TMS320VC5501/5502 DSP Host Port Interface (HPI) Reference Guide

Literature Number: SPRU620C
March 2004

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 manual describes the features and operation of the host port interface (HPI) that is on the TMS320VC5501 and TMS320VC5502 digital signal processors (DSPs) in the TMS320C55x™ (C55x™) DSP generation.

Notational Conventions

This document uses the following conventions.

- In most cases, hexadecimal numbers are shown with the suffix h. For example, the following number is a hexadecimal 40 (decimal 64):

  40h

  Similarly, binary numbers often are shown with the suffix b. For example, the following number is the decimal number 4 shown in binary form:

  0100b

- If a signal or pin is active low, it has an overbar. For example, the $\overline{\text{RESET}}$ signal is active low.

Related Documentation From Texas Instruments

The following documents describe the C55x 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.

TMS320VC5501 Fixed-Point Digital Signal Processor Data Manual (literature number SPRS206) describes the features of the TMS320VC5501 fixed-point DSP and provides signal descriptions, pinouts, electrical specifications, and timings for the device.

TMS320VC5502 Fixed-Point Digital Signal Processor Data Manual (literature number SPRS166) describes the features of the TMS320VC5502 fixed-point DSP and provides signal descriptions, pinouts, electrical specifications, and timings for the device.

TMS320C55x Technical Overview (literature number SPRU393) introduces the TMS320C55x DSPs, the latest generation of fixed-point DSPs in the TMS320C5000™ DSP platform. Like the previous generations, this processor is optimized for high performance and low-power operation. This book describes the CPU architecture, low-power enhancements, and embedded emulation features.

TMS320C55x DSP CPU Reference Guide (literature number SPRU371) describes the architecture, registers, and operation of the CPU for the TMS320C55x DSPs.

TMS320C55x DSP Peripherals Overview Reference Guide (literature number SPRU317) introduces the peripherals, interfaces, and related hardware that are available on TMS320C55x DSPs.

TMS320C55x DSP Algebraic Instruction Set Reference Guide (literature number SPRU375) describes the TMS320C55x DSP algebraic instructions individually. Also includes a summary of the instruction set, a list of the instruction opcodes, and a cross-reference to the mnemonic instruction set.

TMS320C55x DSP Mnemonic Instruction Set Reference Guide (literature number SPRU374) describes the TMS320C55x DSP mnemonic instructions individually. Also includes a summary of the instruction set, a list of the instruction opcodes, and a cross-reference to the algebraic instruction set.

TMS320C55x Optimizing C/C++ Compiler User's Guide (literature number SPRU281) describes the TMS320C55x C/C++ Compiler. This C/C++ compiler accepts ISO standard C and C++ source code and produces assembly language source code for TMS320C55x devices.

TMS320C55x Assembly Language Tools User's Guide (literature number SPRU280) describes the assembly language tools (assembler, linker, and other tools used to develop assembly language code), assembler directives, macros, common object file format, and symbolic debugging directives for TMS320C55x devices.

TMS320C55x DSP Programmer's Guide (literature number SPRU376) describes ways to optimize C and assembly code for the TMS320C55x DSPs and explains how to write code that uses special features and instructions of the DSPs.

Trademarks

TMS320C5000, TMS320C55x, and C55x are trademarks of Texas Instruments.

Other trademarks are the property of their respective owners.

Contents

Figures

Tables

Host Port Interface (HPI)

This guide describes the host port interface (HPI) on the TMS320VC5501 and TMS320VC5502 digital signal processors (DSPs). The HPI enables an external host processor (host) to directly access memory internal to the DSP using an 8-bit or 16-bit interface.

1 Introduction to the HPI

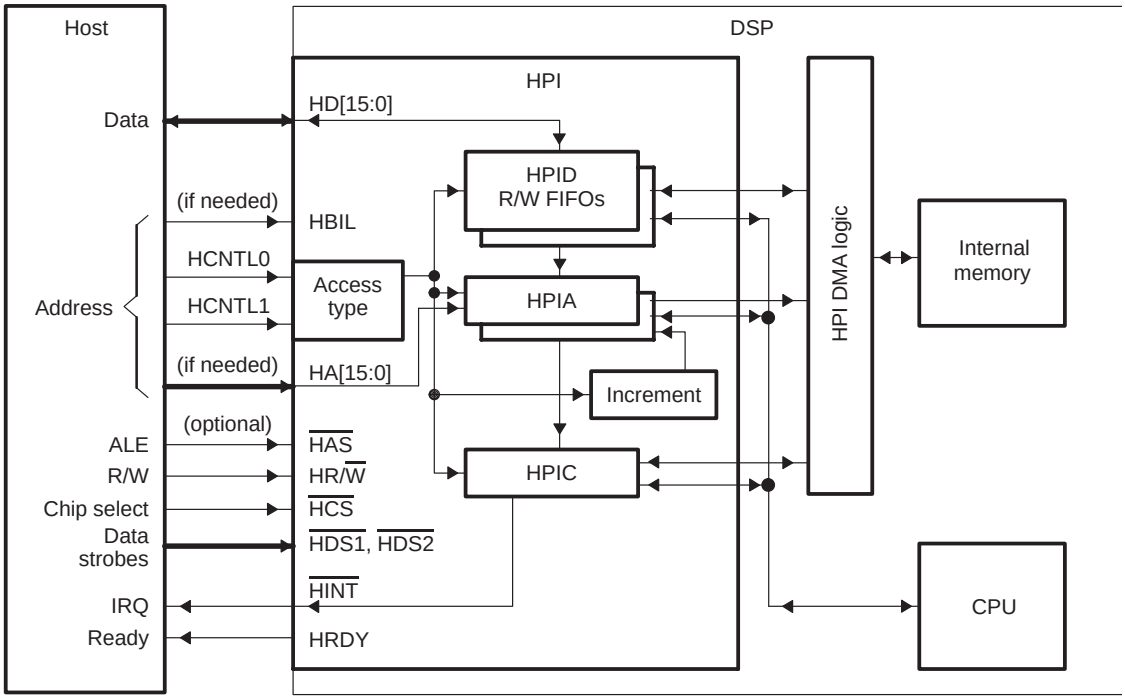
The HPI provides a parallel port through which an external host processor (host) can access memory internal to the TMS320VC5501 and TMS320VC5502 DSPs (see Table 1). The host cannot access addresses 0000h−005Fh because they are reserved for the memory mapped registers of the CPU. The host does not have direct access to memory that is external to the DSP.

Table 1. Internal Memory Accessible to Host and Address Bits Required From Host

Device	Internal Memory Accessible to the Host	Address Bits Required From Host
TMS320VC5501	First 16K words, except addresses 0000h−005Fh	The host must provide 14-bit addresses to the HPI, where each address points to a word (a 16-bit value) in memory.
TMS320VC5502	First 32K words, except addresses 0000h−005Fh	The host must provide 15-bit addresses to the HPI, where each address points to a word (a 16-bit value) in memory.

Figure 1 is a high-level block diagram showing how the HPI connects a host (left side of figure) and the DSP internal memory (right side of figure). Host activity is asynchronous to the clock that drives the HPI, which is the fast peripherals clock (SYSCLK1) of the DSP. The host functions as a master to the HPI. When HPI resources are temporarily busy or unavailable, the HPI can communicate this to the host by deasserting the HPI-ready (HRDY) output signal.

Figure 1. The Position of the HPI in the Host-DSP System



The HPI supports two interface modes:

- **8-bit multiplexed mode.** This mode is selected if the GPIO6 pin of the DSP is sampled high at reset. In this mode, an 8-bit data bus (HD[7:0]) carries both addresses and data. Each host cycle on the bus consists of two consecutive 8-bit transfers. When the host drives an address on the bus, the address is stored in a 16-bit address register (HPIA) in the HPI, so that the bus can then be used for data. The HPI contains two HPIAs (HPIAR and HPIAW), which can be used as separate address registers for read accesses and write accesses (for details, see section 2 on page 18).

- **16-bit nonmultiplexed mode.** This mode is not supported on TMS320VC5501 devices. On TMS320VC5502 devices, this mode is selected if the GPIO6 pin of the DSP is sampled low at reset. In the nonmultiplexed mode, the HPI provides separate address and data buses: a 16-bit address bus (HA[15:0]) and a 16-bit data bus (HD[15:0]). The address registers of the HPI are not used in this mode. Each host cycle on the data bus consists of one 16-bit transfer.

A 16-bit control register (HPIC) is accessible by the DSP CPU and the host. The CPU can use HPIC to send an interrupt request to the host, to clear an interrupt request from the host, and to monitor the HPI. The host can use HPIC to configure and monitor the HPI, to send an interrupt request to the CPU, and to clear an interrupt request from the CPU.

Data flow between the host and the HPI uses a temporary storage register, the 16-bit data register (HPID). Data arriving from the host is held in HPID until the data can be stored elsewhere in the DSP. Data to be sent to the host is held in HPID until the HPI is ready to perform the transfer. When address autoincrementing is used, read and write FIFOs are used to store burst data. If autoincrementing is not used, the FIFO memory acts as a single register (only one location is used).

Note:

To manage data transfers between HPID and the internal memory, the DSP contains dedicated HPI DMA logic. The HPI DMA logic is not programmable; it automatically stores or fetches data using the address provided by the host. The HPI DMA logic is independent of the DMA controller included in the DSP. Information on the DMA controller is in the *TMS320VC5501/5502 DSP Direct Memory Access (DMA) Controller Reference Guide* (literature number SPRU613).

1.1 Summary of the HPI Registers

Table 2 summarizes the registers inside the HPI, including access permissions and access requirements from the perspective of the host and of the DSP CPU. Section 9 (page 57) gives detailed descriptions of all these registers. Section 2 (page 18) explains why there are two address registers (HPIAW and HPIAR) and describes the two HPIA modes that determine how the registers are used by the host.

The host can access only HPIC, HPIAW, HPIAR, and HPID. By driving specific levels on the HCNTL[1:0] signals, the host indicates whether it is performing an HPIC, HPIA, or HPID access. In the case of an HPID access, the HCNTL signals also indicate whether the HPI should perform an automatic address increment after the access or not increment the address after the access. The effects of the HCNTL[1:0] signals are described in more detail in section 3.4 (page 27). The HR/$\overline{W}$ signal indicates whether the host is reading or writing.

The DSP CPU cannot access HPID but has limited access to HPIC, HPIAR, and HPIAW. The CPU has full access to the power and emulation management register, which is used to select an emulation mode for the HPI.

To access a register in the HPI, the CPU accesses it at a designated address in the I/O space of the DSP. Table 2 shows the I/O addresses.

Table 2. Summary of HPI Registers

Register	Host Access		CPU Access	
	Read/Write Permissions	**Access Requirements**	**Read/Write Permissions**	**I/O Address**
Reserved	–	–	–	A000h to A001h
HPWREMU: Power and emulation management register	None	–	Read/Write	A002h
Reserved	–	–	–	A003h
HGPIOINT1: General-purpose I/O interrupt control register 1	None	–	Read/Write	A004h
HGPIOINT2: General-purpose I/O interrupt control register 2	None	–	Read/Write	A005h
HGPIOEN: General-purpose I/O enable register	None	–	Read/Write	A006h
Reserved	–	–	–	A007h
HGPIODIR1: General-purpose I/O direction register 1	None	–	Read/Write	A008h
Reserved	–	–	–	A009h
HGPIODAT1: General-purpose I/O data register 1	None	–	Read/Write	A00Ah
Reserved	–	–	–	A00Bh

† On TMS320VC5502 devices, HGPIODIR3 and HGPIODAT3 are used when the address pins, HA[15:0], are configured for general-purpose I/O. On TMS320VC5501 devices, these register addresses are reserved because there are no address pins.
‡ The single-HPIA mode and the dual-HPIA mode are described in section 2 (page 18).

Table 2. Summary of HPI Registers (Continued)

Register	Host Access		CPU Access	
	Read/Write Permissions	**Access Requirements**	**Read/Write Permissions**	**I/O Address**
HGPIODIR2: General-purpose I/O direction register 2	None	–	Read/Write	A00Ch
Reserved	–	–	–	A00Dh
HGPIODAT2: General-purpose I/O data register 2	None	–	Read/Write	A00Eh
Reserved	–	–	–	A00Fh
HGPIODIR3: General-purpose I/O direction register 3 or Reserved†	None	–	Read/Write	A010h
Reserved	–	–	–	A011h
HGPIODAT3: General-purpose I/O data register 3 or Reserved†	None	–	Read/Write	A012h
Reserved	–	–	–	A013h to A017
HPIC: Control register	Read/Write	HCNTL1 low HCNTL0 low	Read: All bits Write: HINT and DSPINT bits only	A018h
Reserved	–	–	–	A019h
HPIAW: Write address register	Read/Write	❑ Multiplexed mode ❑ HCNTL1 high HCNTL0 low ❑ Single-HPIA mode, or Dual-HPIA mode with HPIAW selected‡	Read-only	A01Ah

† On TMS320VC5502 devices, HGPIODIR3 and HGPIODAT3 are used when the address pins, HA[15:0], are configured for general-purpose I/O. On TMS320VC5501 devices, these register addresses are reserved because there are no address pins.
‡ The single-HPIA mode and the dual-HPIA mode are described in section 2 (page 18).

Table 2. Summary of HPI Registers (Continued)

Register	Host Access		CPU Access	
	Read/Write Permissions	**Access Requirements**	**Read/Write Permissions**	**I/O Address**
Reserved	–	–	–	A01Bh
HPIAR: Read address register	Read/Write	❑ Multiplexed mode ❑ HCNTL1 high HCNTL0 low ❑ Single-HPIA mode, or Dual-HPIA mode with HPIAR selected‡	Read-only	A01Ch
Reserved	–	–	–	A01Dh to A020h
HPID: Data register	Read/Write	With autoincrementing: HCNTL1 low HCNTL0 high No autoincrementing: HCNTL1 high HCNTL0 high	None	None

† On TMS320VC5502 devices, HGPIODIR3 and HGPIODAT3 are used when the address pins, HA[15:0], are configured for general-purpose I/O. On TMS320VC5501 devices, these register addresses are reserved because there are no address pins.
‡ The single-HPIA mode and the dual-HPIA mode are described in section 2 (page 18).

1.2 Summary of the HPI Signals

The external bus selection register (XBSR) in the DSP includes a Parallel/Host Port Mux Mode bit that determines how certain pins are allocated within the DSP. Table 3 describes the effects of this bit on the HPI. At reset, the mux mode selected depends on the GPIO6 pin. If GPIO6 is sampled low at reset, the mux mode bit is forced to 0. If GPIO6 is sampled high at reset, the mux mode bit is forced to 1. For more information about XBSR and GPIO6, see the device-specific data manual: *TMS320VC5501 Fixed-Point Digital Signal Processor Data Manual* (literature number SPRS206) or *TMS320VC5502 Fixed-Point Digital Signal Processor Data Manual* (literature number SPRS166).

Table 3. Effects of the Parallel/Host Port Mux Mode Bit of XBSR

Device	Parallel/Host Port Mux Mode Bit = 0 (Default if GPIO6 Pin Low at Reset)	Parallel/Host Port Mux Mode Bit = 1 (Default if GPIO6 Pin High at Reset)
TMS320VC5501	8-bit multiplexed mode off	8-bit multiplexed mode on
	The pins HBIL, $\overline{\text{HAS}}$, and HD[7:0] are not under the control of the HPI. Host accesses are not possible.	The HPI has control over all pins needed for the 8-bit multiplexed mode.
TMS320VC5502	16-bit multiplexed mode selected	8-bit multiplexed mode selected
	The HPI has control over all pins needed for the 16-bit nonmultiplexed mode. The unneeded pins, HBIL and $\overline{\text{HAS}}$, are not under the control of the HPI.	The HPI has control over all pins needed for the 8-bit multiplexed mode. The unneeded pins, HA[15:0] and HD[15:8], are not under the control of the HPI.

Table 4 summarizes each of the HPI signals. It provides the signal name, the possible states for the signal (input, output, or high-impedance), the connection(s) to be made on the host side of the interface, and a description of the signal's function.

Table 4. HPI Signals

Signal	State†	Host Connection	Description
HPIENA	I	Static/tied high or low.	HPI enable. To enable the HPI for operation, tie this pin high. If this pin is tied low, the HPI is disabled, and all output pins that are not enabled for general-purpose I/O are placed in the high-impedance state.
$\overline{\text{HCS}}$	I	Chip select pin	HPI chip select. $\overline{\text{HCS}}$ must be low for the HPI to be selected by the host. $\overline{\text{HCS}}$ can be kept low between accesses. $\overline{\text{HCS}}$ normally precedes an active $\overline{\text{HDS}}$ (data strobe) signal, but can be connected to an $\overline{\text{HDS}}$ pin for simultaneous select and strobe activity.
$\overline{\text{HDS1}}$ and $\overline{\text{HDS2}}$	I	Read strobe and write strobe pins or any data strobe pin	HPI data strobe pins. These pins are used for strobing data in and out of the HPI (for data strobing details, see section 3.3 on page 25). The direction of the data transfer depends on the logic level of the HR/$\overline{\text{W}}$ signal. The $\overline{\text{HDS}}$ signals are also used to latch control information (if $\overline{\text{HAS}}$ is tied high) on the falling edge. During an HPID write access, data is latched into the HPID register on the rising edge of $\overline{\text{HDS}}$. During read operations, these pins act as output-enable pins of the host data bus.
HCNTL[1:0]	I	Address or control pins	HPI access control inputs. The HPI latches the logic levels of these pins on the falling edge of $\overline{\text{HAS}}$ or internal $\overline{\text{HSTRB}}$ (for details about internal $\overline{\text{HSTRB}}$, see section 3.3 on page 25). The four binary states of these pins determine the access type of the current transfer (HPIC, HPID with autoincrementing, HPIA, HPID).
HR/$\overline{\text{W}}$	I	R/W strobe pin	HPI read/write. On the falling edge of $\overline{\text{HAS}}$ or internal $\overline{\text{HSTRB}}$, HR/$\overline{\text{W}}$ indicates whether the current access is to be a read or write operation. Driving HR/$\overline{\text{W}}$ high indicates the transfer is a read from the HPI, while driving HR/$\overline{\text{W}}$ low indicates a write to the HPI.
HBIL	I	Address or control pins	Byte identification line. In the 8-bit multiplexed mode, the host must use HBIL to identify the first and second bytes of the host cycle. HBIL must be driven low for the first byte and high for the second byte. This signal is ignored when the HPI is operating in the 16-bit nonmultiplexed mode (TMS320VC5502 devices only).

