TMS320C6000 DSP Host Port Interface (HPI) Reference Guide

Literature Number: SPRU578A
September 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 host-port interface (HPI) in the digital signal processors (DSPs) of the TMS320C6000™ DSP family that external processors use to access the memory space.

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.

TMS320C64x Technical Overview (SPRU395) gives an introduction to the TMS320C64x™ DSP and discusses the application areas that are enhanced by the TMS320C64x VelociTI™.

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

Contents

Figures

Tables

Host Port Interface (HPI)

This document describes the host port interface (HPI) in the digital signal processors (DSPs) of the TMS320C6000™ DSP family that external processors use to access the memory space.

1 Overview

The host port interface (HPI) is a parallel port through which a host processor can directly access the CPU memory space. The host device functions as a master to the interface, which increases ease of access. The host and CPU can exchange information via internal or external memory. The host also has direct access to memory-mapped peripherals. Connectivity to the CPU memory space is provided through the direct memory access (DMA) or enhanced DMA (EDMA) controller. Both the host and the CPU can access the HPI control register (HPIC). The host can access the HPI address register (HPIA), the HPI data register (HPID), and the HPIC by using the external data and interface control signals. For the C64x™ DSP, the CPU can also access the HPIA.

Through the HPI, an external host is capable of accessing the entire DSP memory map except the following:

- L2 control registers (C6x1x DSP only)
- Interrupt selector registers
- Emulation logic

Figure 1 is a block diagram of the C620x/C670x DSP; Figure 2 is a block diagram of the C621x/C671x/C64x DSP. Table 1 summarizes the differences between in the C6000™ DSP.

Figure 1. TMS320C620x/C670x DSP Block Diagram

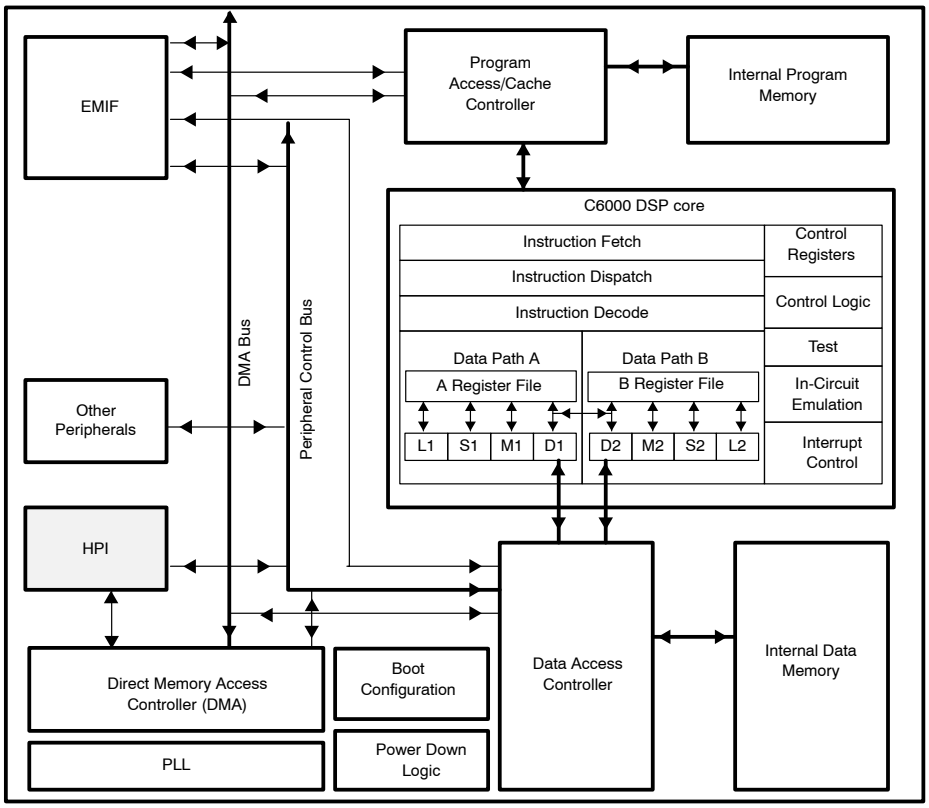


Figure 2. TMS320C621x/C671x/C64x DSP Block Diagram

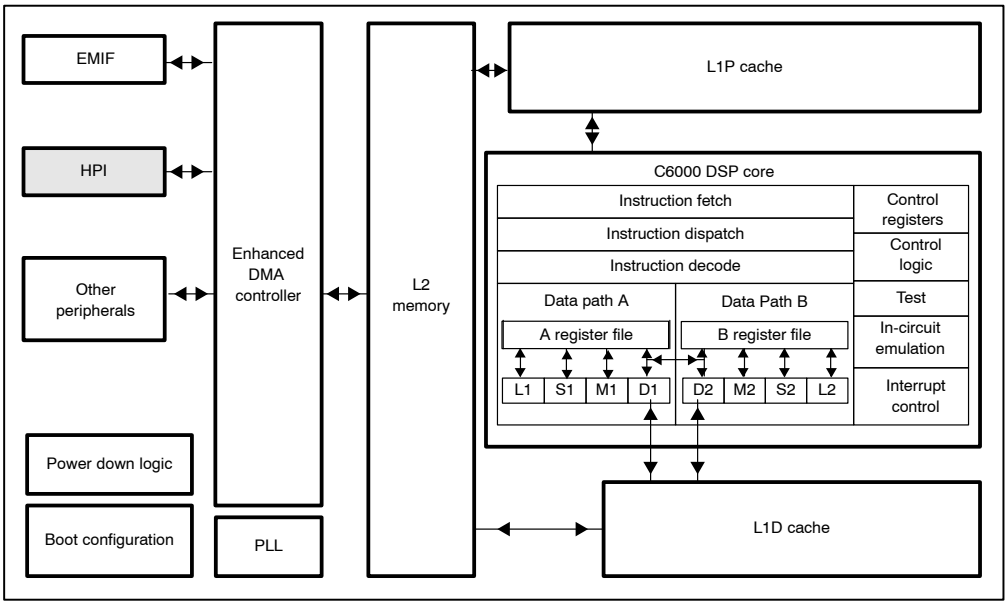


Table 1. Differences Between the C62x/C67x and C64x HPI

	C62x/C67x HPI		C64x HPI	
Features	**C620x/C670x**	**C621x/C671x**	**HPI16**	**HPI32**
Data bus width	16-bit	16-bit	16-bit	32-bit
Byte enable $\overline{\text{HBE}}$[1−0] pins	Yes	No	No	No
HHWIL	Used	Used	Used	Not used
Single halfword access support	Yes	No	No	No
HPIA access	By host only	By host only	By host or CPU. HPIA consists of HPIAR and HPIAW.	By host or CPU. HPIA consists of HPIAR and HPIAW.
$\overline{\text{HRDY}}$ operation	Not-ready after each word access	Not-ready only if internal read/write buffers not ready	Same as C621x/C671x HPI	Same as C621x/C671x HPI
Internal read buffer	No	Yes, 8-deep	Yes, 16-deep	Yes, 16-deep
Internal write buffer	No	Yes, 8-deep	Yes, 32-deep. Flushes after internal timer times out	Yes, 32-deep. Flushes after internal timer times out

2 HPI External Interface

The following sections discuss the external interface of the C6000 HPI. See section 3 for a detailed description of the interface signals.

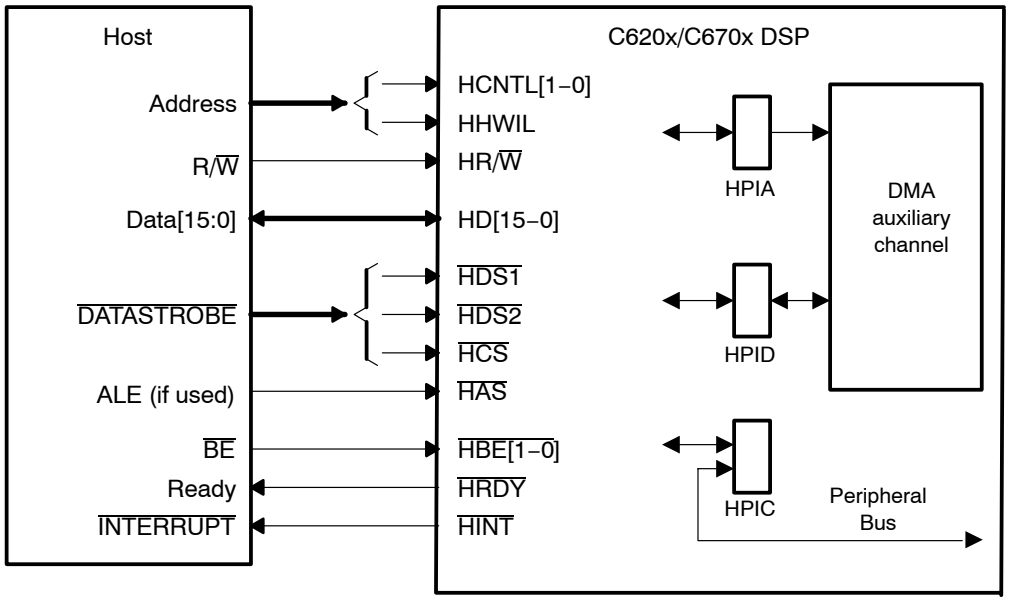
2.1 TMS320C620x/C670x HPI

Figure 3 is a simplified diagram of the C620x/C670x HPI.

The HPI provides 32-bit data to the CPU with an economical 16-bit external interface by automatically combining successive 16-bit transfers. When the host device transfers data through HPID, the DMA auxiliary channel accesses the CPU's address space.

The 16-bit data bus, HD[15−0], exchanges information with the host. Because of the 32-bit-word structure of the chip architecture, all transfers with a host consist of two consecutive 16-bit halfwords. On HPI data (HPID) write accesses, the $\overline{\text{HBE}}$[1−0] byte enables select the bytes to be written. For HPIA, HPIC, and HPID read accesses, the byte enables are not used. The dedicated HHWIL pin indicates whether the first or second halfword is being transferred. An internal control register bit determines whether the first or second halfword is placed into the most significant halfword of a word. For a full word access, the host must not break the first halfword/second halfword (HHWIL low/high) sequence of an ongoing HPI access.

Figure 3. C620x/C670x HPI Block Diagram



The two data strobes ($\overline{\text{HDS1}}$ and $\overline{\text{HDS2}}$), the read/write select (HR/$\overline{\text{W}}$), and the address strobe ($\overline{\text{HAS}}$) enable the HPI to interface to a variety of industry-standard host devices with little or no additional logic. The HPI can easily interface to hosts with a multiplexed or dedicated address/data bus, a data strobe and a read/write strobe, or two separate strobes for read and write.

The HCNTL[1–0] control inputs indicate which HPI register is accessed. Using these inputs, the host can specify an access to the HPIA (which serves as the pointer into the source or destination space), HPIC, or HPID. These inputs, along with HHWIL, are commonly driven directly by host address bus bits or a function of these bits. The host can interrupt the CPU by writing to the HPIC; the CPU can activate the $\overline{\text{HINT}}$ output to interrupt the host.

The host can access HPID with an optional automatic address increment of HPIA. This feature facilitates reading and writing to sequential word locations. In addition, during an HPID read with autoincrement, data is prefetched from the autoincremented address to reduce latency on the subsequent host read request.

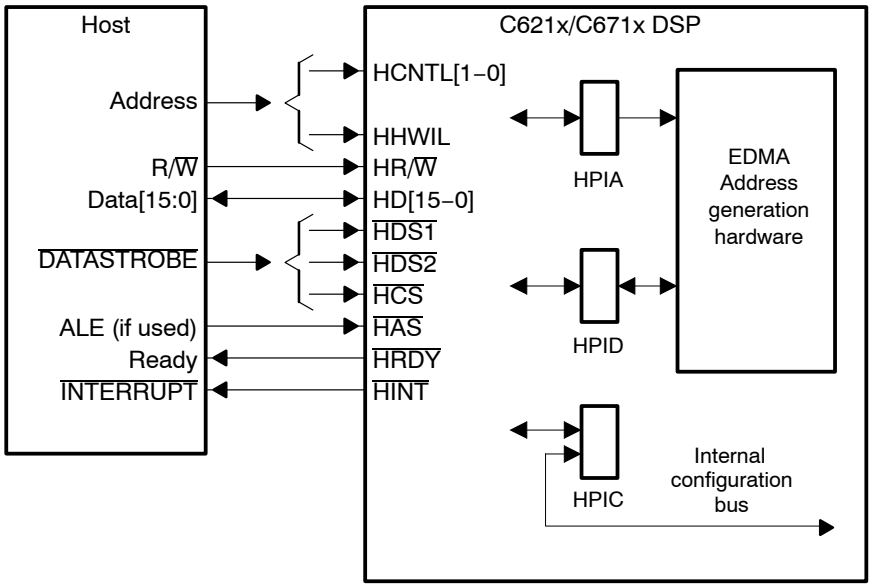
The HPI ready pin ($\overline{\text{HRDY}}$) allows insertion of host wait states. Wait states may be necessary, depending on latency to the point in the memory map accessed via the HPI, as well as on the rate of host access. The rate of host access can force not-ready conditions if the host attempts to access the host port before any previous HPID write access or prefetched HPID read access finishes. In this case, the HPI simply holds off the host via $\overline{\text{HRDY}}$. $\overline{\text{HRDY}}$ provides a convenient way to automatically adjust the host access rate to the rate of data delivery from the DMA auxiliary channel (no software handshake is needed). In the cases of hardware systems that cannot take advantage of the $\overline{\text{HRDY}}$ pin, an HRDY bit in the HPIC is available for use as a software handshake.

