	0
Physical Address		XA5	XA4	XA3	XA2		

3.1 Asynchronous Mode

The asynchronous cycles of the XBUS are identical to the asynchronous cycles provided by the EMIF. During asynchronous peripheral accesses, XRDY acts as an active-high ready input and XBE[3−0]/XA[5−2] operate as address signals XA[5−2]. The remaining asynchronous peripheral signals operate exactly like their EMIF counterparts. The following minimum values apply to the asynchronous parameters:

- SETUP + STROBE + HOLD ≥ 3
  - SETUP ≥ 1
  - STROBE ≥ 1
- If XRDY is used to extend STROBE, then HOLD ≥ 2.

Notes:

1) XRDY is active (low) during host-port accesses.

2) XBE[3−0]/XA[5−2] operate as XBE[3−0] during host-port accesses.

An access to a section of memory that does not return a ready indication is not allowed. This includes accesses to XBUS I/O asynchronous spaces with XRDY pulled inactive or left floating on the device. Possible requestors are programmed DMA channels, or HPI/PCI/XBUS host mastering via the auxiliary DMA. This type of access can create a stall indefinitely.

3.2 Synchronous FIFO Mode

The synchronous FIFO mode of the XBUS offers a glueless and/or low glue interface to standard synchronous FIFOs. The XBUS can interface with up to four write FIFOs without using glue logic (one per XCE space) or three write FIFOs and a single read FIFO (in $\overline{\text{XCE3}}$ only). However, with a minimal amount of glue logic, up to 16 read and write FIFOs can be used per XCE space.

Several FIFOs can be accessed in a single XCE space, if address decode logic is used to access each FIFO separately.

A description of the synchronous FIFO signals is listed in Table 5.

Table 5. Signal Description—Synchronous FIFO Mode

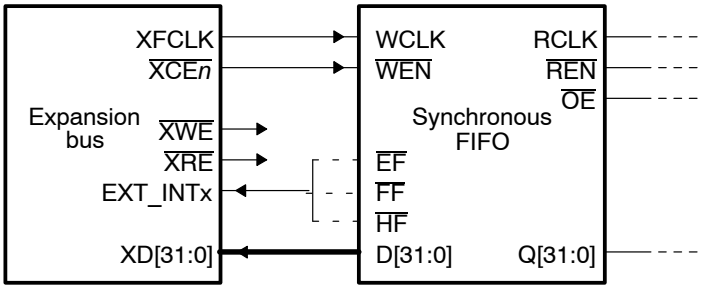
Signal Name	I/O/Z	Signal Purpose	Signal Function	
			Read/Write Mode	Read Mode
XFCLK	O	FIFO clock output	Programmable to either 1/2, 1/4, 1/6, or 1/8 of the CPU clock frequency. If CPU clock = 250 MHz, then XFCLK = 125, 62.5, 41.7, or 31.25 MHz. XFCLK continues to clock even when the DSP releases ownership of the XBUS.	
XD[31−0]	I/O/Z	Data	Data lines	
$\overline{\text{XCE}}$[3−0]	O	FIFO read enable/ write enable/ chip select	Active for both read and write transactions. They should be logically ORed with output control signals externally to create dedicated controls for a FIFO. Also can be used directly as FIFO write enable signal for a single write FIFO per XCE space.	Acts as read enable signal (XCE3 only).
$\overline{\text{XWE}}$	O	FIFO write enable	Write enable signal for FIFO. Must be logically ORed with corresponding XCE signal to ensure that only one FIFO is addressed at a time.	
$\overline{\text{XRE}}$	O	FIFO read enable	Read enable signal for FIFO. Must be logically ORed with corresponding XCE signal to ensure that only one FIFO is addressed at a time.	
$\overline{\text{XOE}}$	O	FIFO output enable	Shared output enable signal. Must be logically ORed with corresponding XCE signal to ensure that only one FIFO is addressed at a time.	Dedicated output enable signal in XCE3, if FIFO read mode is selected. If selected, this signal is disabled for all other modes.
$\overline{\text{XBE}}$[3−0]/ XA[5−2]	O/Z	Expansion bus address	Operate as XA[5−2]. Can be decoded to specify up to 16 different addresses, enabling interface with glue logic to 16 read FIFOs and 16 write FIFOs in a single XCE space.	

3.2.1 Write FIFO Interface

During write accesses to a memory space configured for read/write FIFO mode, the XCE signal and XWE signal are both active for a single rising edge of XFCLK. So, depending on the specific system environment, the write interface can be accomplished either with glue logic or without glue logic (glueless).

The glueless interface can be used if only a single write FIFO is used in a given XCE space (see Figure 5), since the XCE signal is used as the write enable signal. If this is true, the XCE signal is tied directly to the write enable ($\overline{\text{WEN}}$) input of the FIFO. If a read FIFO is also used in the same XCE space, glue logic must be used since the XCE signal also goes low for reads from the read FIFO.

Figure 5. Write FIFO Interface With Glueless Logic



3.2.2 Read/Write FIFO Interface

Figure 6 shows an interface to a read FIFO and a write FIFO in the same XCE space. For this example, the XCE signal is used to gate the appropriate read/write strobes to the FIFOs. The FIFO write timing diagram for this interface is shown in Figure 7.

Figure 6. Read and Write FIFO Interface With Glue Logic

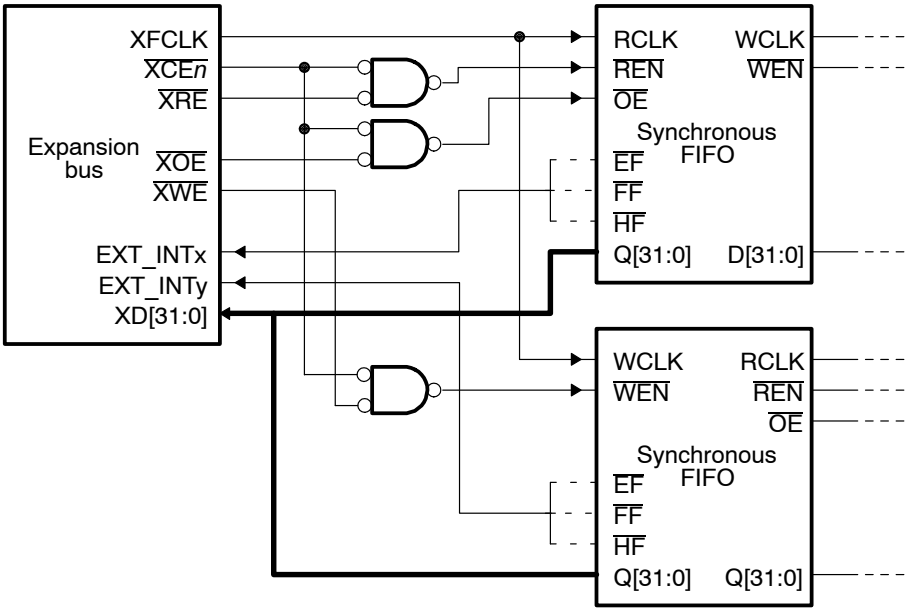
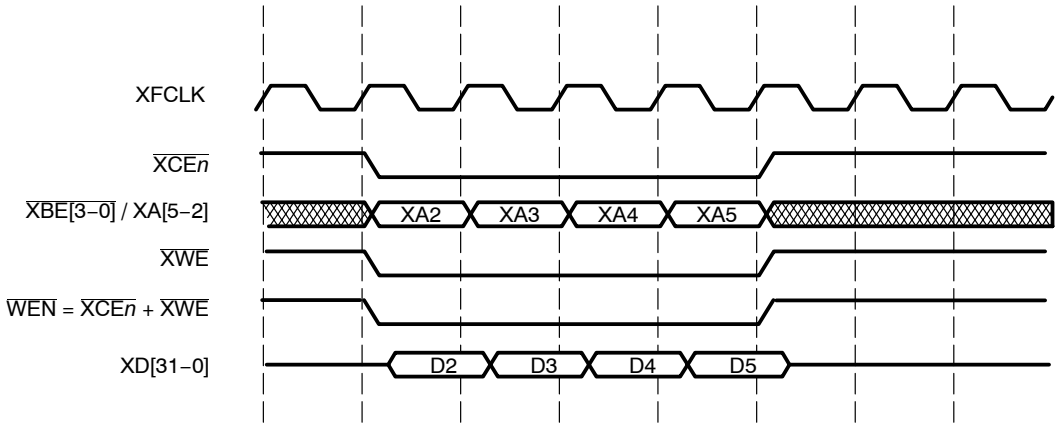


Figure 7. Read and Write FIFO Interface With Glue Logic—FIFO Write Cycles



3.2.3 Read FIFO Interface

The read FIFO interface can be accomplished without glue logic (glueless) in XCE3 space or with a minimal amount of glue logic in any XCE space. If a glueless read FIFO interface is used (specified by boot configuration selection), the $\overline{\text{XOE}}$ signal is only enabled in XCE3 space and is dedicated to use for the FIFO interface. If this mode is selected at boot, the XOE signal is disabled in all other XCE spaces. In this mode, $\overline{\text{XCE3}}$ is used as the read enable signal ($\overline{\text{REN}}$) and $\overline{\text{XOE}}$ is used as the output enable signal ($\overline{\text{OE}}$) of the FIFO. Figure 8 shows this interface and Figure 9 shows the read timing diagram. If the glueless read FIFO mode is not chosen, then a minimal amount of glue logic can be used in any XCE space specified as a FIFO interface. Figure 6 shows the required glue logic. Figure 10 shows the read timing diagram for the case when glue logic is used to read from FIFO.

Figure 8. Read FIFO Interface With Glueless Logic

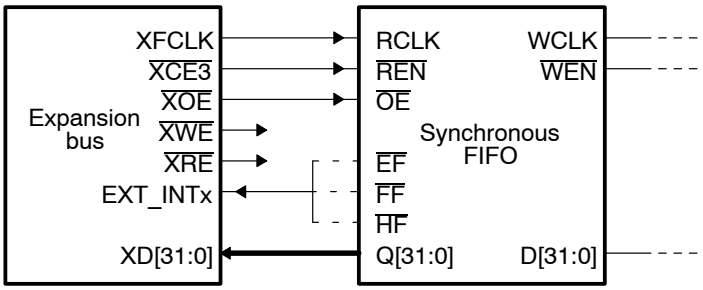


Figure 9. Read FIFO Interface With Glueless Logic—FIFO Read Cycles

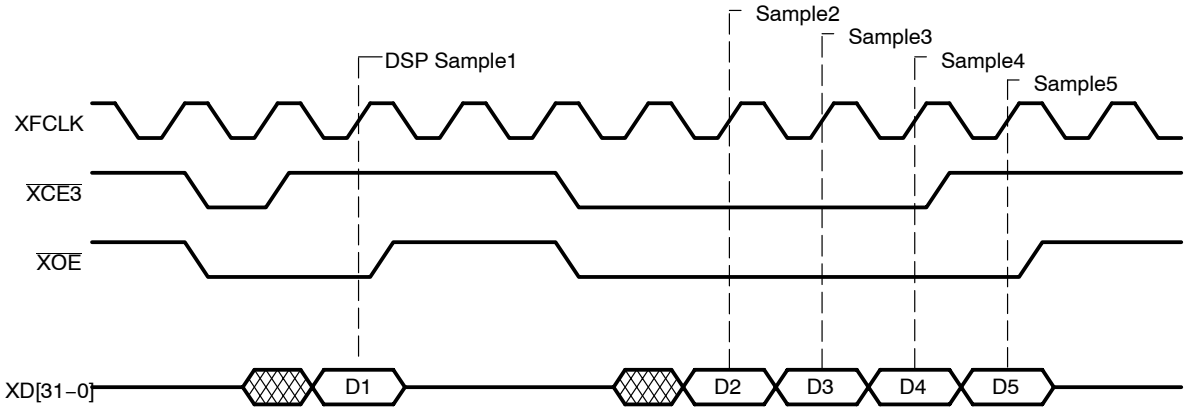
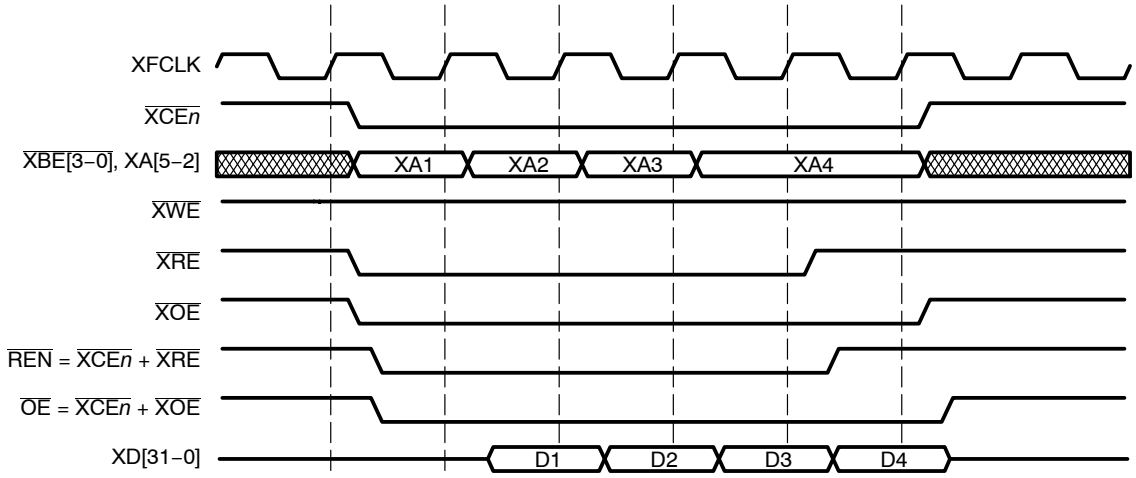


Figure 10. Read FIFO Interface With Glue Logic—FIFO Read Cycles



3.2.4 Programming Offset Register

The programmable offset registers of the FIFO are used to hold the offset values for the flags that indicate the condition of the FIFO contents. The programmable offset registers must be programmed in consecutive cycles and read in consecutive cycles. The writer should not write to the FIFO until the offset registers have been programmed. In addition, the reader should not read from the FIFO until the writer has programmed the offset registers. This should not be a problem, since the FIFO is not read until it has been written.