† I = Input, O = Output, Z = High impedance

Table 4. HPI Signals (Continued)

Signal	State†	Host Connection	Description
$\overline{\text{HAS}}$	I	ALE (address latch enable) or address strobe pin	Address strobe. A host with a multiplexed address/data bus can have $\overline{\text{HAS}}$ connected to its ALE pin. The falling edge of $\overline{\text{HAS}}$ is used to latch the logic levels of the HR/$\overline{\text{W}}$, HCNTL1, and HCNTL0 pins, which are typically connected to host address lines. When used, the $\overline{\text{HAS}}$ signal must precede the falling edge of the internal $\overline{\text{HSTRB}}$ signal. Hosts with separate address/data buses must tie this signal high. In this case, the HPI latches the HR/$\overline{\text{W}}$, HCNTL1, and HCNTL0 levels on the falling edge of the internal $\overline{\text{HSTRB}}$ signal.
HA[15:0]	I	Address bus	HPI address bus. In the 16-bit nonmultiplexed mode, the host drives a 15-bit address on this 16-bit bus. HA[14:0] carry the address, and because only 15 address bits are needed, the bit on HA15 is a don't care. In the 8-bit multiplexed mode, HA[15:0] are not used.
HD[15:8] HD[7:0]	I/O/Z	Data bus	HPI data bus. The HPI data bus carries the data to/from the HPI. In the 16-bit nonmultiplexed mode, all 16 lines are used. In the 8-bit multiplexed mode, lines 7 through 0 are used. HD[15:8] are not present on TMS320VC5501 devices.
HRDY	O/Z	Asynchronous ready pin	HPI-ready signal. When the HPI drives HRDY high, the host has permission to complete the current host cycle. When the HPI drives HRDY low, the HPI is not ready for the current host cycle to complete.
$\overline{\text{HINT}}$	O/Z	Interrupt pin	Host Interrupt. The DSP can interrupt the host processor by writing a 1 to the HINT bit of HPIC. Before subsequent HINT interrupts can occur, the host must clear previous interrupts by writing a 1 to the HINT bit. This pin is active-low and inverted from the HINT bit value in HPIC.

† I = Input, O = Output, Z = High impedance

2 Using the Address Registers (8-Bit Multiplexed Mode Only)

The HPI contains two 16-bit address registers: one for read operations (HPIAR) and one for write operations (HPIAW). These roles are unchanging from the viewpoint of the HPI DMA logic. In the 8-bit multiplexed mode of the HPI, the HPI DMA logic gets the address from HPIAR when reading from internal memory and gets the address from HPIAW when writing to internal memory.

However, unlike the HPI DMA logic, the host can choose how to interact with the two HPIA registers. Using the DUALHPIA bit of HPIC, the host determines whether HPIAR and HPIAW act as a single 16-bit register (single-HPIA mode) or as two independent 16-bit registers (dual-HPIA mode).

2.1 Single-HPIA Mode

If DUALHPIA = 0 in HPIC, HPIAR and HPIAW become a single HPIA register from the perspective of the host. In this mode:

- A host HPIA write cycle (HCNTL[1:0] = 10b, HR/$\overline{W}$ = 0) updates HPIAR and HPIAW with the same value.
- Both HPIA registers are incremented during autoincrement read/write cycles (HCNTL[1:0] = 01b).
- An HPIA read cycle (HCNTL[1:0] = 10b, HR/$\overline{W}$ = 1) returns the content of HPIAR, which should be identical to the content of HPIAW.

To maintain consistency between the contents of HPIAR and HPIAW, the host should always re-initialize the HPIA registers after changing the state of the DUALHPIA bit. In addition, when DUALHPIA = 0, the host must always re-initialize the HPIA registers when it changes the data direction (from an HPID read cycle to an HPID write cycle, or vice versa). Otherwise, the memory location accessed by the HPI DMA logic might not be the location intended by the host.

2.2 Dual-HPIA Mode

The host can take advantage of two independent HPIA registers by choosing the dual-HPIA mode (DUALHPIA = 1 in HPIC). In this mode:

- A host HPIA access (HCNTL[1:0] = 10b) reads/updates either HPIAR or HPIAW, depending on the value of the HPIA read/write select (HPIASEL) bit of HPIC. This bit is programmed by the host. While HPIASEL = 1, only HPIAR is read or updated by the host. While HPIASEL = 0, only HPIAW is read or updated by the host. The HPIASEL bit is only meaningful in the dual-HPIA mode.

Note:

The HPIASEL bit does not affect the HPI DMA logic. Regardless of the value of HPIASEL, the HPI DMA logic uses HPIAR when reading from memory and HPIAW when writing to memory.

- A host HPID access with autoincrementing (HCNTL[1:0] = 01b) causes only the relevant HPIA value to be incremented to the next consecutive memory address. In an autoincrement read cycle, HPIAR is incremented after it has been used to perform the current read from memory. In an autoincrement write cycle, HPIAW is incremented after it has been used for the write operation.

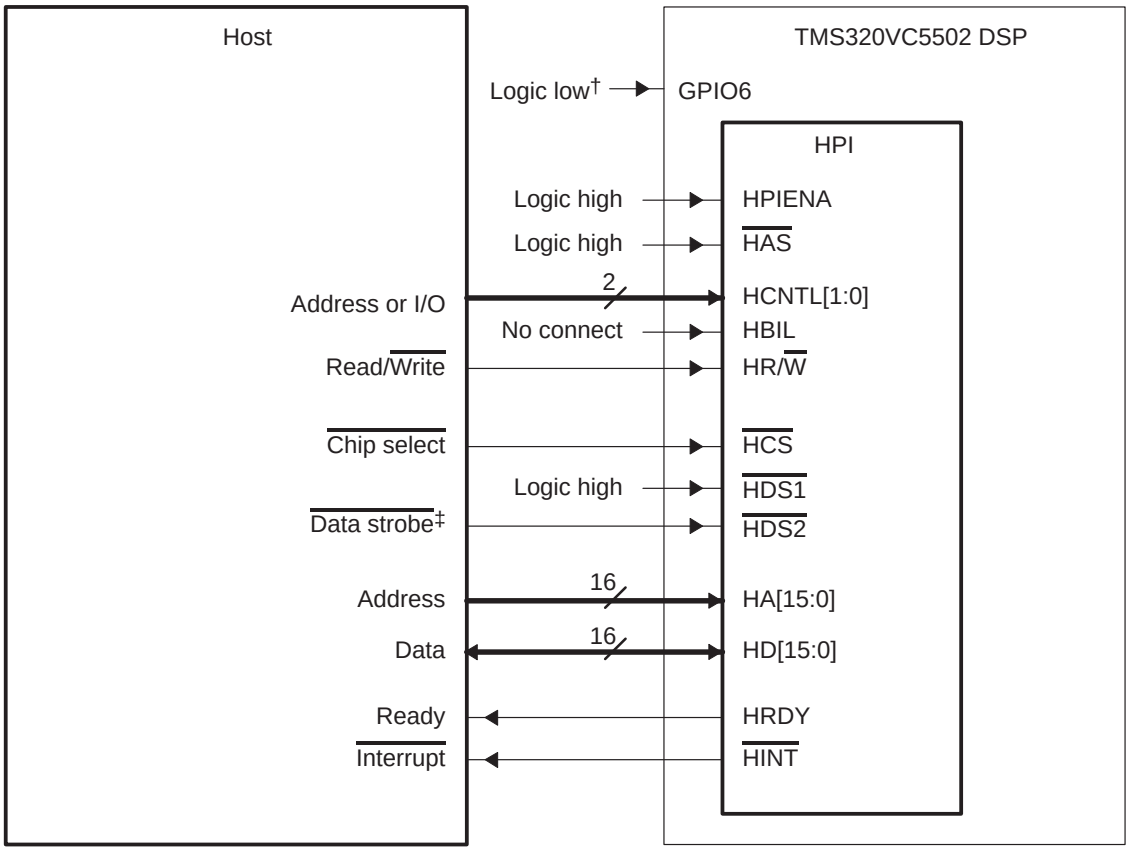
3 HPI Operation

3.1 Host-HPI Signal Connections

Figure 2 shows an example of signal connections for the 16-bit nonmultiplexed mode. Figure 3 and Figure 4 show a similar examples for the 8-bit multiplexed mode. In Figure 3, the $\overline{\text{HAS}}$ signal is used as described in section 3.6 (page 29). In Figure 4, $\overline{\text{HAS}}$ is tied high (not used). The following are key points of comparison between the signal connections in the two interface modes:

- The pin GPIO6 of the DSP must be held low at reset for the 16-bit nonmultiplexed mode (TMS320VC5502 devices only) or held high at reset for the 8-bit multiplexed mode.
- The address strobe ($\overline{\text{HAS}}$) of the HPI is not used in the 16-bit nonmultiplexed mode and is optional in the 8-bit multiplexed mode.
- The byte identification line (HBIL) of the HPI is not used in the 16-bit nonmultiplexed mode but is required in the 8-bit multiplexed mode.
- The address bus (HA[15:0]) of the HPI is used in the 16-bit nonmultiplexed mode but is not used in the 8-bit multiplexed mode.

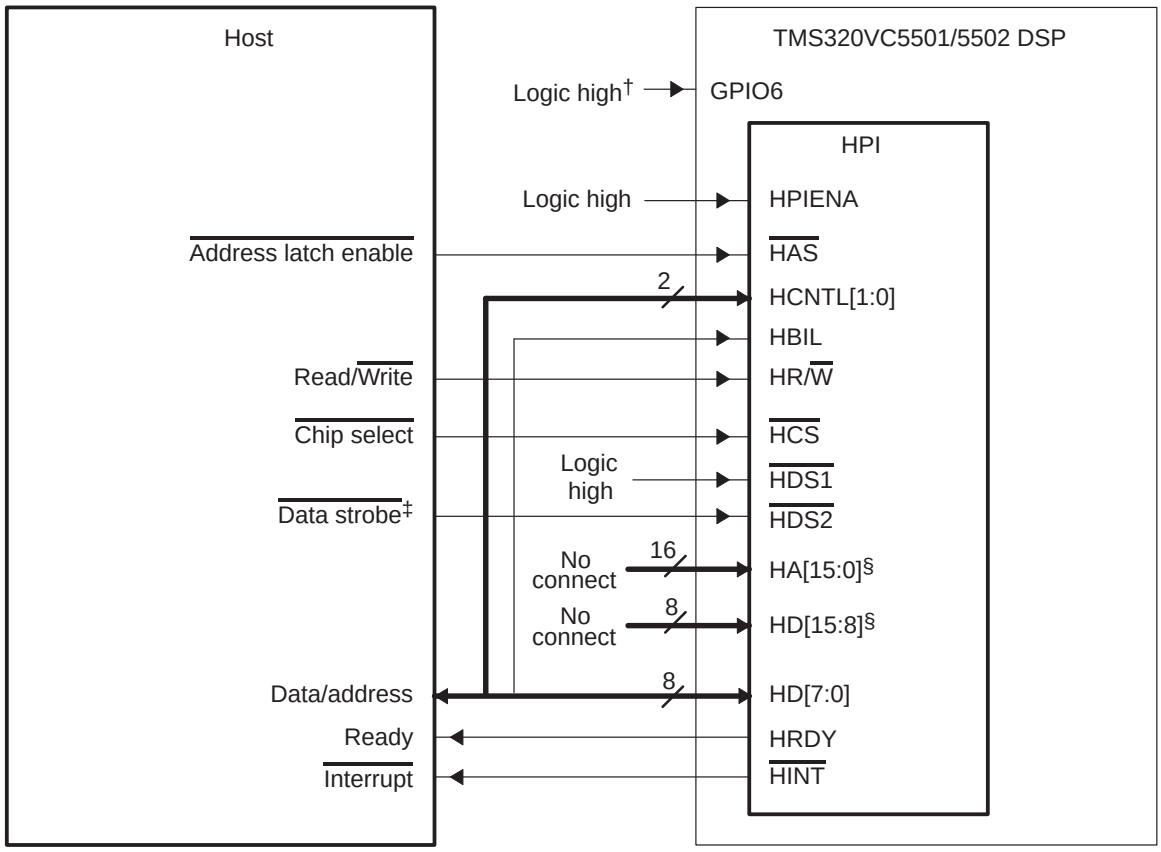
Figure 2. Example of Host-DSP Signal Connections in the 16-Bit Nonmultiplexed Mode



† GPIO6 does not have to be tied high or low. This pin is sampled at reset. On TMS320VC5501 devices, if GPIO6 is sampled low at reset, the HPI is disabled.

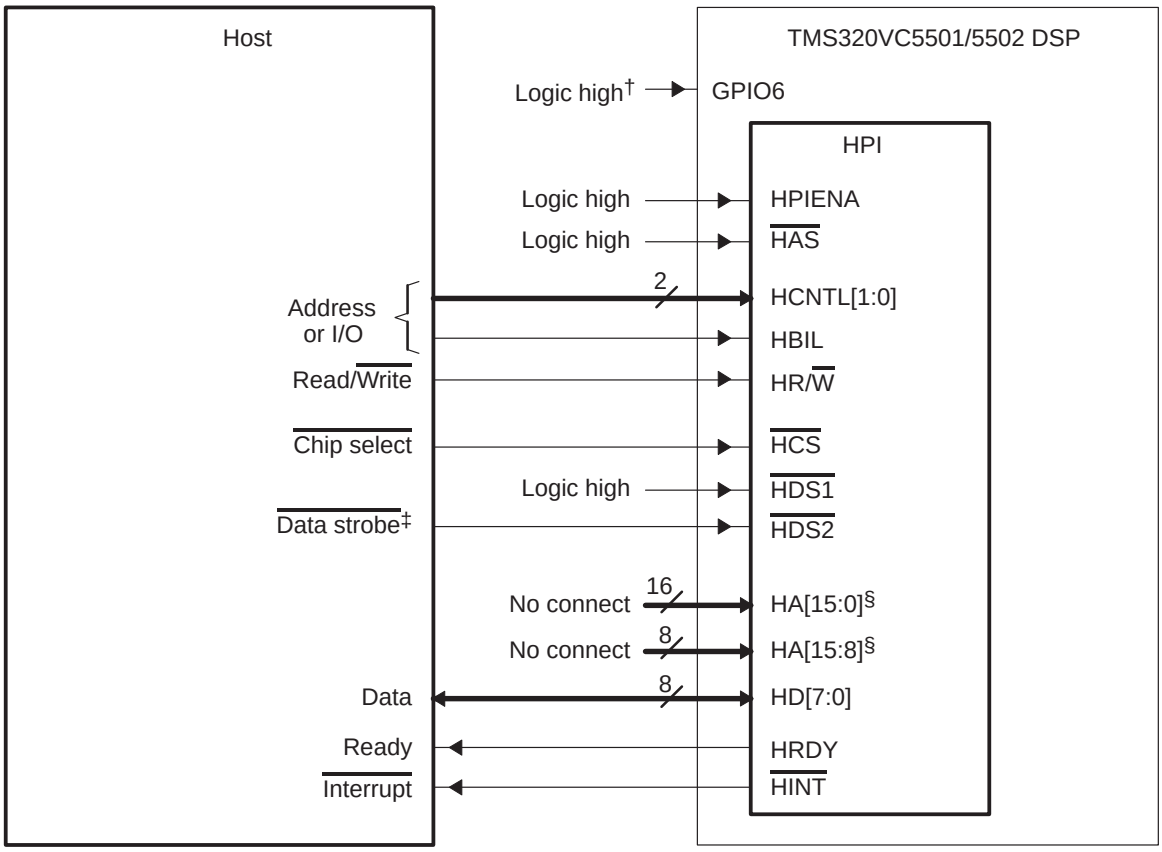
‡ Data strobing options are given in section 3.3 (page 25).

Figure 3. Example of Host-DSP Signal Connections When Using the $\overline{HAS}$ Signal in the 8-Bit Multiplexed Mode



† GPIO6 does not have to be tied high or low. This pin is sampled at reset.
‡ Data strobing options are given in section 3.3 (page 25).
§ HA[15:0] and HD[15:8] are not present on TMS320VC5501 devices.

Figure 4. Example of Host-DSP Signal Connections When the $\overline{HAS}$ Signal is Tied High in the 8-Bit Multiplexed Mode



† GPIO6 does not have to be tied high or low. This pin is sampled at reset.
‡ Data strobing options are given in section 3.3 (page 25).
§ HA[15:0] and HD[15:8] are not present on TMS320VC5501 devices.

3.2 HPI Configuration and Data Flow

The way the host accomplishes accesses varies slightly depending on the selected interface mode. In the 8-bit multiplexed mode:

1) The host writes to the control register (HPIC) to properly configure the HPI. Typically, this means programming the byte-order bit (BOB) and the HPIA-related bits (HPIADUAL and HPIASEL). This step is normally performed once before the initial data access. For information on the control bits in HPIC, see section 9.1 on page 57.

2) The host writes the desired internal DSP memory address to an address register (HPIAR and/or HPIAW). For an introduction to the two HPIA registers and the two ways the host can interact with them, see section 2 on page 18.

3) The host reads from or writes to the data register (HPID). Data transfers between HPID and the internal memory of the DSP are handled by the HPI DMA logic.

In the 16-bit nonmultiplexed mode (TMS320VC5502 devices only), no HPIC programming is required. The host can immediately read from or write to the data register (HPID). When initiating an HPID read or write cycle, the host drives the memory address on the dedicated address lines, HA[15:0]. Only 15 address bits are needed to access the 32K words of internal memory that are accessible to the HPI; therefore, the MSB of the address (on line HA15) is a don't care. Data transfers between HPID and the internal memory of the DSP are handled by the HPI DMA logic.

Regardless of the chosen interface mode, each step of the access uses the same bus. Therefore, the host must drive the appropriate levels on the HCNTL1 and HCNTL0 signals to indicate which register is to be accessed. The host must also drive the appropriate level on the HR/$\overline{W}$ signal to indicate the data direction (read or write) and must drive other control signals as appropriate. When HPI resources are temporarily busy or unavailable, the HPI can communicate this to the host by deasserting the HPI-ready (HRDY) output signal.

When performing an access, the HPI first latches the levels on HCNTL[1:0], HR/$\overline{W}$, and other control signals. In the 8-bit multiplexed mode, this latching can occur on the falling edge of the internal strobe signal (for details, see section 3.3 on page 25) or the falling edge of $\overline{HAS}$ (for details, see section 3.6 on page 29). In the 16-bit nonmultiplexed mode, this latching must occur on the falling edge of the internal strobe signal. After the control information is latched, the HPI initiates an access based on the control signals.

If the host wants to read data from the internal memory, the HPI DMA logic reads the memory address from HPIAR and retrieves the data from the addressed memory location. When the data has been placed in HPID, the HPI drives the data onto its HD bus. The HRDY signal informs the host whether the data on the HD bus is valid (HRDY high) or not valid yet (HRDY low). When the data is valid, the host should latch the data and drive the connected data strobe ($\overline{\text{HDS1}}$ or $\overline{\text{HDS2}}$) inactive, which, in turn, will cause the internal strobe (internal $\overline{\text{HSTRB}}$) signal to transition from low to high.