2.2 TMS320C621x/C671x HPI

The C621x/C671x pin interface is shown in Figure 4. All accesses through the 16-bit data bus HD[15–0] have to be in pairs.

Unlike the C620x HPI interface, which uses the DMA auxiliary channel to perform accesses, the C621x/C671x HPI ties directly into internal address generation hardware. No specific EDMA channel is used for performing C621x/C671x HPI accesses. Instead the internal address generation hardware, which is not visible to you, handles the read/write requests and accesses.

Figure 4. C621x/C671x HPI Block Diagram

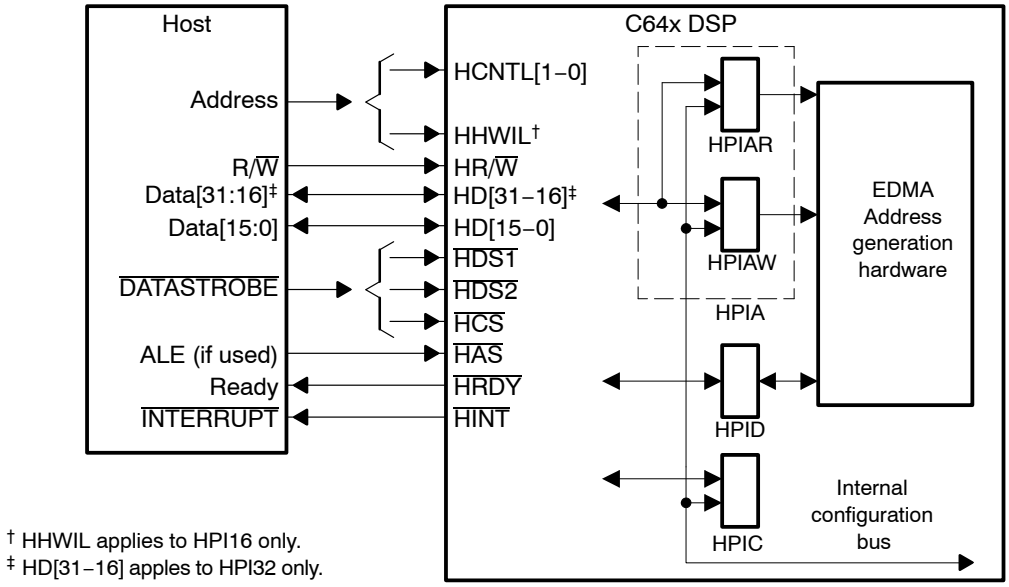


2.3 TMS320C64x HPI16 or HPI32

As shown in Figure 5, the C64x has 32 external data pins HD[31−0]. As a result, the C64x HPI supports either a 16-bit or 32-bit external pin interface. The C64x HPI is called the HPI16 when operating as a 16-bit-wide host port, and it is called the HPI32 when operating as a 32-bit-wide host port. The C64x selects either the HPI16 or the HPI32 via the boot and device configuration pins at reset.

The HPI16 is an enhanced version of the C621x/C671x HPI. The HPI16 provides 32-bit data to the CPU with a 16-bit external interface. In addition to all the C621x/C671x HPI functions, the HPI16 allows the DSP to access the HPI address register (HPIA). As shown in Figure 5, HPIA is separated into two registers, HPIA write (HPIAW) and HPIA read (HPIAR).

Figure 5. C64x HPI Block Diagram



The HPI32 has similar functions to the HPI16. The followings are the only differences between the HPI16 and the HPI32:

- **HHWIL Input:** The HHWIL input is used on the HPI16 to identify the first or second halfword of a word transfer. The HHWIL is not used on the HPI32, as all data transfers are performed in 32-bit words.

- **Data Bus Size:** As the name implies, the HPI16 has a 16-bit data bus. The HPI16 combines successive 16-bit transfers to provide 32-bit data to the CPU. For compatibility with other C6000 devices, the HPI16 uses HD[15−0] as data pins regardless of the endian mode selected at reset. The HPI32 has a 32-bit data bus. With this increased bus width, all transfers consist of one 32-bit word instead of two consecutive 16-bit halfwords. As a result, throughput is increased when the HPI operates in HPI32 mode.

3 Signal Descriptions

The external HPI interface signals implement a flexible interface to a variety of host devices. Table 2 lists the HPI pins and their functions. The remainder of this section discusses the pins in detail.

Table 2. HPI External Interface Signals

Signal Name	Signal Type†	Signal Count	Host Connection	Signal Function
HD[15−0] or HD[31−0]‡	I/O/Z	16 or 32‡	Data bus	
HCNTL[1−0]	I	2	Address or control lines	HPI access type control
HHWIL§	I	1	Address or control lines	Halfword identification input
$\overline{\text{HAS}}$	I	1	Address latch enable (ALE), address strobe, or unused (tied high)	Differentiation between address and data values on multiplexed address/data host
$\overline{\text{HBE[1−0]}}$¶	I	2	Byte enables	Data write byte enables
HR/$\overline{\text{W}}$	I	1	Read/write strobe, address line, or multiplexed address/data	Read/write select
$\overline{\text{HCS}}$	I	1	Address or control lines	Data strobe inputs
$\overline{\text{HDS[1−2]}}$	I	1 1	Read strobe and write strobe or data strobe	Data strobe inputs
$\overline{\text{HRDY}}$	O	1	Asynchronous ready	Ready status of current HPI access
$\overline{\text{HINT}}$	O	1	Host interrupt input	Interrupt signal to host

† I = input, O = output, Z = high impedance
‡ HD[31−16] applies to C64x or HPI32 only.
§ HHWIL does not apply to C64x HPI32.
¶ $\overline{\text{HBE[1−0]}}$ applies to C620x/C670x DSP only.

3.1 Data Bus: HD[15–0] or HD[31–0]

HD[15–0] or HD[31–0] is a parallel, bidirectional, 3-state data bus. HD is placed in the high-impedance state when it is not responding to an HPI read access. Pins HD[31:16] apply to the C64x HPI32 only. See section 2.3.

3.2 Access Control Select: HCNTL[1–0]

HCNTL[1–0] indicate which internal HPI register is being accessed. The states of these two pins select access to the HPI address (HPIA), HPI data (HPID), or HPI control (HPIC) registers. Additionally, HPID can be accessed with an optional automatic address increment. Table 3 describes the HCNTL[1–0] bit functions.

Table 3. HPI Input Control Signals Function Selection

HCNTL1	HCNTL0	Description
0	0	Host reads from or writes to the HPI control register (HPIC).
0	1	Host reads from or writes to the HPI address register (HPIA).
1	0	Host reads or writes to the HPI data register (HPID) in autoincrement mode. The HPI address register (HPIA) is postincremented by a word address (four byte addresses).
1	1	Host reads or writes to the HPI data register (HPID) in fixed address mode. HPI address register (HPIA) is not affected.

3.3 Halfword Identification Select: HHWIL

HHWIL identifies the first or second halfword of a transfer, but not the most-significant or least-significant halfword. The status of the HWOB bit in HPIC determines which halfword is least significant or most significant. HHWIL is low for the first halfword and high for the second halfword.

Since byte enable pins $\overline{\text{HBE}}$[1–0] are removed from the C621x/C671x HPI and C64x HPI, HHWIL in combination with HWOB specify the half-word position in the HPI data register (HPID). This is shown in Table 4 along with the LSB address bits depending on endianness. HHWIL does not apply to the C64x HPI32.

Table 4. HPI Data Write Access

Data-Type Little-Endian (LE)/ Big-Endian (BE)	HWOB bit	First Write (HHWIL=0) / Logical LSB Address Bits	Second Write (HHWIL=1) / Logical LSB Address Bits
Halfword: Little endian (LE) Big endian (BE)	0	MS halfword LE = 10 BE = 00	LS halfword LE = 00 BE = 10
Halfword: Little endian (LE) Big endian (BE)	1	LS halfword LE = 00 BE = 10	MS halfword LE = 10 BE = 00
Word: Little endian (LE) Big endian (BE)	0	MS halfword LE = 00 BE = 00	LS halfword LE = 00 BE = 00
Word: Little endian (LE) Big endian (BE)	1	LS halfword LE = 00 BE = 00	MS halfword LE = 00 BE = 00

3.4 Address Strobe Input: $\overline{\text{HAS}}$

$\overline{\text{HAS}}$ allows HCNTL[1–0], HR/$\overline{\text{W}}$, and HHWIL to be removed earlier in an access cycle, which allows more time to switch bus states from address to data information. This feature facilitates interface to multiplexed address and data buses. In this type of system, an address latch enable (ALE) signal is often provided and is normally the signal connected to $\overline{\text{HAS}}$.

Hosts with a multiplexed address and data bus connect $\overline{\text{HAS}}$ to their ALE pin or an equivalent pin. HHWIL, HCNTL[1–0], and HR/$\overline{\text{W}}$ are latched on the falling edge of $\overline{\text{HAS}}$. When used, $\overline{\text{HAS}}$ must precede the latest of $\overline{\text{HCS}}$, $\overline{\text{HDS1}}$, or $\overline{\text{HDS2}}$. Hosts with separate address and data buses can tie $\overline{\text{HAS}}$ high. In this case, HHWIL, HCNTL[1–0], and HR/$\overline{\text{W}}$ are latched by the latest falling edge of $\overline{\text{HDS1}}$, $\overline{\text{HDS2}}$, or $\overline{\text{HCS}}$ while $\overline{\text{HAS}}$ stays inactive (high).

3.5 Byte Enables: $\overline{\text{HBE}}$[1–0] (C620x/C670x DSP only)

On HPID writes, the value of $\overline{\text{HBE}}$[1:0] indicates which bytes of the 32-bit word are written. On HPID writes, $\overline{\text{HBE}}$0 enables the least significant byte in the halfword and $\overline{\text{HBE}}$1 enables the most significant byte in the halfword. Table 5 lists the valid combinations of byte enables. For byte writes, only one $\overline{\text{HBE}}$ in either of the halfword accesses can be enabled. For halfword data writes, both the $\overline{\text{HBE}}$s must be held active(low) in either (but not both) halfword access. For word accesses, both $\overline{\text{HBE}}$s must be held active (low) in both halfword accesses. No other combinations are valid. The selection of byte enables and the endianness of the CPU (selected via the LENDIAN pin) determine the logical address implied by the access.

The HPI only performs word reads. Therefore, the byte enables $\overline{\text{HBE}}$[1–0] are don't care during a host read access.

Table 5. Byte Enables for HPI Data Write Access (C620x/C670x HPI only)

		$\overline{\text{HBE}}$[1–0]			
	HWOB = 0	First Write HHWIL = 0	Second Write HHWIL = 1	Effective Logical Address LSBs (Binary)	
Data Write Type	HWOB = 1	Second Write HHWIL = 1	First Write HHWIL = 0	Little Endian	Big Endian
Byte		11	10	00	11
Byte		11	01	01	10
Byte		10	11	10	01
Byte		01	11	11	00
Halfword		11	00	00	10
Halfword		00	11	10	00
Word		00	00	00	00

3.6 Read/Write Select: HR/$\overline{W}$

HR/$\overline{W}$ is the host read/write select input. The host must drive HR/$\overline{W}$ high to read and low to write HPI. A host without either a read/write select output or a read or write strobe can use an address line for this function.

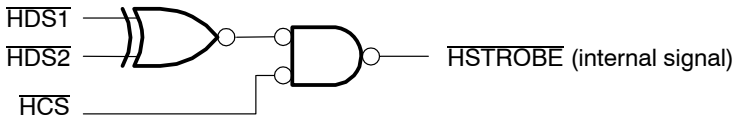
3.7 Strobes: $\overline{HCS}$, $\overline{HDS1}$, $\overline{HDS2}$

$\overline{HCS}$, $\overline{HDS1}$, and $\overline{HDS2}$ allow connection to a host that has either:

- A single strobe output with read/write select (HR/$\overline{W}$)
- Separate read and write strobe outputs. In this case, read or write select can be done by using different addresses.

Figure 6 shows the equivalent circuit of the $\overline{HCS}$, $\overline{HDS1}$, and $\overline{HDS2}$ inputs.