For programming (or reading) the offset registers, back-to-back accesses must be done. For example, the first XFCLK edge with the program input to the FIFO low, programs the PAE register; the second XFCLK edge, programs the PAF register. Also, for 9-bit or 18-bit FIFOs, it is common to require two or three write cycles to fully program each register. The first write programs the low bits, the second write programs the middle bits, and the third write programs the high bits.

A general-purpose output (DMACx or TOUTx) can be used to control whether FIFO reads/writes are done to the FIFO memory or to the programmable offset register of the memory. The XA[5–2] signals can also be decoded to control when the FIFO offset register is accessed.

3.2.5 Flag Monitoring

To efficiently control bursts to and from the dedicated FIFO interfaces, the interrupt signals EXT_INT4, EXT_INT5, EXT_INT6, and EXT_INT7 are used as flags to control DMA transfers. The flag polarity used to start the transfer can be programmed in the DMA channel secondary control register (SECCTL). The CPU EXT_INT and DMA EXT_INT polarity is controlled separately.

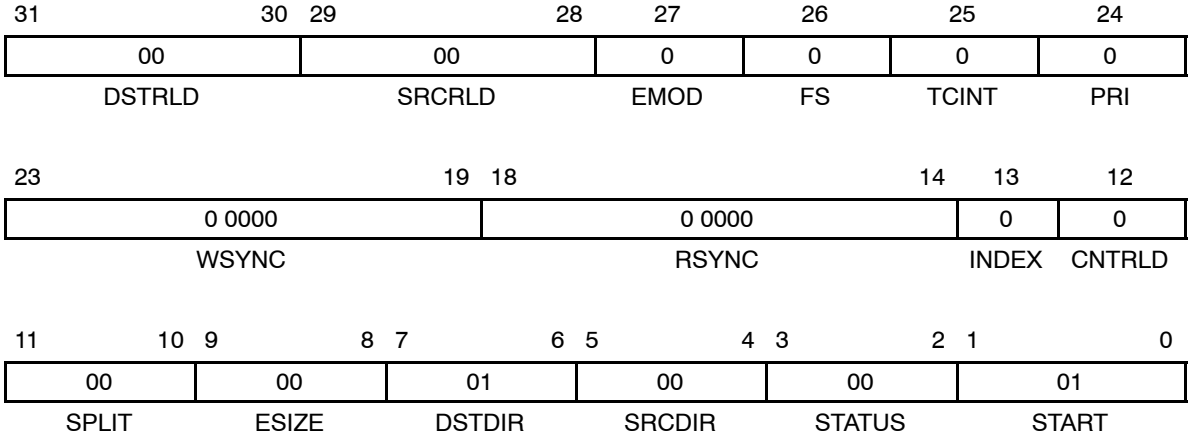
3.3 Single Frame Transfer Example

Peripherals located on the I/O port of the XBUS are accessible only via DMA transactions. This example shows you how to transfer a single frame of 256 words from a FIFO located in XCE0 into internal data memory at 8000 0000h. This example sets up the source and destination registers, transfer counter register, and starts the DMA with an incrementing destination address (DSTDIR = 01) and a nonchanging source address (SRCDIR = 00). The source address does not change since the FIFO is located in a fixed-memory location. The content of relevant DMA registers is listed in Table 6 and the content of the DMA channel primary control register (PRICTL) is shown in Figure 11.

Table 6. DMA Registers Content for Single Frame Transfer Example

Register	Contents
DMA channel primary control register (PRICTL)	0000 0041h (see Figure 11)
DMA channel source address register (SRC)	4000 0000h
DMA channel destination address register (DST)	8000 0000h
DMA channel transfer counter register (XFRCNT)	0000 0100h

Figure 11. DMA Channel Primary Control Register (PRICTL) Content for Single Frame Transfer Example



3.4 Multiple Frame Transfer With Frame Synchronization Example

This example shows you how to transfer 10 frames of 256 words from a FIFO located in XCE0 into internal data memory at 8000 0000h. This example sets up the source and destination registers, transfer counter register, global count reload register, and starts the DMA with an incrementing destination address (DSTDIR = 01) and a nonchanging source address (SRCDIR = 00). The source address does not change since the FIFO is located in a fixed-memory location. Frame synchronization is enabled (FS = 1) and an active (high) EXT_INT4 is used for the synchronization event (RSYNC = 0100). The content of relevant DMA registers is listed in Table 7 and the content of the DMA channel primary control register (PRICTL) and secondary control register (SECCTL) are shown in Figure 12 and Figure 13.

Table 7. DMA Registers Content for Multiple Frame Transfer Example

Register	Content
DMA channel primary control register (PRICTL)	0401 0041h (see Figure 12)
DMA channel secondary control register (SECCTL)	0008 0000h (see Figure 13)
DMA channel source address register (SRC)	4000 0000h
DMA channel destination address register (DST)	8000 0000h
DMA channel transfer counter register (XFRCNT)	000A 0100h
DMA global count reload register A (GBLCNTA)	0000 0100h

Figure 12. DMA Channel Primary Control Register (PRICTL) Content for Multiple Frame Transfer Example

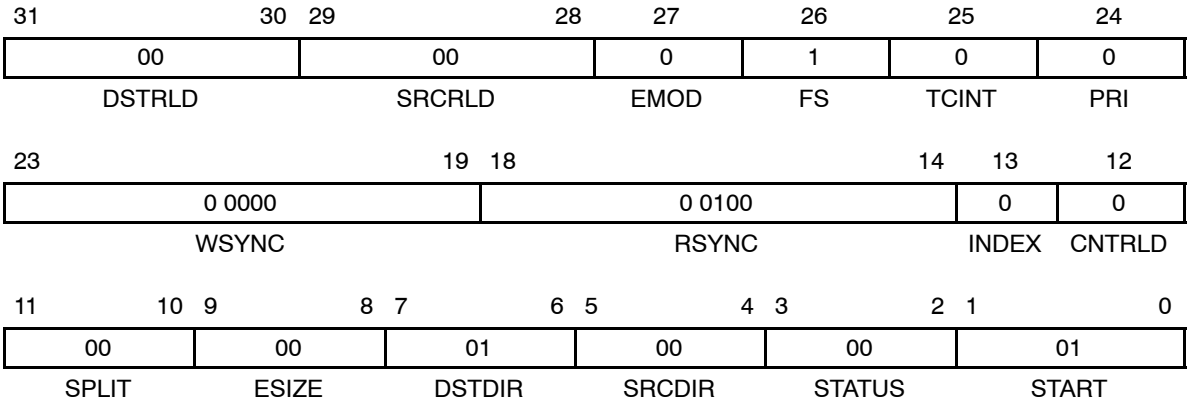
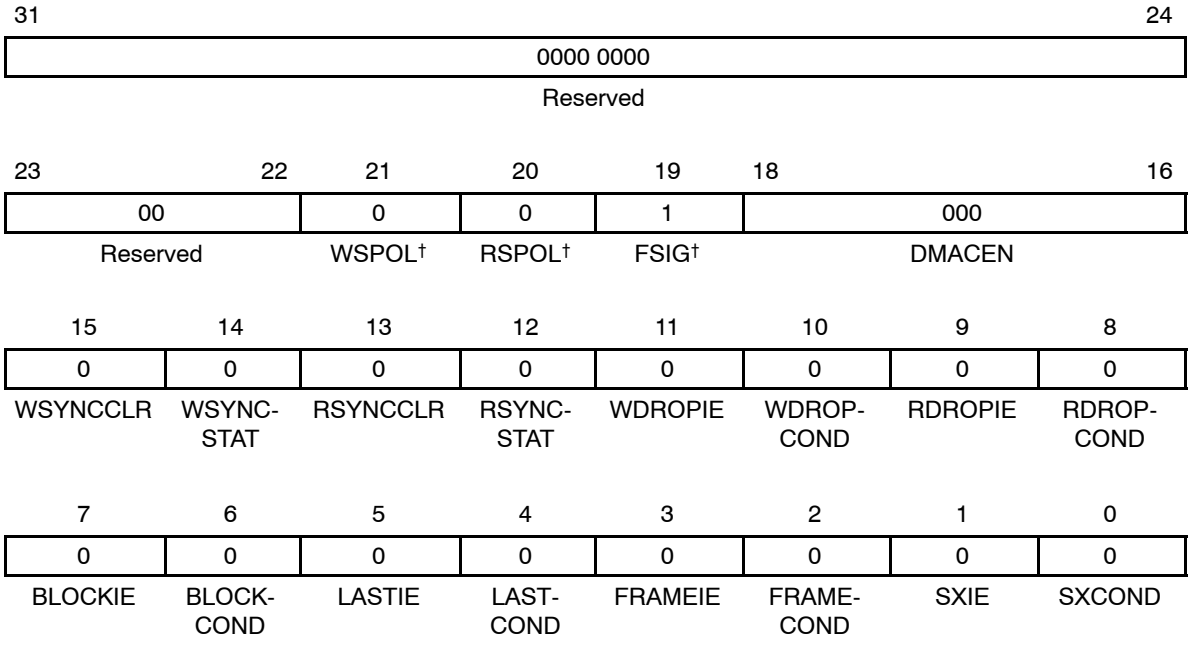


Figure 13. DMA Channel Secondary Control Register (SECCTL) Content for Multiple Frame Transfer Example



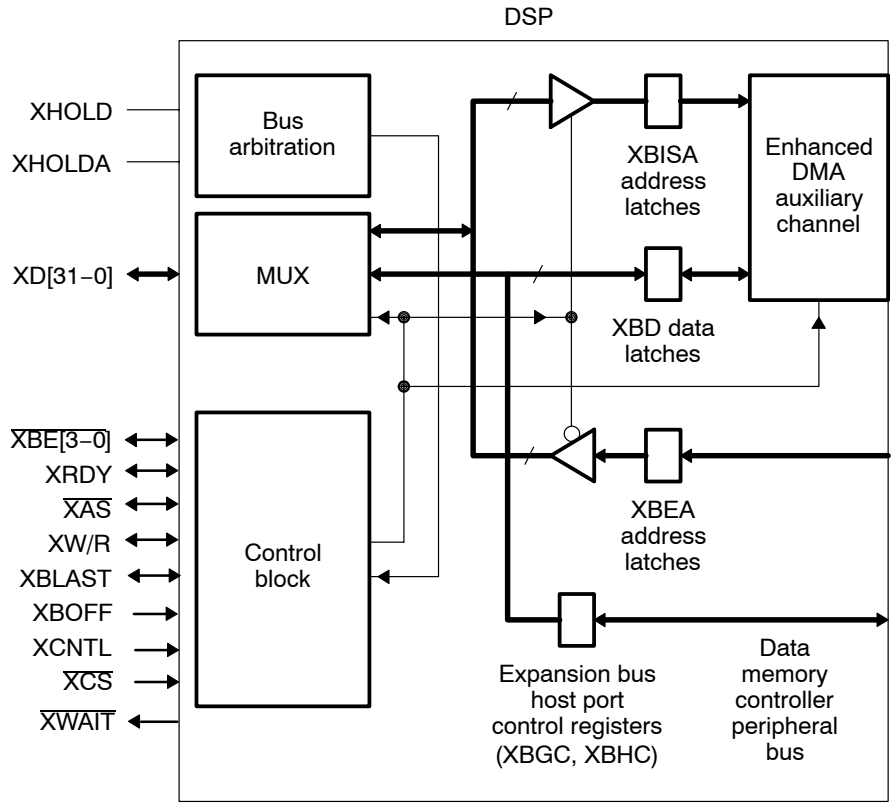
† Available only on C6202 and C6203 devices.