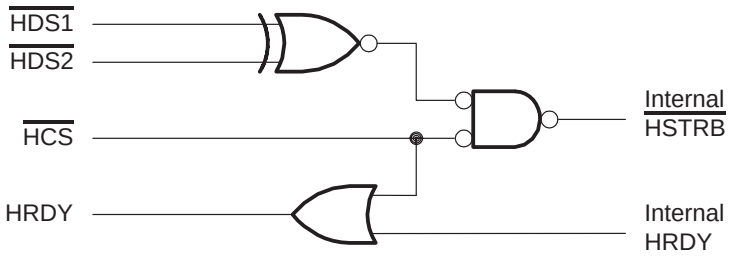
If the host wants to write data to the internal memory, the operation is similar. After the host determines that the HPI is ready to latch the data (HRDY is high), it must cause internal $\overline{\text{HSTRB}}$ to transition from low to high, which causes the data to be latched into HPID. Once the data is in HPID, the HPI DMA logic reads the memory address from HPIAW and transfers the data from HPID to the addressed memory location.

The CPU, the HPI DMA logic, and the DMA controller all can access the internal memory of the DSP. In the case when the CPU and the HPI DMA logic simultaneously request access to the same DARAM block in internal memory, CPU requests always have priority over HPI DMA logic requests. The same is true when the CPU and the DMA controller access the same DARAM block. Both HPI DMA logic and DMA controller requests to a DARAM block are serviced when there are no more CPU requests. If both the HPI DMA logic and the DMA controller access the same half of the on-chip DARAM, the requests from those two modules are serviced in a round-robin fashion. The bottom half of DARAM consists of memory blocks DARAM0−DARAM3, and the top half consists of blocks DARAM4−DARAM7. See the device-specific data manual for the start and end addresses of each DARAM block.

3.3 $\overline{\text{HDS2}}$, $\overline{\text{HDS1}}$, and $\overline{\text{HCS}}$: Data Strobing and Chip Selection

As illustrated in Figure 5, the strobing logic is a function of three key inputs: the chip select pin ($\overline{\text{HCS}}$) and two data strobe signals ($\overline{\text{HDS1}}$ and $\overline{\text{HDS2}}$). The internal strobe signal, which is referred to as internal $\overline{\text{HSTRB}}$ throughout this document, functions as the actual strobe signal inside the HPI. $\overline{\text{HCS}}$ must be low (HPI selected) during strobe activity on the $\overline{\text{HDS}}$ pins. If $\overline{\text{HCS}}$ remains high (HPI not selected), activity on the $\overline{\text{HDS}}$ pins is ignored.

Figure 5. HPI Strobe and Select Logic



Strobe connections between the host and the HPI depend in part on the number and types of strobe pins available on the host. Table 5 describes some options for connecting to the HDS pins.

Notice in Figure 5 that HRDY is also gated by HCS. If HCS goes high (HPI not selected), HRDY goes high, regardless of whether the current internal transfer is completed in the DSP.

Table 5. Options for Connecting Host and HPI Data Strobe Pins

Available Host Data Strobe Pins	Connections to HPI Data Strobe Pins
Host has separate read and write strobe pins, both active-low	Connect one strobe pin to HDS1 and the other to HDS2†
Host has one active-low strobe pin	Connect the strobe pin to HDS1 or HDS2, and connect the other pin to logic level 1.
Host has one active-high strobe pin	Connect the strobe pin to HDS1 or HDS2, and connect the other strobe pin to logic level 0.

† The HR/W signal could be driven by a host address line in this case.

Note:

1) The HCS input and one HDS strobe input can be tied together and driven with a single strobe signal from the host. This technique selects the HPI and provides the strobe simultaneously. When using this method, be aware that HRDY is gated by HCS as previously described.
2) It is not recommended to tie both HDS1 and HDS2 to static logic levels and use HCS as a strobe.

3.4 HCNTL[1:0] and HR/$\overline{W}$: Indicating the Cycle Type

The cycle type consists of:

- The access type that the host selects by driving the appropriate levels on the HCNTL[1:0] pins of the HPI. Table 6 describes the four available access types.
- The transfer direction that the host selects with the HR/$\overline{W}$ pin. The host must drive the HR/$\overline{W}$ signal high (read) or low (write).

A summary of cycle types is in Table 7. The HPI samples the HCNTL levels either at the falling edge of $\overline{HAS}$ (if $\overline{HAS}$ is used in the 8-bit multiplexed mode) or at the falling edge of the internal strobe signal $\overline{HSTRB}$ (if $\overline{HAS}$ is not used and is tied high). In the 16-bit nonmultiplexed mode, the HPIA registers are not used and, therefore, only HPID accesses without autoincrementing and HPIC accesses are valid.

Table 6. Access Types Selectable With the HCNTL Signals

HCNTL1	HCNTL0	Access Type
0	0	HPIC access. The host requests to access the HPI control register (HPIC).
0	1	HPID access with autoincrementing. *This is only a valid option in the multiplexed mode.* The host requests to access the HPI data register (HPID) and to have the appropriate HPI address register (HPIAR and/or HPIAW) automatically incremented by 1 after the access.
1	0	HPIA access. *This is only a valid option in the multiplexed mode.* The host requests to access the appropriate HPI address register (HPIAR and/or HPIAW).
1	1	HPID access without autoincrementing. The host requests to access the HPI data register (HPID) but requests no automatic post-increment of the HPI address register.

Table 7. Cycle Types Selectable With the HCNTL and HR/$\overline{W}$ Signals

HCNTL1	HCNTL0	HR/$\overline{W}$	Cycle Type
0	0	0	HPIC write cycle
0	0	1	HPIC read cycle
0	1	0	HPID write cycle with autoincrementing
0	1	1	HPID read cycle with autoincrementing
1	0	0	HPIA write cycle
1	0	1	HPIA read cycle
1	1	0	HPID write cycle without autoincrementing
1	1	1	HPID read cycle without autoincrementing

3.5 HBIL: Identifying the First and Second Bytes in Multiplexed-Mode Transfers

In the 8-bit multiplexed mode, each host cycle consists of two consecutive byte transfers. For each transfer, the host must specify the cycle type with HCNTL[1:0] and HR/$\overline{W}$, and the host must use HBIL to indicate whether the first or second byte is being transferred. For HPID and HPIA accesses, HBIL must always be driven low for the first byte transfer and high for the second byte transfer. Results are undefined if the sequence is broken. For examples of using HBIL, see the figures in sections 3.6 and 3.7. (Section 3.6 begins on page 29.)

When the host sends the two bytes of a 16-bit word in this manner, the host can send the most and the least significant bytes of the word in either order (most significant byte first or most significant byte second). However, the host must inform the HPI of the selected order before beginning the host cycle. This is done by programming the byte order (BOB) bit of HPIC. Although BOB is written at bit 0 in HPIC, it's current value is readable at both bit 0 and bit 8 (BOBSTAT). Thus, the host can determine the current byte-order configuration by checking the least significant bit of either half of HPIC.

There is one case when the 8-bit multiplexed mode does not require a dual-byte cycle with HBIL low for the first byte and HBIL high for the second byte. Either byte of the HPIC can be accessed with a single-byte cycle. During such a cycle, the host drives HBIL high or low, and as with a dual-byte cycle, the current value of the BOB bit determines which byte of HPIC is accessed. For an example timing diagram of this case, see section 3.8 on page 33.

In the 16-bit nonmultiplexed mode, each host cycle is one word transfer. The HBIL signal is ignored and 16 bits of data are transferred for each active cycle of the internal strobe signal (internal $\overline{\text{HSTRB}}$).

3.6 $\overline{\text{HAS}}$: Forcing the HPI to Latch Control Information Early in the 8-Bit Multiplexed Mode

The $\overline{\text{HAS}}$ signal is only valid in the 8-bit multiplexed mode. This signal is an address strobe that allows control information to be removed earlier in a host cycle, which allows more time to switch bus states from address to data information.

Figure 3 (page 22) shows an example of signal connections when $\overline{\text{HAS}}$ is used for multiplexed transfers. Figure 6 and Figure 7 (following) show typical HPI signal activity when $\overline{\text{HAS}}$ is used.

The process for using $\overline{\text{HAS}}$ is as follows:

1) The host selects the access type.

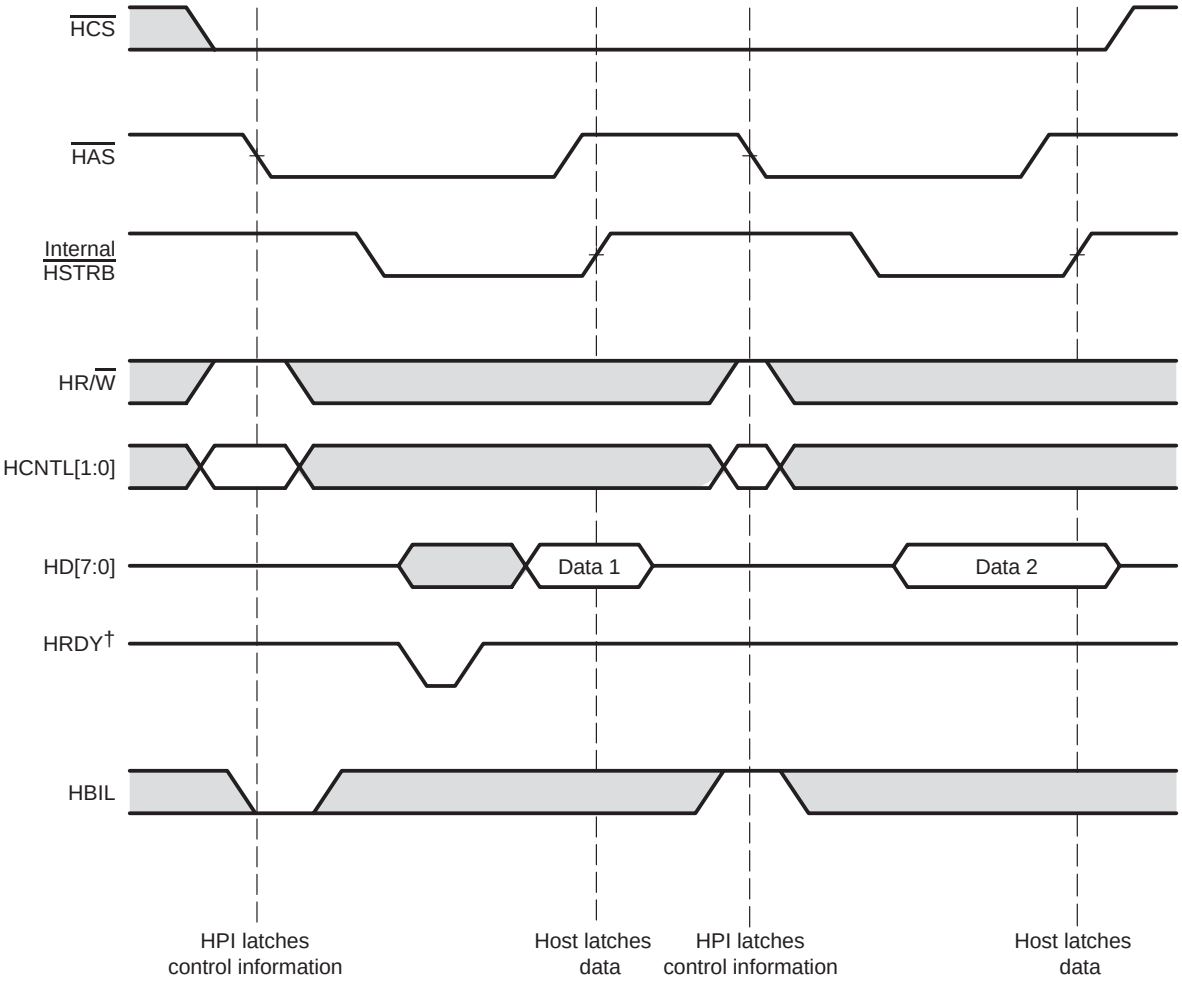
   The host drives the appropriate levels on the HCNTL [1:0] and HR/$\overline{\text{W}}$ signals, and indicates which byte (first or second) will be transferred by driving HBIL high or low.

2) The host drives $\overline{\text{HAS}}$ low.

   On the falling edge of $\overline{\text{HAS}}$, the HPI latches the states of the HCNTL[1:0], HR/$\overline{\text{W}}$, and HBIL. The high to low transition of $\overline{\text{HAS}}$ must precede the falling edge of the internal strobe signal (internal $\overline{\text{HSTRB}}$), which is derived from $\overline{\text{HCS}}$, $\overline{\text{HDS1}}$, and $\overline{\text{HDS2}}$ as described in section 3.3 (page 25).

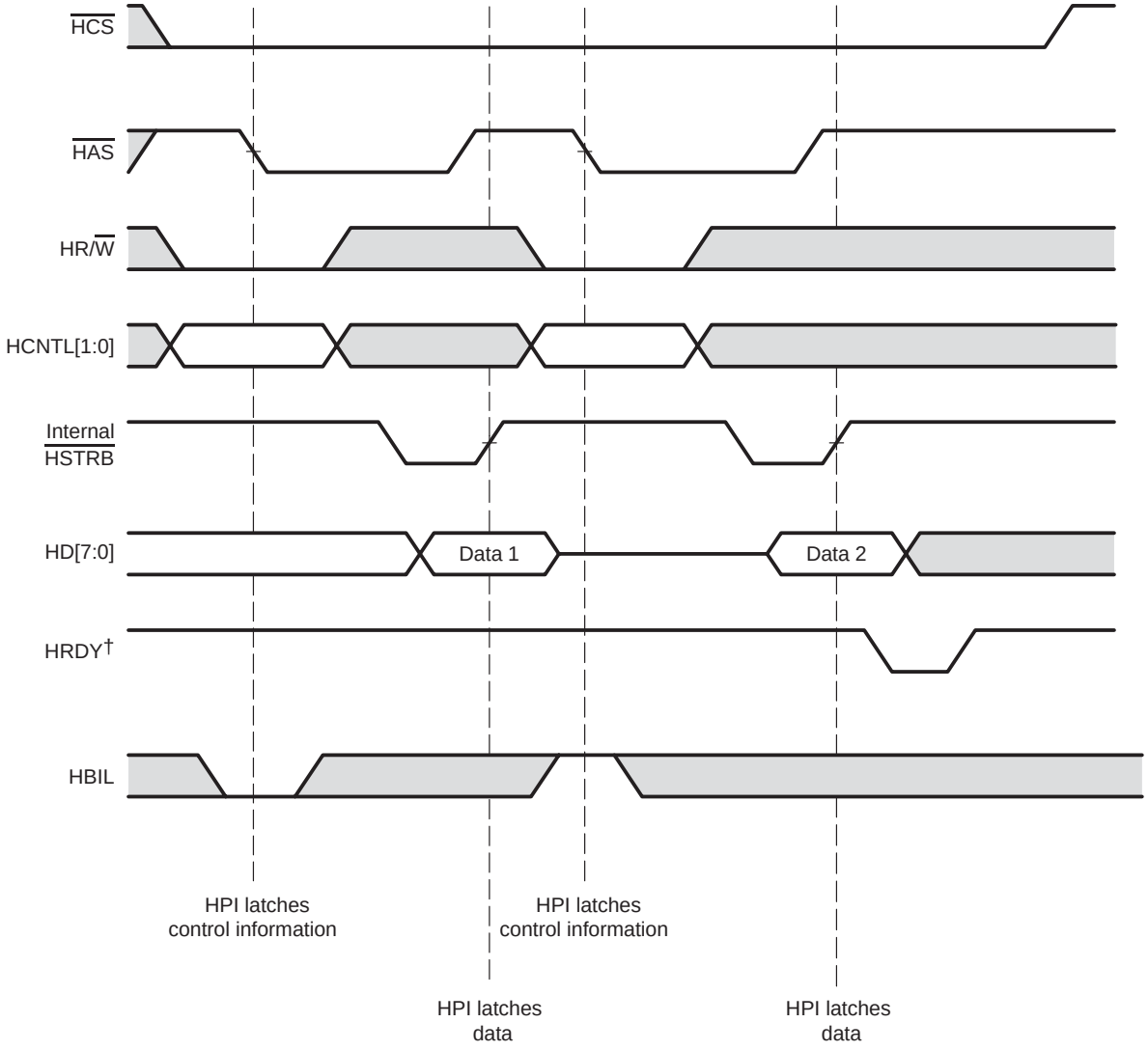
The $\overline{\text{HAS}}$ input is not gated by $\overline{\text{HCS}}$, therefore allowing time for the host to perform the subsequent access. The $\overline{\text{HAS}}$ signal may be brought high after internal $\overline{\text{HSTRB}}$ goes low, indicating that the data access is about to occur. $\overline{\text{HAS}}$ is not required to be driven high at any time during the cycle, but eventually must transition high before the host uses it for another access with different values for HCNTL [1:0], HR/$\overline{\text{W}}$, and HBIL.

Figure 6. Multiplexed-Mode Host Read Cycle Using $\overline{HAS}$



† Depending on the type of write operation (HPID without autoincrementing, HPIA, HPIC, or HPID with autoincrementing) and the state of the FIFO, transitions on HRDY may or may not occur. For more information, see section 3.10, *Hardware Handshaking Using the HPI-Ready (HRDY) Signal*, on page 36.

Figure 7. Multiplexed-Mode Host Write Cycle Using $\overline{HAS}$



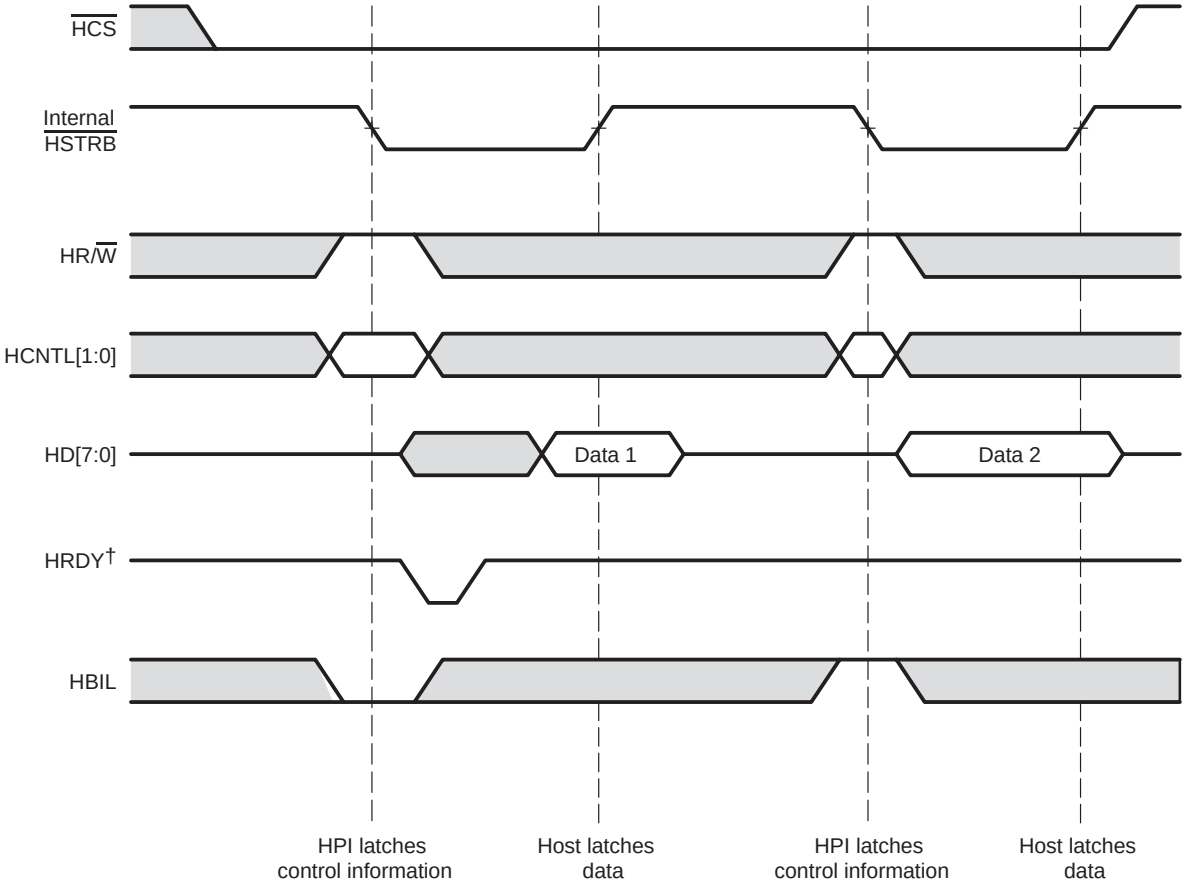
† Depending on the type of write operation (HPID without autoincrementing, HPIA, HPIC, or HPID with autoincrementing) and the state of the FIFO, transitions on HRDY may or may not occur. For more information, see section 3.10, *Hardware Handshaking Using the HPI-Ready (HRDY) Signal*, on page 36.