Figure 6. Select Input Logic



Used together, $\overline{HCS}$, $\overline{HDS1}$, and $\overline{HDS2}$ generate an active (low) internal $\overline{HSTROBE}$ signal. $\overline{HSTROBE}$ is active (low) only when both $\overline{HCS}$ is active and either (but not both) $\overline{HDS1}$ or $\overline{HDS2}$ is active. The falling edge of $\overline{HSTROBE}$ when $\overline{HAS}$ is tied inactive (high) samples HCNTL[1–0], HHWIL, and HR/$\overline{W}$. Therefore, the latest of $\overline{HDS1}$, $\overline{HDS2}$, or $\overline{HCS}$ controls the sampling time. $\overline{HCS}$ serves as the enable input for the HPI and must be low during an access. However, because the $\overline{HSTROBE}$ signal determines the actual boundaries between accesses, $\overline{HCS}$ can stay low between successive accesses as long as both $\overline{HDS1}$ and $\overline{HDS2}$ transition appropriately.

Hosts with separate read and write strobes connect these strobes to either $\overline{HDS1}$ or $\overline{HDS2}$. Hosts with a single data strobe connect it to either $\overline{HDS1}$ or a $\overline{HDS2}$, tying the unused pin high. Regardless of $\overline{HDS1}$ and $\overline{HDS2}$ connections, HR/$\overline{W}$ is required to determine the direction of transfer. Because $\overline{HDS1}$ and $\overline{HDS2}$ are internally exclusive-NORed, hosts with an active high data strobe can connect this strobe to either $\overline{HDS1}$ or $\overline{HDS2}$ with the other signal tied low.

HSTROBE is used for four purposes:

- On a read, the falling edge of HSTROBE initiates HPI read accesses for all access types.
- On a write, the rising edge of HSTROBE initiates HPI write accesses for all access types.
- The falling edge latches the HPI control inputs, including HHWIL, HR/W, and HCNTL[1–0]. HAS also affects latching of control inputs. See section 3.4 for a description of HAS.
- The rising edge of HSTROBE latches the HBE[1–0] input (C620x/C670x DSP only) as well as the data to be written.

HCS gates the HRDY output. In other words, a not-ready condition is indicated by the HRDY pin being driven high only if HCS is active (low). Otherwise HRDY is active (low).

3.8 Ready: HRDY

When active (low), HRDY indicates that the HPI is ready for a transfer to be performed. When inactive, HRDY indicates that the HPI is busy completing the internal portion of a current read access or a previous HPID read prefetch or write access. HCS enables HRDY; HRDY is always low when HCS is high.

3.9 Interrupt to Host: HINT

HINT is the host interrupt output that is controlled by the HINT bit in the HPIC. This signal is described in more detail in section 8.3.3. The HINT bit is set to 0 when the chip is being reset. Thus, the HINT pin is high at reset.

4 HPI Bus Access

4.1 HPI Bus Access for C620x/C670x HPI

HPI access timing in different cases for the C620x/C670x HPI are shown in Figure 7, Figure 8, Figure 9, and Figure 10. $\overline{\text{HSTROBE}}$ represents the internally generated strobe described in Figure 6. Control signals HCNTL[1−0], HR/$\overline{\text{W}}$, HHWIL, and $\overline{\text{HBE}}$[1−0] are inputs typically driven by the host. HCNTL[1−0] and HR/$\overline{\text{W}}$ should have the same values for both halfword accesses. HHWIL must be low for the first halfword transfer and high for the second. For correct operation, the host should monitor and detect HRDY low before any HPI transfer (this includes HPID, HPIA, and HPIC accesses).

4.1.1 Latching Control Signals

The control signals are latched differently, depending upon whether $\overline{\text{HAS}}$ is used. If $\overline{\text{HAS}}$ is tied high and not used (Figure 7 and Figure 9), the falling edge of $\overline{\text{HSTROBE}}$ latches the control signals. If $\overline{\text{HAS}}$ is used (Figure 8 and Figure 10), the falling edge of $\overline{\text{HAS}}$ latches these control signals. In the latter case, the falling edge of $\overline{\text{HAS}}$ must precede the falling edge of $\overline{\text{HSTROBE}}$.

4.1.2 HPID Read

On a read, data is valid at some time after the falling edge of $\overline{\text{HSTROBE}}$. If valid data is not already present in the HPID, $\overline{\text{HRDY}}$ goes not-ready (high). Once data is available, $\overline{\text{HRDY}}$ goes ready (low). Data is set up at the falling edge of $\overline{\text{HRDY}}$ and held until the rising edge of $\overline{\text{HSTROBE}}$. Therefore the host should not end a read cycle ($\overline{\text{HSTROBE}}$ rising edge) until $\overline{\text{HRDY}}$ is detected ready (low).

$\overline{\text{HRDY}}$ goes not-ready (high) in one of the following conditions:

- After $\overline{\text{HCS}}$ falling edge, if prefetch from the previous autoincrement address mode access has not completed (Case 1 in Figure 7 and Figure 8).
- After $\overline{\text{HSTROBE}}$ falling edge, if $\overline{\text{HAS}}$ is not used and valid data is not in HPID (Case 2 in Figure 7 and Figure 8).
- After $\overline{\text{HAS}}$ falling edge, if $\overline{\text{HAS}}$ is used, and valid data is not in HPID (Case 2 in Figure 7 and Figure 8).
- After $\overline{\text{HSTROBE}}$ rising edge, if the read access is in autoincrement address mode.

Fixed Address Mode HPID Read (HCNTL[1−0] = 11b)

For a fixed address mode read, the HPI sends the read request to the DMA auxiliary channel and HRDY becomes not-ready (high). HRDY remains not-ready (high) until the requested data is loaded into HPID. Since the DMA auxiliary channel performs word reads, the data is already present in the HPID at the beginning of the second halfword read access. Thus, the second halfword HPID read never encounters a not-ready condition, and HRDY remains low.

Autoincrement Address Mode HPID Read (HCNTL[1−0] = 10b)

For the first autoincrement address mode read, the HPI sends the read request to the DMA auxiliary channel and HRDY goes not-ready, as shown in Case 2 in Figure 7 and Figure 8. The data pointed to by the next address is fetched immediately upon completion of the current read. HRDY becomes not-ready (high) when the HPI is busy pre-fetching data.

4.1.3 HPID Write

On a write, the host must set up data and HBE[1−0] on the rising edge of HSTROBE. The C620x/C670x HPI provides 32-bit data to the CPU through a 16-bit external interface by automatically combining two successive halfword transfers.

During an HPID write access, two halfword portions of the HPID are transferred from the host. At the end of this write access, (with the second rising edge of HSTROBE), the contents of HPID are transferred as a 32-bit word to the address specified by HPIA.

The host should not end (with an HSTROBE rising edge) a write cycle until HRDY is detected ready (low). HRDY goes not-ready (high) in one of the following conditions:

- After HCS falling edge, if the previous write access has not yet completed
- After HSTROBE rising edge to service the HPID write

See Figure 9 and Figure 10 for details on HPID write.

4.1.4 HPIC or HPIA Access

For correct operation, an HPIC or HPIA write must occur before switching from HPID read to HPID write, or conversely. The host should monitor and detect HRDY low before reading or writing to the HPIA/HPIC.

Figure 7. HPI Read Timing ($\overline{HAS}$ Not Used, Tied High)

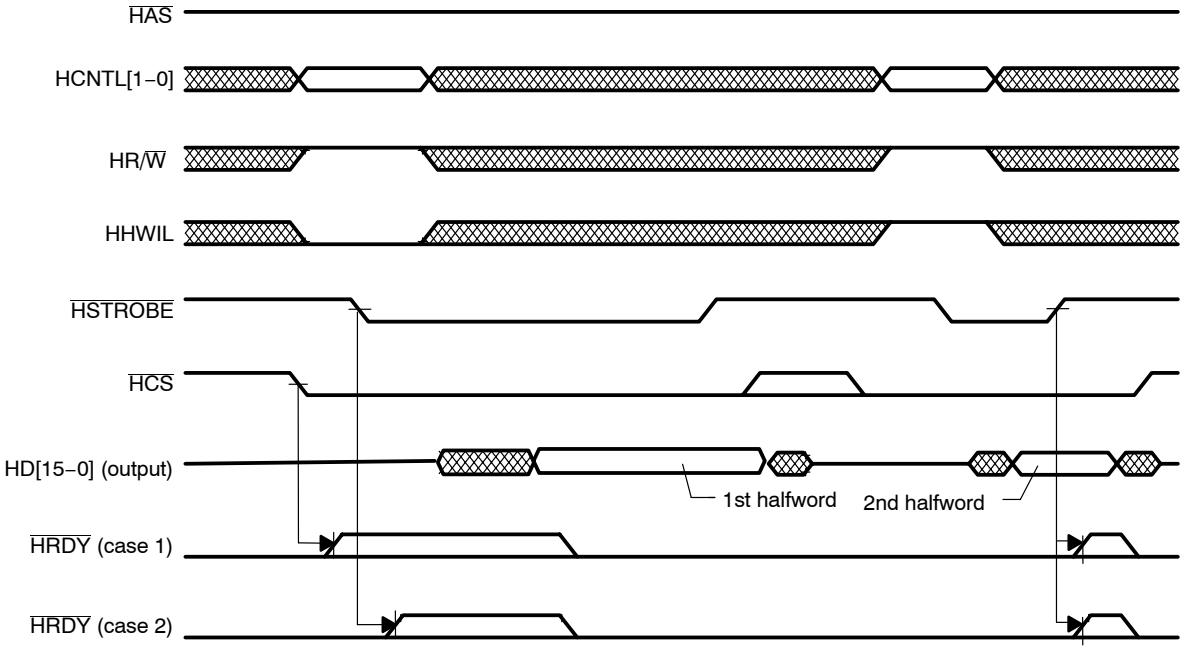
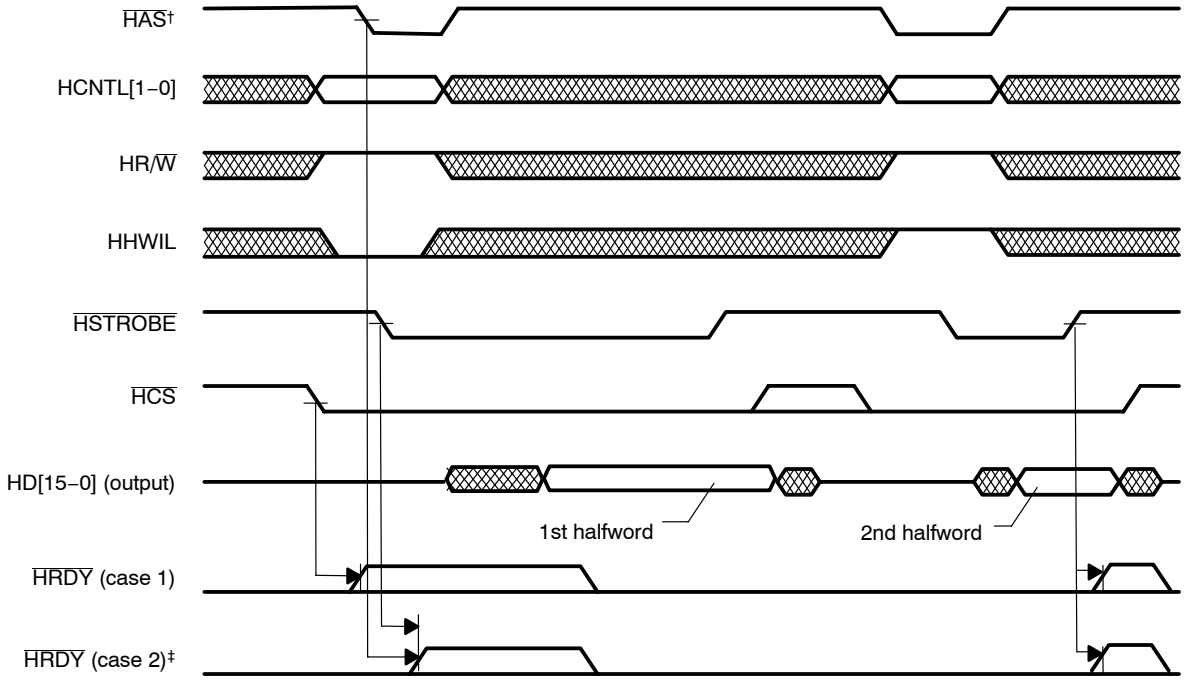


Figure 8. HPI Read Timing (HAS Used)



† For correct operation, strobe the HAS signal only once per HSTROBE cycle.
‡ For C620x/C670x DSP, if HAS is used, HRDY goes not ready after HAS falling edge. For all other devices, HRDY goes not-ready after HSTROBE falling edge, even if HAS is used.