4 Expansion Bus Host Port Operation

The XBUS host port has two modes of operation that enable interfaces to external processors, PCI bridge devices, or other external peripherals. These two operation modes are the synchronous host port mode and the asynchronous host port mode. The synchronous host port mode can interface with minimum glue logic to PCI bridge devices and many common microprocessors. The asynchronous host port mode enables interfacing to genuine asynchronous devices. The XBUS host port block diagram is shown in Figure 14.

Using pull-up/pull-down resistors on the data bus during reset sets the host port operational mode, the DSP bootmode, and endianness.

Figure 14. Expansion Bus Host Port Interface Block Diagram



4.1 Synchronous Host Port Mode

In this mode, the host port has address and data signals multiplexed and is i960Jx compatible. This allows a minimum glue logic interface to the PCI bus, since major PCI interface device manufacturers adopted the i960 bus for the local bus on their devices.

The synchronous host port can also easily interface to many other common processors, and essentially act in a slave only mode. This is done by not initiating transactions on the XBUS. The XBUS has the capability to initiate and receive burst transfers.

A description of the synchronous host port signals is listed in Table 8.

Table 8. Signal Description—Synchronous Host Port Mode

Signal Name	I/O/Z	Signal Purpose	Signal Function
XCLKIN	I	Clock input	XBUS clock (maximum clock speed is 1/4 of the CPU clock speed).
$\overline{\text{XCS}}$	I	Chip select	Selects the DSP as a target of an external master.
XHOLD	I/O/Z	Hold request	Case 1 (Internal bus arbiter is enabled) XHOLD is asserted by external device to request use of the XBUS. The DSP asserts XHOLDA when control is granted. Case 2 (Internal bus arbiter is disabled) The DSP wakes up from reset as slave on the bus. XHOLD is asserted by the DSP to request use of the XBUS. The XBUS arbiter asserts XHOLDA when control is granted.
XHOLDA	I/O/Z	Hold acknowledge	Case 1 (Internal bus arbiter is disabled) The DSP wakes up from reset as slave on the bus. The XBUS arbiter asserts XHOLDA when control is granted in response to XHOLD. The bus should not be granted to the DSP unless requested by XHOLD. Case 2 (Internal bus arbiter is enabled) The DSP wakes up from reset as master of the bus. XHOLDA is asserted by the DSP when control is granted in response to XHOLD.
XD[31–0]	I/O/Z	Address/ data bus	Data
XBLAST	I/O/Z	Burst last	Signal driven by the current XBUS master to indicate the last transfer in a bus access. Input polarity selected at boot. Output polarity is always active low.

Table 8. Signal Description—Synchronous Host Port Mode (Continued)

Signal Name	I/O/Z	Signal Purpose	Signal Function
$\overline{\text{XAS}}$	I/O/Z	Address strobe	Indicates a valid address and the start of a new bus access. Asserted for the first clock of a bus access.
XCNTL	I	Control signal	This signal selects between XBD and XBISA. XCNTL = 0: access is made to the expansion bus data register (XBD). XCNTL = 1: access is made to the expansion bus internal slave address register (XBISA).
$\overline{\text{XBE}}$[3–0]/ XA[5–2]	I/O/Z	Byte enables	During host-port accesses these signals operate as $\overline{\text{XBE}}$[3–0]. $\overline{\text{BE3}}$ byte enable 3: XD[31–24] $\overline{\text{BE2}}$ byte enable 2: XD[23–16] $\overline{\text{BE1}}$ byte enable 1: XD[15–8] $\overline{\text{BE0}}$ byte enable 0: XD[7–0] For XBD access (XCNTL = 0): 8-bit data must be byte-aligned 16-bit data must be halfword-aligned 32-bit data must be word-aligned. For XBISA access (XCNTL = 1), all $\overline{\text{XBE}}$[3–0] must be active low.
XW/R	I/O/Z	Read/write	Read/write enable. Polarity of this signal is configured during boot.
XRDY	I/O/Z	Ready out Ready in	Active (low) during host-port access. XRDY is an input when the DSP owns the bus. When the DSP does not own the bus, XRDY is not driven until a request is made to the DSP.
XBOFF	I	Bus back-off	When asserted, suspends the current access and the DSP releases ownership of the XBUS.
$\overline{\text{XWAIT}}$	O	Wait	Ready output when the DSP initiates transfers on the XBUS.

4.1.1 TMS320C62x Master on the Expansion Bus

When the C62x DSP is the master of the XBUS, it can initiate a burst read or write to a peripheral on the bus.

When the DSP controls the bus, data flow is governed in a manner similar to a DMA transfer; however, the XBUS host channel regulates the actual data transfer. The event flow is:

1) The DSP must initialize the expansion bus external address register (XBEA), which dictates where data is accessed in the external slave memory map.

2) The expansion bus internal master address register (XBIMA) must be set to specify the source or destination address in the DSP memory map where the transaction starts.

3) The XFRCT bits of the expansion bus host port interface control register (XBHC) are set to control the number of 32-bit words being transferred.

   Note: Only 32-bit transfers are supported by the XBUS when the DSP is the master in synchronous host port mode.

4) The START bits of XBHC are written, controlling whether the external access is a read or write burst.

An interrupt is generated at the completion of the transfer, if specified by the INTSRC bit in XBHC.

Figure 15 and Figure 16 show examples of timing diagrams for a burst read and write when the DSP is mastering the bus. In this case, internal bus arbiter is disabled (XHOLD is output and XHOLDA is input) and the DSP wakes up from reset as slave on the XBUS.

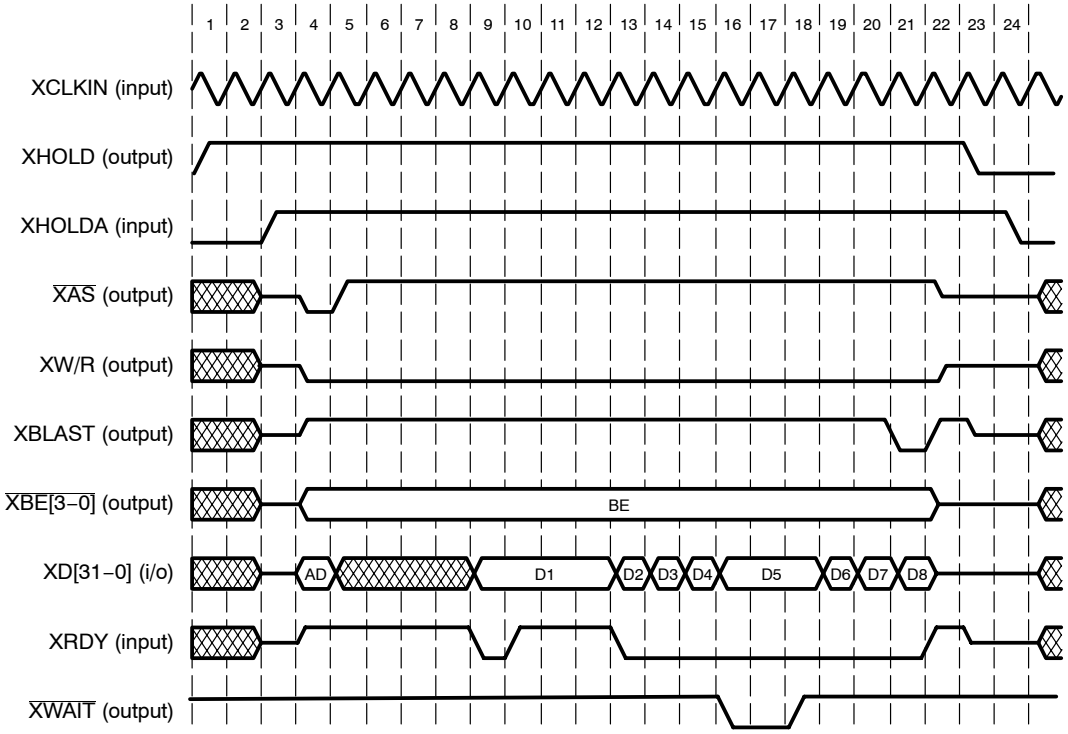
The $\overline{\text{XWAIT}}$ signal prevents data overflow/underflow when the DSP is a master on the XBUS. The $\overline{\text{XWAIT}}$ signal, which is multiplexed with the $\overline{\text{XWE}}$ signal, can be thought of as a ready output when the DSP initiates transfers on the XBUS. When the DSP has initiated a transaction, the DSP indicates that it is not ready to deliver/receive new data by asserting the $\overline{\text{XWAIT}}$ signal low.

The $\overline{\text{XWAIT}}$ signal is an output only signal in synchronous host port mode.

4.1.1.1 Burst Read Transfer

The timing diagram in Figure 15 can be referenced for a visual description of the steps required to complete a burst read initiated by the DSP and throttled by the $\overline{\text{XWAIT}}$ and XRDY signals.

Figure 15. Read Transfer Initiated by the DSP and Throttled by $\overline{\text{XWAIT}}$ and XRDY Signals (Internal Bus Arbiter Disabled)



Note: $\overline{\text{XWAIT}}$ is an output only signal in synchronous host port mode.

The step-by-step description of the events marked above the waveforms in Figure 15:

1) The DSP requests the XBUS by asserting XHOLD output.
2) The DSP waits for the XBUS.
3) The external bus arbiter asserts the XHOLDA signal, and the DSP starts driving the bus. The $\overline{\text{XAS}}$, XW/R, XBLAST, $\overline{\text{XBE[3–0]}}$ signals become outputs, and the XRDY signal becomes an input.

4) Address phase: During this phase, $\overline{\text{XAS}}$ is asserted and the address is presented on the XBUS.

5) Data phase: The external device is not ready to deliver data, as indicated by XRDY high.

6) Same as step 5.

7) Same as step 5.

8) Same as step 5.

9) The external device presents requested data (D1), and asserts XRDY.

10) The external device is not ready to deliver next data. The XRDY is negated.

11) Same as step 10

12) Same as step 10

13) The external device presents next data (D2), and asserts XRDY.

14) The external device presents next data (D3), and XRDY stays asserted.

15) The external device presents next data (D4), and XRDY stays asserted.

16) The external device presents next data (D5), and XRDY stays asserted. The DSP can not accept the new data (D5), and asserts $\overline{\text{XWAIT}}$.

17) The external device recognizes $\overline{\text{XWAIT}}$, and keeps the D5 on the XBUS. The XRDY is asserted and indicates that the external device is ready waiting for the DSP to accept the data.

18) The DSP deasserts $\overline{\text{XWAIT}}$, and accepts D5.

19) The external device presents next data (D6), and XRDY stays asserted.

20) The external device presents next data (D7), and XRDY stays asserted.

21) The external device presents the last data (D8), and the DSP asserts the XBLAST.

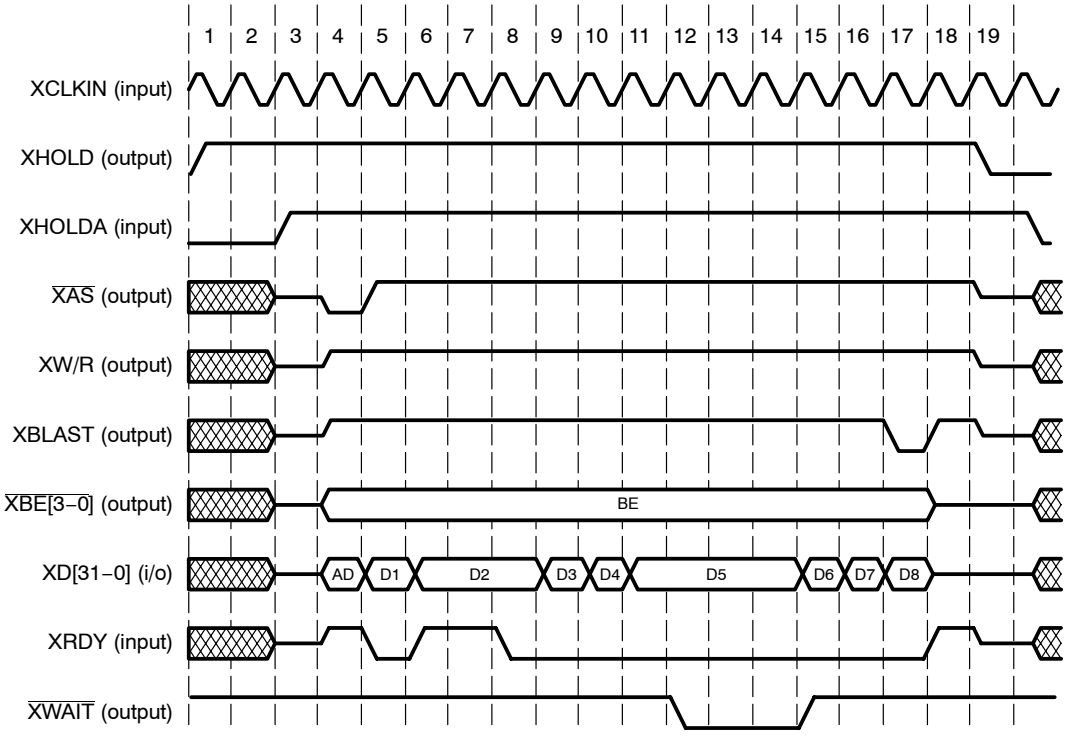
22) The recovery cycle.