3.7 Performing a Multiplexed Access Without $\overline{HAS}$

In cases where the host processor has dedicated signals (address lines or bit I/O) capable of driving the control lines, the $\overline{HAS}$ signal is not required. Dedicated pins can be directly connected to HCNTL[1:0], HR/$\overline{W}$, and HBIL.

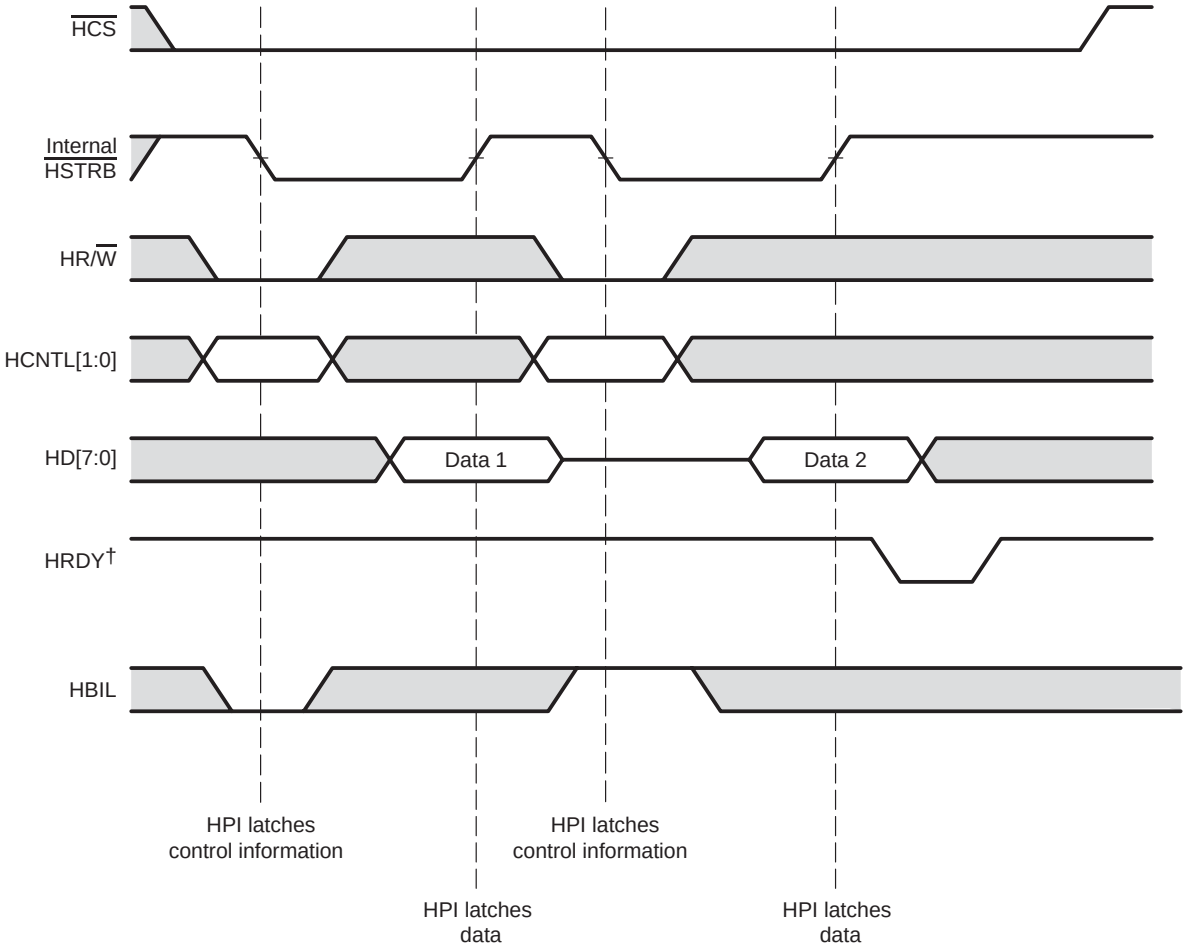
Figure 4 (page 23) shows an example of signal connections when $\overline{HAS}$ is not used for multiplexed transfers. When $\overline{HAS}$ is not used, it must be tied high (inactive). Figure 8 and Figure 9 (following) show typical HPI signal activity when $\overline{HAS}$ is tied high. In these cases, the falling edge of internal $\overline{HSTRB}$ is used to latch the HCNTL[1:0], HR/$\overline{W}$, and HBIL states into the HPI. Internal $\overline{HSTRB}$ is derived from $\overline{HCS}$, $\overline{HDS1}$, and $\overline{HDS2}$ as described in section 3.3 (page 25).

Figure 8. Multiplexed-Mode Host Read Cycle With $\overline{HAS}$ Tied High



† Depending on the type of write operation (HPID without autoincrementing, HPIA, HPIC, or HPID with autoincrementing) and the state of the FIFO, transitions on HRDY may or may not occur. For more information, see section 3.10, *Hardware Handshaking Using the HPI-Ready (HRDY) Signal*, on page 36.

Figure 9. Multiplexed-Mode Host Write Cycle With $\overline{HAS}$ Tied High

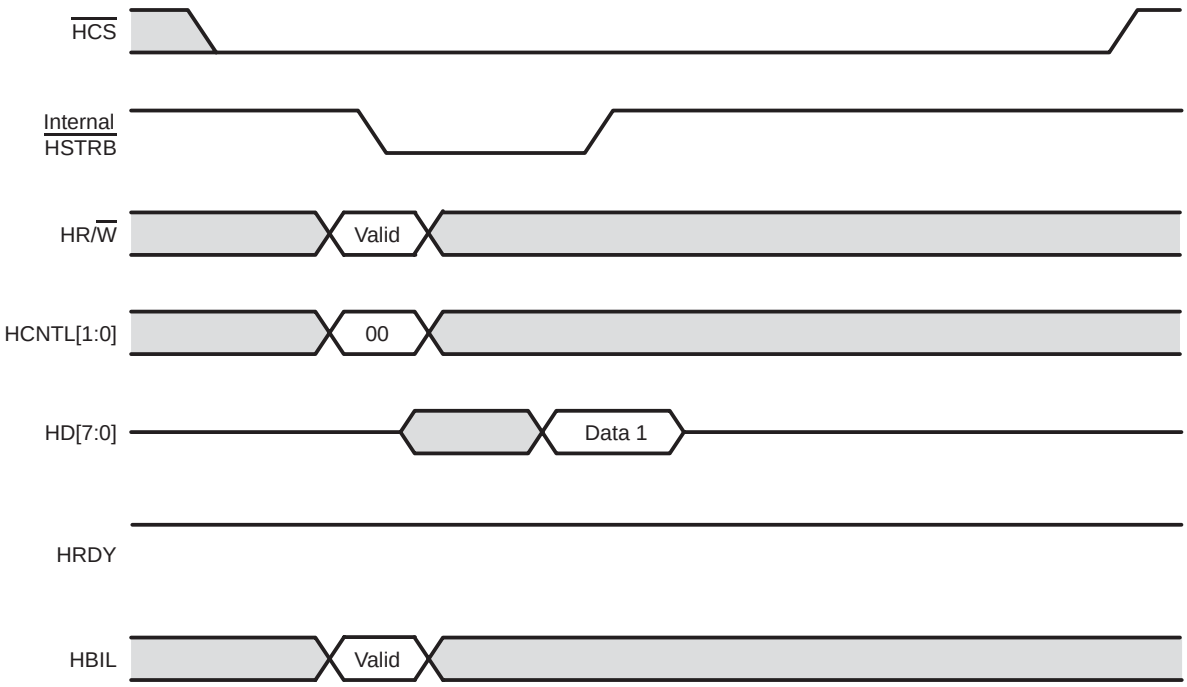


† Depending on the type of write operation (HPID without autoincrementing, HPIA, HPIC, or HPID with autoincrementing) and the state of the FIFO, transitions on HRDY may or may not occur. For more information, see section 3.10, *Hardware Handshaking Using the HPI-Ready (HRDY) Signal*, on page 36.

3.8 Single-Byte HPIC Cycle in the 8-Bit Multiplexed Mode

Figure 10 shows the special case (see section 3.5 on page 28) in which the host performs a single-byte cycle to access the HPIC. The host drives HBIL high or low, and the current value of the BOB bit determines whether the low byte or the high byte of HPIC is accessed. Although the example in Figure 10 has the $\overline{HAS}$ signal tied high, this type of HPIC cycle can also be done using the $\overline{HAS}$ signal to force early latching of control information.

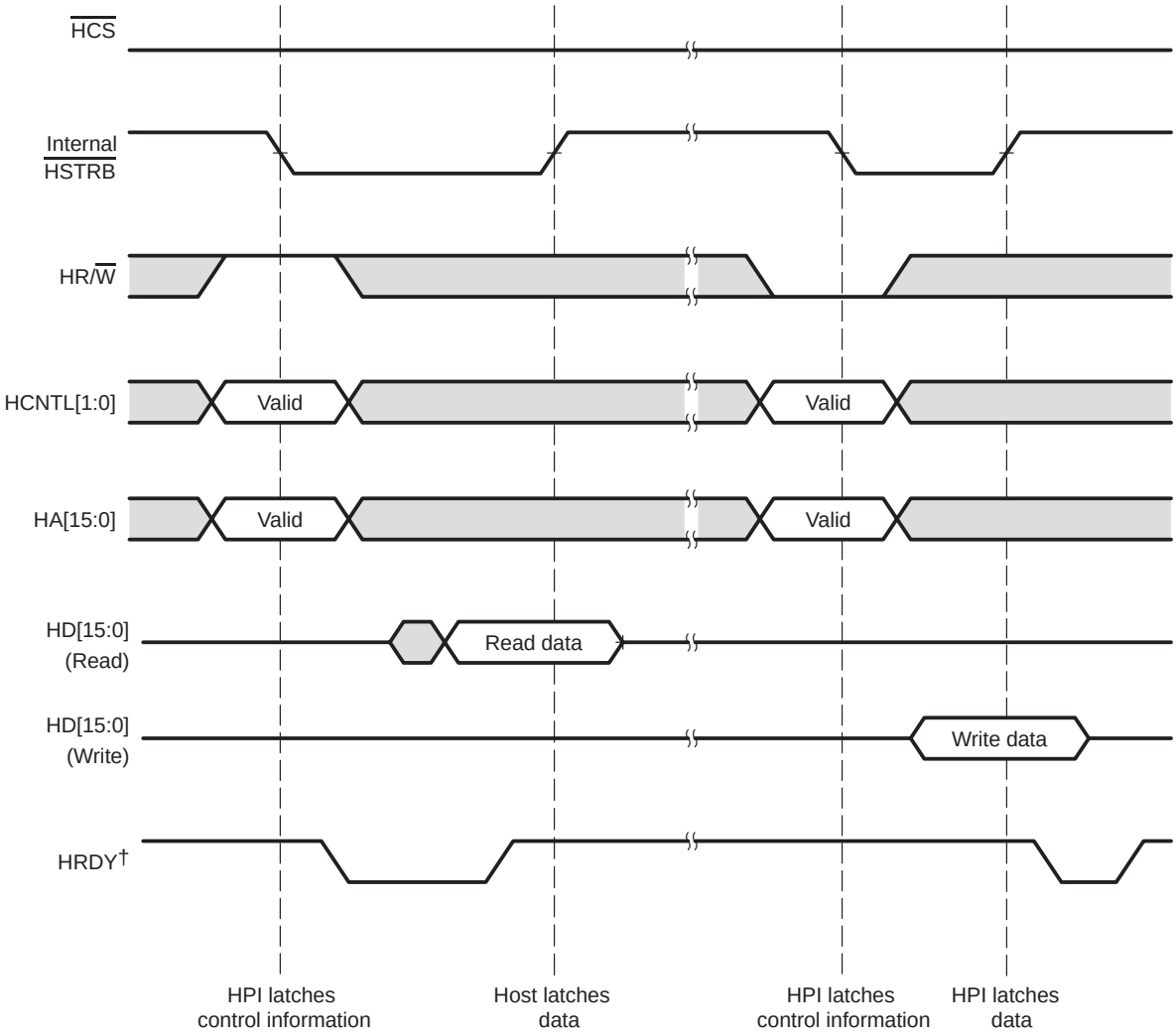
Figure 10. Multiplexed-Mode Single-Byte HPIC Cycle With $\overline{HAS}$ Tied High (Read or Write)



3.9 Host Cycles in the 16-Bit Nonmultiplexed Mode

Figure 2 (page 21) shows an example of signal connections for nonmultiplexed transfers. Figure 11 (following) shows typical read cycle and a typical write cycle in the 16-bit nonmultiplexed mode. In each cycle consists of a single 16-bit transfer. The falling edge of internal $\overline{HSTRB}$ is used to latch the HCNTL[1:0], HR/$\overline{W}$, and address information into the HPI. Internal $\overline{HSTRB}$ is derived from $\overline{HCS}$, $\overline{HDS1}$, and $\overline{HDS2}$ as described in section 3.3 (page 25).

Figure 11. Nonmultiplexed-Mode Host Read Cycle and Host Write Cycle



† Depending on the type of write operation (HPID without autoincrementing, HPIA, HPIC, or HPID with autoincrementing) and the state of the FIFO, transitions on HRDY may or may not occur. For more information, see section 3.10, *Hardware Handshaking Using the HPI-Ready (HRDY) Signal*, on page 36.

3.10 Hardware Handshaking Using the HPI-Ready (HRDY) Signal

The HPI uses its ready signal, HRDY, to tell the host whether it is ready to complete an access. During a read cycle, the HPI is ready (drives HRDY high) when it has data available for the host. During a write cycle, the HPI is ready (drives HRDY high) when it is ready to latch data from the host. If the HPI is not ready, it can drive HRDY low to insert wait states. These wait states indicate to the host that read data is not yet valid (read cycle) or that the HPI is not ready to latch write data (write cycle). The number of wait states that must be inserted by the HPI is dependent upon the state of the resource that is being accessed. See the device-specific data manual for latency information.

Note:

In some cases, if the host does not have an input pin to connect to the HRDY pin, the host can check the readiness of the HPI by polling the HRDY bit in the control register (HPIC). For details, see section 4 on page 42.

When the HPI is not ready to complete the current cycle (HRDY low), the host can begin a new host cycle by forcing the HPI to latch new control information. However, once the cycle has been initiated, the host must wait until HRDY goes high before causing a rising edge on the internal strobe signal (internal $\overline{\text{HSTRB}}$) to complete the cycle. If internal $\overline{\text{HSTRB}}$ goes high when the HPI is not ready, the cycle will be terminated with invalid data being returned (read cycle) or written (write cycle).

One reason the HPI may drive HRDY low is a not-ready condition in one of its first-in, first-out buffers (FIFOs). For example, any HPID access that occurs while the write FIFO is full or the read FIFO is empty may result in some number of wait states being inserted by the HPI. The FIFOs are explained in section 6 (page 47).

The following sections describe the behavior of HRDY during HPI register accesses in the multiplexed and nonmultiplexed HPI modes. Note that the nonmultiplexed mode is not supported on TMS320VC5501 devices. In all cases, the chip select signal, $\overline{\text{HCS}}$, must be asserted for HRDY to go low.

3.10.1 HRDY Behavior During Multiplexed-Mode Read Operations

Figure 12 shows an HPIC (HCNTL[1:0] = 00b) or HPIA (HCNTL[1:0] = 10b) read cycle during multiplexed HPI operation. Neither an HPIC read cycle nor an HPIA read cycle causes HRDY to go low.

Figure 12. HRDY Behavior During an HPIC or HPIA Read Cycle in the Multiplexed Mode

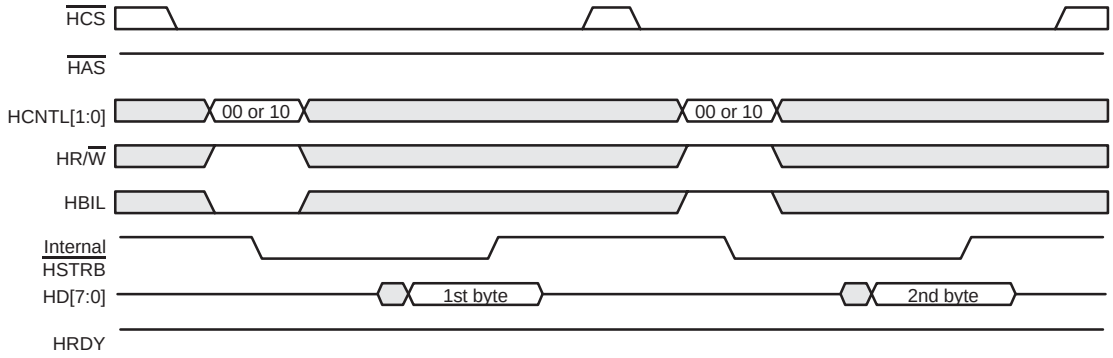


Figure 13 includes an HPID read cycle without autoincrementing in the mutliplexed mode. The host writes the memory address during the HPIA (HCNTL[1:0] = 10b) write cycle, and the host reads the data during the HPID (HCNTL[1:0] = 11b) read cycle. HRDY goes low for each HPIA byte access, but HRDY goes low for only the first byte access in each HPID read cycle.