Figure 9. HPI Write Timing ($\overline{HAS}$ Not Used, Tied High)

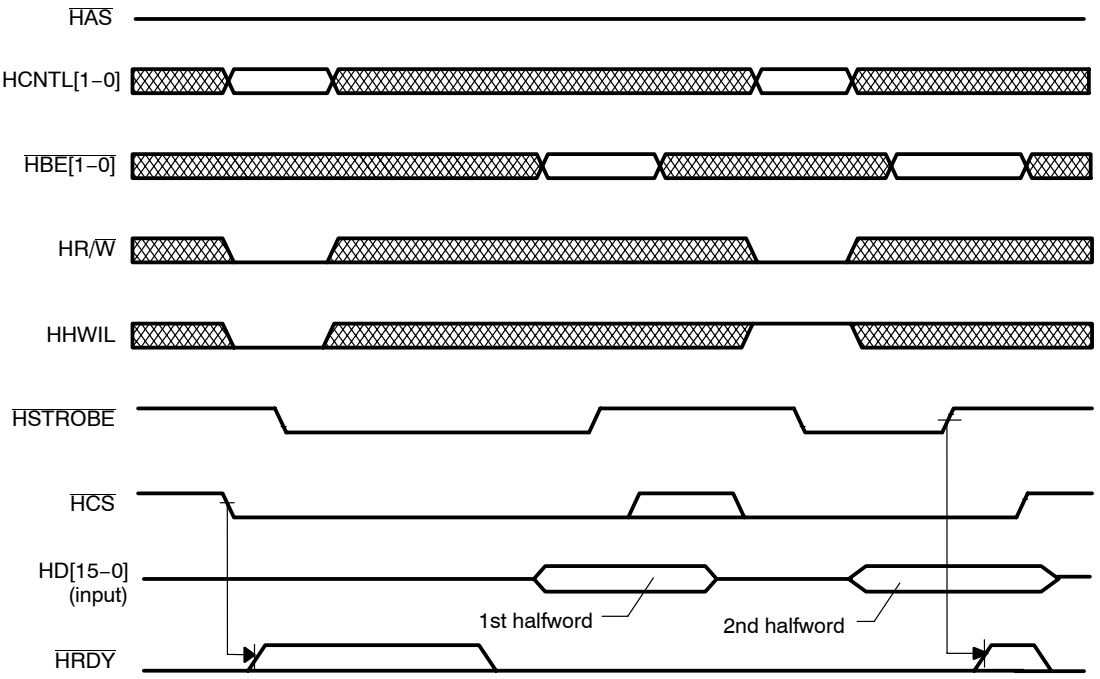
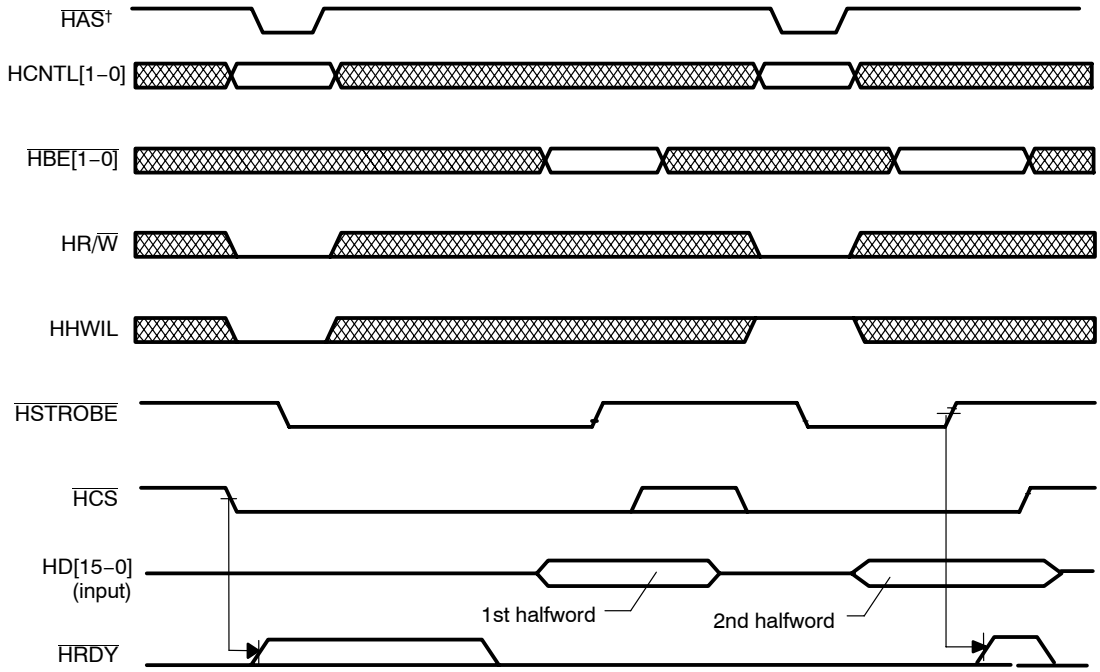


Figure 10. HPI Write Timing (HAS Used)



† For correct operation, strobe the HAS signal only once per HSTROBE cycle.

4.2 HPI Bus Access for C621x/C671x HPI

The C621x/C671x HPI shares the same bus interface as the C620x/C670x HPI. In addition, the C621x/C671x HPI has internal read and write buffers to improve throughput in both read and write accesses. For correct operation, the host should monitor and detect HRDY low before any HPI transfer (this includes HPID, HPIA, and HPIC accesses).

4.2.1 Latching Control Signals

Similar to the C620x/C670x HPI, the C621x/C671x HPI latches control signals differently depending upon whether HAS is used. See section 4.1.1.

4.2.2 HPID Read

Fixed Address Mode HPID Read (HCNTL[1–0] = 11b)

For a fixed address mode read, the HPI sends the read request to the EDMA internal address generation hardware and $\overline{\text{HRDY}}$ becomes not-ready (high). This is shown in Case 2 in Figure 7 and Figure 8. $\overline{\text{HRDY}}$ remains not-ready until the requested data is loaded into HPID. Since the EDMA internal address generation hardware performs word reads, the data is already present in the HPID at the beginning of the second halfword read access. Thus, the second halfword HPID read never encounters a not-ready condition, and $\overline{\text{HRDY}}$ remains low.

Autoincrement Address Mode HPID Read (HCNTL[1–0] = 10b)

The C621x/C671x HPI has an internal read buffer that helps to improve throughput in autoincrement mode. For the first host read of the HPID, the HPI deasserts $\overline{\text{HRDY}}$ high (not-ready) after $\overline{\text{HSTROBE}}$ falling edge, as shown in Case 2 in Figure 7 and Figure 8. During this time, the EDMA internal address generation hardware begins fetching enough words to fill the internal read buffer. As soon as the first valid data is ready, the HPI presents it on the HD bus and $\overline{\text{HRDY}}$ goes ready. If the next data is already present in the internal read buffer when the current read finishes, $\overline{\text{HRDY}}$ remains ready at the beginning of the next host read ($\overline{\text{HSTROBE}}$ falling edge). The HPI continues to perform subsequent data fetches and places the data in the internal read buffer before the actual host read occurs. This reduces the subsequent host read data access time. A read cycle is terminated by a host write to the HPIA or HPIC, at which point the HPI automatically empties the internal read buffer, and $\overline{\text{HRDY}}$ goes not-ready. Note that a read cycle does not have to be terminated immediately when a read burst completes. The normal process of reinitializing the HPIA register for a new address range will force the internal read buffer to flush in anticipation of a new read command.

For all modes of HPI read, the host should not end ($\overline{\text{HSTROBE}}$ rising edge) a read cycle until $\overline{\text{HRDY}}$ is detected ready (low). $\overline{\text{HRDY}}$ goes not-ready (high) in one of the following conditions:

- After $\overline{\text{HCS}}$ falling edge
- After $\overline{\text{HSTROBE}}$ falling edge for first halfword transfer (HHWIL low)
- After $\overline{\text{HSTROBE}}$ rising edge for second halfword transfer (HHWIL high).

4.2.3 HPID Write

On a write, the host must set up data on the rising edge of $\overline{\text{HSTROBE}}$. All C621x/C671x HPI writes must consist of two successive halfword transfers.

Fixed Address Mode HPID Write (HCNTL[1–0] = 11b)

This operation is identical to the C620x/C670x HPI fixed address mode HPID write. See section 4.1.3.

Autoincrement Address Mode HPID Write (HCNTL[1–0] = 10b)

The C621x/C671x HPI has an internal write buffer that helps to improve throughput in autoincrement mode. At the end of a word write in autoincrement mode, the data is copied from the HPID to the internal write buffer to wait for service by the EDMA internal address generation hardware. The DSP does not actually service this host write until the internal write buffer is half full, or if the write cycle is terminated. A write cycle is terminated by a host access to the HPIA or the HPIC, at which point the DSP services the HPI by transferring all remaining elements from the internal write buffer to their destinations.

Since the data is copied to the internal write buffer, the HPID is immediately ready for the next data write from the host. Thus under normal conditions, $\overline{\text{HRDY}}$ remains ready at the beginning of the next host write access ($\overline{\text{HSTROBE}}$ active).

The host should not end (with an $\overline{\text{HSTROBE}}$ rising edge) a write cycle until $\overline{\text{HRDY}}$ is detected ready (low). $\overline{\text{HRDY}}$ goes not-ready (high) in one of the following conditions:

- After $\overline{\text{HCS}}$ falling edge
- After $\overline{\text{HSTROBE}}$ rising edge for second halfword transfer (HHWIL high)
- After $\overline{\text{HSTROBE}}$ falling edge for first halfword transfer (HHWIL low) if the internal write buffer is full.

4.2.4 HPIC or HPIA Access

For correct operation, an HPIC or HPIA write must occur before switching from HPID read to HPID write, or conversely. An HPIC or HPIA register write terminates a burst read/write access in autoincrement mode. $\overline{\text{HRDY}}$ may go not-ready while the internal read buffer is being emptied (for reads), or while the internal write buffer is being serviced by the DSP (for writes).

4.3 HPI Bus Access for C64x HPI

With a 32-bit data bus, the C64x HPI is an enhanced version of the C621x/C671x HPI. The C64x HPI can be configured at reset to operate in either HPI16 or HPI32 mode.

The HPI16 operation is similar to the C621x/C671x 16-bit HPI. See section 4.2. In addition to the operation described in section 4.2, the C64x HPI internal write buffer will flush when an internal timer times out after 128 CPU clock cycles.

The HPI32 operation is similar to the HPI16 operation, with exceptions due to the expanded 32-bit data bus. Since the HPI32 data bus is expanded to 32 bits, all read and write transfers consist of one 32-bit word access (HD[0–31]) instead of two consecutive 16-bit halfword accesses. Also, the HHWIL signal is not used on the HPI32 interface. The HPI32 read and write timings are shown in Figure 11, Figure 12, Figure 13, and Figure 14. The remaining HPI32 operations are identical to the HPI16.

When performing reads with autoincrement, the C64x HPI differs slightly from the C621x/C671x HPI in that the C64x HPI does not indicate ready ($\overline{\text{HRDY}}$ low) until the internal read buffer has filled with the 16-word prefetch. Thus, accesses to slow regions of memory, such as internal peripheral registers or slow external memory, might take a significant amount of time. For best performance, accesses to these regions should be done in fixed mode unless multiple words are desired.

Figure 11. HPI32 Read Timing ($\overline{HAS}$ Not Used, Tied High) for C64x HPI

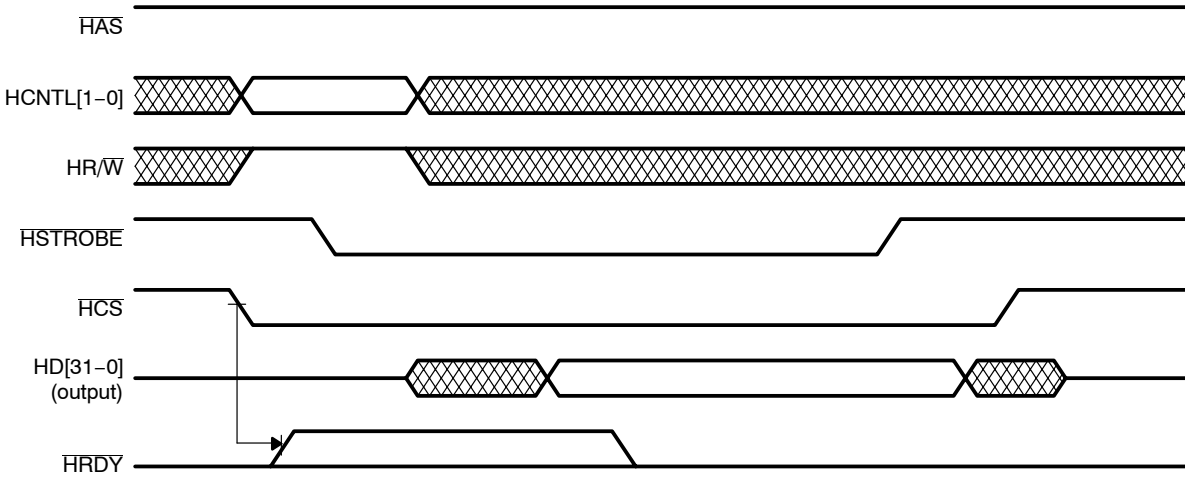
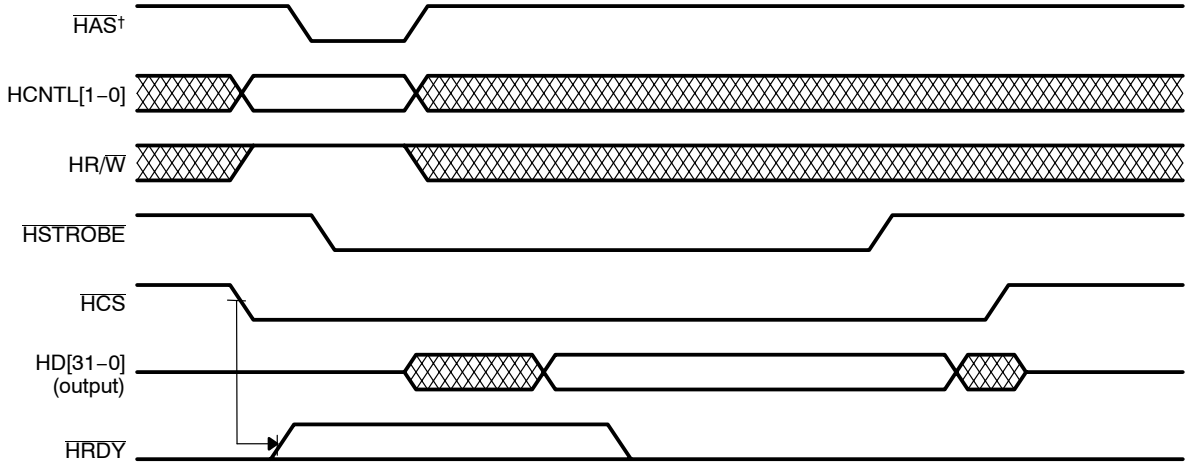