23) The DSP negates the XBUS request (XHOLD), and turns off the outputs.

4.1.1.2 Burst Write Transfer

The timing diagram in Figure 16 can be referenced for a visual description of the steps required to complete a burst write initiated by the DSP and throttled by the $\overline{\text{XWAIT}}$ and XRDY signals.

Figure 16. Write Transfer Initiated by the DSP and Throttled by $\overline{\text{XWAIT}}$ and XRDY Signals (Internal Bus Arbiter Disabled)



Note: $\overline{\text{XWAIT}}$ is an output only signal in synchronous host port mode.

The step-by-step description of the events marked above the waveforms in Figure 16:

1) The DSP requests the XBUS (XHOLD asserted).
2) The DSP waits for the XHOLDA signal to be asserted by the external arbiter.
3) The external bus arbiter asserts the XHOLDA signal, the $\overline{\text{XAS}}$, XW/R, XBLAST, and $\overline{\text{XBE[3–0]}}$ signals become outputs, and the XRDY signal becomes an input.

4) Address phase: During this phase, the $\overline{\text{XAS}}$ is asserted and the address is presented on the XBUS.

5) Data phase: During this phase, data (D1) is presented by the DSP and the external device is ready to accept the data, which is indicated by XRDY being active.

6) The DSP presents next data (D2). The external device indicates not ready condition, which is indicated by XRDY being inactive.

7) The DSP is holding data D2 on the XBUS since the external device is still not ready.

8) External device finally accepts the D2.

9) The DSP presents next data (D3). The external device is ready to take D3.

10) The DSP presents next data (D4). The external device is ready to take D4.

11) The DSP presents next data (D5). The external device is ready to take D5.

12) The DSP is not ready to present D6 and asserts $\overline{\text{XWAIT}}$. The external device is waiting for the DSP to present new data.

13) Same as step12.

14) Same as step 12.

15) The DSP presents next data (D6), and negates $\overline{\text{XWAIT}}$. The external device is ready to take D6.

16) The DSP presents next data (D7). The external device is ready to take D7.

17) The DSP presents the last data (D8), and asserts XBLAST. The external device is ready to take D8.

18) Recovery cycle

19) The DSP removes the bus request (XHOLD), and is turns off the outputs.

To prevent contention on the XBUS, one recovery state between the last data transfer and next address cycle is inserted.

4.1.1.3 Preventing Deadlocks with Backoff

The XBUS has the XBOFF signal to prevent deadlocks while the DSP is performing a master transfer. When asserted, XBOFF suspends the current access and causes the DSP to release ownership of the XBUS. Figure 17 is the timing diagram for the XBOFF signal.

The backoff is only recognized during active master transfers when XRDY indicates a not-ready status and one of the following conditions exists:

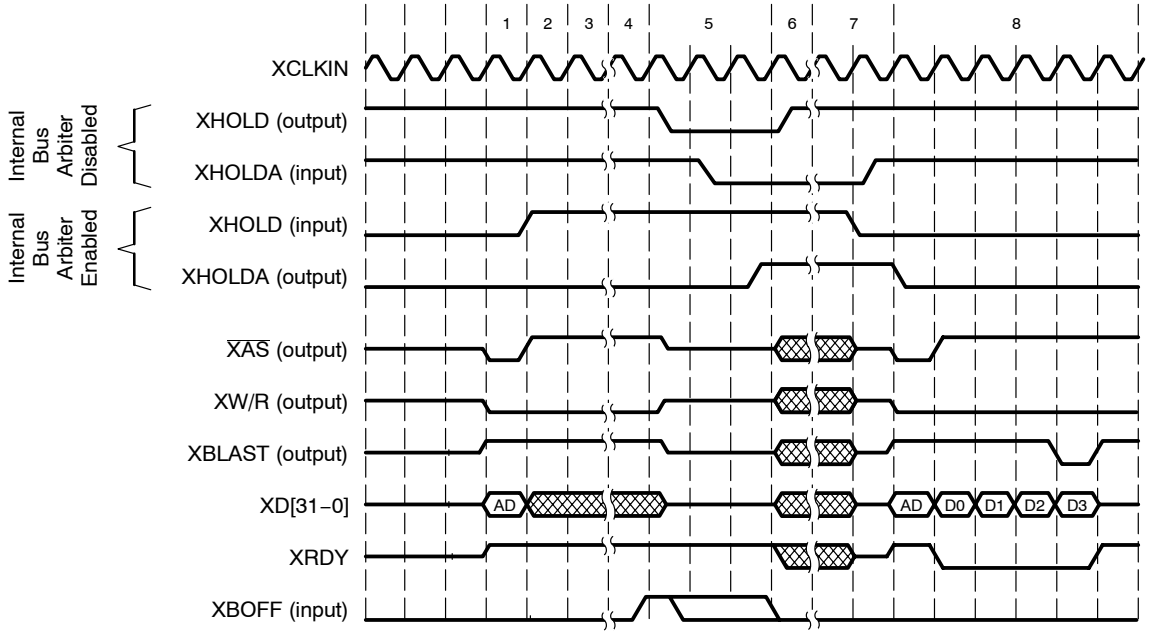
1) The external device is requesting the XBUS (XHOLD = 1), when the internal bus arbiter is enabled (XARB = 1).

or

2) The DSP is the XBUS master (XHOLD = 1 and XHOLDA = 1), and the internal bus arbiter is disabled (XARB = 0).

The backoff request is not serviced until all current master transfers are completed internally. This allows read data to be flushed out of the pipeline. The XBOFF signal is not recognized during I/O port transfers.

Figure 17. External Device Requests the Bus From the DSP Using XBOFF



The timing diagram in Figure 17 can be referenced for a visual description of the steps involved in release of the XBUS ownership as initiated by the XBOFF signal. Figure 17 shows the backoff condition for both internal bus arbiter enabled and internal bus arbiter disabled. The step-by-step description of the events marked above the waveforms in Figure 17:

1) The DSP is the XBUS master and initiates address phase of a read transaction. The $\overline{\text{XAS}}$ signal is active and valid address is presented.

2) The XRDY signal is high, indicating that the external device is not ready to perform the transaction. Also, the external device drives XHOLD active, indicating a bus request.

3) The DSP is still holding the XBUS waiting for XRDY to become low.

4) The external device asserts XBOFF, indicating a potential deadlock condition.

5) The DSP responds by releasing the XBUS. When the internal bus arbiter is enabled, the DSP asserts XHOLDA. When the internal bus arbiter is disabled the DSP deasserts XHOLD. It can take a several clock cycles before the DSP responds to XBOFF. Figure 17 shows the fastest response time, one cycle.

6) The XBUS ownership changes. The new master drives the XBUS. XBOFF is deasserted.

7) The external device releases the bus after performing the desired transactions.

8) The XHOLDA is removed, and the DSP resumes the XBUS ownership.

9) The DSP performs a burst read of four words.

The DSP automatically tries to restart the transfer interrupted by a backoff from the point where the interruption took place. The transfer restart is completely transparent.

4.1.2 TMS320C62x Slave on the Expansion Bus

The external host can access the different expansion bus host port registers by driving the XCNTL signal as follows:

- XCNTL = 0; reads or writes the expansion bus data register (XBD).
- XCNTL = 1; reads or writes the expansion bus internal slave address register (XBISA).

Every transaction initiated by the host on the XBUS is a two-step process. First, the host has to set XBISA, and then transfer the data to/from the address pointed by XBISA. Bursts longer than one word must take place with autoincrementing of XBISA. Therefore, the AINC bit must always be cleared to 0.

Initial access made to the expansion bus in host slave mode should be done in the previous order. After reset, the first access from the host should be an XBISA write followed by an XBD read/write. Undefined operation may occur if an XBISA read or an XBD read/write occurs before an XBISA write.

To read/write from the DSP memory space, the host must follow the following sequence:

1) The host writes the transfer source/destination address to XBISA and clears the AINC bit.
2) The host reads/writes the address specified by XBISA. Read or write is determined by the XBISA XW/R signal. XBISA is autoincremented since bit 1 of XBISA must be cleared by the external host.
3) If the transfer is a burst, dictated by the BLAST signal, data is continuously read or written. If a multiword burst is being done, all $\overline{\text{XBE}n}$ signals must be low, because only 32-bit word bursts are allowed. If less than 32 bits are transferred (specified by $\overline{\text{XBE}n}$ signals), then only single element transfers are allowed.

The $\overline{\text{XWAIT}}$ signal is not used in slave mode.

4.1.2.1 Cycle Description

Each access initiated by the external host can be broken up into distinct categories:

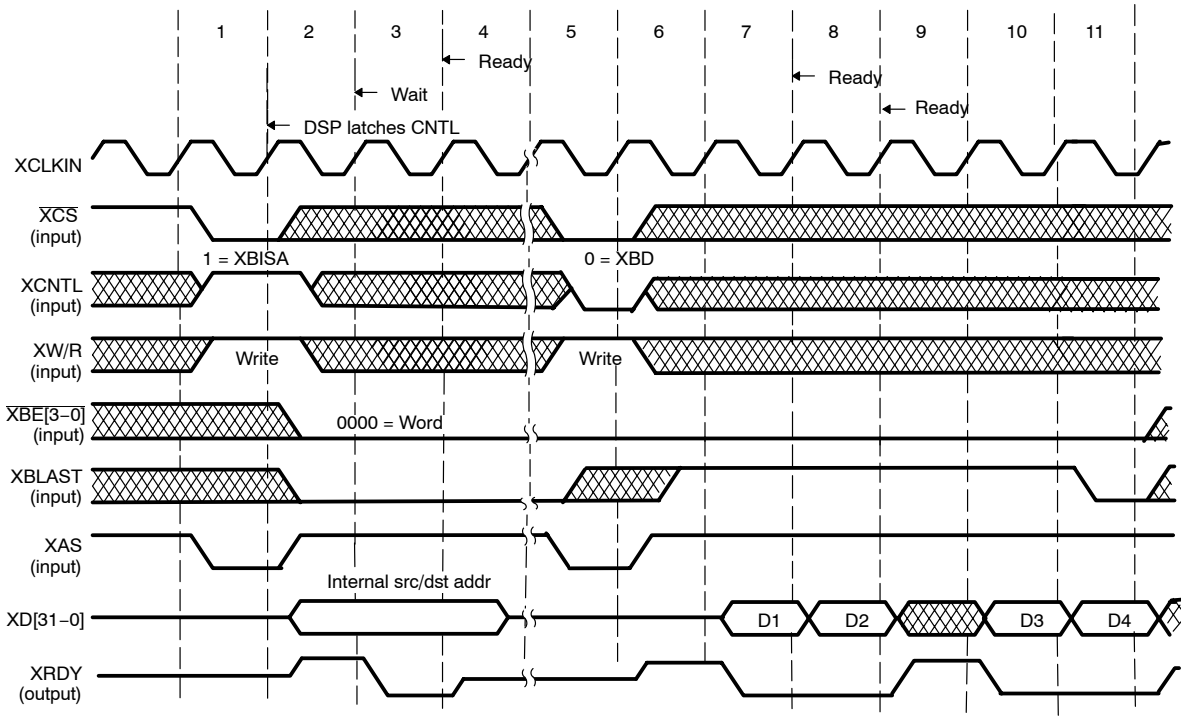
- **Address phase (Ta):** During the address phase, the DSP is selected with the $\overline{\text{XCS}}$ input and the address phase is started with a low pulse on the $\overline{\text{XAS}}$ signal. During this phase, the DSP determines if the external master is doing a read or write cycle (XW/R input) and which XBUS register is being accessed (via the XCNTL input).
- **Wait/data phase (Tw/Td):** Immediately after the address phase, the transaction enters either the wait phase or data phase. For a read cycle, there is at least one wait phase before the DSP presents the data to the external host. This is controlled via the XRDY output of the DSP. If the XRDY signal is high, this indicates to the external host that the DSP is not ready to receive data for a write, or is not ready to present data for a read, and is in the wait phase. The data phase is entered when the DSP asserts XRDY signal, indicating that read data should be latched by the external host or that write data has been latched by the DSP.
- **Recovery phase (Tr):** The recovery phase is entered after the final data phase of a burst access, or after the data phase of a single access. When the DSP is a slave, if the external master has a multiplexed address/data bus, it is recommended that the external master insert at least one recovery phase between a read data phase and a subsequent address phase in order to avoid potential bus contention.

4.1.2.2 Burst Write Transfer

The timing diagram in Figure 18 can be referenced for a visual description of the steps required to complete a burst write initiated by an external host and throttled by the XRDY signal.

The boot configuration for XBLAST and XW/R: BLPOL = 0 and RWPOL = 0. See Table 8 (page 27) for more details.