Figure 13. HRDY Behavior During a Data Read Operation in the Multiplexed Mode (Case 1: HPIA Write Cycle Followed by Nonautoincrement HPID Read Cycle)

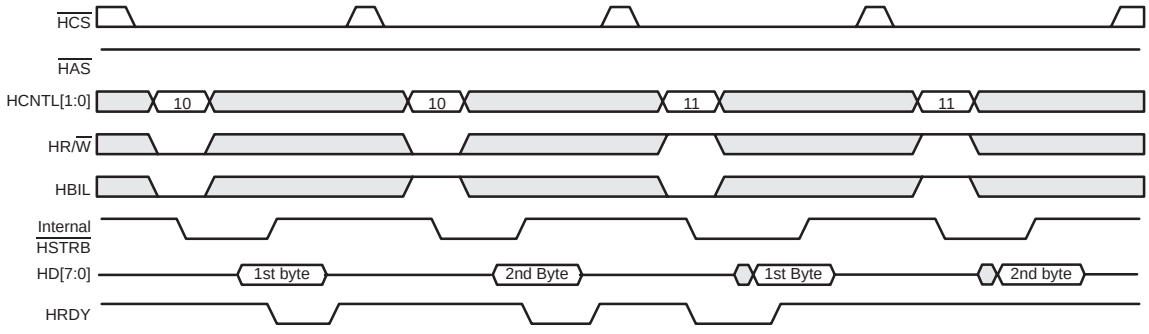
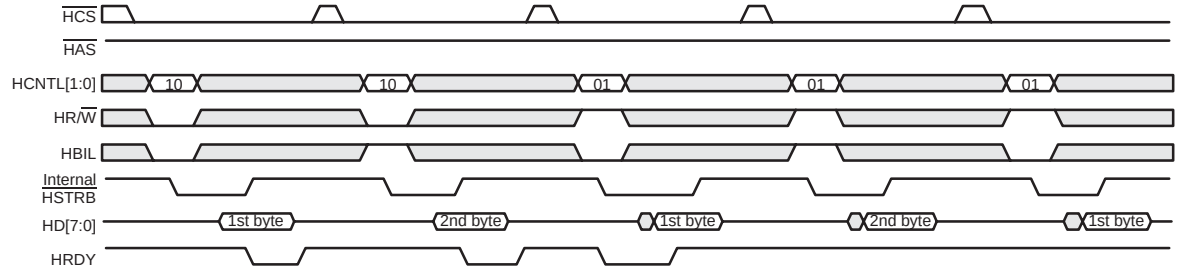


Figure 14 includes an autoincrement HPID read cycle in the multiplexed mode. The host writes the memory address while asserting HCNTL[1:0] = 10b and reads the data while asserting HCNTL[1:0] = 01b. During the first HPID read cycle, HRDY goes low for only the first byte access, and subsequent HPID read cycles do not cause HRDY to go low.

Figure 14. HRDY Behavior During a Data Read Operation in the Multiplexed Mode (Case 2: HPIA Write Cycle Followed by Autoincrement HPID Read Cycles)



3.10.2 HRDY Behavior During Multiplexed-Mode Write Operations

Figure 15 shows an HPIC (HCNTL[1:0] = 00b) write cycle during multiplexed HPI operation. An HPIC write cycle does not cause HRDY to go low.

Figure 15. HRDY Behavior During an HPIC Write Cycle in the Multiplexed Mode

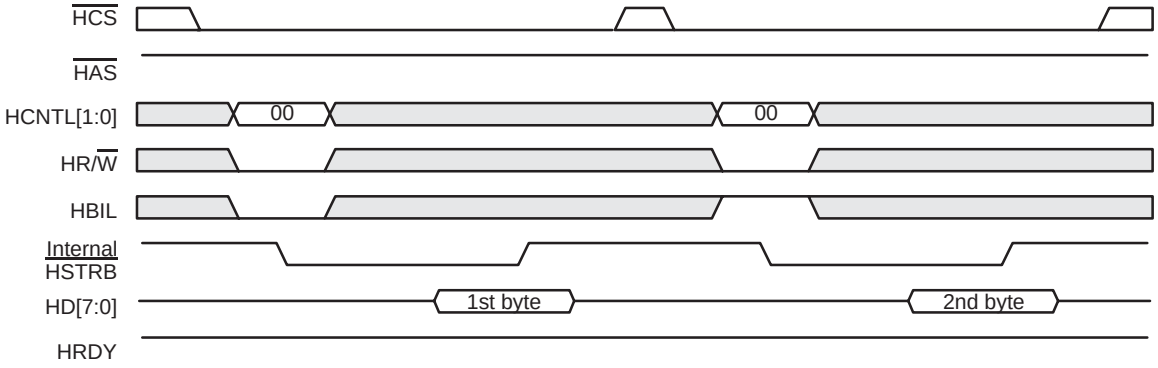


Figure 16 includes a HPID write cycle without autoincrementing in the multiplexed mode. The host writes the memory address while HCNTL[1:0] = 10b and writes the data while HCNTL[1:0] = 11b. During the HPID write cycle, HRDY goes low only for the second byte access.

Figure 16. HRDY Behavior During a Data Write Operation in the Multiplexed Mode (Case 1: No Autoincrementing)

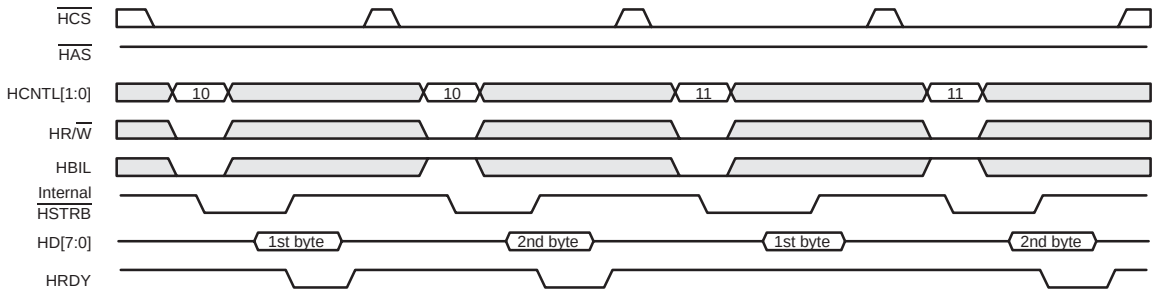


Figure 17 shows autoincrement HPID write cycles in the multiplexed mode when the write FIFO is empty prior to the HPIA write. The host writes the memory address while HCNTL[1:0] = 10b and writes the data while HCNTL[1:0] = 01b. HRDY does not go low during any of the HPID write cycles.

Figure 17. HRDY Behavior During a Data Write Operation in the Multiplexed Mode (Case 2: Autoincrementing Selected, FIFO Empty Before Write)

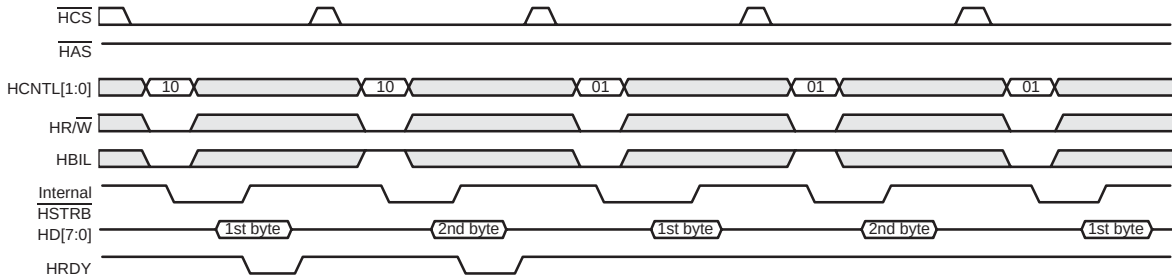
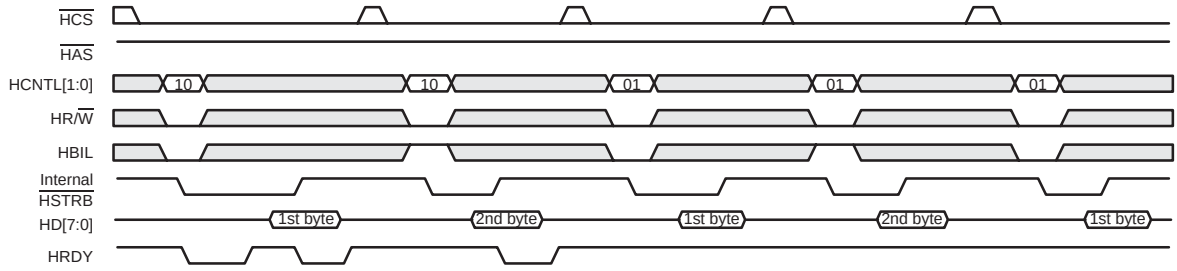


Figure 18 shows a case similar to that of Figure 17. However, in the case of Figure 18, the write FIFO is not empty when the HPIA access is made. HRDY goes low twice for the first byte access of the HPIA write cycle. The first HRDY low period is due to the non-empty FIFO. The data currently in the FIFO must first be written to the memory. This results in HRDY going low immediately after the falling edge of the data strobe ($\overline{\text{HSTRB}}$). The second and third HRDY low periods occur for the writes to the HPIA. HRDY remains high for the HPID accesses.

Figure 18. HRDY Behavior During a Data Write Operation in the Multiplexed Mode (Case 3: Autoincrementing Selected, FIFO Not Empty Before Write)



3.10.3 HRDY Behavior During Nonmultiplexed-Mode Read Operations

Figure 19 shows an HPIC (HCNTL[1:0] = 00b) read cycle during nonmultiplexed HPI operation. An HPIC access does not cause HRDY to go low.

Figure 19. HRDY Behavior During an HPIC Read Cycle in the Nonmultiplexed Mode

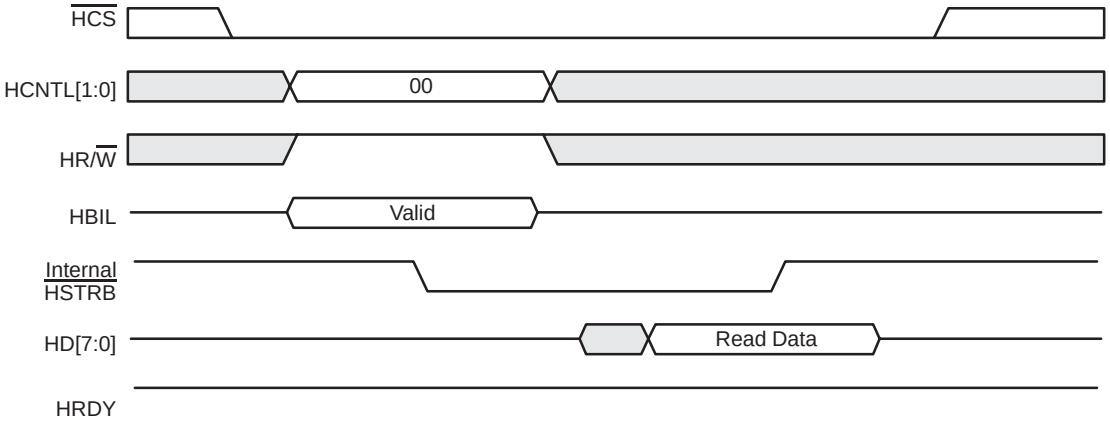
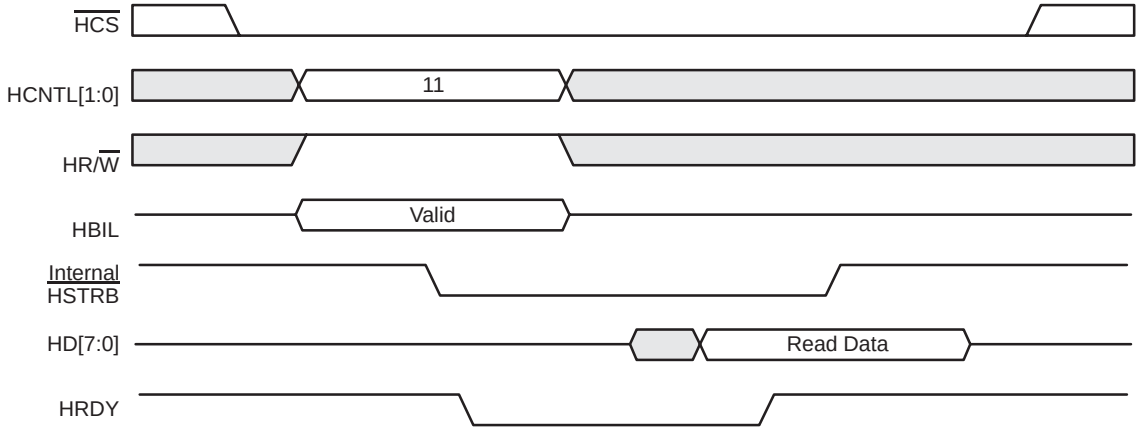


Figure 20 shows an HPID (HCNTL[1:0] = 11b) read cycle when the HPI pins are configured for nonmultiplexed operation. HRDY goes low once for the single 16-bit access.

Figure 20. HRDY Behavior During a Data Read Operation in the Nonmultiplexed Mode



3.10.4 HRDY Behavior During Nonmultiplexed-Mode Write Operations

Figure 21 shows an HPIC write cycle during nonmultiplexed HPI operation. An HPIC write access does not cause HRDY to go low.

Figure 21. HRDY Behavior During an HPIC Write Cycle in the Nonmultiplexed Mode

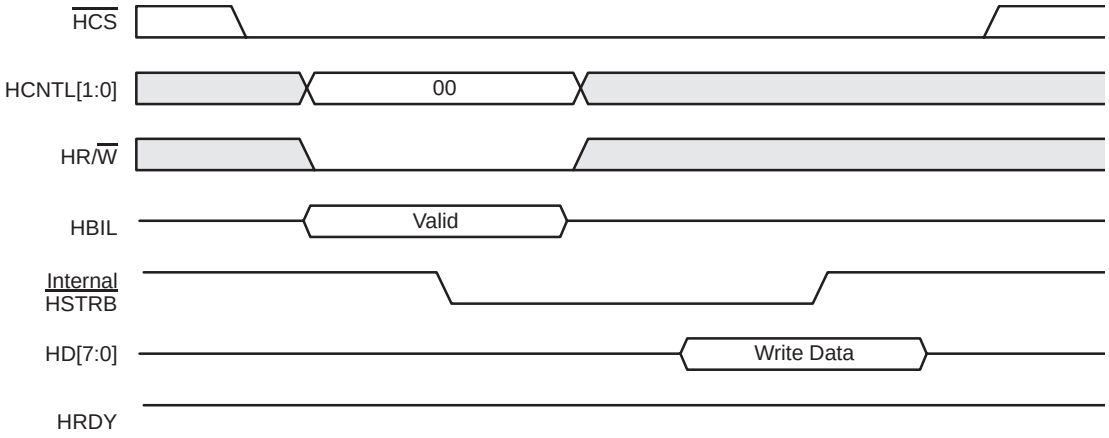
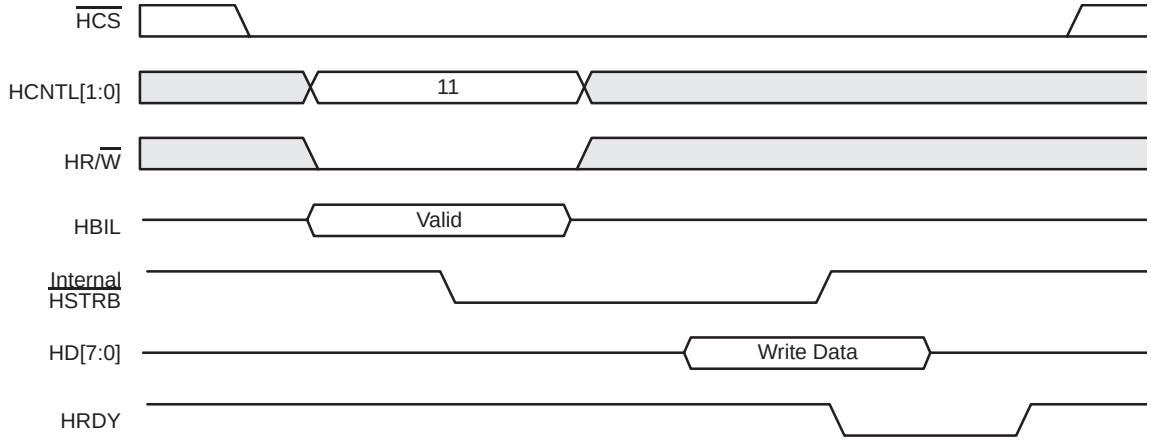


Figure 22 shows an HPID (HCNTL[1:0] = 11b) write cycle when the HPI pins are configured for nonmultiplexed operation. HRDY goes low once for the single 16-bit access.

Figure 22. HRDY Behavior During a Data Write Operation in the Nonmultiplexed-Mode



4 Software Handshaking Using the HPI-Ready (HRDY) Bit

In addition to the HRDY output signal, the HPI contains an HRDY bit, which is in the control register (HPIC). This bit is useful for software polling when the host does not have an input pin to connect to the HRDY pin. In some cases, the host can read the HPIC register and, based on the status of the HRDY bit, determine whether the HPI is ready with read data (during a read cycle) or ready to latch write data (during a write cycle). Section 4.1 explains which read cycles and write cycles allow for polling of the HRDY bit in the 8-bit multiplexed mode. Section 4.2 explains the same for the 16-bit nonmultiplexed mode.

In the 8-bit multiplexed mode, when the host is performing HPID host cycles with an automatic address increment between accesses, the value in the HRDY bit pertains to the availability of space in the write FIFO or the availability of data in the read FIFO. If the previous host cycle was a read cycle, the HRDY bit pertains to the read FIFO. If the previous host cycle was a write cycle, the HRDY bit pertains to the write FIFO. If the previous host cycle set the FETCH bit of HPIC, the HRDY bit pertains to the read FIFO. If the host has performed no data accesses yet, the HRDY bit pertains to the write FIFO by default.

The HRDY bit reflects the level of an internal HRDY signal that is not gated by the chip select ($\overline{\text{HCS}}$) input. The HRDY bit could be cleared in response to one of the following conditions:

- A prefetch was issued (FETCH = 1 in HPIC). HRDY is low until a flush occurs and new data is loaded in the read FIFO. When the data is available, the HRDY bit is set.
- The previous cycle was an autoincrement HPID write cycle and the write FIFO became full. When space is available in the write FIFO, the HRDY bit is set.
- The previous cycle was a non-autoincrement HPID write cycle and the write FIFO is not empty. This condition indicates that the data has not yet been written to memory.
- The previous cycle was an HPID read cycle and the read FIFO is empty. *Exception:* If the previous cycle was a *non-autoincrement* HPID read cycle, when internal $\overline{\text{HSTRB}}$ becomes high (inactive), the HRDY bit stays 1 even though the FIFO is empty. This exception is to accommodate hosts that require the HPI to indicate that it is ready before the host begins the next cycle.
- The previous cycle was an HPID read cycle and a read FIFO flush is in progress.

4.1 Polling the HRDY Bit in the 8-Bit Multiplexed Mode

Read cycles. Only the FETCH command and autoincrement HPID read cycles may be used to perform reads in this mode while using HRDY polling. Non-autoincrement HPID read cycles may not be performed. This is because during cycles without autoincrementing, the host must extend the read cycle until the read FIFO is flushed and the read data is retrieved from the DSP memory. Therefore, the host cannot create the HPIC cycles needed to poll HRDY because the host bus is busy with the current read access. The difference in a cycle with autoincrementing is that the host can release the host bus while the read data is automatically loaded into the read FIFO (due to the FETCH command and subsequent autoincrement read cycles).