Figure 12. HPI32 Read Timing ($\overline{HAS}$ Used) for C64x HPI



† For correct operation, strobe the $\overline{HAS}$ signal only once per $\overline{HSTROBE}$ cycle.

Figure 13. HPI32 Write Timing ($\overline{HAS}$ Not Used, Tied High) for C64x HPI

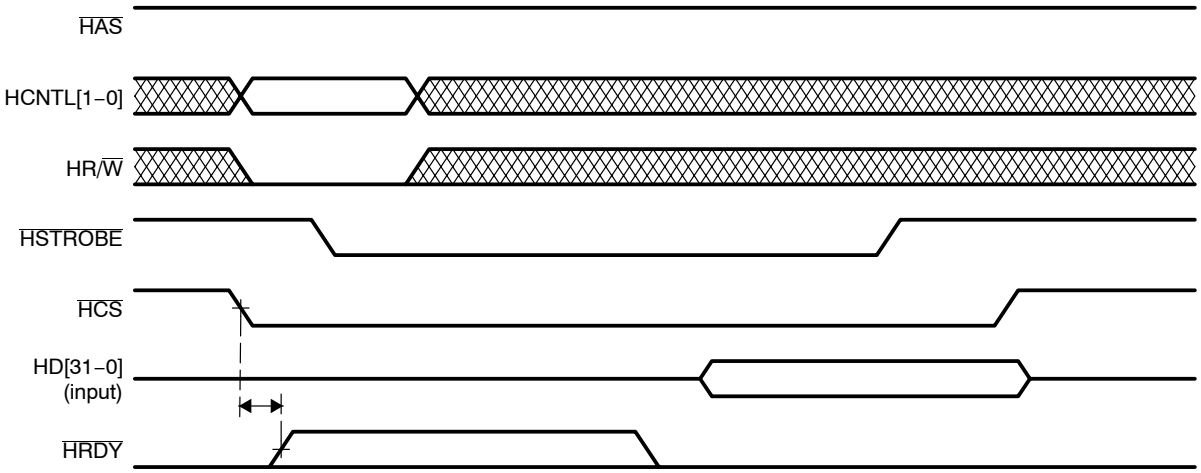
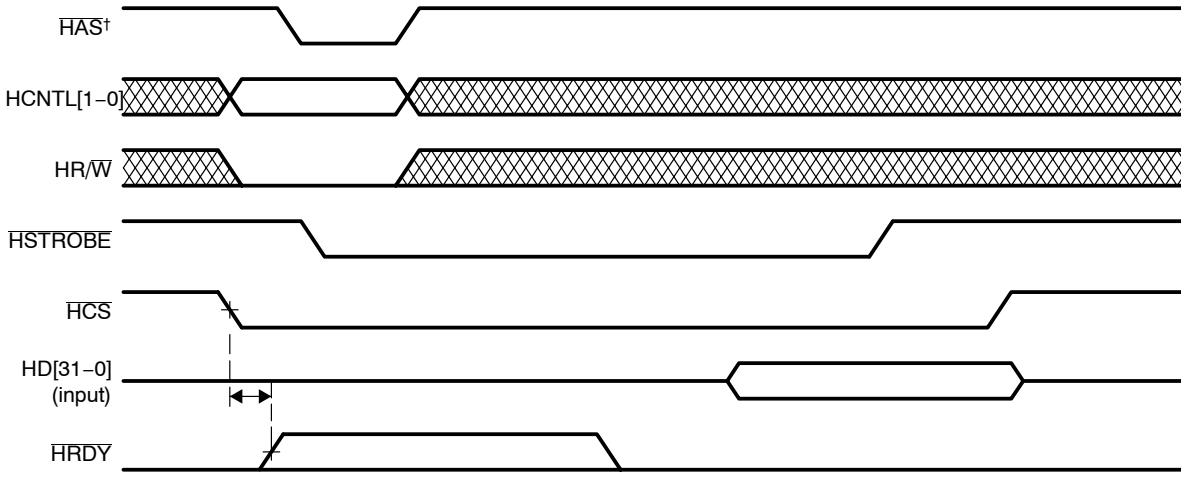


Figure 14. HPI32 Write Timing ($\overline{HAS}$ Used) for C64x HPI



† For correct operation, strobe the $\overline{HAS}$ signal only once per $\overline{HSTROBE}$ cycle.

5 Host Access Sequences

The host begins HPI accesses by performing the following tasks in this order:

1) Initializing the HPI control register (HPIC)
2) Initializing the HPI address register (HPIA)
3) Writing data to or reading data from the HPI data register (HPID)

Reading from or writing to HPID initiates an internal cycle that transfers the desired data between HPID and the DMA auxiliary channel in the C620x/C670x DSP or the internal address generation hardware in the C621x/C671x/C64x DSP. For the 16-bit HPI, host access of any HPI register requires two halfword accesses on the HPI bus: the first with HHWIL low and the second with HHWIL high. Typically, the host must not break the first halfword/second halfword (HHWIL low/high) sequence. If this sequence is broken, data can be lost, and undesired operation can result. The first halfword access may have to wait for a previous HPI request to finish. Previous requests include HPID writes and prefetched HPID reads. Thus, the HPI deasserts $\overline{\text{HRDY}}$ (drives $\overline{\text{HRDY}}$ high) until the HPI can begin this request. The second halfword access always has $\overline{\text{HRDY}}$ active because all previous accesses have been completed for the first halfword access. The C64x HPI32 combines two halfword transfers into a single-word transfer.

5.1 Initialization of HPIC and HPIA

Before any data access, the HPIC and HPIA must be initialized. On the C62x/C67x DSP, only the host has access to HPIA. On the C64x DSP, either the host or the CPU can be used to initialize HPIC and HPIA. The following sections discuss the host initialization sequence for the 16-bit-wide host port (C62x/C67x HPI and C64x HPI16) and the 32-bit-wide host port (HPI32).

5.1.1 Initialization of HPIC and HPIA (C62x/C67x HPI and C64x HPI16)

Before accessing data, the HWOB bit of HPIC and the HPIA register must be initialized in this order (because the HWOB bit affects the HPIA access). After initializing the HWOB bit, the host (or the C64x CPU) can write to HPIA with the correct halfword alignment. Table 6 and Table 7 summarize the initialization sequence for HWOB = 1 and HWOB = 0, respectively. In these examples, HPIA is set to 8000 1234h. In all these accesses, the HRDY bits in HPIC are set. A question mark in these tables indicates that the value is unknown.

Table 6. Initialization of HPIC and HPIA With HWOB = 1

Event	Value During Access					Value After Access		
	HD	$\overline{\text{HBE[1–0]}}$	HR/$\overline{\text{W}}$	HCNTL[1–0]	HHWIL	HPIC	HPIA†	HPID
Host writes HPIC 1st halfword	0001	xx	0	00	0	0009 0009	????????	????????
Host writes HPIC 2nd halfword	0001	xx	0	00	1	0009 0009	????????	????????
Host writes HPIA 1st halfword	1234	xx	0	01	0	0009 0009	???? 1234	????????
Host writes HPIA 2nd halfword	8000	xx	0	01	1	0009 0009	8000 1234	????????

Legend: ? = value is unknown
† For the C64x DSP, a host write to HPIA updates both HPIAR and HPIAW internally.

Table 7. Initialization of HPIC and HPIA With HWOB = 0

Event	Value During Access					Value After Access		
	HD	$\overline{\text{HBE[1–0]}}$	HR/$\overline{\text{W}}$	HCNTL[1–0]	HHWIL	HPIC	HPIA†	HPID
Host writes HPIC 1st halfword	0000	xx	0	00	0	0008 0008	????????	????????
Host writes HPIC 2nd halfword	0000	xx	0	00	1	0008 0008	????????	????????
Host writes HPIA 1st halfword	8000	xx	0	01	0	0008 0008	8000 ????	????????
Host writes HPIA 2nd halfword	1234	xx	0	01	1	0008 0008	8000 1234	????????

Legend: ? = value is unknown
† For the C64x DSP, a host write to HPIA updates both HPIAR and HPIAW internally.

5.1.2 Initialization of HPIC and HPIA (C64x HPI32)

For the HPI32, either the host or the CPU can be used to initialize HPIC and HPIA. All accesses are 32-bits wide. The HWOB bit in HPIC is not used. Therefore, it may not be necessary to initialize HPIC if the default value is desired. Table 8 summarizes the HPIC and HPIA initialization sequence for HPI32.

Table 8. Initialization of HPIC and HPIA for HPI32

Event	Value During Access			Value After Access		
	HD	HR/$\overline{W}$	HCNTL[1–0]	HPIC	HPIA	HPID
Host writes HPIC	0000 0000	0	00	0008 0008	????????	????????
Host writes HPIA	8000 1234	0	01	0008 0008	8000 1234	????????

Legend: ? = value is unknown

5.2 HPID Read Access in Fixed Address Mode

Assume that once the HPI is initialized, the host wishes to perform a read access to an address in fixed address mode. Assume that the host wants to read the word at address 8000 1234h and that the word value at that location is 789A BCDEh. The following sections discuss an HPID read access in fixed address mode for the 16-bit-wide host port (C62x/C67x HPI and C64x HPI16) and the 32-bit-wide host port (HPI32).

5.2.1 HPID Read in Fixed Address Mode (C62x/C67x HPI and C64x HPI16)

The host must read the 32-bit HPID in two 16-bit halfwords. Table 9 and Table 10 summarize this access for HWOB = 1 and HWOB = 0, respectively. On the first halfword access, the HPI waits for any previous requests to finish. During this time, $\overline{HRDY}$ pin is held high. Then, the HPI sends the read request to the DMA auxiliary channel (C620x/C670x DSP) or the internal address generation hardware (C621x/C671x/C64x DSP). If no previous requests are pending, this read request occurs on the falling edge of $\overline{HSTROBE}$. $\overline{HRDY}$ pin remains high until the requested data is loaded into HPID. Because all internal reads are word reads, at the beginning of the second read access, the data is already present in HPID. Thus, the second halfword HPID read never encounters a not-ready condition, and $\overline{HRDY}$ pin remains active. The byte enables are not important in this instance, because the HPI performs only word reads.

Table 9. Data Read Access to HPI in Fixed Address Mode With HWOB = 1

	Value During Access						Value After Access		
Event	**HD**	**$\overline{\text{HBE}}$[1–0]**	**HR/$\overline{\text{W}}$**	**HCNTL[1–0]**	**$\overline{\text{HRDY}}$**	**HHWIL**	**HPIC**	**HPIA**	**HPID**
Host reads 1st halfword	????	xx	1	11	1	0	0001 0001	8000 1234	????????
Data not ready									
Host reads 1st halfword	BCDE	xx	1	11	0	0	0009 0009	8000 1234	789A BCDE
Data ready									
Host reads 2nd halfword	789A	xx	1	11	0	1	0009 0009	8000 1234	789A BCDE

Legend: ? = value is unknown

Table 10. Data Read Access to HPI in Fixed Address Mode With HWOB = 0

	Value During Access						Value After Access		
Event	**HD**	**$\overline{\text{HBE}}$[1–0]**	**HR/$\overline{\text{W}}$**	**HCNTL[1–0]**	**$\overline{\text{HRDY}}$**	**HHWIL**	**HPIC**	**HPIA**	**HPID**
Host reads 1st halfword	????	xx	1	11	1	0	0000 0000	8000 1234	????????
Data not ready									
Host reads 1st halfword	789A	xx	1	11	0	0	0008 0008	8000 1234	789A BCDE
Data ready									
Host reads 2nd halfword	BCDE	xx	1	11	0	1	0008 0008	8000 1234	789A BCDE

Legend: ? = value is unknown

5.2.2 HPID Read in Fixed Address Mode (C64x HPI32)

The host access sequence to HPID of the HPI32 is similar to the sequence for the HPI16. The difference is that an HPI32 host access is done in one 32-bit word instead of two 16-bit halfwords. Table 11 shows an example of this read access in fixed address mode. In this example, the host reads the word at address 8000 1234h with a value of 789A BCDEh.

