TMS320C642x DSP
Host Port Interface (HPI)

User's Guide

Literature Number: SPRUEM9A
May 2008–Revised July 2008

Contents

List of Figures

List of Tables

Read This First

About This Manual

This document describes the features and operation of the host port interface (HPI) in the TMS320C642x digital signal processor (DSP).

Notational Conventions

This document uses the following conventions.

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

Related Documentation From Texas Instruments

The following documents describe the TMS320C642x Digital Signal Processor (DSP). 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.

The current documentation that describes the C642x DSP, related peripherals, and other technical collateral, is available in the C6000 DSP product folder at: www.ti.com/c6000.

SPRUEM3 — ***TMS320C642x DSP Peripherals Overview Reference Guide.*** Provides an overview and briefly describes the peripherals available on the TMS320C642x Digital Signal Processor (DSP).

SPRAA84 — ***TMS320C64x to TMS320C64x+ CPU Migration Guide.*** Describes migrating from the Texas Instruments TMS320C64x digital signal processor (DSP) to the TMS320C64x+ DSP. The objective of this document is to indicate differences between the two cores. Functionality in the devices that is identical is not included.

SPRU732 — ***TMS320C64x/C64x+ DSP CPU and Instruction Set Reference Guide.*** Describes the CPU architecture, pipeline, instruction set, and interrupts for the TMS320C64x and TMS320C64x+ digital signal processors (DSPs) of the TMS320C6000 DSP family. The C64x/C64x+ DSP generation comprises fixed-point devices in the C6000 DSP platform. The C64x+ DSP is an enhancement of the C64x DSP with added functionality and an expanded instruction set.

SPRU871 — ***TMS320C64x+ DSP Megamodule Reference Guide.*** Describes the TMS320C64x+ digital signal processor (DSP) megamodule. Included is a discussion on the internal direct memory access (IDMA) controller, the interrupt controller, the power-down controller, memory protection, bandwidth management, and the memory and cache.

Host Port Interface (HPI)

1 Introduction

The host port interface (HPI) provides a parallel port interface through which an external host processor can directly access the TMS320C642x DSP processor's resources (configuration and program/data memories). The external host device is asynchronous to the CPU clock and functions as a master to the HPI interface. The HPI enables a host device and the C642x DSP processor to exchange information via internal or external memory. Dedicated address (HPIA) and data (HPID) registers within the HPI provide the data path between the external host interface and the processor resources. An HPI control register (HPIC) is available to the host and the CPU for various configuration and interrupt functions.

1.1 Purpose of the Peripheral

The HPI enables an external host processor (host) to directly access program/data memory on the processor using a parallel interface. The primary purpose is to provide a mechanism to move data to and from the processor. In addition to data transfer, the host can also use the HPI to bootload the processor by downloading program and data information to the processor's memory after power-up.

1.2 Features

The HPI supports the following features:

- Multiplexed address/data
- Dual 16-bit halfword cycle access (internal data word is 32-bits wide)
- 16-bit-wide host data bus interface
- Internal data bursting using 8-word read and write first-in, first-out (FIFO) buffers
- HPI control register (HPIC) accessible by both the DSP CPU and the external host
- HPI address register (HPIA) accessible by both the DSP CPU and the external host
- Separate HPI address registers for read (HPIAR) and write (HPIAW) with configurable option for operating as a single HPI address register
- HPI data register (HPID)/FIFOs providing data-path between external host interface and CPU resources
- Multiple strobes and control signals to allow flexible host connection
- Asynchronous HRDY output to allow the HPI to insert wait states to the host
- Software control of data prefetching to the HPID/FIFOs
- Processor-to-Host interrupt output signal controlled by HPIC accesses
- Host-to-Processor interrupt controlled by HPIC accesses
- Register controlled HPIA and HPIC ownership and FIFO timeout
- Memory-mapped peripheral identification register (PID)
- Bus holders on host data and address buses (these are actually external to HPI module)