Figure 18. Expansion Bus Master Writes a Burst of Data to the DSP



The step-by-step description of the events marked above the waveforms in Figure 18:

1) The $\overline{\text{XCS}}$ and $\overline{\text{XAS}}$ signals are low and the XCNTL signal is high, indicating XBISA as the destination for the following transaction. The XW/R is high, specifying that a write access is taking place.

2) The DSP begins driving the XRDY output in response to a transfer initiated by the external host. A high XRDY indicates that the DSP is not ready.

3) The data is written to XBISA when the DSP asserts the XRDY output low.

4) The DSP inserts one or more not-ready cycles (XRDY = 1) between the address phase and the first data phase.

5) The $\overline{\text{XAS}}$ and XCNTL signals are low (and $\overline{\text{XCS}}$ is low), indicating XBD as the destination for the following transaction. The XW/R is high, specifying that a write access is taking place.

6) The DSP inserts one not-ready cycle (XRDY = 1).

7) The XBUS master presents the valid data. The data is written to XBD on the rising edge of the XCLKIN when XRDY is active-low.

8) Same as step 7.

9) The DSP is not ready to accept next data, which is indicated by XRDY high.

10) Same as step 7.

11) The XBUS master indicates that the last write transaction is taking place by asserting the XBLAST signal. The data is written to XBD on the rising edge of the XCLKIN.

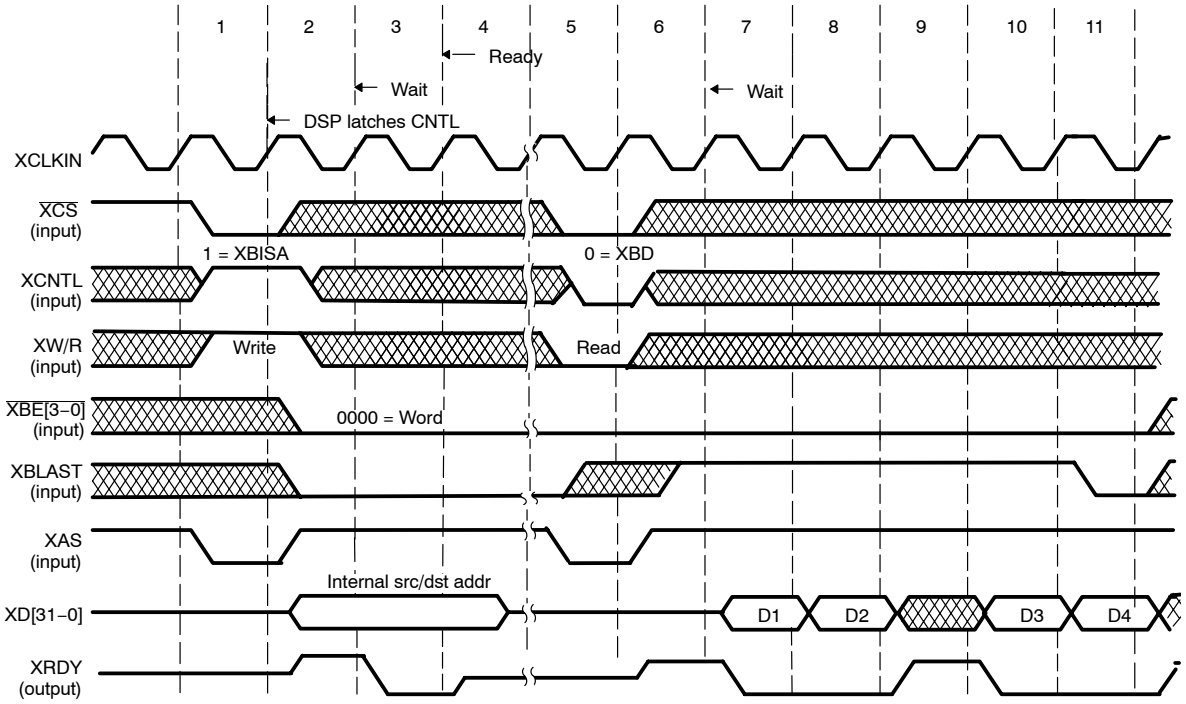
The $\overline{\text{XWAIT}}$ signal is not used in slave mode.

4.1.2.3 Burst Read Transfer

The timing diagram in Figure 19 can be referenced for a visual description of the steps required to complete a burst read initiated by an external host and throttled by the XRDY signal.

The boot configuration for XBLAST and XRW: BLPOL = 0 and RWPOL = 0. See Table 8 (page 27) for more details.

Figure 19. Expansion Bus Master Reads a Burst of Data From the DSP



The step-by-step description of the events marked above the waveforms in Figure 19:

1) The $\overline{\text{XCS}}$ and $\overline{\text{XAS}}$ signals are low and the XCNTL signal is high, indicating XBISA as the destination for the following transaction. The XW/R is high, specifying that a write access is taking place.

2) The DSP begins driving the XRDY output in response to a transfer initiated by the external host. A high XRDY indicates that the DSP is not ready.

3) The data is written to XBISA when the DSP asserts the XRDY output low.

4) The DSP inserts one or more not-ready cycles (XRDY = 1) between the address phase and the first data phase.

5) The $\overline{\text{XAS}}$ and XCNTL signals are low (and $\overline{\text{XCS}}$ is low), indicating XBD as the destination for the following transaction. The XW/R is low, specifying that a read access is taking place.

6) The DSP inserts one not-ready cycle (XRDY = 1).

7) The DSP presents the valid data, and drives XRDY low.

8) Same as step 7.

9) The DSP is not ready to present the next data, which is indicated by XRDY high.

10) Same as step 7.

11) The XBUS master indicates that the last read transaction is taking place by asserting the XBLAST signal.

The $\overline{\text{XWAIT}}$ signal is not used in slave mode.

4.2 Asynchronous Host Port Mode

The asynchronous host port mode is slave only, it uses a 32-bit data path and is similar to the host port interface (HPI) on the C6201 DSP. The asynchronous host port mode is used to interface to asynchronous microprocessor buses.

A description of the signals when the XBUS operates in the asynchronous host port mode is listed in Table 9.

Table 9. Signal Description—Asynchronous Host Port Mode

Signal Name	I/O/Z	Signal Purpose	Signal Function
$\overline{\text{XCS}}$	I	Chip select	Selects the DSP as a target of an external master.
XD[31−0]	I/O/Z	Data bus	
$\overline{\text{XBE}}$[3−0]	I	Byte enables	Functionality of these signals is the same as on the DSP HPI (during a read, XBE does not matter). During a write: $\overline{\text{BE3}}$ byte enable 3: XD[31−24] $\overline{\text{BE2}}$ byte enable 2: XD[23−16] $\overline{\text{BE1}}$ byte enable 1: XD[15−8] $\overline{\text{BE0}}$ byte enable 0: XD[7−0] 8-bit data must be byte-aligned. 16-bit data must be halfword-aligned. 32-bit data must be word-aligned.
XCNTL	I	Control signal	This signal selects between XBD and XBISA. XCNTL=0, access is made to the expansion bus data register (XBD). XCNTL=1, access is made to the expansion bus internal slave address register (XBISA).
XW/R	I	Read/write	Read/write enable. Polarity of this signal is configured during boot.
XRDY	O/Z	Ready out	Ready signal indicates normally not-ready condition. This signal is always driven in asynchronous host mode when the DSP does not own the bus.

The XCNTL signal selects which internal register the host is accessing. The state of this pin selects if access is made to the expansion bus internal slave address register (XBISA) or the expansion bus data register (XBD).

If the XBUS host port operates in the asynchronous mode, every transaction initiated by the host on the XBUS is a two-step process. The host first has to set XBISA, and then transfer the data to/from the address pointed to by XBISA. The data transfer can take place with or without autoincrementing XBISA. Whether or not XBISA gets autoincremented is determined by the AINC bit in XBISA.

In order to read/write from the DSP memory spaces, the host must follow the following sequence:

1) Host writes the address to XBISA, and sets AINC accordingly in XBISA.

2) Host reads/writes the address specified by XBISA. Read or write is determined by the XW/R signal. XBISA may be autoincremented, depending upon what is written to the AINC bit during step 1.

Initial access made to the expansion bus in host slave mode should be done in the order indicated above. After reset, the first access from the host should be an XBISA write followed by an XBD read/write. Undefined operation may occur if an XBISA read or an XBD read/write occurs before an XBISA write.

If the XBUS host port is configured to operate in asynchronous mode, the $\overline{\text{XCS}}$ signal is used for four purposes:

1) To select the XBUS host port as a target of an external master.

2) On a read, the falling edge of $\overline{\text{XCS}}$ initiates read accesses.

3) On a write, the rising edge of $\overline{\text{XCS}}$ initiates write accesses.

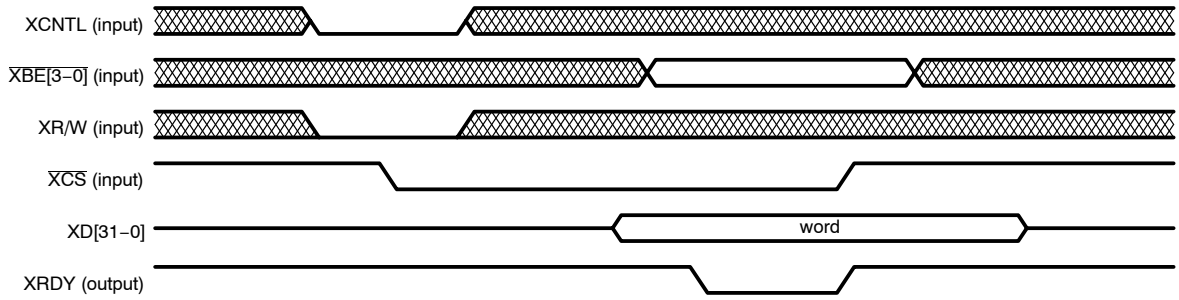
4) The $\overline{\text{XCS}}$ falling edge latches XBUS host port control inputs including: XW/R and XCNTL.

The XRDY signal of the DSP functions differently than the C6201 HPI READY signal. The XRDY signal indicates normally not ready condition (active-low READY signal is internally ORed with $\overline{\text{XCS}}$ signal in order to obtain XRDY). XRDY should be polled during reads/writes to/from the XBISA or XBD.

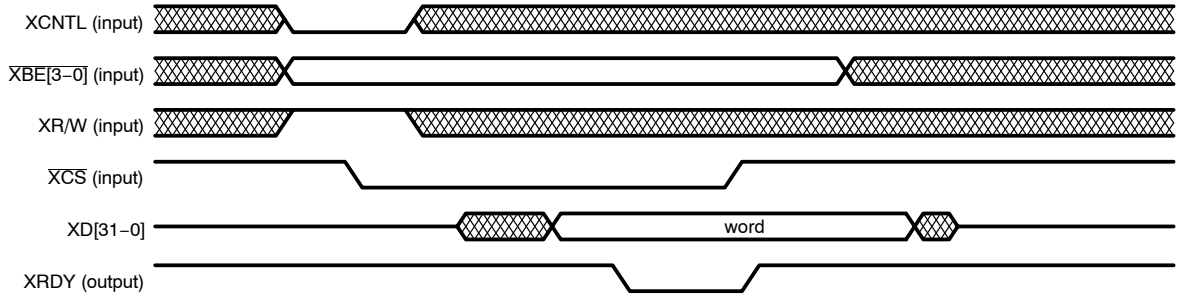
Asynchronous host port read and write timing diagrams are shown in Figure 20.

Figure 20. Asynchronous Host Port Mode Timing Diagrams

(a) Write Timing



(b) Read Timing



4.3 Special Circumstance of XBUS Host Memory Accesses

When the XBUS host port executes a read from the DSPs memory space, it does so by performing burst prefetches of 3 words. This results in the DMA auxiliary channel reading 3 higher-word addresses that you may not have explicitly requested. This occurs only under the following situations:

- ❏ An external master performs an autoincremented read from the XBUS configured for host slave mode (both synchronous or asynchronous).
- ❏ The XBUS configured as the master in synchronous host port mode writes from the DSP to the external space via the XBUS.

The accesses described above can cause the following undesired operations:

1) Accesses to undesired CE spaces:
   - When reading the top 3 words of EMIF CE0, the resulting prefetches can cause an inadvertent access to CE1 that may cause an undesired read to a device or a stall, if the inadvertent access is to an asynchronous memory space with ARDY left floating or pulled inactive (not-ready).
   - The above example also applies to CE2 with the resulting prefetches possibly causing an inadvertent access to CE3.

   Associated design tip: If not using ARDY or XRDY, always pull to the ready state to avoid stalls. If you always want to detect bad software setups, always pull to the not-ready state to detect system stalls.