Table 11. Data Read Access in Fixed Address Mode for HPI32

	Value During Access				Value After Access		
Event	**HD**	**HR/$\overline{W}$**	**HCNTL[1–0]**	**$\overline{HRDY}$**	**HPIC**	**HPIA**	**HPID**
Host reads HPIC Data not ready	????????	1	00	1	0000 0000	8000 1234	????????
Host writes HPID Data ready	789A BCDE	0	11	0	0008 0008	8000 1234	789A BCDE

Legend: ? = value is unknown

5.3 HPID Read Access in Autoincrement Mode

The autoincrement feature results in efficient sequential host accesses. For both HPID read and write accesses, this removes the need for the host to load incremented addresses into HPIA. For read accesses, the data pointed to by the next address is fetched immediately after the completion of the current read. Because the intervals between successive reads are used to prefetch data, the latency for the next access is reduced. For the C62x/C67x HPI, prefetching also occurs after a host write of FETCH = 1 to HPIC. If the next HPI access is an HPID read, then the data is not refetched and the prefetched data is sent to the host. Otherwise, the HPI must wait for the prefetch to finish.

5.3.1 HPID Read in Autoincrement Mode (C62x/C67x HPI and C64x HPI16)

Table 12 summarizes a read access with autoincrement. After the first halfword access is complete (with the rising edge of the first $\overline{HSTROBE}$), the address increments to the next word, or 8000 1238h in this example. Assume that the data at that location is 8765 4321h. This data is prefetched and loaded into HPID. For the C62x/C67x HPI, prefetching begins on the rising edge of $\overline{HSTROBE}$ in the second halfword read. The C64x HPI has an internal read buffer that allows prefetching to occur to fill the internal buffer upon the first HPID read access ($\overline{HSTROBE}$ falling edge). See section 4.2.2 for details.

Table 12. Read Access to HPI in Autoincrement Mode With HWOB = 1

	Value During Access						Value After Access		
Event	**HD**	**$\overline{\text{HBE}}$[1–0]**	**HR/$\overline{\text{W}}$**	**HCNTL[1–0]**	**$\overline{\text{HRDY}}$**	**HHWIL**	**HPIC**	**HPIA**	**HPID**
Host reads 1st halfword	????	xx	1	10	1	0	0001 0001	8000 1234	????????
Data not ready									
Host reads 1st halfword	BCDE	xx	1	10	0	0	0009 0009	8000 1234	789A BCDE
Data ready									
Host reads 2nd halfword	789A	xx	1	10	0	1	0009 0009	8000 1234	789A BCDE
Prefetch†	????	xx	x	xx	1	x	0001 0001	80001238	789A BCDE
Data not ready									
Prefetch†	????	xx	x	xx	0	x	0009 0009	8000 1238	8765 4321
Data ready									

Legend: ? = value is unknown
† For the C64x HPI, prefetch occurs immediately after the first host read request. (Row 1 in this table).

Table 13. Read Access to HPI in Autoincrement Mode With HWOB = 0

	Value During Access						Value After Access		
Event	**HD**	**$\overline{\text{HBE}}$[1–0]**	**HR/$\overline{\text{W}}$**	**HCNTL[1–0]**	**$\overline{\text{HRDY}}$**	**HHWIL**	**HPIC**	**HPIA**	**HPID**
Host reads 1st halfword	????	xx	1	10	1	0	0000 0000	8000 1234	????????
Data not ready									
Host reads 1st halfword	789A	xx	1	10	0	0	0008 0008	8000 1234	789A BCDE
Data ready									
Host reads 2nd halfword	BCDE	xx	1	10	0	1	0008 0008	8000 1234	789A BCDE
Prefetch†	????	xx	x	xx	1	x	0000 0000	8000 1238	789A BCDE
Data not ready									
Prefetch†	????	xx	x	xx	0	x	0008 0008	8000 1238	8765 4321
Data ready									

Legend: ? = value is unknown
† For the C64x HPI, prefetch occurs immediately after the first host read request. (Row 1 in this table).

5.3.2 HPID Read in Autoincrement Mode (C64x HPI32)

Table 14 summarizes a read access with autoincrement for the HPI32. In autoincrement mode, the first HPID read access causes the HPI to not only fetch the current data, but also to prefetch extra data to fill the internal read buffer. This throughput improvement internal read buffer is discussed in section 4.2.2.

Table 14. Read Access to HPI in Autoincrement Mode for HPI32

	Value During Access				Value After Access		
Event	**HD**	**HR/$\overline{W}$**	**HCNTL[1–0]**	**$\overline{HRDY}$**	**HPIC**	**HPIA**	**HPID**
Host reads HPID	????????	1	10	1	0000 0000	8000 1234	????????
Data not ready							
Prefetch data to fill internal read buffer							
Host reads HPID	789A BCDE	1	10	0	0008 0008	8000 1234	789A BCDE
Data ready							
Address autoincrement	????????	?	??	0	0008 0008	8000 1238	8765 4321
Next data ready							

Legend: ? = value is unknown

5.4 HPID Write Access in Fixed Address Mode

5.4.1 HPID Write in Fixed Address Mode (C62x/C67x HPI and C64x HPI16)

During a write access to the HPI, the first halfword portion of HPID (the least-significant halfword or most-significant halfword, as selected by HWOB) is overwritten by the data coming from the host, and the first $\overline{\text{HBE}}[1-0]$ pair is latched while the HHWIL pin is low. The second halfword portion of HPID is overwritten by the data coming from the host, and the second $\overline{\text{HBE}}[1-0]$ pair is latched on the rising edge of $\overline{\text{HSTROBE}}$ while the HHWIL pin is high. At the end of this write access (with the second rising edge of $\overline{\text{HSTROBE}}$), HPID is transferred as a 32-bit word to the address specified by HPIA with the four related byte enables.

Table 15 and Table 16 summarize an HPID write access with HWOB = 1 and HWOB = 0, respectively. The host writes 5566h to the 16 LSBs of location 8000 1234h, which is already pointed to by HPIA. This location is assumed to start with the value 0. The HPI delays the host until any previous transfers are completed by setting $\overline{\text{HRDY}}$ high. If there are no pending writes waiting in HPID, then write accesses normally proceed without a not-ready time. For the C620x/C670x HPI, the $\overline{\text{HBE}}[1-0]$ pins are enabled only for the transfer of the 16 LSBs. For the C621x/C671x HPI and C64x HPI16, the $\overline{\text{HBE}}[1-0]$ pins do not exist. Only word writes are allowed, and all 16-bit write accesses must be made in pairs. The entire 32-bit word is transferred. Table 17 shows a 32-bit write access with HWOB = 1.

Table 15. 16-Bit Data Write Access to HPI in Fixed Address Mode With HWOB = 1

	Value During Access						Value After Access			
Event	**HD†**	**$\overline{\text{HBE}}[1-0]$**	**HR/$\overline{\text{W}}$**	**HCNTL[1–0]**	**$\overline{\text{HRDY}}$**	**HHWIL**	**HPIC**	**HPIA**	**HPID**	**Location 8000 1234**
Host writes HPID 1st halfword	5566	00	0	11	1	0	0001 0001	8000 1234	????????	0000 0000
Waiting for previous access to complete										
Host writes HPID 1st halfword	5566	00	0	11	0	0	0009 0009	8000 1234	???? 5566	0000 0000
Host writes HPID 2nd halfword	wxyz	11	0	11	0	1	0009 0009	8000 1234	wxyz 5566	0000 0000
Waiting for access to complete	????	??	?	??	1	?	0001 0001	8000 1234	wxyz 5566	0000 5566

Legend: ? = value is unknown

† For C620x/C670x HPI, wxyz represents a “don’t care” value on the HD pins. The $\overline{\text{HBE}}[1-0]$ value indicates that only 16-bits are transferred. For C621x/C671x HPI and C64x HPI, wxyz should be 0000 on the HD pins. The entire 32-bit word is transferred.

Table 16. 16-Bit Data Write Access to HPI in Fixed Address Mode With HWOB = 0

Event	Value During Access						Value After Access			Location
	HD†	$\overline{\text{HBE}}$[1–0]	HR/$\overline{\text{W}}$	HCNTL[1–0]	$\overline{\text{HRDY}}$	HHWIL	HPIC	HPIA	HPID	8000 1234
Host writes HPID 1st halfword	wxyz	11	0	11	1	0	0000 0000	8000 1234	????????	0000 0000
Waiting for previous access to complete										
Host writes HPID 1st halfword	wxyz	11	0	11	0	0	0008 0008	8000 1234	wxyz ????	0000 0000
Host writes HPID 2nd halfword	5566	00	0	11	0	1	0008 0008	8000 1234	wxyz 5566	0000 0000
Waiting for access to complete	????	??	?	??	1	?	0008 0008	8000 1234	wxyz 5566	0000 5566

Legend: ? = value is unknown

† For C620x/C670x HPI, wxyz represents a “don’t care” value on the HD pins. The $\overline{\text{HBE}}$[1–0] value indicates that only 16-bits are transferred. For C621x/C671x HPI and C64x HPI, wxyz should be 0000 on the HD pins. The entire 32-bit word is transferred.

Table 17. 32-Bit Data Write Access to HPI in Fixed Address Mode With HWOB = 1

Event	Value During Access						Value After Access			Location
	HD†	$\overline{\text{HBE}}$[1–0]	HR/$\overline{\text{W}}$	HCNTL[1–0]	$\overline{\text{HRDY}}$	HHWIL	HPIC	HPIA	HPID	8000 1234
Host writes HPID 1st halfword	5566	00	0	11	1	0	0001 0001	8000 1234	????????	0000 0000
Waiting for previous access to complete										
Host writes HPID 1st halfword	5566	00	0	11	0	0	0009 0009	8000 1234	???? 5566	0000 0000
Host writes HPID 2nd halfword	wxyz	00	0	11	0	1	0009 0009	8000 1234	wxyz 5566	0000 0000
Waiting for access to complete	????	??	?	??	1	?	0001 0001	8000 1234	wxyz 5566	wxyz 5566

Legend: ? = value is unknown

† For C620x/C670x HPI, wxyz represents a “don’t care” value on the HD pins. The $\overline{\text{HBE}}$[1–0] value indicates that only 16-bits are transferred. For C621x/C671x HPI and C64x HPI, wxyz should be 0000 on the HD pins. The entire 32-bit word is transferred.

5.4.2 HPID Write in Fixed Address Mode (C64x HPI32)

HPID write of the HPI32 is similar to the HPI16. However, the host can write to HPID in one 32-bit write access. Table 18 summarizes an HPID write access for HPI32 in fixed address mode.

Table 18. Data Write Access to HPI in Fixed Address Mode for HPI32

	Value During Access				Value After Access			
Event	**HD**	**HR/$\overline{W}$**	**HCNTL[1–0]**	**$\overline{HRDY}$**	**HPIC**	**HPIA**	**HPID**	**Location 8000 1234**
Host writes HPID Waiting for previous access to complete.	0000 5566	0	11	1	0000 0000	8000 1234	????????	0000 0000
Host writes HPID.	0000 5566	0	11	0	0008 0008	8000 1234	0000 5566	0000 0000
Ready								
Waiting for access to complete.	????????	?	??	0	0008 0008	8000 1234	8765 4321	0000 5566

Legend: ? = value is unknown

5.5 HPID Write Access in Autoincrement Mode

5.5.1 HPID Write in Autoincrement Mode (C62x/C67x HPI and C64x HPI16)

Table 19 and Table 20 summarize a host data write with autoincrement for HWOB = 1 and HWOB = 0, respectively. These examples are identical to the ones in section 5.4, except for the HCNTL[1–0] value and a subsequent write at address 8000 1238h. The increment occurs on the rising edge of $\overline{HSTROBE}$ on the next HPID write access. If the next access is an HPIA or HPIC access or an HPID read, the autoincrement does not occur.

For the C64x HPI in autoincrement mode, data written by the host is immediately copied from HPID to the internal write buffer. Therefore if the internal write buffer is not full, $\overline{HRDY}$ remains ready, and row 4 and 7 in Table 19 and Table 20 do not apply. In addition, the DSP only services the HPI write access in autoincrement mode when the internal write buffer is half full, or when the write cycle is terminated. Locations 8000 1234h and 8000 1238h in Table 13 and Table 14 do not get updated to the correct values (0000 5566h, 3300 0000h) until the internal write buffer is serviced.