1.3 Functional Block Diagram

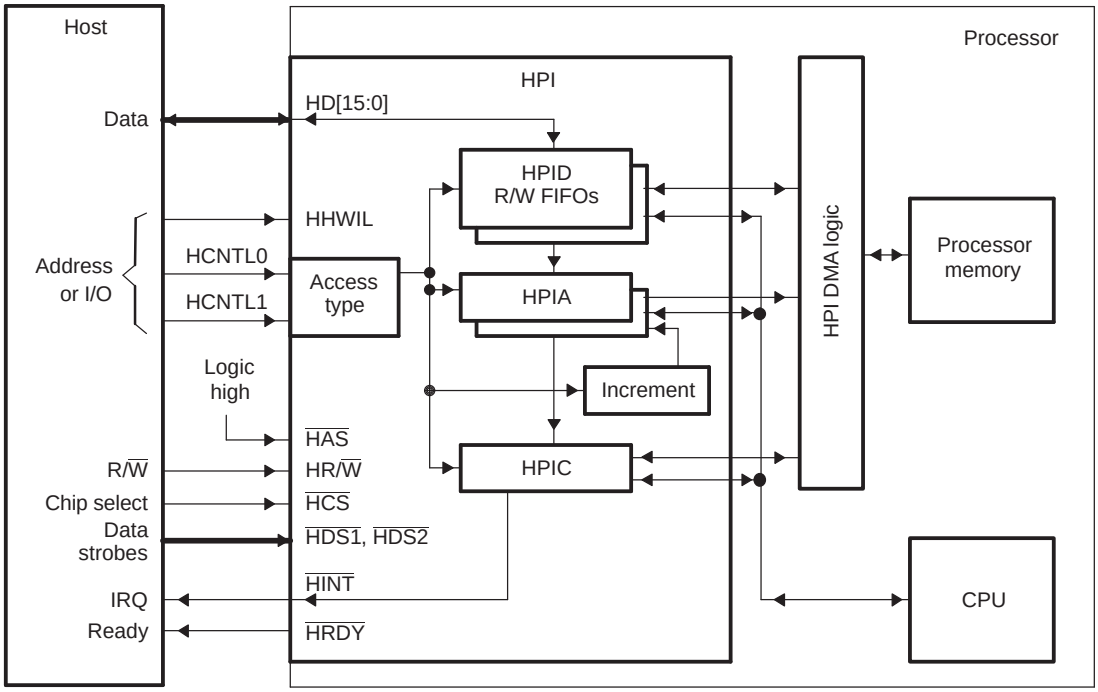
Figure 1 is a high-level block diagram showing how the HPI connects a host (left side of figure) and the processor internal memory (right side of figure). Host activity is asynchronous to the internal processor clock that drives the HPI. The host functions as a master to the HPI. When HPI resources are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

The HPI supports multiplexed operation meaning the data bus is used for both address and data. Each host cycle consists of two consecutive 16-bit transfers. When the host drives an address on the bus, the address is stored in a 32-bit address register (HPIA) in the HPI, so that the bus can then be used for data. The HPI contains two address registers (HPIAR and HPIAW), which can be used as separate address registers for read accesses and write accesses (for details, see Section 2.7.1).

A control register (HPIC) is accessible by the CPU and the host. The CPU uses HPIC to send an interrupt request to the host, to clear an interrupt request from the host, and to monitor the HPI. The host uses 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 32-bit data register (HPID). Data arriving from the host is held in HPID until the data can be stored elsewhere in the processor. 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).

Figure 1. HPI Block Diagram



1.4 Industry Standard(s) Compliance Statement

The HPI is not an industry standard interface that is developed and monitored by an international organization. It is a generic parallel interface that can be configured to gluelessly interface to a variety of parallel devices.

1.5 Terminology Used in This Document

The following is a brief explanation of some terms used in this document:

Term	Meaning
CPU	DSP CPU
host	External host device
HPI DMA logic	Logic used to communicate between the HPI and the DMA system that moves data to and from memory. This is independent of the EDMA system on the processor
processor	the entire digital media system-on-chip

2 Peripheral Architecture

2.1 Clock Control

The HPI clock is derived from SYSCLK5, which is the PLL1 clock divided by 6. For detailed information on the PLLs and clock distribution on the processor, see the *TMS320C642x DSP Phase-Locked Loop Controller (PLLC) User's Guide* (SPRUES0).