2) Unintended port crossings or illegal accesses to a reserved location:
   - When reading the top 3 words of EMIF CE1, the access can cross into either program memory (PMEM) block 0 when in Map 0, or to the internal peripheral bus (PBus) region storing EMIF control registers when in Map 1. This is an illegal port crossing.
   - When reading the top 3 words of EMIF CE3, the access can cross into reserved address space. This is an illegal access.
   - When reading the top 3 words of PMEM block 0, the access can cross into PMEM block 1. This is an illegal port crossing.
   - When reading the top 3 words of PMEM block 1, the access can cross into reserved address space. This is an illegal access.
   - When reading the top 3 words of data memory (DMEM) block 1, the access can cross into reserved address space. This is an illegal access.
   - When reading anywhere in the PBus space, you may prefetch ahead to three undesired control registers. This can cause an illegal access when accessing a reserved register address. If the register access has side effects (like reading the McBSP DRR, clearing RRDY), then you may inadvertently cause these side effects.

Note: A restriction does not exist when crossing between DMEM block 0 and block 1 because they both use the same DMA port.

Associated design tips:

- When reading internal peripheral registers:
  - For reads from an external master, use fixed-mode addressing. As a broader statement, it is good practice to use fixed mode also when writing to peripheral registers as sometimes there are gaps between them.
  - Do not use the XBUS host port configured in synchronous master mode to directly copy peripheral register values to external slaves. This is an atypical operation. If you must do so, copy the register data to an internal space with the CPU and then copy those internal locations to external space.
- When reading the top 3 locations of an EMIF CEx, internal program block or DMEM block 1, use fixed-mode addressing. Note this procedure does not have to be followed when accessing the top 3 words of DMEM block 0, this is because DMEM block 0 and block 1 are in the same DMA port.

5 Expansion Bus Arbitration

Two signals, XHOLD and XHOLDA, are provided for expansion bus arbitration. The internal bus arbiter is disabled or enabled depending on the value on the expansion data bus during reset.

The XARB bit in the expansion bus global control register (XBGC) indicates if the internal bus arbiter is enabled or disabled as shown in Table 10. If the internal bus arbiter is enabled (XARB = 1), the DSP wakes up from reset as the bus master. If the internal bus arbiter is disabled (XARB = 0), the DSP wakes up from reset as the bus slave. The DMA controller releases the XBUS between frames if a DMA block transfer is in progress. When the DSP releases the XBUS, the host port signals become tristated, except for the I/O port signals ($\overline{\text{XWE}}$/XWAIT, $\overline{\text{XOE}}$, $\overline{\text{XRE}}$, $\overline{\text{XCE}}$[3–0], and XFCLK) that are not affected.

Table 10. XHOLD and XHOLDA Signal Functionality Based on XARB Bit Value

XARB Bit Value	Internal Bus Arbiter	XHOLD	XHOLDA	Section
0	Disabled	Output	Input	5.2
1	Enabled	Input	Output	5.1

5.1 Internal Bus Arbiter Enabled

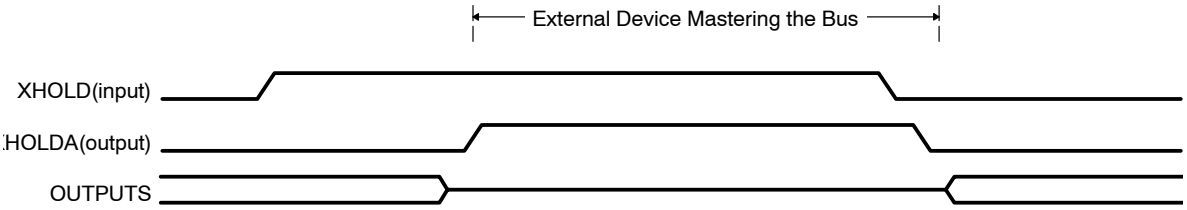
When the internal bus arbiter is enabled (XARB = 1), the DSP owns the XBUS by default. The DSP wakes up from reset as the master of the XBUS, and all other devices must request the bus from the DSP. This mode is preferred when connecting one DSP to a PCI interface device.

When the DSP owns the XBUS, both XHOLD (input) and XHOLDA (output) are low. XHOLD is asserted by an external device to request use of the XBUS. The DSP asserts XHOLDA when a bus request is granted. The XBUS is not granted unless requested by XHOLD.

Figure 21 shows the XHOLD and XHOLDA functionality when the internal bus arbiter is enabled. The DSP grants the XBUS to the requester only if no internal transfer requests to the XBUS are pending.

During a synchronized slave single-word write to XBD, if XHOLD input is deasserted quickly enough after XBLAST = 0 (in the same cycle), the transfer gets corrupted. No slave single-word write with the simultaneous removal of the XHOLD and assertion of the XBLAST should be allowed (XARB = 1). In these cases, the XHOLD should be registered before connecting it to the XHOLD input on the DSP.

Figure 21. XHOLD/XHOLDA Timing Diagram for Bus Arbitration With Internal Bus Arbiter Enabled



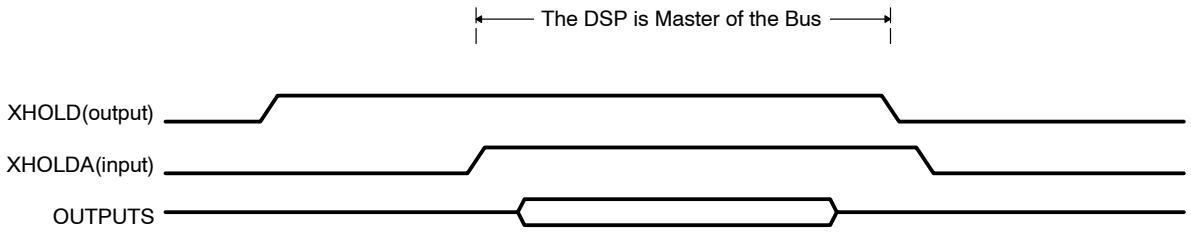
5.2 Internal Bus Arbiter Disabled

When the internal bus arbiter is disabled (XARB = 0), the DSP acts as slave on the XBUS by default. This mode is preferred when the DSP is interfacing to an external host, or if multiple DSPs are connected to a PCI interface device.

When the DSP owns the XBUS, both XHOLD (output) and XHOLDA (input) are high. To request use of the XBUS (for example to access a FIFO), the DSP asserts XHOLD. The external XBUS arbiter asserts XHOLDA when control is granted. The XBUS should not be granted to the DSP unless requested by XHOLD.

Figure 22 shows the XHOLD and XHOLDA functionality when the internal bus arbiter is disabled.

Figure 22. XHOLD/XHOLDA Timing Diagram for Bus Arbitration With Internal Bus Arbiter Disabled



When the internal bus arbiter is disabled and the XBUS master transfer is initiated by writing to the START bits of XBHC, the DSP asserts its XHOLD request. If the host initiates a transfer to the DSP instead of granting the DSP access to the XBUS, the DSP drops its XHOLD request as shown in Figure 23.

The DSP drops the bus request only if the pending request is for a transfer to the XBUS host port. The DSP reasserts the bus request for pending master transfers after the host completes its transfer (see Figure 23). Table 11 shows possible scenarios that can happen when the internal bus arbiter is disabled.

Figure 23. XHOLD Timing Diagram When the External Host Starts a Transfer to the DSP Instead of Granting the DSP Access to the Expansion Bus With Internal Bus Arbiter Disabled

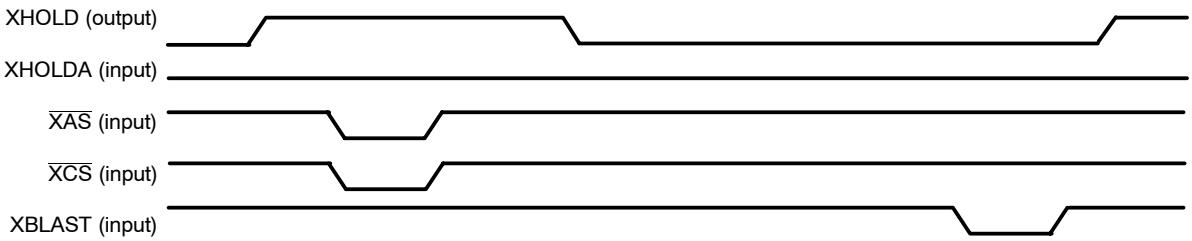


Table 11. Possible Expansion Bus Arbitration Scenarios With Internal Bus Arbiter Disabled

XBOFF Asserted	Current External Host Activity	Current DSP State	Actions
N/A	Host transfer to the XBUS in progress	DMA request to XBUS I/O port pending	If the DMA request comes before or at the same time when the host started the transfer, the DSP asserts the XHOLD and keeps it asserted during the host transfer. If the DMA request came after the host started the transfer, the DSP waits for the host transfer to complete and then asserts XHOLD.
		DMA request to XBUS I/O port, and auxiliary DMA requests are pending	After the DSP gets the XBUS, the pending auxiliary DMA request is executed first (since for the XBUS, the auxiliary DMA channel always has priority over the other DMA channels). After the DMA transfer is completed, the DSP starts the auxiliary DMA transfer and does not drop the XHOLD between these two transfers.
		Auxiliary DMA request pending	If the auxiliary DMA request comes prior to the host starting the transfer, the DSP asserts the XHOLD and keeps it asserted until the host starts the transfer. Once the host starts the transfer, the DSP drops the request (see Figure 15). The DSP reasserts the XHOLD after the host completes the transfer. If the auxiliary DMA request comes after the host is started the transfer, the DSP waits for the host transfer to complete and asserts the XHOLD.

Table 11. Possible Expansion Bus Arbitration Scenarios With Internal Bus Arbiter Disabled (Continued)

XBOFF Asserted	Current External Host Activity	Current DSP State	Actions
No	None	DMA request to XBUS I/O port pending	The DSP asserts the XHOLD, and once it gets the XBUS, the transfer starts.
		DMA request to XBUS I/O port, and auxiliary DMA requests are pending	After the DSP gets the XBUS, the pending auxiliary DMA request is executed first (since for the XBUS, the auxiliary DMA channel always has priority over the rest of the DMA channels). After the auxiliary DMA transfer is completed, the DSP starts the DMA transfer and does not drop the XHOLD between these two transfers.
		Auxiliary DMA request pending	The DSP asserts the XHOLD, and once it gets the XBUS the transfer starts.
Yes	N/A	DMA transfer to XBUS I/O port in progress	XBOFF is ignored if a DMA transfer to the XBUS I/O port is in progress.
		Auxiliary DMA transfer in progress	The DSP releases ownership of the XBUS as soon as possible. After that, the DSP requests the XBUS to complete the transfer interrupted by the XBOFF.
		Auxiliary DMA transfer in progress, and DMA request to XBUS I/O port pending	The DSP stops the current auxiliary DMA transfer in progress, and starts executing the pending DMA transfer to the XBUS I/O port. After the pending DMA transfer is completed, the DSP releases the XBUS to the external device. Some time afterwards, the DSP requests the XBUS to complete the transfer interrupted by the XBOFF.

5.3 Expansion Bus Requestor Priority

The auxiliary DMA channel for the XBUS is always given the highest priority, followed by the standard DMA priority (DMA0 highest) as listed in Table 12.

Table 12. Expansion Bus Requestor Priority

Priority	Description
Highest	Auxiliary channel
	DMA0
	DMA1
	DMA2
Lowest	DMA3

In many situations, the priority between the auxiliary channel and the standard DMA channels is first-come, first-served, because the auxiliary channel cannot preempt the standard DMA channels during a frame transfer and the standard DMA channels cannot preempt the auxiliary channel. The standard DMA channels can preempt each other.

The auxiliary channel can only acquire the bus between DMA frames or if no other DMA activity is occurring. For example, if an unsynchronized DMA transfer is set up to perform 4 frames of 32 elements each, and an auxiliary transfer becomes pending, either by an external host asserting the XHOLD request signal if the internal arbiter is enabled or by the DSP attempting to begin a master transfer by writing to the start bits of XBHC (internal arbiter enabled or disabled), the auxiliary request will be ignored during the frame transfer to the expansion memory. After the first frame, however, the auxiliary request is recognized and the DMA transfer to the expansion memory stops to allow the host transfer to begin.

To allow host transfers sufficient access to the XBUS, DMA transactions should be set up so that the frame length is as short as possible. The size of frame transfers to the XBUS I/O port define the longest amount of time that host transactions can be blocked from accessing the XBUS.

6 Boot Configuration Control via Expansion Bus

The pull-up/pull-down resistors on the expansion bus XD[31–0] pins are used to determine the boot and device configurations during reset. The boot and device configurations include:

- Boot-mode of the device
- Lendian mode selection
- FIFO mode
- Internal expansion bus arbiter enable/disable
- Expansion bus host port mode
- Polarity of read/write XW/R and XBLAST control signals
- Memory type used in each XBUS memory space.

Several device settings are configured at reset to determine how the device operates. These settings include the boot configuration, the input clock mode, endian mode, and other device-specific configurations. The boot configuration is determined by the BOOTMODE[4–0] values. Table 13 lists all the values for BOOTMODE[4–0], as well as the associated memory maps and boot processes. For example, the value 00000b on BOOTMODE[4–0] selects memory map 0, indicates that the memory type at address 0 is synchronous DRAM (SDRAM) organized as four 8-bit-wide banks, and that no boot process is selected. SDWID is a bit in the EMIF SDRAM control register.