Write cycles. As long as the HRDY bit is sampled high, any type of write cycle may be performed by the host. This includes autoincrement HPID write cycles and non-autoincrement HPID write cycle. It is possible to do either type of HPID cycle because the write data goes into the FIFO, and the internal transfer to DSP memory takes place after the host has ended the host bus cycle. This leaves the host bus inactive and therefore available to the host for HPIC reads to poll the HRDY bit.

4.2 Polling the HRDY Bit in the 16-Bit Nonmultiplexed Mode

Read cycles. In this mode, no read cycles can be performed while using HRDY polling. This is because in the 16-bit nonmultiplexed mode, there are no FIFOs and there is no read-data prefetching. All host reads must be extended until the read data is retrieved from DSP memory. Therefore, the host cannot create the HPIC cycles needed to poll HRDY because the host bus is busy with the current read access. Since no read cycles can be performed in this mode, polling is only useful in a system which uses the 16-bit nonmultiplexed mode for write cycles only. A typical example of such a system is one where the host is used simply to load DSP program code into DSP memory and never performs reads of DSP memory.

Write cycles. In this mode, write cycles can be performed while using HRDY polling. This is because each host write goes into the HPID register, and the internal transfer to DSP memory takes place after the host has ended the host bus cycle. This leaves the host bus inactive and therefore available to the host for HPIC reads to poll the HRDY bit.

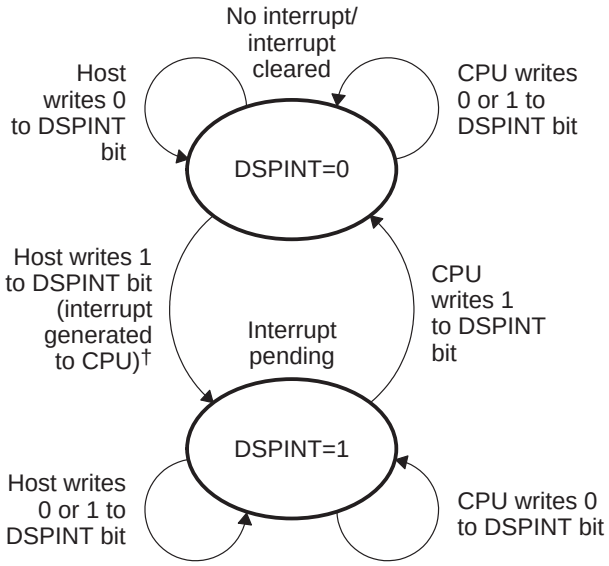
5 Interrupts Between the Host and the CPU

The host can interrupt the CPU of the DSP via the DSPINT bit of the HPIC, as described here in section 5.1. The CPU can send an interrupt to the host by using the HINT bit of HPIC, as described here in section 5.2. The host may also be able to interrupt the CPU with the $\overline{\text{HAS}}$ pin. When available for general-purpose I/O, the $\overline{\text{HAS}}$ pin can be configured as an interrupt pin. See section 9.8 on page 73 for details about this option.

5.1 DSPINT Bit: Host-to-CPU Interrupts

The DSPINT bit of HPIC allows the host to send an interrupt request to the CPU. The use of the DSPINT bit is summarized in Figure 23 and detailed following the figure.

Figure 23. Host-to-CPU Interrupt State Diagram



† When the DSPINT bit transitions from 0 to 1, an interrupt is generated to the CPU. No new interrupt can be generated until the CPU has cleared the bit (DSPINT = 0).

To interrupt the CPU, the host must:

1) Drive both HCNTL1 and HCNTL0 low to request a write to HPIC.

2) Write 1 to the DSPINT bit of HPIC.

When the host sets the DSPINT bit, the HPI generates a interrupt pulse that sets the corresponding flag bit in an interrupt flag register of the CPU. If this maskable interrupt is properly enabled in the CPU, the CPU executes the corresponding interrupt service routine (ISR).

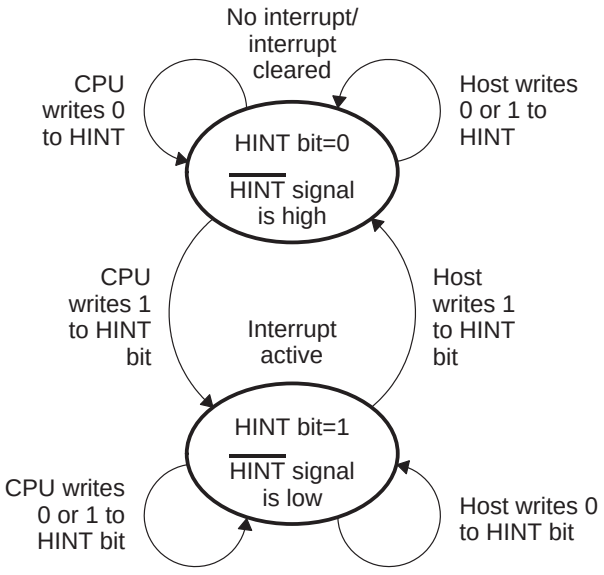
Before the host can use DSPINT to generate a subsequent interrupt to the CPU, the CPU must acknowledge the current interrupt by writing a 1 to the DSPINT bit. When the CPU writes 1, DSPINT is forced to 0. The host should verify that DSPINT = 0 before generating subsequent interrupts. While DSPINT = 1, host writes to the DSPINT bit do not generate an interrupt pulse.

Following is additional information about the DSPINT bit: Writes of 0 have no effect. A hardware reset immediately clears DSPINT and thus clears an active host-to-CPU interrupt. The host cannot use the DSPINT bit to awaken the HPI if the HPI has been placed in its idle mode (see section 8.1 on page 55 for information about entering the idle mode).

5.2 HINT Bit: CPU-to-Host Interrupts

The HINT bit of HPIC allows the CPU to send an interrupt request to the host. The use of the HINT bit is summarized in Figure 24 and detailed following the figure.

Figure 24. CPU-to-Host Interrupt State Diagram



If the CPU writes 1 to the HINT bit of HPIC, the HPI drives the $\overline{\text{HINT}}$ signal low, indicating an interrupt condition to the host. Before the CPU can use the HINT bit generate a subsequent interrupt to host, the host must acknowledge the current interrupt by writing 1 to the HINT bit. When the host does do, the HPI clears the HINT bit (HINT = 0), and this drives the $\overline{\text{HINT}}$ signal high. The CPU should read HPIC and make sure HINT = 0 before generating subsequent interrupts.

Following is additional information about the HINT bit: Writes of 0 have no effect. A hardware reset immediately clears the HINT bit and thus clears an active CPU-to-host interrupt.

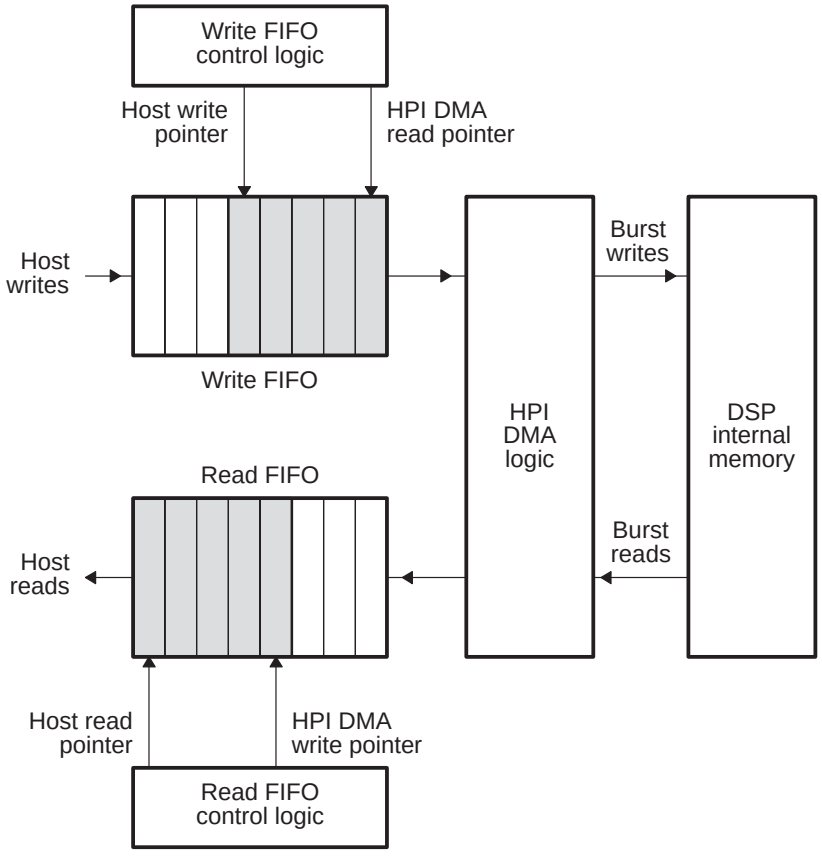
6 FIFOs and Bursting (8-Bit Multiplexed Mode Only)

In the 8-bit multiplexed mode, the data register (HPID) is a port through which the host accesses two first-in, first-out buffers (FIFOs). As shown in Figure 25, a read FIFO supports host read cycles, and a write FIFO supports host write cycles. Both read and write FIFOs are eight words deep (each word 16 bits). If the host is performing multiple reads or writes to consecutive memory addresses (autoincrement HPID cycles), the FIFOs are used for bursting. The HPI DMA logic reads or writes a burst of four words at a time when accessing one of the FIFOs.

Bursting is essentially invisible to the host because the host interface signaling is not affected. Its benefit to the host is that the HRDY signal is deasserted less often when there are multiple reads or writes to consecutive addresses.

In the 16-bit nonmultiplexed mode, the FIFO memory acts as a single register (only one location is used) and bursting is not used.

Figure 25. FIFOs in the HPI



6.1 Read Bursting

When the host writes to the read-address register (HPIAR), the read FIFO is flushed. Any host read data that was in the read FIFO is discarded (the read FIFO pointers are reset). If an HPI DMA write to the read FIFO is in progress at the time of a flush request, the HPI allows this write to complete and then performs the flush.

After any read FIFO flush, no read cycles are being initiated. Read bursting can begin in one of two ways: the host initiates an HPID read cycle with autoincrementing, or the host initiates issues a FETCH command (writes 1 to the FETCH bit in HPIC).

If the host initiates an HPID read cycle with autoincrementing, the HPI DMA logic performs two 4-word burst operations to fill the read FIFO. The host is initially held off by the deassertion of the HRDY signal until data is available to be read from the read FIFO. Once the FIFO is full, the host can read data from the read FIFO by performing subsequent reads of HPID with autoincrementing. Once the initial read has been performed, the HPI DMA logic continues to perform 4-word burst operations to consecutive memory addresses every time there are four empty word locations in the read FIFO. The HPI DMA logic continues to prefetch data to keep the read FIFO full, until the occurrence of an event that causes a read FIFO flush (see section 6.3 on page 50).

As mentioned, the second way that read bursting may begin is with a FETCH command. The host should always precede the FETCH command with the initialization of the HPIAR register or a non-autoincrement access, so that the read FIFO is flushed beforehand. When the host initiates a FETCH command, the HPI DMA logic begins to prefetch data to keep the read FIFO full, as described in the previous paragraph. The FETCH bit in the HPIC register does not actually store the value that is written to it; rather, the decoding of a host write of 1 to this bit is considered a FETCH command.

The FETCH command can be helpful if the host does not use the HRDY signal. The host can initiate prefetching by writing 1 to the FETCH bit and then poll the HRDY bit, which is also in HPIC. When the HRDY bit is 1, the host can perform an HPID read cycle.

Both types of continuous or burst reads described in the preceding paragraphs begin with a write to the HPIA register, which causes a read FIFO flush. This is the typical way of initiating read cycles, because the initial read address needs to be specified.

An HPID read cycle without autoincrementing does not initiate any prefetching activity. Instead, it causes the read FIFO to be flushed and causes the HPI DMA logic to perform a single-word read from the DSP memory. As soon as the host activates a read cycle without autoincrementing, prefetching activity ceases until the occurrence of a FETCH command or an autoincrement read cycle. A non-autoincrement read cycle always should be preceded by another non-autoincrement cycle or the direct initialization of HPIAR, so that the read FIFO is flushed beforehand.

6.2 Write Bursting

A write to the write-address register (HPIAW) causes the write FIFO to be flushed. This means that any write data in the write FIFO is forced to its destination in the DSP memory (the HPI DMA logic performs burst operations until the write FIFO is empty). When the FIFO has been flushed, the only action that will cause the HPI DMA logic to perform burst writes is a host write to HPID with autoincrementing. The initial host-write data is stored in the write FIFO. An HPI DMA write is not requested until there are four words in the write FIFO. As soon as four words have been written to the FIFO via HPID write cycles with autoincrementing, the HPI DMA logic performs a 4-word burst operation to the DSP memory. The burst operations continue as long as there are at least four words in the FIFO. If the FIFO becomes full (eight words are waiting in the FIFO), the HPI holds off the host by deasserting HRDY until at least one empty word location is available in the FIFO.

Because excessive time might pass between consecutive burst operations, the HPI has a time-out counter. If there are fewer than four words in the write FIFO and the time-out counter expires, the HPI DMA logic empties the FIFO immediately by performing a 2- or 3-word burst, or a single-word write, as necessary. Every time new data is written to the write FIFO, the time-out counter is automatically reset to begin its count again. The time-out period is 16 clock cycles, where the clock is the fast peripherals clock (SYSCLK1) of the DSP.

An HPID write cycle without autoincrementing does not initiate any bursting activity. Instead, it causes the write FIFO to be flushed and causes the HPI DMA logic to perform a single-word write to the DSP memory. As soon as the host activates a write cycle without autoincrementing, bursting activity ceases until the occurrence of an autoincrement write cycle. A non-autoincrement write cycle always should be preceded by the initialization of HPIAW or by another non-autoincrement access, so that the write FIFO is flushed beforehand.

6.3 FIFO Flush Conditions

When specific conditions occur within the HPI, the read or write FIFO must be flushed to prevent the reading of stale data from the FIFOs. When a read FIFO flush condition occurs, all current host accesses and direct memory accesses (DMAs) to the read FIFO are allowed to complete. This includes DMAs that have been requested but not yet initiated. The read FIFO pointers are then reset, causing any read data to be discarded.

Similarly, when a write FIFO flush condition occurs, all current host accesses and DMAs to the write FIFO are allowed to complete. This includes DMAs that have been requested but not yet initiated. All posted writes in the FIFO are then forced to completion with a final burst or single-word write, as necessary.

If the host initiates an HPID host cycle during a FIFO flush, the cycle is held off with the deassertion of HRDY until the flush is complete and the FIFO is ready to be accessed.

The following conditions cause the read and write FIFOs to be flushed:

Read FIFO Flush Conditions:

- A value from the host is written to the read-address register (HPIAR).
- The host performs an HPID read cycle without autoincrementing.

Write FIFO Flush Conditions:

- A value from the host is written to the write-address register (HPIAW).
- The host performs an HPID write cycle without autoincrementing.
- The write-burst time-out counter expires.

When operating with DUALHPIA = 0 (all HPIA writes and increments affect both HPIAR and HPIAW), any read or write flush condition listed above will cause both read and write FIFOs to be flushed. In addition, the following scenarios cause both FIFOs to be flushed when DUALHPIA = 0:

- The host performs an HPID write cycle with autoincrementing while the read FIFO is not empty (the read FIFO still contains data from prefetching or an HPID read cycle with autoincrementing).
- The host performs an HPID read cycle with autoincrementing while the write FIFO is not empty (there is still posted write data in the write FIFO).

This is useful in providing protection against reading stale data by reading a memory address when a previous write cycle has not been completed at the same address. Similarly, this protects against overwriting data at a memory address when a previous read cycle has not been completed at the same address.

When operating with DUALHPIA = 1 (HPIAR and HPIAW are independent), there is no such protection. However, when DUALHPIA = 1, data flow can occur in both directions without flushing both FIFOs simultaneously, thereby improving HPI bandwidth.

6.4 FIFO Behavior When a Hardware Reset or an HPI Software Reset Occurs

A hardware reset ($\overline{\text{RESET}}$ pin driven low) or an HPI software reset (HPIRST = 1 in HPIC) causes the FIFOs to be reset. The FIFO pointers are cleared, so that all data in the FIFOs are discarded. In addition, all associated FIFO logic is reset.

If a host cycle is active when a hardware or HPI software reset occurs, the HRDY signal is asserted (driven high), allowing the host to complete the cycle. When the cycle is complete, HRDY is deasserted (driven low). Any access interrupted by a reset may result in corrupted read data or a lost write data (if the write does not actually update the intended memory or register). Although data may be lost, the host interface protocol is not violated. While either of reset condition is true, and the host is idle (internal $\overline{\text{HSTRB}}$ is held high), the FIFOs are held in reset, and host transactions are held off with an inactive HRDY signal.

7 Using HPI Pins for General-Purpose I/O

Table 8 summarizes which HPI pins can be used for general-purpose I/O (GPIO) in TMS320VC5501 and TMS320VC5502 devices for the possible mode conditions. It also indicates whether GPIO functionality is controlled via HGPIO registers in the HPI or via PGPIO registers at the DSP system level. An overview of the HGPIO registers follows the table. For descriptions of the PGPIO registers, see the device-specific data manual.

Note:

Depending on the pins you enable for general-purpose I/O, host accesses may be limited or impossible. See section 3 on page 20 to check which signals are required for a particular type of host access.

Table 8. General-Purpose I/O Control of HPI Pins

Device	Condition	GPIO Pin Control
TMS320VC5501	If the 8-bit multiplexed mode is on ...	HBIL, $\overline{\text{HAS}}$, and HD[7:0] are controlled via HGPIO registers in the HPI.
	If the 8-bit multiplexed mode is off ...	HBIL, $\overline{\text{HAS}}$, and HD[7:0] are not available to the HPI. GPIO control for these pins is handled via PGPIO registers at the DSP system level.
	Regardless of whether the 8-bit multiplexed mode is on or off ...	HCNTL0, HCNTL1, and $\overline{\text{HINT}}$ are controlled via HGPIO registers in the HPI.
		If the HPIENA signal is low (HPI disabled), $\overline{\text{HDS1}}$, $\overline{\text{HDS2}}$, $\overline{\text{HCS}}$, and HR/$\overline{\text{W}}$ are also controlled via HGPIO registers. If the HPIENA signal is high (HPI enabled), the GPIO functionality of these pins is disabled.
		HRDY and HPIENA are not available for GPIO.