2.2 Memory Map

The HPI can be used by the host to access the following processor resources:

- HPI configuration registers
- Most on-chip device memory, peripherals, and memory-mapped registers (see Section 4, *System Interconnect*, in the device-specific data manual for more detailed information)
- DDR2 Memory Controller configuration register file and memory address ranges
- Power and Sleep Controller (PSC) registers
- PLL1 and PLL2 registers

Consult the device-specific data manual for the memory address ranges of the above resources.

2.3 Signal Descriptions

Table 1 shows the a description of the HPI signals.

2.4 Pin Multiplexing

On the C642x DSP extensive pin multiplexing is used to accommodate the largest number of peripheral functions in the smallest possible package. Pin multiplexing is controlled using a combination of hardware configuration at device reset and software programmable register settings. Refer to the device-specific data manual to determine how pin multiplexing affects the HPI.

2.5 Protocol Description

The HPI does not conform to any industry standard protocol. Details on the nature of address, data, and control transactions are found in the following sections.

2.6 Endianness Considerations

The HPI operation is independent of the C642x DSP endianness mode; therefore, there are no endianness considerations for the HPI.

Table 1. HPI Pins

Pin	Type	Host Connection	Function
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 internal HSTRB (for details about internal HSTRB, see Section 2.7.4). The four binary states of these pins determine the access type of the current transfer (HPIC, HPIA, HPID with and without autoincrementing).
HCS	I	Chip select pin	**HPI chip select**. HCS must be low for the HPI to be selected by the host. HCS can be kept low between accesses. HCS normally precedes an active HDS (data strobe) signal, but can be connected to an HDS pin for simultaneous select and strobe activity.
HR/W	I	R/W strobe pin	**HPI read/write**. On the falling edge of internal HSTRB, HR/W indicates whether the current access is to be a read or write operation. Driving HR/W high indicates the transfer is a read from the HPI, while driving HR/W low indicates a write to the HPI.
HHWIL	I	Address or control pins	**Halfword identification line**. The host uses HHWIL to identify the first and second halfwords of the host cycle. HHWIL must be driven low for the first halfword and high for the second halfword.
HAS	I	None	**Address strobe**. Connect to logic high.
HINT	O/Z	Interrupt pin	**Host Interrupt**. The CPU can interrupt the host processor by writing a 1 to the HINT bit of HPIC. Before subsequent HINT interrupts can occur, the host must acknowledge interrupts by writing a 1 to the HINT bit. This pin is active-low (that is, when an interrupt is asserted from the host, the state of this signal is low) and inverted from the HINT bit value in HPIC.
HDS1 and 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 2.7.4). The direction of the data transfer depends on the logic level of the HR/W signal. The HDS signals are also used to latch control information on the falling edge. During an HPID write access, data is latched into the HPID register on the rising edge of HDS. During read operations, these pins act as output-enable pins of the host data bus.
HD[15:0]	I/O/Z	Data bus	**HPI data bus**. The HPI data bus carries the address and data to/from the HPI.
HRDY	O/Z	Asynchronous ready pin	**HPI-ready signal**. When the HPI drives HRDY low, the host has permission to complete the current host cycle. When the HPI drives HRDY high, the HPI is not ready for the current host cycle to complete.

2.7 Architecture and Operation

2.7.1 Using the Address Registers

The HPI contains two 32-bit address registers: one for read operations (HPIAR) and one for write operations (HPIAW). These roles are unchanging from the viewpoint of the HPI logic. The HPI DMA logic gets the address from HPIAR when reading from processor resources (see Section 2.2) and gets the address from HPIAW when writing to processor resources (see Section 2.2).

However, unlike the HPI logic, the host can choose how to interact with the two HPI address registers. Using the DUALHPIA bit in the HPI control register (HPIC), the host determines whether HPIAR and HPIAW act as a single 32-bit register (single-HPIA mode) or as two independent 32-bit registers (dual-HPIA mode).

Note that the addresses loaded into the HPI address registers must be byte addresses, and must be 32-bit word aligned (with the least-significant two bits equal to zero), for use in addressing memory space within the C642x DSP.