For C6000 DSPs with multiple memory maps, the boot configuration determines the type of memory located at the reset address for processor operation, address 0, as shown in Table 13. When the boot configuration selects MAP 1, this memory is internal; when the boot configuration selects MAP 0, the memory is external. When external memory is selected, the boot configuration also determines the type of memory at the reset address. These options effectively provide alternative reset values to the appropriate EMIF control registers.

Table 13. Boot Configuration Summary

BOOTMODE	Memory Map	Memory at Address 0	Boot Process
00000	MAP 0	SDRAM: four 8-bit devices (SDWID = 0)	None
00001	MAP 0	SDRAM: two 16-bit devices (SDWID = 1)	None
00010	MAP 0	32-bit asynchronous with default timing	None
00011	MAP 0	1/2 × rate SBSRAM	None
00100	MAP 0	1 × rate SBSRAM	None
00101	MAP 1	Internal	None

Table 13. Boot Configuration Summary (Continued)

BOOTMODE	Memory Map	Memory at Address 0	Boot Process
00110	MAP 0	External: default values	Host boot (HPI/XBUS/PCI)
00111	MAP 1	Internal	Host boot (HPI/XBUS/PCI)
01000	MAP 0	SDRAM: four 8-bit devices (SDWID = 0)	8-bit ROM with default timings
01001	MAP 0	SDRAM: two16-bit devices (SDWID = 1)	8-bit ROM with default timings
01010	MAP 0	32-bit asynchronous with default timing	8-bit ROM with default timings
01011	MAP 0	1/2 × rate SBSRAM	8-bit ROM with default timings
01100	MAP 0	1 × rate SBSRAM	8-bit ROM with default timings
01101	MAP 1	Internal	8-bit ROM with default timings
01110-01111	–	Reserved	
10000	MAP 0	SDRAM: four 8-bit devices (SDWID = 0)	16-bit ROM with default timings
10001	MAP 0	SDRAM: two 16-bit devices (SDWID = 1)	16-bit ROM with default timings
10010	MAP 0	32-bit asynchronous with default timing	16-bit ROM with default timings
10011	MAP 0	1/2× rate SBSRAM	16-bit ROM with default timings
10100	MAP 0	1× rate SBSRAM	16-bit ROM with default timings
10101	MAP 1	Internal	16-bit ROM with default timings
10110-10111	–	Reserved	
11000	MAP 0	SDRAM: four 8-bit devices (SDWID = 0)	32-bit ROM with default timings
11001	MAP 0	SDRAM: two 16-bit devices (SDWID = 1)	32-bit ROM with default timings
11010	MAP 0	32-bit asynchronous with default timing	32-bit ROM with default timings
11011	MAP 0	1/2 × rate SBSRAM	32-bit ROM with default timings
11100	MAP 0	1 × rate SBSRAM	32-bit ROM with default timings
11101	MAP 1	Internal	32-bit ROM with default timings
11110-11111	–	Reserved	

6.1 Boot and Device Configuration

The pull-up/pull-down resistors on the XBUS are used to determine the boot configuration (pins XD[4−0]) and other device configurations (pins XD[31−5]) at reset. The XD[4−0] pins directly map to BOOTMODE[4−0] described in Table 13. Reserved configuration pins should be pulled-down. Detailed descriptions of boot and device configurations are shown in Table 14.

Table 14. Boot and Device Configuration Description

XD Pin	Type	Value	Description
30-28	MTYPE3		XCE3 memory type.
		000-001	Reserved
		010	32-bit wide asynchronous interface
		011-100	Reserved
		101	32-bit wide FIFO interface
		110-111	Reserved
27	Reserved	–	Reserved configuration pins should be pulled-down.
26-24	MTYPE2		XCE2 memory type.
		000-001	Reserved
		010	32-bit wide asynchronous interface
		011-100	Reserved
		101	32-bit wide FIFO interface
		110-111	Reserved
23	Reserved	–	Reserved configuration pins should be pulled-down.
22-20	MTYPE1		XCE1 memory type.
		000-001	Reserved
		010	32-bit wide asynchronous interface
		011-100	Reserved
		101	32-bit wide FIFO interface
		110-111	Reserved

Table 14. Boot and Device Configuration Description (Continued)

XD Pin	Type	Value	Description
19	Reserved	–	Reserved configuration pins should be pulled-down.
18-16	MTYPE0		XCE0 memory type.
		000-001	Reserved
		010	32-bit wide asynchronous interface
		011-100	Reserved
		101	32-bit wide FIFO interface
		110-111	Reserved
15-14	Reserved	–	Reserved configuration pins should be pulled-down.
13	BLPOL		Determines polarity of the XBLAST signal when the DSP is a slave on the XBUS. When the DSP initiates a transfer on the expansion bus, XBLAST is always active low.
		0	XBLAST is active low.
		1	XBLAST is active high.
12	RWPOL		Determines polarity of XBUS read/write signal.
		0	Write is active-high.
		1	Read is active-high.
11	HMOD		Host mode.
		0	External host interface operates in asynchronous slave mode.
		1	External host interface operates in synchronous master/slave mode.
10	XARB		XBUS arbiter (status in XBGC).
		0	Internal XBUS arbiter is disabled.
		1	Internal XBUS arbiter is enabled.
9	FMOD		FIFO mode (status in XBGC).
		0	Glue logic is used for FIFO read interface in all XCE spaces operating in FIFO mode. XOE can be used in all XCE spaces.
		1	XOE is reserved for use only in $\overline{\text{XCE3}}$ for FIFO read mode. XOE is disabled in all other XCE spaces.

Table 14. Boot and Device Configuration Description (Continued)

XD Pin	Type	Value	Description
8	LEND		Little-endian mode select pin.
		0	System operates in big-endian mode.
		1	System operates in little-endian mode.
7-5	Reserved	–	Reserved configuration pins should be pulled-down.
4-0	BOOT-MODE	00000-11111	Determines the bootmode of the device, see Table 13.

6.2 Boot Processes

The boot process is determined by the boot configuration selected by the BOOTMODE[4−0] pins and described in Table 13. Up to three types of boot processes are available:

- **No boot process:** The CPU begins direct execution from the memory located at address 0. If SDRAM is used in the system, the CPU is held until SDRAM initialization is complete. Operation is undefined if invalid code is located at address 0.

- **ROM boot process**: The program located in external ROM is copied to address 0 by the DMA controller. Although the boot process begins when the device is released from external reset, this transfer occurs while the CPU is internally held in reset. This boot process also lets you choose the width of the ROM. In this case, the EMIF automatically assembles consecutive 8-bit bytes or 16-bit halfwords to form the 32-bit instruction words to be moved. These values are expected to be stored in little-endian format in the external memory, typically a ROM device.

  The transfer is automatically done by the DMA controller as a single-frame block transfer from the ROM to address 0. After completion of the block transfer, the CPU is removed from reset and allowed to run from address 0.

  The DMA controller copies 64K bytes from CE1 to address 0, using default ROM timings. After the transfer, the CPU begins executing from address 0.

- **Host boot process**: The XBUS can be used for the host boot. The type of host interface is determined by a set of latched signals during reset. The CPU is held in reset while the remainder of the device is released. During this period, an external host can initialize the CPU's memory space as necessary through the host interface, including internal configuration registers, such as those that control the EMIF or other peripherals. Once the host is finished with all necessary initialization, it must set the DSPINT to complete the boot process. This transition causes the boot configuration logic to remove the CPU from its reset state. The CPU then begins execution from address 0. The DSPINT condition is not latched by the CPU, because it occurs while the CPU is still in reset. Also, DSPINT wakes the CPU from internal reset only if the host boot process is selected. All memory may be written to and read by the host. This allows for the host to verify what it sends to the processor, if required.

7 Registers

Control of the XBUS and the peripheral interfaces is maintained through registers within the XBUS listed in Table 15. See the device-specific datasheet for the memory address of these registers.

The external master on the XBUS uses the XCNTL signal to select which internal register is being accessed. The state of this pin selects whether access is made to the XBUS internal slave address register (XBISA) or expansion bus data register (XBD). In addition to that, the external master has access to the entire memory map of the DSP, including memory-mapped registers.

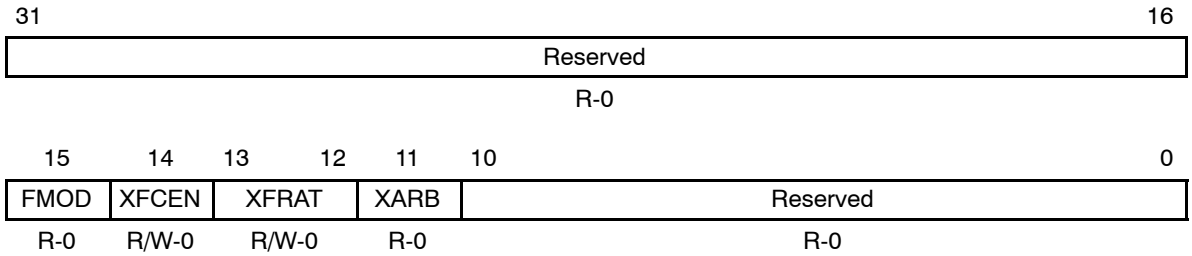
Table 15. Expansion Bus Registers

		Read/Write Access		
Acronym	**Register Name**	**DSP**	**Host**	**Section**
XBGC	Expansion bus global control register			7.1
XCECTL0-3	Expansion bus XCE space control registers			7.2
XBHC	Expansion bus host port interface control register	R/W	—	7.3
XBIMA	Expansion bus internal master address register	R/W	—	7.4
XBEA	Expansion bus external address register	R/W	—	7.5
XBD	Expansion bus data register	—	R/W	7.6
XBISA	Expansion bus internal slave address register	—	R/W	7.7

7.1 Expansion Bus Global Control Register (XBGC)

The expansion bus global control register (XBGC) configures parameters of the XBUS that are common to all interfaces. XBGC must not be modified while I/O port transactions are in progress. The XBGC is shown in Figure 24 and described in Table 16.

Figure 24. Expansion Bus Global Control Register (XBGC)



Legend: R = Read only; R/W = Read/Write; -*n* = value after reset

Table 16. Expansion Bus Global Control Register (XBGC) Field Descriptions

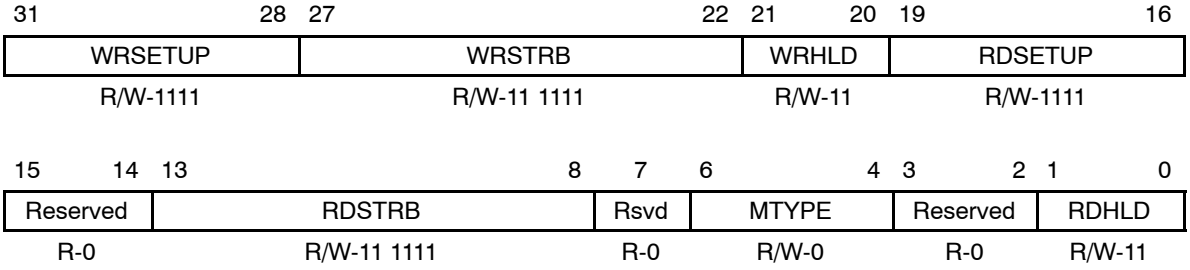
Bit	*field*†	*symval*†	Value	Description
31-16	Reserved	–	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.
15	FMOD			FIFO boot-mode selection bit.
		GLUE	0	Glue logic is used for FIFO read interface in all XCE spaces operating in FIFO mode.
		GLUELESS	1	Glueless read FIFO interface. If $\overline{\text{XCE3}}$ is selected for FIFO mode, then XOE acts as FIFO output enable and $\overline{\text{XCE3}}$ acts as FIFO read enable. XOE is disabled in all other XCE spaces regardless of MTYPE setting in XCECTL.
14	XFCEN			FIFO clock enable bit. The FIFO clock enable cannot be changed while a DMA request to XCE space is active.
		DISABLE	0	XFCLK is held high.
		ENABLE	1	XFCLK is enabled to clock.
13-12	XFRAT		0–3h	FIFO clock rate bits. The FIFO clock setting cannot be changed while a DMA request to XCE space is active. The XFCLK should be disabled before changing the XFRAT bits. There is no delay required between enabling/disabling XFCLK and changing the XFRAT bits.
		ONEEIGHTH	0	XFCLK = 1/8 CPU clock rate
		ONESIXTH	1h	XFCLK = 1/6 CPU clock rate
		ONEFOURTH	2h	XFCLK = 1/4 CPU clock rate
		ONEHALF	3h	XFCLK = 1/2 CPU clock rate
11	XARB			Arbitration mode select bit.
		DISABLE	0	Internal arbiter is disabled. DSP wakes up from reset as the bus slave.
		ENABLE	1	Internal arbiter is enabled. DSP wakes up from reset as the bus master.
10-0	Reserved	–	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.