Table 8. General-Purpose I/O Control of HPI Pins (Continued)

Device	Condition	GPIO Pin Control
TMS320VC5502	If the 8-bit multiplexed mode is selected ...	HBIL, $\overline{\text{HAS}}$, and HD[7:0] are controlled via HGPIO registers in the HPI.
		HD[15:8] and HA[15:0] are not available for GPIO because they are used by the external memory interface (EMIF).
	If the 16-bit multiplexed mode is selected ...	HBIL and $\overline{\text{HAS}}$ are not available to the HPI but can be controlled via PGPIO registers at the DSP system level.
		HA[15:0] and HD[15:0] are controlled via HGPIO registers in the HPI.
	Regardless of whether the multiplexed or nonmultiplexed mode is selected ...	HCNTL0, HCNTL1, and $\overline{\text{HINT}}$ are controlled via HGPIO registers in the HPI.
		If the HPIENA signal is low (HPI disabled), $\overline{\text{HDS1}}$, $\overline{\text{HDS2}}$, $\overline{\text{HCS}}$, and HR/$\overline{\text{W}}$ are also controlled via HGPIO registers. If the HPIENA signal is high (HPI enabled), the GPIO functionality of these pins is disabled.
		HRDY and HPIENA are not available for GPIO.

For pins under the control of the HPI, GPIO functionality is configured and controlled with the following registers:

- GPIO enable register (HGPIOEN). Bits in this register are used to configure pins of the HPI for either their HPI function or for GPIO. Details about HGPIOEN are in section 9.4 (page 63).
- GPIO direction registers (HGPIODIR1, HGPIODIR2, HGPIODIR3). Each of the pins that you can enable for GPIO has a direction bit in one of these registers. When GPIO is enabled for the pin, the direction bit determines whether the pin is an input pin or an output pin. For details about these registers, see sections 9.5 through 9.7. (Section 9.5 begins on page 66.)

- GPIO data registers (HGPIODAT1, HGPIODAT2, HGPIODAT3). Each of the pins that you can enable for GPIO has a data bit in this register. When GPIO is enabled for the pin, the data bit is used to monitor or drive the pin. If the pin is configured as an input pin, the CPU reads the data bit to determine the signal level on the pin. If the pin is configured as an output pin, the CPU writes to the data bit to drive the signal high or low. For details about these registers, see sections 9.5 through 9.7.

- GPIO interrupt control registers (HGPIOINT1, HGPIOINT2). When the $\overline{\text{HAS}}$ signal is configured as a general-purpose input pin, these registers allow you to enable $\overline{\text{HAS}}$ as an interrupt pin and to invert the polarity of the input. Details are in section 9.8 (page 73).

8 Power, Emulation, and Reset Considerations

8.1 Conserving Power

The DSP is divided into a number of idle domains. To minimize power consumption, you can choose which domains are active and which domains are idle at any given time. The current state of all domains is collectively called the *idle configuration*. If a particular idle configuration turns off the master port domain, the HPI might enter an inactive, low-power mode, depending on other factors described in the device-specific data manual.

When the HPI is successfully requested to enter this idle mode, all HPI activity stops immediately. The HPI must communicate with the host to ensure that all host accesses are complete and that host activity is halted before the CPU places the HPI into its idle mode.

Keep in mind that idle domains other than the master port domain can affect the HPI. For example, if the clock generator domain is idle, the HPI has no clocks for operation.

8.2 Emulation Modes

The FREE and SOFT bits of the power and emulation management register (HPWREMU) determine the response of the HPI to an emulation suspend condition. If FREE = 1, the HPI is not affected, and the SOFT bit has no effect. If FREE = 0 and SOFT = 0, the HPI is not affected. If FREE = 0 and SOFT = 1:

- The HPI DMA logic halts after the current host and HPI DMA operations are completed.
- The external host interface functions as normal throughout the emulation suspend condition. The host may access the control register (HPIC). In the 8-bit multiplexed mode, the host may also access the HPIA registers and may perform data reads until the read FIFO is empty or data writes until the write FIFO is full. As in normal operation, HRDY is driven low during a host cycle that cannot be completed due to the write FIFO being full or the read FIFO being empty. If this occurs, HRDY continues to be driven low, holding off the host, until the emulation suspend condition is over, and the FIFOs are serviced by the HPI DMA logic, allowing the host cycle to complete.
- When the emulation suspend condition is over, the appropriate requests by the HPI DMA logic are made to process any posted host writes in the write FIFO or to fill the read FIFO as necessary. HPI operation then continues as normal.

8.3 Effects of a Hardware Reset on the HPI

When the entire TMS320VC5501/5502 DSP is reset with the $\overline{\text{RESET}}$ pin:

- If the internal strobe signal, internal $\overline{\text{HSTRB}}$, is high (host is inactive), HRDY is driven low and remains low until the reset condition is over.
- If internal $\overline{\text{HSTRB}}$ is low (host cycle is active), HRDY is driven high, allowing the host to complete the cycle. When internal $\overline{\text{HSTRB}}$ goes high (cycle is complete), HRDY is driven low and remains low until the reset condition is over. If the active cycle was a write cycle, the memory or register may not have been correctly updated. If the active cycle was a read cycle, the fetched value may not be valid.
- The HPI registers are reset to their default values. These default values are shown under each bit field of the register figures in section 9.
- The read and write FIFOs and the associated FIFO logic are reset (this includes a flush of the FIFOs).
- Host-to-DSP and DSP-to-host interrupts are cleared.
- The logic for controlling the general-purpose I/O functionality of the HPI is reset.

8.4 HPI Software Reset

The control register (HPIC) provides an HPI software reset bit (HPIRST) that is used to reset the read and write FIFOs. When the CPU sets the HPIRST bit:

- If the internal strobe signal, internal $\overline{\text{HSTRB}}$, is high (host is inactive), HRDY is driven low and remains low until the reset condition is over.
- If internal $\overline{\text{HSTRB}}$ is low (host cycle is active), direct memory accesses (DMAs) of the FIFOs are allowed to complete. Then HRDY is driven high, allowing the host to complete the cycle. When internal $\overline{\text{HSTRB}}$ goes high (cycle is complete), HRDY is driven low and remains low until the reset condition is over. If the active cycle was a write cycle, the memory or register may not have been correctly updated. If the active cycle was a read cycle, the fetched value may not be valid.
- After any remaining DMAs of the FIFOs are complete, the read and write FIFOs and the associated FIFO logic are reset. The FIFO pointers are cleared, so that any data in the FIFOs are discarded. The CPU reads 0 in HPIRST until the FIFOs are fully reset. Writing 0 to HPIRST before the FIFO reset is complete will not stop the FIFO reset from occurring.

An HPI software reset does not reset any HPI registers other than the FIFOs.

9 HPI Registers

This section describes the registers inside the HPI, which are listed in Table 9. All registers except the data register (HPID) are accessible to the CPU via 16-bit word addresses in the I/O space of the DSP. Only the control register (HPIC), the data register (HPID), and the address registers (HPIAR and HPIAW) are accessible to the host.

Table 9. Registers of the HPI

Register	Description	For Details, See ...
HPIC	Control register	Page 57
HPID	Data register	Page 61
HPIAR	Address register for host read cycles	Page 61
HPIAW	Address register for host write cycles	Page 61
HGPIOEN	General-purpose I/O enable register	Page 63
HGPIODIR1	General-purpose I/O direction register 1	Page 66
HGPIODAT1	General-purpose I/O data register 1	Page 66
HGPIODIR2	General-purpose I/O direction register 2	Page 67
HGPIODAT2	General-purpose I/O data register 2	Page 67
HGPIODIR3	General-purpose I/O direction register 3	Page 71
HGPIODAT3	General-purpose I/O data register 3	Page 71
HGPIOINT1	General-purpose I/O interrupt control register 1	Page 73
HGPIOINT2	General-purpose I/O interrupt control register 2	Page 73
HPWREMU	Power and emulation management register	Page 74

9.1 Control Register (HPIC)

HPIC is a 16-bit register that stores configuration and control information for the HPI. The fields within the register are shown in Figure 26 and described in Table 10. As shown in the figure, the host and the CPU do not have the same access permissions. The host owns HPIC and thus has full read/write access. The CPU has primarily read-only access, but the exceptions are:

- The CPU can write 1 to the HINT bit to generate an interrupt to the host.
- The CPU can write 1 to the DSPINT bit to clear/acknowledge an interrupt from the host.
- The CPU can write 1 to HPIRST to cause an HPI software reset.

Figure 26. Control Register (HPIC)

Host access permissions

15	12	11	10	9	8
Reserved		HPIASEL	Reserved	DUALHPIA	BOBSTAT
R−0		R/W−0	R−0	R/W−0	R−0

7	6	5	4	3	2	1	0
HPIRST	Reserved†		FETCH	HRDY	HINT	DSPINT	BOB
R−0	R/W−0		R/W−0	R−1	R/W1C−0	R/W−0	R/W−0

CPU access permissions

15	12	11	10	9	8
Reserved		HPIASEL	Reserved	DUALHPIA	BOBSTAT
R−0		R−0	R−0	R−0	R−0

7	6	5	4	3	2	1	0
HPIRST	Reserved		FETCH	HRDY	HINT	DSPINT	BOB
R/W−0	R−0		R−0	R−0	R/W−0	R/W1C−0	R−0

Legend: R = Read; W = Write; W1C = Write 1 to clear; −*n* = Value after hardware reset
† The host must write 0 to these reserved bits.

Table 10. Control Register (HPIC) Bits

Bit	Field	Value	Description
15−12	Reserved		Read-only reserved bits. Reads return 0s.
11	HPIASEL		HPIA read/write select bit (configured by the host). This bit is applicable only in the dual-HPIA mode (DUALHPIA = 1). **Note:** HPIASEL does not affect the HPI DMA logic. Regardless of the value of HPIASEL, the HPI DMA logic uses HPIAW when writing to memory and HPIAR when reading from memory.
		0	In the next HPIA host cycle, the host will access HPIAW (the write address register).
		1	In the next HPIA host cycle, the host will access HPIAR (the read address register).
10	Reserved		Read-only reserved bits. Reads return 0.

Table 10. Control Register (HPIC) Bits (Continued)

Bit	Field	Value	Description
9	DUALHPIA		Dual-HPIA mode bit (configured by the host)
		0	Single-HPIA mode. From the host's perspective there is one 16-bit HPIA register. A host HPIA write cycle places the same value in both HPIAR and HPIAW. During autoincrementing both HPIAR and HPIAW are incremented. A host HPIA read cycle retrieves the value from HPIAR.
		1	Dual-HPIA mode. The host sees two 16-bit HPIA registers: HPIAR for read addresses and HPIAW for write addresses.
8	BOBSTAT		BOB status bit. BOBSTAT reflects the value of the BOB bit (see bit 0).
		0	BOB = 0 (first byte is most significant)
		1	BOB = 1 (first byte is least significant)
7	HPIRST		HPI software reset bit (set by CPU)
		0	**Host:** Reads of HPIRST always return 0. **CPU:** Once the CPU has written 1 to HPIRST, reads return 0 until the FIFOs are completely reset. Writing 0 before the reset process is complete will not stop the reset from occurring.
		1	**CPU:** Writing 1 causes the read and write FIFOs and the associated FIFO logic to be reset. As described in section 8.4 (page 56), an active host cycle is allowed to complete before the reset process begins.
6−5	Reserved	0	The host must write 0s to these bits. The CPU cannot modify these bits.
4	FETCH		Host data fetch command bit (set by host)
		0	**CPU/Host:** Reads of FETCH always return 0.
		1	**Host:** Write 1 to tell the HPI DMA logic to pre-fetch data into the read FIFO.
3	HRDY		HPI-ready indicator (read-only)
		0	**Host:** Internal HRDY is low. The HPI is not ready to complete a host cycle. **CPU:** Reads of HRDY always return 0.
		1	**Host:** Internal HRDY is high. The HPI is ready to complete a host cycle.

Table 10. Control Register (HPIC) Bits (Continued)

Bit	Field	Value	Description
2	HINT		Host interrupt bit (set by the CPU, cleared by the host)
		0	**CPU/Host:** Writing 0 has no effect.
		1	**CPU:** Writing 1 to HINT generates a CPU-to-host interrupt. HINT remains 1 until it is cleared by the host or by a hardware reset.
			Host: Writing 1 to HINT clears HINT to 0, to acknowledge the CPU-to-host interrupt.
1	DSPINT		DSP interrupt bit (set by the host, cleared by the CPU)
		0	**CPU/Host:** Writing 0 has no effect.
		1	**CPU:** Writing 1 to DSPINT clears DSPINT to 0, to acknowledge the host-to-CPU interrupt.
			Host: Writing 1 to DSPINT generates a host-to-CPU interrupt. DSPINT remains 1 until it is cleared by the CPU or by a hardware reset.
0	BOB		Byte order bit (configured by the host). This bit is applicable only in the 8-bit multiplexed mode. BOB must be initialized by the host before the first data or address register access. The status of BOB is also reflected in BOBSTAT (see bit 8).
			For host write cycle:
		0	The first byte received from the bus is most significant (written to the high half of HPID/HPIC/HPIAR/HPIAW). The second byte is least significant (written to the low half of HPID/HPIC/HPIAR/HPIAW).
		1	The first byte received from the bus is least significant (written to the low half of HPID/HPIC/HPIAR/HPIAW). The second byte is most significant (written to the high half of HPID/HPIC/HPIAR/HPIAW).
			For host read cycle:
		0	The first byte transmitted on the bus is most significant (taken from the high half of HPID/HPIC/HPIAR/HPIAW). The second byte is least significant (taken from the low half of HPID/HPIC/HPIAR/HPIAW).
		1	The first byte transmitted on the bus is least significant (taken from the low half of HPID/HPIC/HPIAR/HPIAW). The second byte is most significant (taken from the high half of HPID/HPIC/HPIAR/HPIAW).

9.2 Data Register (HPID)

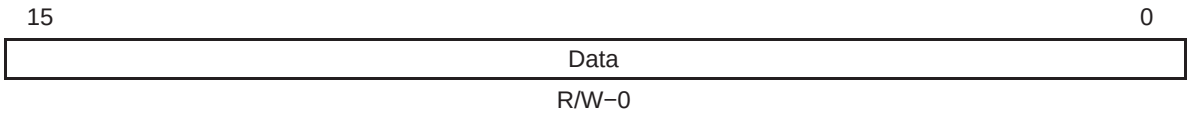
The 16-bit register HPID provides the data path between the host and the HPI DMA logic. During a host write cycle, the host fills HPID with 16 bits, and then the HPI DMA logic transfers the 16-bit value to the internal memory of the DSP. During a host read cycle, the HPI DMA logic fills HPID with 16 bits from the internal memory, and then the HPI transfers the 16-bit value to the host. A host cycle is a single 16-bit transfer (in the 16-bit nonmultiplexed mode) or two consecutive 8-bit transfers (in the 8-bit multiplexed mode).

As shown in Figure 27, the host has full read/write access to HPID. The CPU cannot access HPID.

In the 8-bit multiplexed mode, HPID is actually a port through which the host accesses two first-in, first-out buffers (FIFOs). The read FIFO and the write FIFO play a significant role in providing a higher data throughput. For information about the FIFOs, see section 6 on page 47.

Figure 27. Data Register (HPID)

Host access permissions (CPU cannot access HPID)



Legend: R = Read; W = Write; −*n* = Value after hardware reset

9.3 Address Registers (HPIAR and HPIAW)

Table 11 shows how much of the internal memory of the TMS320VC5501 and TMS320VC5502 devices is accessible to the host through the HPI. The host cannot access addresses 0000h−005Fh, which are reserved for the memory mapped registers of the CPU. In the 8-bit multiplexed mode, when the host wants to access a location in the internal memory of the DSP, the host must supply the address of that location. To do this, the host writes the address to an HPI address (HPIA) register. The HPI DMA logic reads the address from the HPIA register when fetching or storing data.

Table 11. Internal Memory Accessible to Host and Address Bits Required From Host

Device	Internal Memory Accessible to the Host	Address Bits Required From Host
TMS320VC5501	First 16K words, except addresses 0000h−005Fh	The host must write a 14-bit address to the HPI address register.
TMS320VC5502	First 32K words, except addresses 0000h−005Fh	The host must write a 15-bit address to the HPI address register.

There are two 16-bit HPIA registers: HPIAR for read operations and HPIAW for write operations. The HPI can be configured such that HPIAR and HPIAW act as a single 16-bit HPIA (single-HPIA mode) or as two separate 16-bit HPIAs (dual-HPIA mode) from the perspective of the host. For details about these HPIA modes, see section 2 on page 18.

Figure 28 shows the format of an address register in a TMS320VC5501 device, and Figure 29 shows the format in a TMS320VC5502 device. One fewer address bit is required for TMS320VC5501 devices because half as much of the internal memory is accessible. As shown in the figures, the host has full read/write access to the HPIAs, while the CPU can only read the HPIAs.

In the 16-bit nonmultiplexed mode (TMS320VC5502 devices only), the HPIAs are not used.

Figure 28. Format of an Address Register (HPIAR or HPIAW) − TMS320VC5501 Device

Host access permissions

15	14	13	0
—†		Address	
R/W−0		R/W−0	

CPU access permission

15	14	13	0
—†		Address	
R−0		R−0	

Legend: R = Read; W = Write; −*n* = Value after hardware reset

† Always write 0s to these reserved bits. Because the segment of memory accessible via the HPI is 16K words, only 14 address bits (13−0) are needed; bits 15 and 14 are not needed.

Figure 29. Format of an Address Register (HPIAR or HPIAW) − TMS320VC5502 Device

Host access permissions

15	14	0
—†	Address	
R/W−0	R/W−0	

CPU access permission

15	14	0
—†	Address	
R−0	R−0	

Legend: R = Read; W = Write; −*n* = Value after hardware reset

† Always write 0 to this reserved bit. Because the segment of memory accessible via the HPI is 32K words, only 15 address bits (14−0) are needed; bit 15 is not needed.

9.4 General-Purpose I/O Enable Register (HGPIOEN)

Figure 30 and Table 12 summarize the fields of HGPIOEN. The CPU programs this register; the host cannot access it.

Each GPIO enable bit in this register corresponds to one or more pins of the HPI. To enable a pin or bank of pins for general-purpose I/O, the CPU must set the corresponding GPIO enable bit. Then the CPU can control each enabled pin with the appropriate GPIO direction register (to select input or output) and the appropriate GPIO data register (to read or drive the signal level). In addition, if the $\overline{\text{HAS}}$ pin is enabled for GPIO, it can be configured to be used by the host to generate CPU interrupts (see section 9.8 on page 73).

The following limitations apply to the use of HGPIOEN:

- The EN11 and EN12 bits are used only in the 16-bit nonmultiplexed mode. In the 8-bit multiplexed mode, pins HA[15:0] are not available.
- The EN8 bit is used only in the 16-bit nonmultiplexed mode. In the 8-bit multiplexed mode, pins HD[15:8] are not available.
- The EN2 and EN4 bits are used only in the 8-bit multiplexed mode. In the 16-bit nonmultiplexed mode, the $\overline{\text{HAS}}$ and HBIL pins are not available.
- The EN0 bit can be used only when the HPI is disabled by pulling the HPIENA pin low. When HPIENA is low, software can configure the $\overline{\text{HCS}}$, $\overline{\text{HDS1}}$, $\overline{\text{HDS2}}$, and HR/$\overline{\text{W}}$ pins for GPIO by writing 1 to the EN0 bit. When HPIENA is high, the EN0 bit should be loaded with 0 because these pins cannot be used for GPIO while the HPI is in use.