2.7.1.1 Single-HPIA Mode

When DUALHPIA = 0 in HPIC, HPIAR and HPIAW become a single HPI address register (HPIA) 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 HPI address 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 reinitialize the HPI address registers after changing the state of the DUALHPIA bit. In addition, when DUALHPIA = 0, the host must always reinitialize the HPI address registers when it changes the data direction (from an HPID read cycle to an HPID write cycle, or conversely). Otherwise, the memory location accessed by the HPI DMA logic might not be the location intended by the host.

2.7.1.2 Dual-HPIA Mode

When DUALHPIA = 1 in HPIC, HPIAR and HPIAW are two independent HPI address registers from the perspective of the host. 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 in 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.

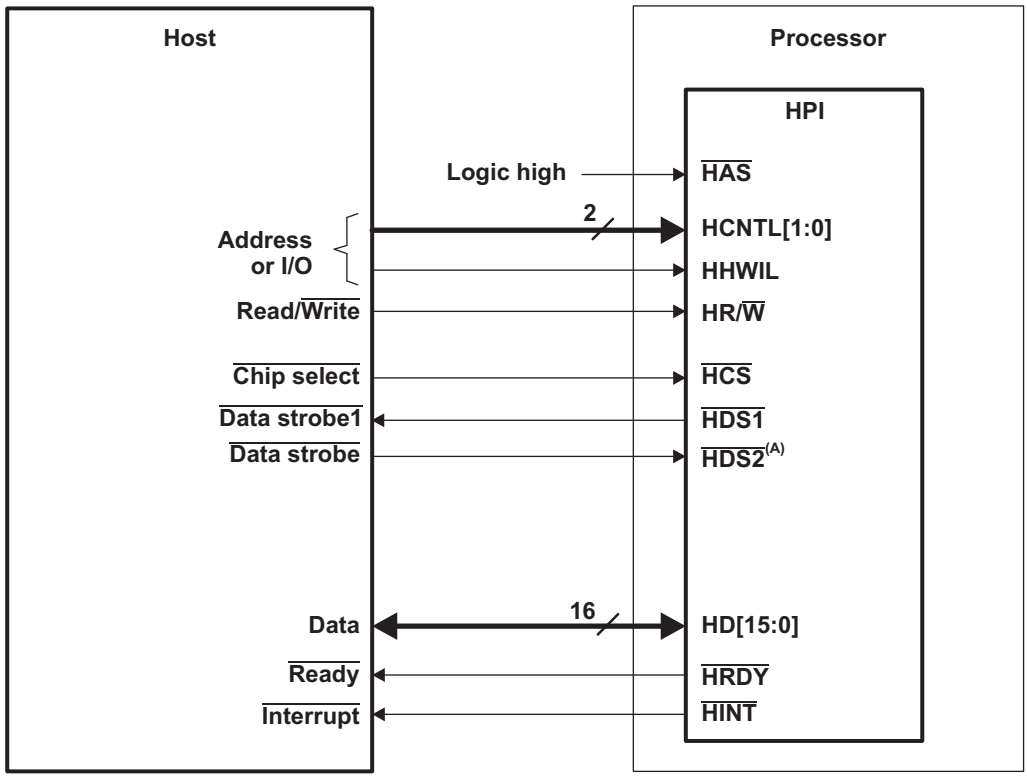
  Note: The HPIASEL bit does not affect the HPI DMA logic. Regardless of the value of HPIASEL, the HPI DMA logic uses HPIAR when reading from memory and HPIAW when writing to memory.

- 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.

2.7.2 Host-HPI Signal Connections

Figure 2 shows an example of a signal connection between the HPI and a host.

Figure 2. Example of Host-Processor Signal Connections



A Data strobing options are given in Section 2.7.4

2.7.3 HPI Configuration and Data Flow

The host accomplishes a multiplexed access in the following manner:

1. The host writes to the HPI control register (HPIC) to properly configure the HPI. Typically, this means programming the halfword order bit (HWOB) and the HPIA-related bits (DUALHPIA and HPIASEL). This step is normally performed once before the initial data access.
2. The host writes the desired internal processor memory address to an address register (HPIAR and/or HPIAW). For an introduction to the two HPI address registers and the two ways the host can interact with them, see Section 2.7.1.
3. The host reads from or writes to the data register (HPID). Data transfers between HPID and the internal memory of the processor are handled by the HPI DMA logic and are transparent to the CPU.