† For CSL implementation, use the notation XBUS_XBGC_*field_symval*

7.2 Expansion Bus XCE Space Control Registers (XCECTL0–3)

The expansion bus XCE space control registers (XCECTL0–3) correspond to the four XCE memory spaces supported by the XBUS. The XCECTL is shown in Figure 25 and described in Table 17.

Figure 25. Expansion Bus XCE Space Control Register (XCECTL)



Legend: R = Read only; R/W = Read/Write; -*n* = value after reset

Table 17. Expansion Bus XCE Space Control Register (XCECTL) Field Descriptions

Bit	*field*†	*symval*†	Value	Description
31-28	WRSETUP	OF(*value*)	0-Fh	Write setup width. Number of CLKOUT1 cycles of setup time for byte-enable/address ($\overline{\text{XBE}}$/XA) and chip enable ($\overline{\text{XCE}}$) before write strobe falls. For asynchronous read accesses, this is also the setup time of $\overline{\text{XOE}}$ before $\overline{\text{XRE}}$ falls.
27-22	WRSTRB	OF(*value*)	0-3Fh	Write strobe width. The width of write strobe ($\overline{\text{XWE}}$) in CLKOUT1 cycles.
21-20	WRHLD	OF(*value*)	0-3h	Write hold width. Number of CLKOUT1 cycles that byte-enable/address ($\overline{\text{XBE}}$/XA) and chip enable ($\overline{\text{XCE}}$) are held after write strobe rises. For asynchronous read accesses, this is also the hold time of $\overline{\text{XCE}}$ after $\overline{\text{XRE}}$ rising.
19-16	RDSETUP	OF(*value*)	0-Fh	Read setup width. Number of CLKOUT1 cycles of setup time for byte-enable/address ($\overline{\text{XBE}}$/XA) and chip enable ($\overline{\text{XCE}}$) before read strobe falls. For asynchronous read accesses, this is also the setup time of $\overline{\text{XOE}}$ before $\overline{\text{XRE}}$ falls.
15-14	Reserved	–	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.

† For CSL implementation, use the notation XBUS_XCECTL_*field_symval*

Table 17. Expansion Bus XCE Space Control Register (XCECTL) Field Descriptions (Continued)

Bit	*field*†	*symval*†	Value	Description
13-8	RDSTRB	OF(*value*)	0-3Fh	Read strobe width. The width of read strobe ($\overline{\text{XRE}}$) in CLKOUT1 cycles.
7	Reserved	–	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.
6-4	MTYPE		0–7h	Memory type is configured during boot using pull-up or pull-down resistors on the expansion bus.
		–	0-1h	Reserved
		32BITASYN	2h	32-bit wide asynchronous interface
		–	3h-4h	Reserved
		32BITFIFO	5h	32-bit wide FIFO interface
		–	6h-7h	Reserved
3-2	Reserved	–	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.
1-0	RDHLD	OF(*value*)	0-3h	Read hold width. Number of CLKOUT1 cycles that byte-enable/address ($\overline{\text{XBE}}$/XA) and chip enable ($\overline{\text{XCE}}$) are held after read strobe rises. For asynchronous read accesses, this is also the hold time of $\overline{\text{XCE}}$ after $\overline{\text{XRE}}$ rising.

† For CSL implementation, use the notation XBUS_XCECTL_*field_symval*

7.3 Expansion Bus Host Port Interface Control Register (XBHC)

The expansion bus host port interface control register (XBHC) configures expansion bus host port parameters. The XBHC is shown in Figure 26 and described in Table 18.

The START bits are not cleared to 0 after a transfer is completed. Writing 00 to the the START bits while a transfer-in-progress is stalled by XRDY high, aborts the transfer. When a transfer is aborted, the XFRCT bits and the address registers, XBIMA and XBEA, reflect the state of the aborted transfer. Using this state information, the transfer can be restarted. Writing values other than 00 to the START bits is not recommended.

Figure 26. Expansion Bus Host Port Interface Control Register (XBHC)

31	16
XFRCT	
R/W-0	

15	6	5	4	3	2	1	0
Reserved		INTSRC	START		Rsvd	DSPINT	Rsvd
R-0		R/W-0	R/W-0		R-0	R/W-0	R-0

Legend: R = Read only; R/W = Read/Write; -*n* = value after reset

Table 18. Expansion Bus Host Port Interface Control Register (XBHC) Field Descriptions

Bit	*field*†	*symval*†	Value	Description
31-16	XFRCT	OF(*value*)	0–FFFFh	Transfer counter bits control the number of 32-bit words transferred between the expansion bus and an external slave when the CPU is mastering the bus (range of up to 64K).
15-6	Reserved	–	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.

† For CSL implementation, use the notation XBUS_XBHC_*field_symval*

Table 18. Expansion Bus Host Port Interface Control Register (XBHC) Field Descriptions (Continued)

Bit	*field*†	*symval*†	Value	Description
5	INTSRC			The interrupt source bit selects between the DSPINT bit of the expansion bus internal slave address register (XBISA) and the XFRCT counter. An XBUS host port interrupt can be caused either by the DSPINT bit or by the XFRCT counter.
		INTSRC	0	Interrupt source is the DSPINT bit of XBISA. When a zero is written to the INTSRC bit, the DSPINT bit of XBISA is copied to the DSPINT bit of XBHC.
		INTSRC	1	Interrupt is generated at the completion of the master transfer initiated by writing to the START bits.
4-3	START		0–3h	Start bus master transaction bit.
		ABORT	0	Writing 00 to the the START field while an active transfer is stalled by XRDY high, aborts the transfer. When a transfer is aborted, the XBUS registers reflect the state of the aborted transfer. Using this state information, you can restart the transfer.
		WRITE	1h	Starts a burst write transaction from the address pointed to by the expansion bus internal master address register (XBIMA) to the address pointed to by the expansion bus external address register (XBEA).
		READ	2h	Starts a burst read transaction from the address pointed to by the expansion bus external address register (XBEA) to the address pointed to by the expansion bus internal master address register (XBIMA).
		–	3h	Reserved
2	Reserved	–	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.

† For CSL implementation, use the notation XBUS_XBHC_*field_symval*

Table 18. Expansion Bus Host Port Interface Control Register (XBHC) Field Descriptions (Continued)

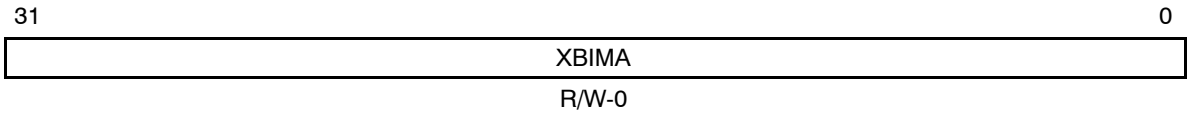
Bit	*field*†	*symval*†	Value	Description
1	DSPINT			The expansion bus to DSP interrupt (set either by the external host or the completion of a master transfer) is cleared when this bit is set. The DSPINT bit must be manually cleared before another one can be set.
		NONE	0	DSP interrupt bit is not cleared.
		CLR	1	DSP interrupt bit is cleared.
0	Reserved	–	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.

† For CSL implementation, use the notation XBUS_XBHC_*field_symval*

7.4 Expansion Bus Internal Master Address Register (XBIMA)

The expansion bus internal master address register (XBIMA) specifies the source or destination address in the DSP memory map where the transaction starts. XBIMA is set by the DSP when the DSP wants to initiate a transfer on the XBUS. Since all transfers have a width of one word, XBIMA is incremented by 4 after each transfer. XBIMA is used when the host port operates in synchronous mode. XBIMA is shown in Figure 27 and described in Table 19.

Figure 27. Expansion Bus Internal Master Address Register (XBIMA)



Legend: R/W = Read/Write; -*n* = value after reset

Table 19. Expansion Bus Internal Master Address Register (XBIMA) Field Descriptions

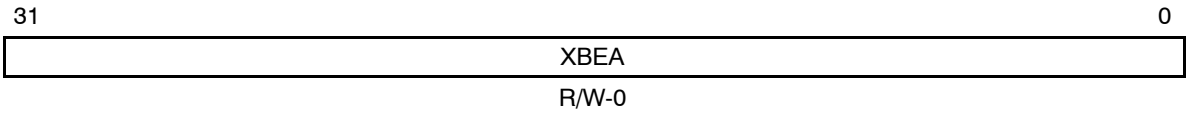
Bit	Field	*symval*†	Value	Description
31-0	XBIMA	OF(*value*)	0-FFFF FFFFh	Specifies the source or destination address in the DSP memory map where the transaction starts.

† For CSL implementation, use the notation XBUS_XBIMA_XBIMA_*symval*

7.5 Expansion Bus External Address Register (XBEA)

The expansion bus external address register (XBEA) specifies the source or destination address in the external slave memory map where the data is accessed. XBEA is set by the DSP when the DSP is ready to initiate a transfer on the XBUS. The content of XBEA appears on the XD[31−0] lines during the address phase of the transfer initiated by the DSP. Since all transfers have a width of one word, XBEA is incremented by 4 after each transfer. XBEA is used when the host port operates in synchronous mode. XBEA is shown in Figure 28 and described in Table 20.

Figure 28. Expansion Bus External Address Register (XBEA)



Legend: R/W = Read/Write; *-n* = value after reset

Table 20. Expansion Bus External Address Register (XBEA) Field Descriptions

Bit	Field	*symval*†	Value	Description
31-0	XBEA	OF(*value*)	0-FFFF FFFFh	Specifies the source or destination address in the external slave memory map where the data is accessed.

† For CSL implementation, use the notation XBUS_XBEA_XBEA_*symval*

7.6 Expansion Bus Data Register (XBD)

The expansion bus data register (XBD) contains the data that was read from the memory accessed by the XBUS host port, if the current access is a read. If the current access is a write, XBD contains the data that is written to the memory. The host can perform single 32-bit, 16-bit, or 8-bit accesses to XBD. Bursts longer than one word to XBD should always be 32-bits wide. XBD is used when the XBUS host port operates either in synchronous or asynchronous mode. XBD is shown in Figure 29 and described in Table 21.

Figure 29. Expansion Bus Data Register (XBD)

Legend: H = Host access; R/W = Read/Write; -*n* = value after reset

Table 21. Expansion Bus Data Register (XBD) Field Descriptions

Bit	Field	Value	Description
31-0	XBD	0-FFFF FFFFh	Contains the data that was read from the memory accessed by the XBUS host port, if the current access is a read; contains the data that is written to the memory, if the current access is a write.

7.7 Expansion Bus Internal Slave Address Register (XBISA)

The expansion bus internal slave address register (XBISA) is used when the external XBUS master initiates a data transfer. XBISA is shown in Figure 30 and described in Table 22. The XBSA bits, a 30-bit word address, control the memory location in the DSP memory map being accessed by the external mastering data transactions. The AINC bit is used by the host to enable or disable autoincrement of the XBSA bits. The DSPINT bit is used to trigger the interrupt (by setting the DSPINT bit). The host can only perform 32-bit accesses to XBISA.

XBISA is used when the host port operates either in synchronous or asynchronous mode. The DSP does not have access to the XBISA content. Burst transfers in the synchronous host-port mode are always expected to occur with autoincrement (AINC should be cleared to 0). In autoincrement mode (AINC = 0), operation is undefined if an external host attempts to access the last two word locations in the internal program/data RAM. This is because the DSP tries to prefetch data from reserved locations. Operation is also undefined if an external host attempts to cross a block boundary in a single DMA transfer. For more information, see the *Program and Data Memory Controller/Direct Memory Access (DMA) Controller Reference Guide*, SPRU577.

Figure 30. Expansion Bus Internal Slave Address Register (XBISA)

31	2	1	0
XBSA		AINC	DSPINT
HR/W-0		HR/W-0	HR/W-0

Legend: H = Host access; R/W = Read/Write; -*n* = value after reset

Table 22. Expansion Bus Internal Slave Address Register (XBISA) Field Descriptions

Bit	Field	Value	Description
31-2	XBSA	0-3FFF FFFFh	This 30-bit word address specifies the memory location in the DSP memory map being accessed by the host.
1	AINC		Autoincrement mode enable bit. (Asynchronous mode only)
		0	The expansion bus data register (XBD) is accessed with autoincrement of XBSA bits.
		1	The expansion bus data register (XBD) is accessed without autoincrement of XBSA bits.
0	DSPINT	0–1	The external master to DSP interrupt bit. Used to wake up the DSP from reset. The DSPINT bit is cleared by the corresponding DSPINT bit in the expansion bus host port interface control register (XBHC).

Revision History

Table 23 lists the changes made since the previous version of this document.

Table 23. Document Revision History

Page	Additions/Modifications/Deletions
56	Changed XD pin 7 to reserved in Table 14.

This page is intentionally left blank.

Index

N

O

R

S

T

W

X