Table 19. Write Access to HPI in Autoincrement Mode With HWOB = 1

Event	Value During Access						Value After Access			Location 8000 1234	Location 8000 1238
	HD†	$\overline{HBE}$	HR/$\overline{W}$	HCNTL	$\overline{HRDY}$	HHWIL	HPIC	HPIA	HPID		
Host writes HPID 1st halfword	5566	00	0	10	1	0	0001 0001	8000 1234	????????	0000 0000	0000 0000
Waiting for previous access to complete											
Host writes HPID 1st halfword	5566	00	0	10	0	0	0009 0009	8000 1234	???? 5566	0000 0000	0000 0000
Ready											
Host writes HPID 2nd halfword	wxyz	11	0	10	0	1	0009 0009	8000 1234	wxyz 5566	0000 0000	0000 0000
Host writes HPID 1st halfword	nopq	11	0	10	1	0	0001 0001	8000 1234	wxyz 5566	0000 5566	0000 0000
Waiting for previous access to complete											
Host writes HPID 1st halfword	nopq	11	0	10	0	0	0009 0009	8000 1238	wxyz nopq	0000 5566	0000 0000
Host writes HPID 2nd halfword	33rs	01	0	10	0	1	0009 0009	8000 1238	33rs nopq	0000 5566	0000 0000
Waiting for access to complete	????	??	?	??	1	?	0001 0001	8000 1238	33rs nopq	0000 5566	3300 0000

Legend: ? = value is unknown

† For C620x/C670x HPI, wxyz, rs, and nopq represent don't care values on the HD pins. For C621x/C671x HPI and C64x HPI, wxyz + 0000, rs = 00, and nopq = 0000 on the HD pins. The entire 32-bit word is transferred.

Table 20. Write Access to HPI in Autoincrement Mode With HWOB = 0

Event	Value During Access						Value After Access			Location 8000 1234	Location 8000 1238
	HD†	$\overline{HBE}$	HR/$\overline{W}$	HCNTL	$\overline{HRDY}$	HHWIL	HPIC	HPIA	HPID		
Host writes HPID 1st halfword	wxyz	11	0	10	1	0	0000 0000	8000 1234	????????	0000 0000	0000 0000
Waiting for previous access to complete											
Host writes HPID 1st halfword	wxyz	11	0	10	0	0	0008 0008	8000 1234	wxyz ???	0000 0000	0000 0000
Ready											
Host writes HPID 2nd halfword	5566	00	0	10	0	1	0008 0008	8000 1234	wxyz 5566	0000 0000	0000 0000
Host writes HPID 1st halfword	33rs	01	0	10	1	0	0000 0000	8000 1234	wxyz 5566	0000 5566	0000 0000
Waiting for previous access to complete											
Host writes HPID 1st halfword	33rs	01	0	10	0	0	0008 0008	8000 1238	33rs 5566	0000 5566	0000 0000
Host writes HPID 2nd halfword	nopq	11	0	10	0	1	0008 0008	8000 1238	33rs nopq	0000 5566	0000 0000
Waiting for access to complete	????	??	?	??	1	?	0000 0000	8000 1238	33rs nopq	0000 5566	3300 0000

Legend: ? = value is unknown

† For C620x/C670x HPI, wxyz, rs, and nopq represent don't care values on the HD pins. For C621x/C671x HPI and C64x HPI, wxyz + 0000, rs = 00, and nopq = 0000 on the HD pins. The entire 32-bit word is transferred.

5.5.2 HPID Write in Autoincrement Mode (C64x HPI32)

As described in section 5.5.1, data written in autoincrement mode by the C64x HPI host is immediately copied from HPID to the internal write buffer. Therefore if the internal write buffer is not full, $\overline{\text{HRDY}}$ remains ready. The DSP only services the HPI write access in autoincrement mode when the internal write buffer is half full, or when the write cycle is terminated. Locations 8000 1234h and 8000 1238h in Table 19 and Table 20 do not get updated to the correct values (0000 5566h, 3300 0000h) until the internal write buffer is serviced. Table 21 summarizes an HPID write in autoincrement mode for the HPI32.

Table 21. Write Access to HPI in Autoincrement Mode for HPI32

	Value During Access					Value After Access				
Event	**HD**	**HR/$\overline{\text{W}}$**	**HCNTL**	**$\overline{\text{HRDY}}$**	**HHWIL**	**HPIC**	**HPIA**	**HPID‡**	**Location 8000 1234†**	**Location 8000 1238†**
Host writes HPID	0000 5566	0	10	1	0	0000 0000	8000 1234	????????	0000 0000	0000 0000
Waiting for previous access to complete										
Host writes HPID	0000 5566	0	10	0	0	0008 0008	8000 1234	0000 5566	0000 0000	0000 0000
Ready										
Host writes HPID	3300 0000	0	10	0	1	0008 0008	8000 1238	3300 0000	0000 0000	0000 0000
Ready										

Legend: ? = value is unknown

† Location 8000 1234h and 8000 1238h do not get updated until the HPI internal write buffer is serviced. This occurs when the internal write buffer is half full, or when the write cycle terminates.

‡ The data in HPID is immediately copied to the internal write buffer if it is not full. Therefore, $\overline{\text{HRDY}}$ is ready.

5.6 Single Halfword Cycles (C620x/C670x HPI only)

In normal operation, every transfer must consist of two halfword accesses. However, the C620x/C670x HPI allows single halfword accesses to speed up operation. These can be useful in performing the following tasks:

- Writes to and reads from HPIC: In Table 6 (page 34), the entire HPIC was written to correctly after the first write. When writing HPIC, the host does not have to be concerned about HHWIL, nor does it have to perform two consecutive writes to both halfwords. Similarly, the host can choose to read HPIC only once, because both halves contain the same value.
- Writes to and reads from HPIA: In Table 6 (page 34), the portion of HPIA accesses selected by HHWIL and HWOB is updated automatically after each halfword access. Thus, to change either the upper or the lower 16 bits of HPIA, the host must select the half to modify through a combination of the HHWIL and HWOB bits. The host can also choose to read only half of HPIA.
- HPID read accesses: Read accesses are actually triggered by the first halfword access (HHWIL low). Thus, if on reads the host is interested only in the first halfword (the least or most significant halfword, as selected by the HWOB bit), the host does not need to request the second address. However, prefetching does not occur unless the second halfword is also read. A subsequent read of the first halfword (HHWIL low) or a write of a new value to HPIA overrides any previous prefetch request. On the other hand, a read of just the second halfword (HHWIL high) is not allowed and results in undefined operation.
- Write accesses: Write accesses are triggered by the second halfword access (HHWIL word high). Thus, if the host desires to change only the portion of HPID selected by HHWIL high (and the associated byte enables) during consecutive write accesses, only a single cycle is needed. This technique's primary use is for memory fills: the host writes both halfwords of the first write access with $\overline{\text{HBE}}[1-0]$ = 00. On subsequent write accesses, the host writes the same value to the portion of HPID selected by HHWIL as the first write access. In this case, the host performs autoincrementing writes (HCNTL[1–0] = 10) on all write accesses.

6 HPI Transfer Priority Queue (C621x/C671x/C64x HPI only)

All C621x/C671x HPI transfers are placed in the high-priority transfer queue, Q1. All C64x HPI transfers can be programmed to any of the four priority levels, with the medium-priority level set as the default. See the *Enhanced DMA (EDMA) Controller Reference Guide* (SPRU234) for details on transfer priority.

7 Memory Access Through the HPI During Reset

During reset when $\overline{\text{HCS}}$ is active low, $\overline{\text{HRDY}}$ is inactive high; when $\overline{\text{HCS}}$ is inactive, $\overline{\text{HRDY}}$ is active. The HPI cannot be used while the device is in reset. However, certain boot modes can allow the host to write to the CPU memory space (including configuring EMIF configuration registers to define external memory before accessing it) upon the rising edge of the $\overline{\text{RESET}}$ signal. Although the device is not in reset during these boot modes, the CPU itself is in reset until the boot completes.

8 HPI Registers

The registers that the HPI uses for communication between the host device and the CPU are listed in Table 22 (C62x/C67x DSP) and Table 23 (C64x DSP). See the device-specific datasheet for the memory address of these registers.

Table 22. HPI Registers for C62x/C67x DSP

		Read/Write Access		
Acronym	**Register Name**	**Host**	**CPU**	**Section**
HPID	HPI data register	R/W	–	8.1
HPIA	HPI address register	R/W	–	8.2
HPIC	HPI control register	R/W	R/W	8.3

Table 23. HPI Registers for C64x DSP

		Read/Write Access		
Acronym	**Register Name**	**Host**	**CPU**	**Section**
HPID	HPI data register	R/W	–	8.1
HPIAW†	HPI address write register	R/W	R/W	8.2
HPIAR†	HPI address read register	R/W	R/W	8.2
HPIC	HPI control register	R/W	R/W	8.3
TRCTL	HPI transfer request control register	–	R/W	8.4

† Host access to the HPIA updates both HPIAW and HPIAR. The CPU can access HPIAW and HPIAR, independently.

8.1 HPI Data Register (HPID)

The HPI data register (HPID) contains the data that was read from the memory accessed by the HPI, if the current access is a read; HPID contains the data that is written to the memory, if the current access is a write.

8.2 HPI Address Register (HPIA)

The HPI address register (HPIA) contains the address of the memory accessed by the HPI at which the current access occurs. This address is a 32-bit word-aligned byte address with all 32-bits readable/writable. The two LSBs always function as 0, regardless of the value read from their location. The C62x/C67x HPIA is only accessible by the host, it is not mapped to the DSP memory.

The C64x HPIA is separated into two registers internally: the HPI address write register (HPIAW) and the HPI address read register (HPIAR). The HPIA is accessible by both the host and the CPU. By separating the HPIA into HPIAW and HPIAR internally, the CPU can update the read and write memory address independently to allow the host to perform read and write to different address ranges. When reading HPIA from the CPU, the value returned corresponds to the address currently being used by the HPI and DMA to transfer data inside the DSP. For an HPI write, HPIAR contains the starting address of the transfer while HPIAW contains the address currently being used and is updated after each burst of data in the transfer. For an HPI read, HPIAW contains the starting address of the transfer while HPIAR contains the address currently being used and is updated after each burst of data in the transfer. HPIA does not contain the address for the current transfer at the external pins. Thus, reading HPIA does not indicate the status of a transfer, and should not be relied upon to do so.

For the C64x HPI, a host access to HPIA is identical to the operation of the C62x/C67x HPI. The HCNTL[1–0] control bits are set to 01b to indicate an access to HPIA. A host write to HPIA updates both HPIAW and HPIAR internally. A host read of HPIA returns the value in the most-recently-used HPIAx register. For example, if the most recent HPID access was a read, then an HPIA read by the external host returns the value in HPIAR; if the most recent HPID access was a write, then an HPIA read by the external host returns the value in HPIAW.

Systems that update HPIAR/HPIAW internally via the CPU must not allow HPIA updates via the external bus and conversely. The HPIAR/HPIAW registers can be read independently by both the CPU and the external host. The system must not allow HPID accesses via the external host while the DSP is updating the HPIAR/W registers internally. This can be controlled by any convenient means, including the use of general-purpose input/output pins to perform handshaking between the host and the DSP.

8.3 HPI Control Register (HPIC)

The HPI control register (HPIC) is normally the first register accessed to set configuration bits and initialize the interface. The HPIC is shown in Figure 15, Figure 16, Figure 17, and Figure 18 and described in Table 24. From the host's view (Figure 15 and Figure 16), HPIC is organized as a 32-bit register with two identical halves, meaning the high halfword and low halfword contents are the same. On a host write, both halfwords must be identical, except when writing the DSPINT bits in HPI16 mode (see section 8.3.2). In HPI16 mode when setting DSPINT = 1, the host must only write 1 to the lower 16-bit halfword or upper 16-bit halfword, but not both. On C64x DSP in HPI16 mode, the value of DSPINT in the first halfword write is latched. The DSPINT bit must be cleared to 0 in the second halfword write. In HPI32 mode, the upper and lower halfwords must always be identical.

From the C6000 CPU view (Figure 17 and Figure 18), HPIC is a 32-bit register with only 16 bits of useful data. Only CPU writes to the lower halfword affect HPIC values and HPI operation.

On C64x DSP, the HWOB bit is writable by the CPU. Therefore, care must be taken when writing to HPIC in order not to write an undesired value to HWOB.