Figure 30. General-Purpose I/O Enable Register (HGPIOEN)

CPU access permissions (Host cannot access HGPIOEN)

15	13	12	11	10	9	8
Reserved†		EN12	EN11	Reserved†		EN8
R/W−0		R/W−0	R/W−0	R/W−0		R/W−0

7	6	5	4	3	2	1	0
EN7	EN6	Reserved†	EN4	Reserved†	EN2	EN1	EN0‡
R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0

Legend: R = Read; W = Write; *−n* = Value after hardware reset
† Write 0s to the reserved bits of HGPIOEN.
‡ When the HPIENA signal is high, the EN0 bit should be loaded with 0 because the $\overline{\text{HCS}}$, $\overline{\text{HDS1}}$, $\overline{\text{HDS2}}$, and HR/$\overline{\text{W}}$ pins cannot be used for general-purpose I/O while the HPI is in use.

Table 12. General-Purpose I/O Enable Register (HGPIOEN) Bits

Bit	Field	Value	Description
15−13	Reserved	0	Write 0s to these bits.
12	EN12		GPIO enable bit for pins HA[15:8]
		0	HA[15:8] are HPI address bus lines.
		1	HA[15:8] are enabled for GPIO.
11	EN11		GPIO enable bit for pins HA[7:0]
		0	HA[7:0] are HPI address bus lines.
		1	HA[7:0] are enabled for GPIO.
10−9	Reserved	0	Write 0s to these bits.
8	EN8		GPIO enable bit for pins HD[15:8]
		0	HD[15:8] are HPI data bus lines.
		1	HD[15:8] are enabled for GPIO.
7	EN7		GPIO enable bit for pins HD[7:0]
		0	HD[7:0] are HPI data bus lines.
		1	HD[7:0] is enabled for GPIO.

Table 12. General-Purpose I/O Enable Register (HGPIOEN) Bits (Continued)

Bit	Field	Value	Description
6	EN6		GPIO enable bit for the $\overline{\text{HINT}}$ pin
		0	$\overline{\text{HINT}}$ is the host interrupt pin.
		1	$\overline{\text{HINT}}$ is enabled for GPIO.
5	Reserved	0	Write 0 to this bit.
4	EN4		GPIO enable bit for the HBIL pin
		0	HBIL is the byte identification line.
		1	HBIL is enabled for GPIO.
3	Reserved	0	Write 0 to this bit.
2	EN2		GPIO enable bit for the $\overline{\text{HAS}}$ pin
		0	$\overline{\text{HAS}}$ is the address strobe pin.
		1	$\overline{\text{HAS}}$ is enabled for GPIO.
1	EN1		GPIO enable bit for pins HCNTL[1:0]
		0	HCNTL[1:0] are the access control pins.
		1	HCNTL[1:0] are enabled for GPIO.
0	EN0		GPIO enable bit for pins $\overline{\text{HCS}}$, $\overline{\text{HDS1}}$, $\overline{\text{HDS2}}$, and HR/$\overline{\text{W}}$. When the HPIENA signal is high (HPI enabled), EN0 should be loaded with 0 because these pins cannot be used for GPIO.
		0	$\overline{\text{HCS}}$, $\overline{\text{HDS1}}$, $\overline{\text{HDS2}}$, and HR/$\overline{\text{W}}$ are the chip select, data strobe, and read/write pins, respectively.
		1	$\overline{\text{HCS}}$, $\overline{\text{HDS1}}$, $\overline{\text{HDS2}}$, and HR/$\overline{\text{W}}$ are enabled for GPIO. This setting is invalid if HPIENA is high.

9.5 General-Purpose I/O Direction Register 1 (HGPIODIR1) and General-Purpose I/O Data Register 1 (HGPIODAT1)

The format for HGPIODIR1 and HGPIODAT1 is shown in Figure 31. The direction bits of HGPIODIR1 are summarized in Table 13, and the data bits of HGPIODAT1 are summarized in Table 14. The CPU programs these registers; the host cannot access them.

The direction bits in HGPIODIR1 are used to configure each HPI data bus pin as either an input pin or an output pin. The pins can be configured independently as inputs (direction bits are 0s) or outputs (direction bits are 1s), in any combination.

HGPIODAT1 contains data bits for monitoring or driving individual pins of the HPI data bus, HD[15:0]. The CPU reads from or writes to a data bit, depending on whether the corresponding HD pin has been configured as an input pin or an output pin in HGPIODIR1:

Pin Direction	Function of Data Bit
Input	Reading 0 indicates the input signal is low. Reading 1 indicates the input signal is high. Writes have no effect on the data bit.
Output	Writing 0 drives the output signal low. Writing 1 drives the output signal high.

The CPU can use these two registers only if the HD pins are enabled for general-purpose I/O in HGPIOEN. Pin bank HD[15:8] and pin bank HD[7:0] are independently enabled or disabled.

The bits that control HD[15:8] can be used only in the 16-bit nonmultiplexed mode. In the 8-bit multiplexed mode, pins HD[15:8] are not available.

Figure 31. Format of Registers HGPIODIR1 and HGPIODAT1

CPU access permissions (Host cannot access HGPIODIR1 or HGPIODAT1)

15	14	13	12	11	10	9	8
HD15	HD14	HD13	HD12	HD11	HD10	HD9	HD8
R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0

7	6	5	4	3	2	1	0
HD7	HD6	HD5	HD4	HD3	HD2	HD1	HD0
R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0

Legend: R = Read; W = Write; −*n* = Value after hardware reset

Table 13. General-Purpose I/O Direction Register 1 (HGPIODIR1) Bits

Bit	Field	Description	
15−0	HD15−HD0	Direction bits for pins HD[15:0]. The function of each direction bit is summarized as follows, where x is a number from 0 to 15:	
		HDx Direction Bit	**HDx Pin Direction**
		0	Input
		1	Output

Table 14. General-Purpose I/O Data Register 1 (HGPIODAT1) Bits

Bit	Field	Description	
15−0	HD15−HD0	Data bits for pins HD[15:0]. The function of each data bit is summarized as follows, where x is a number from 0 to 15:	
		HDx Data Bit	**Signal on HDx Pin**
		0	Low
		1	High

9.6 General-Purpose I/O Direction Register 2 (HGPIODIR2) and General-Purpose I/O Data Register 2 (HGPIODAT2)

The format for HGPIODIR2 and HGPIODAT2 is shown in Figure 32. The direction bits of HGPIODIR2 are summarized in Table 15, and the data bits of HGPIODAT2 are summarized in Table 16. The CPU programs these registers; the host cannot access them.

The direction bits in HGPIODIR2 are used to configure a variety of HPI control pins as either an input pin or an output pin. The pins can be configured independently as inputs or outputs, in any combination.

HGPIODAT2 contains data bits for monitoring or driving individual control pins of the HPI. The CPU reads from or writes to a data bit, depending on whether the corresponding control pin has been configured as an input pin or an output pin in HGPIODIR2:

Pin Direction	Function of Data Bit
Input	Reading 0 indicates the input signal is low. Reading 1 indicates the input signal is high. Writes have no effect on the data bit.
Output	Writing 0 drives the output signal low. Writing 1 drives the output signal high.

The CPU can use these two registers only if the corresponding control pins are enabled for general-purpose I/O in HGPIOEN. The bits that control the $\overline{\text{HAS}}$ and HBIL pins can be used only in the 8-bit multiplexed mode. In the 16-bit nonmultiplexed mode, the $\overline{\text{HAS}}$ and HBIL pins are not available. The $\overline{\text{HCS}}$, $\overline{\text{HDS1}}$, $\overline{\text{HDS2}}$, and HR/$\overline{\text{W}}$ pins can be used for GPIO only when the HPIENA pin is pulled low.

Figure 32. Format of Registers HGPIODIR2 and HGPIODAT2

CPU access permissions (Host cannot access HGPIODIR2 or HGPIODAT2)

15						9	8
Reserved†							HINT
R/W−0							R/W−0

7	6	5	4	3	2	1	0
HCNTL0	HCNTL1	HBIL	HRW	HDS2	HDS1	HCS	HAS
R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0

Legend: R = Read; W = Write; −*n* = Value after hardware reset
† Write 0s to these reserved bits.

Table 15. General-Purpose I/O Direction Register 2 (HGPIODIR2) Bits

Bit	Field	Value	Description
15−9	Reserved	0	Write 0s to these bits.
8	HINT		$\overline{\text{HINT}}$ pin direction bit
		0	Input
		1	Output
7	HCNTL0		HCNTL0 pin direction bit
		0	Input
		1	Output
6	HCNTL1		HCNTL1 pin direction bit
		0	Input
		1	Output
5	HBIL		HBIL pin direction bit. (This pin is available only in the 8-bit multiplexed mode.)
		0	Input
		1	Output
4	HRW		HR/$\overline{\text{W}}$ pin direction bit
		0	Input
		1	Output
3	HDS2		$\overline{\text{HDS2}}$ pin direction bit
		0	Input
		1	Output
2	HDS1		$\overline{\text{HDS1}}$ pin direction bit
		0	Input
		1	Output
1	HCS		$\overline{\text{HCS}}$ pin direction bit
		0	Input
		1	Output

Table 15. General-Purpose I/O Direction Register 2 (HGPIODIR2) Bits (Continued)

Bit	Field	Value	Description
0	HAS		$\overline{\text{HAS}}$ pin direction bit. (This pin is available only in the 8-bit multiplexed mode.)
		0	Input
		1	Output

Table 16. General-Purpose I/O Data Register 2 (HGPIODAT2) Bits

Bit	Field	Value	Description
15−9	Reserved	0	Write 0s to these bits.
8	HINT		$\overline{\text{HINT}}$ pin data bit
		0	Low
		1	High
7	HCNTL0		HCNTL0 pin data bit
		0	Low
		1	High
6	HCNTL1		HCNTL1 pin data bit
		0	Low
		1	High
5	HBIL		HBIL pin data bit. (This pin is available only in the 8-bit multiplexed mode.)
		0	Low
		1	High
4	HRW		HR/$\overline{\text{W}}$ pin data bit
		0	Low
		1	High
3	HDS2		$\overline{\text{HDS2}}$ pin data bit
		0	Low
		1	High

Table 16. General-Purpose I/O Data Register 2 (HGPIODAT2) Bits (Continued)

Bit	Field	Value	Description
2	HDS1		$\overline{\text{HDS1}}$ pin data bit
		0	Low
		1	High
1	HCS		$\overline{\text{HCS}}$ pin data bit
		0	Low
		1	High
0	HAS		$\overline{\text{HAS}}$ pin data bit. (This pin is available only in the 8-bit multiplexed mode.)
		0	Low
		1	High

9.7 General-Purpose I/O Direction Register 3 (HGPIODIR3) and General-Purpose I/O Data Register 3 (HGPIODAT3)

Note:

On TMS320VC5501 devices, the register addresses for HGPIODIR3 and HGPIODAT3 are reserved. On these devices, HGPIODIR3 and HGPIODAT3 are not needed because the pins they would control, HA[15:0], are not present.

The format for HGPIODIR3 and HGPIODAT3 is shown in Figure 33. The direction bits of HGPIODIR3 are summarized in Table 17, and the data bits of HGPIODAT3 are summarized in Table 18. The CPU programs these registers; the host cannot access them.

The direction bits in HGPIODIR3 are used to configure each HPI address bus pin as either an input pin or an output pin. The pins can be configured independently as inputs or outputs, in any combination.

HGPIODAT3 contains data bits for monitoring or driving individual pins of the HPI address bus, HA[15:0]. The CPU reads from or writes to a data bit, depending on whether the corresponding HA pin has been configured as an input pin or an output pin in HGPIODIR3:

Pin Direction	Function of Data Bit
Input	Reading 0 indicates the input signal is low. Reading 1 indicates the input signal is high. Writes have no effect on the data bit.
Output	Writing 0 drives the output signal low. Writing 1 drives the output signal high.

The use of these two registers depends on two conditions:

- The HA pins must be enabled for general-purpose I/O in HGPIOEN. Pin bank HA[15:8] and pin bank HA[7:0] are independently enabled or disabled.
- The HPI must be in the 16-bit nonmultiplexed mode. In the 8-bit multiplexed mode, pins HA[15:0] are not available.

Figure 33. Format of Registers HGPIODIR3 and HGPIODAT3

CPU access permissions (Host cannot access HGPIODIR3 or HGPIODAT3)

15	14	13	12	11	10	9	8
HA15	HA14	HA13	HA12	HA11	HA10	HA9	HA8
R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0

7	6	5	4	3	2	1	0
HA7	HA6	HA5	HA4	HA3	HA2	HA1	HA0
R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0

Legend: R = Read; W = Write; −*n* = Value after hardware reset

Table 17. General-Purpose I/O Direction Register 3 (HGPIODIR3) Bits

Bit	Field	Description	
15−0	HA15−HA0	Direction bits for pins HA[15:0]. The function of each direction bit is summarized as follows, where x is a number from 0 to 15:	
		HAx Direction Bit	**HAx Pin Direction**
		0	Input
		1	Output

Table 18. General-Purpose I/O Data Register 3 (HGPIODAT3) Bits

Bit	Field	Description	
15−0	HA15−HA0	Data bits for pins HA[15:0]. The function of each data bit is summarized as follows, where x is a number from 0 to 15:	
		HAx Data Bit	**Signal on HAx Pin**
		0	Low
		1	High

9.8 General-Purpose I/O Interrupt Control Registers (HGPIOINT1 and HGPIOINT2)

The format for HGPIOINT1 and HGPIOINT2 is shown in Figure 34. The CPU programs these registers; the host cannot access them.

The HAS bit of HGPIOINT1 (see Table 19) determines whether the $\overline{\text{HAS}}$ pin is enabled to be an interrupt pin. The HAS bit of HGPIOINT2 (see Table 20) determines whether the interrupt signal is inverted on $\overline{\text{HAS}}$.

When configured properly, $\overline{\text{HAS}}$ can be driven high or low by the host to generate an interrupt to the CPU. This interrupt is the same one that can be generated when the host changes the DSPINT bit from 0 to 1 in HPIC.

$\overline{\text{HAS}}$ can only be used for interrupts in the 8-bit multiplexed mode. In the 16-bit nonmultiplexed mode, $\overline{\text{HAS}}$ is not available.

To properly configure $\overline{\text{HAS}}$ for interrupts, the CPU must:

- ❏ Enable $\overline{\text{HAS}}$ for general-purpose I/O in HGPIOEN.
- ❏ Make sure $\overline{\text{HAS}}$ is configured as an input pin in HGPIODIR2.
- ❏ Select inversion or no inversion in HGPIOINT2.
- ❏ Set the interrupt enable bit for $\overline{\text{HAS}}$ in HGPIOINT1.

Figure 34. Format of Registers HGPIOINT1 and HGPIOINT2

CPU access permissions (Host cannot access HGPIOINT1 or HGPIOINT2)

15	3	2	1	0
Reserved†		HAS	Reserved†	
R/W−0		R/W−0	R/W−0	

Legend: R = Read; W = Write; −*n* = Value after hardware reset

† Write 0s to these reserved bits.

Table 19. General-Purpose I/O Interrupt Control Register 1 (HGPIOINT1) Bits

Bit	Field	Value	Description
15−3	Reserved	0	Write 0s to these bits.
2	HAS		$\overline{\text{HAS}}$ pin interrupt enable bit (This pin is available only in the 8-bit multiplexed mode.)
		0	The $\overline{\text{HAS}}$ pin cannot generate an interrupt.
		1	The $\overline{\text{HAS}}$ pin is enabled as a potential interrupt pin. If the $\overline{\text{HAS}}$ pin is enabled for general-purpose I/O and is configured as an input pin, it can be used to generate an interrupt to the CPU.
1−0	Reserved	0	Write 0s to these bits.

Table 20. General-Purpose I/O Interrupt Control Register 2 (HGPIOINT2) Bits

Bit	Field	Value	Description
15−3	Reserved	0	Write 0s to these bits.
2	HAS		$\overline{\text{HAS}}$ pin interrupt invert bit (This pin is available only in the 8-bit multiplexed mode.)
		0	Do not invert. Driving $\overline{\text{HAS}}$ low generates a DSP interrupt request to the CPU.
		1	Invert. Driving $\overline{\text{HAS}}$ high generates a DSP interrupt request to the CPU.
1−0	Reserved	0	Write 0s to these bits.

9.9 Power and Emulation Management Register (HPWREMU)

Figure 35 shows the format of HPWREMU. The bits in this register form a combination that determines the emulation mode of the HPI, as described in Table 21.

The CPU has full read/write access to HPWREMU. The host cannot access this register.

Figure 35. Power and Emulation Management Register (HPWREMU)

CPU access permissions (Host cannot access HPWREMU)

15	2	1	0
Reserved†		SOFT	FREE
R/W−0		R/W−0	R/W−0

Legend: R = Read; W = Write; −*n* = Value after hardware reset
† Write 0 to these reserved bits.

Table 21. Power and Emulation Management Register (HPWREMU) Bits

Bit	Field	Value	Description
15−2	Reserved	0	Write 0s to these bits.
1	SOFT		When FREE = 0, this emulation bit determines whether the HPI runs free or makes a soft stop in response to an emulation suspend condition.
		0	Run free. The HPI is not affected by an emulation suspend condition.
		1	Soft stop. In response to an emulation suspend condition, the HPI DMA logic halts after the current host and HPI DMA operations are completed. For more details, see section 8.2 on page 55.
0	FREE		This emulation bit determines whether the HPI runs free or consults the SOFT bit when the HPI detects an emulation suspend condition.
		0	The SOFT bit determines the response of the HPI.
		1	Run free. The HPI is not affected by an emulation suspend condition.

This page is intentionally left blank.

Revision History

This document was revised to SPRU620C from SPRU620B, which was dated November 2003. Notable changes that were made since the last revision are listed in the following table.

Page	Additions/Modifications/Deletions
11	In the first part of section 1, *Introduction to the HPI*, modified the Note regarding the HPI DMA logic. The Note now differentiates the HPI DMA logic from the DMA controller of the DSP.
25	Modified the last paragraph of section 3.2, *HPI Configuration and Data Flow*, to explain what happens when the CPU, the HPI DMA logic, and the DMA controller compete with each other for dual-access RAM (DARAM) inside the DSP.
30−33, 35	Added the following footnote the timing diagrams of Figure 6 through Figure 9, and Figure 11: Depending on the type of write operation (HPID without autoincrementing, HPIA, HPIC, or HPID with autoincrementing) and the state of the FIFO, transitions on HRDY may or may not occur. For more information, see section 3.10, *Hardware Handshaking Using the HPI-Ready (HRDY) Signal*, on page 36.
36	Modified section 3.10, *Hardware Handshaking Using the HPI-Ready (HRDY) Signal.* This section now includes timing diagrams of HRDY behavior for various types of read and write operations initiated by the host.

This page is intentionally left blank.

Index

E

F

G

H

I

L

M

N

O

P

R

S

T

W

X