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 ($\overline{HRDY}$) output signal.

When performing an access, the HPI first latches the levels on HCNTL[1:0], HR/$\overline{W}$, and other control signals. This latching can occur on the falling edge of the internal strobe signal (for details, see Section 2.7.4). After the control information is latched, the HPI initiates an access based on the control signals.

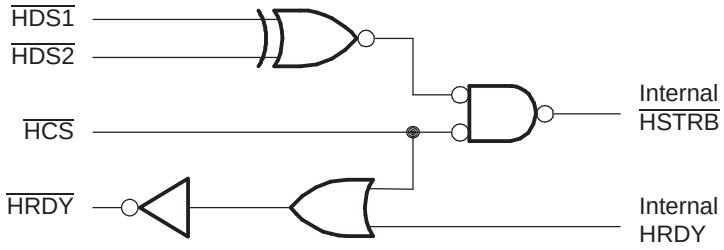
If the host wants to read data from processor resources (see Section 2.2), the HPI DMA logic reads the resource address from HPIAR and retrieves the data from the addressed memory location. When the data has been placed in HPID, the HPI drives the data onto its HD bus. The $\overline{HRDY}$ signal informs the host whether the data on the HD bus is valid ($\overline{HRDY}$ low) or not valid yet ($\overline{HRDY}$ high). When the data is valid, the host should latch the data and drive the connected data strobe ($\overline{HDS1}$ or $\overline{HDS2}$) inactive, which, in turn, will cause the internal strobe (internal $\overline{HSTRB}$) signal to transition from low to high.

If the host wants to write data to processor resources (see Section 2.2), the operation is similar. After the host determines that the HPI is ready to latch the data ($\overline{HRDY}$ is low), it must cause internal $\overline{HSTRB}$ 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.

2.7.4 HDS2, HDS1, and HCS: Data Strobing and Chip Selection

As shown in Figure 3, the strobing logic is a function of three key inputs: the chip select pin (HCS) and two data strobe signals (HDS1 and HDS2). The internal strobe signal, which is referred to as internal HSTRB throughout this document, functions as the actual strobe signal inside the HPI. HCS must be low (HPI selected) during strobe activity on the HDS pins. If HCS remains high (HPI not selected), activity on the HDS pins is ignored.

Figure 3. 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 2 describes some options for connecting to the HDS pins.

Notice in Figure 3 that HRDY is also gated by HCS. If HCS goes high (HPI not selected), HRDY goes low, regardless of whether the current internal transfer is completed in the processor.

Note: 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.

It is not recommended to tie both HDS1 and HDS2 to static logic levels and use HCS as a strobe.

Table 2. 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[(1)]. Since such a host might not provide a R/W line, take care to satisfy HR/W timings as stated in the device data manual. This could possibly be done using a host address line.
Host has separate read and write strobe pins, both active-high	Connect one strobe pin to HDS1 and the other to HDS2[(1)]. Since such a host might not provide a R/W line, take care to satisfy HR/W timings as stated in the device data manual. This could possibly be done using a host address line.
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.

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

2.7.5 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 3 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 4. The HPI samples the HCNTL levels at the falling edge of the internal strobe signal $\overline{HSTRB}$.

Table 3. Access Types Selectable With the HCNTL Signals

HCNTL1[(1)]	HCNTL0[(1)]	Access Type
0	0	**HPIC access.** The host requests to access the HPI control register (HPIC).
0	1	**HPID access with autoincrementing.** 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.** 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.

(1) Note that the encoding of HCNTL0 and HCNTL1 for the different types of HPI accesses varies on many TI DSPs; therefore, you should use caution to ensure that the correct encoding of these inputs is used for your device. The encoding of these signals as described in this document applies only to the C642x DSP.

Table 4. Cycle Types Selectable With the HCNTL and HR/$\overline{W}$ Signals

HCNTL1[(1)]	HCNTL0[(1)]	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

(1) Note that the encoding of HCNTL0 and HCNTL1 for the different types of HPI accesses varies on many TI DSPs; therefore, you should use caution to ensure that the correct encoding of these inputs is used for your device. The encoding of these signals as described in this document applies only to the C642x DSP.