Figure 15. HPI Control Register (HPIC)—Host Reference View (C62x/C67x DSP)

31	21	20	19	18	17	16
Reserved		FETCH	HRDY	HINT	DSPINT	HWOB
HR-0		HR/W-0	HR-1	HR/W-0	HR/W-0	HR/W-0

15	5	4	3	2	1	0
Reserved		FETCH	HRDY	HINT	DSPINT	HWOB
HR-0		HR/W-0	HR-1	HR/W-0	HR/W-0	HR/W-0

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

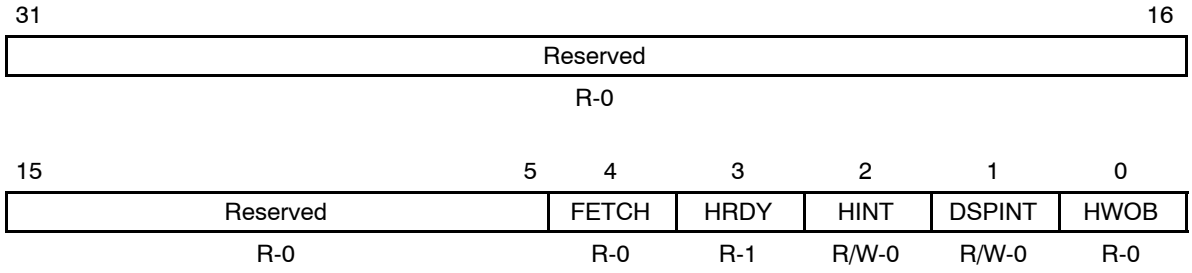
Figure 16. HPI Control Register (HPIC)—Host Reference View (C64x DSP)

31	30 24	23	22 21	20	19	18	17	16
Reserved†	Reserved	Reserved†	Reserved	FETCH	HRDY	HINT	DSPINT	HWOB
HR/W-0	HR-0	HR-0	HR-0	HR/W-0	HR-1	HR/W-0	HR/W-0	HR/W-0

15	14 8	7	6 5	4	3	2	1	0
Reserved†	Reserved	Reserved†	Reserved	FETCH	HRDY	HINT	DSPINT	HWOB
HR/W-0	HR-0	HR-0	HR-0	HR/W-0	HR-1	HR/W-0	HR/W-0	HR/W-0

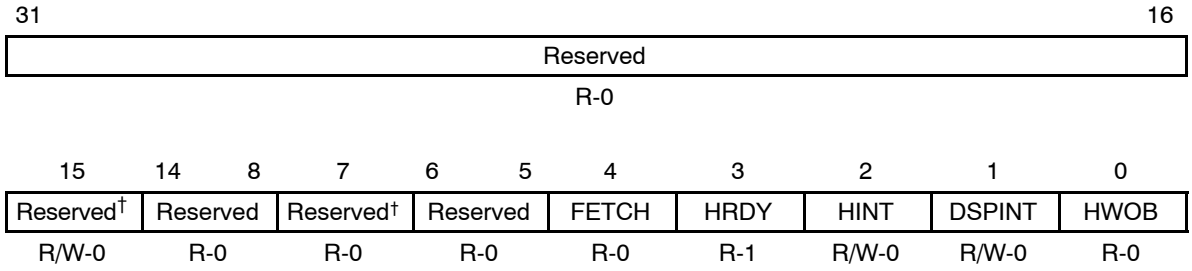
Legend: H = Host access; R = Read only; R/W = Read/Write; -*n* = value after reset
† These bits are writable fields and must be written with 0; otherwise, operation is undefined.

Figure 17. HPI Control Register (HPIC)—CPU Reference View (C62x/C67x DSP)



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

Figure 18. HPI Control Register (HPIC)—CPU Reference View (C64x DSP)



Legend: R = Read only; R/W = Read/Write; -*n* = value after reset
† These bits are writable fields and must be written with 0; otherwise, operation is undefined.

Table 24. HPI Control Register (HPIC) Field Descriptions

Bit	*field*†	*symval*†	Value	Description
31–21	Reserved	–	0	Reserved. The reserved bit location is always read as 0.
20, 4	FETCH			Host fetch request bit.
		0	0	The value read by the host or CPU is always 0.
		1	1	The host writes a 1 to this bit to request a fetch into HPID of the word at the address pointed to by HPIA. The 1 is never actually written to this bit, however.
19, 3	HRDY			Ready signal to host bit. Not masked by $\overline{\text{HCS}}$ (as the $\overline{\text{HRDY}}$ pin is).
		0	0	The internal bus is waiting for an HPI data access request to finish.
		1	1	
18, 2	HINT			DSP-to-host interrupt bit. The inverted value of this bit determines the state of the CPU $\overline{\text{HINT}}$ output.
		0	0	CPU $\overline{\text{HINT}}$ output is logic 1.
		1	1	CPU $\overline{\text{HINT}}$ output is logic 0.
17, 1	DSPINT			The host processor-to-CPU/DMA interrupt bit.
		0	0	
		1	1	
16, 0	HWOB			Halfword ordering bit affects both data and address transfers. Only the host can modify this bit. HWOB must be initialized before the first data or address register access. For HPI32, HWOB is not used and the value of HWOB is irrelevant.
		0	0	The first halfword is most significant.
		1	1	The first halfword is least significant.
15–5	Reserved	–	0	Reserved. The reserved bit location is always read as 0.

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

8.3.1 Software Handshaking Using $\overline{\text{HRDY}}$ and FETCH Bit

The $\overline{\text{HRDY}}$ pin can indicate to a host that an HPID access has not finished. For example, the current HPID access can be waiting for a previous HPID access write to finish or for a previous HPID prefetched read to finish. Also, the current HPID read access can be waiting for its requested data to arrive. The HRDY and FETCH bits in the HPI control register (HPIC) allow for a software handshake that allows an HPI connection in systems in which a hardware ready control is not desired.

The FETCH and HRDY bits can be used to perform a read transfer as follows:

1) The host polls the HPIC register for HRDY = 1.
2) The host writes the desired HPIA value. This step is skipped if HPIA is already set to the desired value.
3) The host writes a 1 to the FETCH bit.
4) The host polls again for HRDY = 1.
5) The host performs an HPID read operation. In this case, the HPI is already in the ready state (HRDY = 1).
6) If this was a read with postincrement, go to step 4. For a read from the same location, go to step 3.For a read to a different address, go to step 2.

The HRDY bit can be used alone for write operations as follows:

1) The host polls for HRDY = 1.
2) The host writes the desired HPIA value. (This step is skipped if HPIA is already set to the desired value.)
3) The host performs an HPID write operation. For another write operation, go to step 1.

8.3.2 Host Device Using DSPINT Bit to Interrupt the CPU

The host can interrupt the CPU by writing to one of the DSPINT bits in HPIC. The DSPINT bit is tied directly to the internal DSPINT signal. By writing DSPINT = 1 when DSPINT = 0, the host causes a low-to-high transition on the DSPINT signal. If you program the selection of the DSPINT interrupt with the interrupt selector, the CPU detects the transition of DSPINT as an interrupt condition. Unlike a host write, a CPU write of DSPINT = 1 when DSPINT = 0 has no effect. The CPU can clear the DSPINT bits by writing a 1 to DSPINT when DSPINT = 1. Writing DSPINT = 0 (in HPIC) via the host or the CPU does not affect either the DSPINT bit or DSPINT signal in any case.

8.3.3 CPU Using HINT Bit to Interrupt the Host

The CPU can send an active interrupt condition on the $\overline{\text{HINT}}$ signal by writing to the HINT bit in HPIC. The HINT bit is inverted and tied directly to the $\overline{\text{HINT}}$ pin. The CPU can set $\overline{\text{HINT}}$ active by writing HINT = 1. The host can clear the $\overline{\text{HINT}}$ to inactive by writing a 1 to HINT. Writing HINT = 0 (in HPIC) via the host or the CPU does not affect either the HINT bit or the $\overline{\text{HINT}}$ signal.

In HPI16 mode, the HINT bit is read twice on the host interface side. The first and second halfword reads by the host can yield different data if the CPU changes the state of this bit between the two read operations.

8.4 HPI Transfer Request Control Register (TRCTL) (C64x DSP only)

The HPI transfer request control register (TRCTL) controls how the HPI submits its requests to the EDMA subsystem. The TRCTL is shown in Figure 19 and described in Table 25.

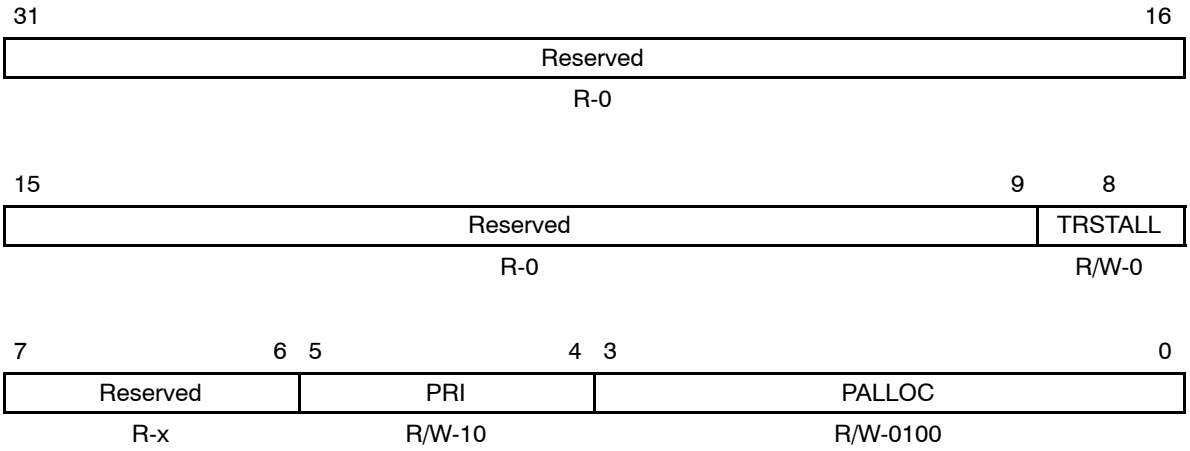
To safely change the PALLOC or PRI bits in TRCTL, the TRSTALL bit needs to be used to ensure a proper transition. The following procedure must be followed to change the PALLOC or PRI bits:

1) Set the TRSTALL bit to 1 to stop the HPI from submitting TR requests on the current PRI level. In the same write, the desired new PALLOC and PRI fields may be specified.
2) Clear all EDMA event enables (EER) corresponding to both old and new PRI levels to stop EDMA from submitting TR requests on both PRI levels. Do not manually submit additional events via the EDMA.
3) Do not submit new QDMA requests on either old or new PRI level.
4) Stop L2 cache misses on either old or new PRI level. This can be done by forcing program execution or data accesses in internal memory. Another way is to have the CPU executing a tight loop that does not cause additional cache misses.
5) Poll the appropriate PQ bits in the priority queue status register (PQSR) of the EDMA until both queues are empty (see the *Enhanced DMA (EDMA) Controller Reference Guide*, SPRU234).
6) Clear the TRSTALL bit to 0 to allow the HPI to continue normal operation.

Requestors are halted on the old HPI PRI level so that memory ordering can be preserved. In this case, all pending requests corresponding to the old PRI level must be let to complete before HPI is released from stall state.

Requestors are halted on the new PRI level to ensure that at no time can the sum of all requestor allocations exceed the queue length. By halting all requestors at a given level, you can be free to modify the queue allocation counters of each requestor.

Figure 19. HPI Transfer Request Control Register (TRCTL)



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

Table 25. HPI Transfer Request Control Register (TRCTL) Field Descriptions

Bit	*field*†	*symval*†	Value	Description
31–9	Reserved	–	0	Reserved. The reserved bit location is always read as 0.
8	TRSTALL			Forces the HPI to stall all HPI requests to the EDMA. This bit allows the safe changing of the PALLOC and PRI fields.
			0	Allows HPI requests to be submitted to the EDMA.
			1	Halts the creation of new HPI requests to the EDMA.
7–6	Reserved	–	0	Reserved. The reserved bit location is always read as 0.
5–4	PRI		0–3h	Controls the priority queue level that HPI requests are submitted to.
			0	Urgent priority
			1h	High priority
			2h	Medium priority
			3h	Low priority
3–0	PALLOC		0–Fh	Controls the total number of outstanding requests that can be submitted by the HPI to the EDMA. Valid values of PALLOC are 1 to 15, all other values are reserved; the default value is 4. HPI may have the programmed number of outstanding requests.

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

Revision History

Table 26 lists the changes made since the previous version of this document.

Table 26. Document Revision History

Page	Additions/Modifications/Deletions
49	Added sixth and seventh sentences in first paragraph of Section 8.3: On C64x DSP in HPI16 mode, the value of DSPINT in the first halfword write is latched. The DSPINT bit must be cleared to 0 in the second halfword write.
49	Changed HINT bit read/write designation in Figure 15: HR/W-0
49	Changed HINT bit read/write designation in Figure 16: HR/W-0
52	Deleted second sentence of Section 8.3.2.
53	Changed first sentence in second paragraph of Section 8.3.3: In HPI16 mode, the HINT bit is read twice on the host interface side.

This page is intentionally left blank.

Index

I

L

M

N

O

P

R

S

T

W