2.7.6 HHWIL: Identifying the First and Second Halfwords in Multiplexed Mode Transfers

Each host cycle consists of two consecutive halfword transfers. For each transfer, the host must specify the cycle type with HCNTL[1:0] and HR/$\overline{W}$, and the host must use HHWIL to indicate whether the first or second halfword is being transferred. For HPID and HPIA accesses, HHWIL must always be driven low for the first halfword transfer and high for the second halfword transfer. Results are undefined if the sequence is broken. For examples of using HHWIL, see Section 2.7.7.

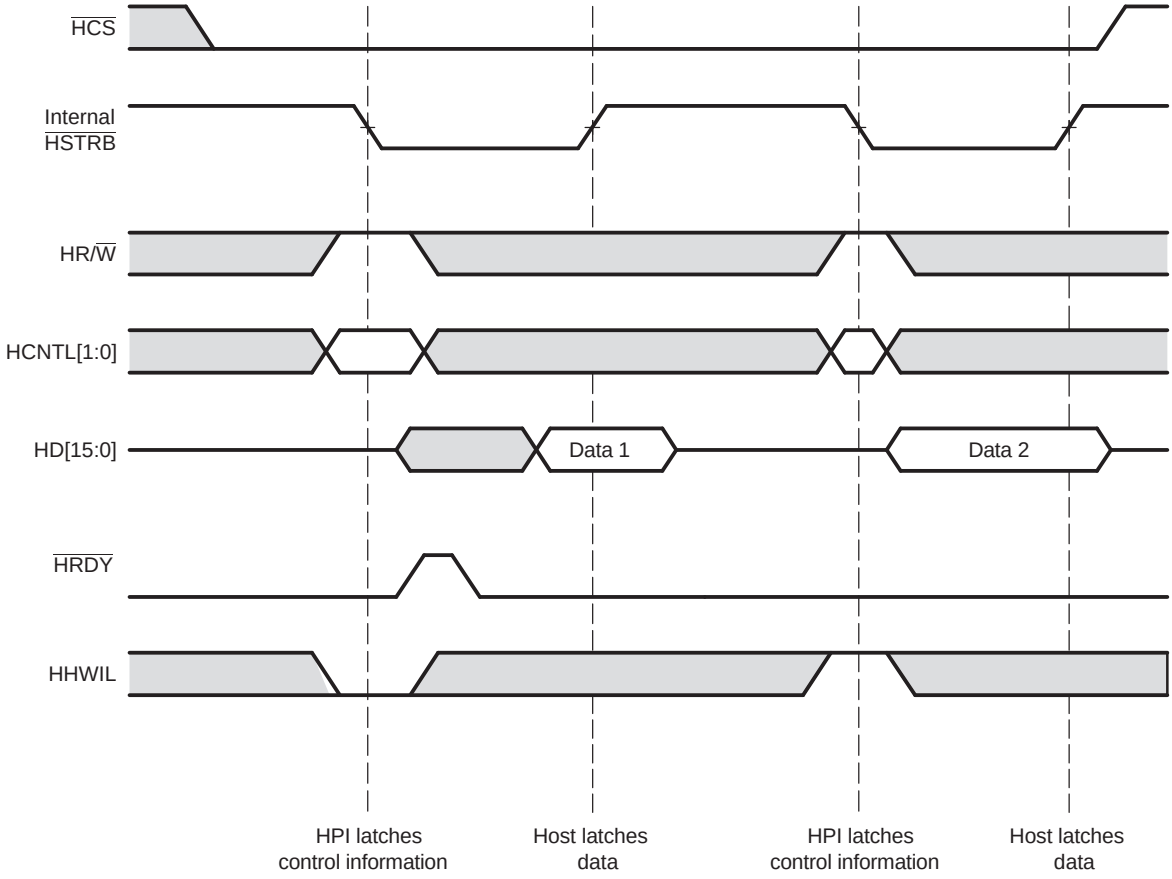
When the host sends the two halfwords of a 32-bit word in this manner, the host can send the most-significant and the least-significant halfwords of the word in either order (most-significant halfword first or most-significant halfword second). However, the host must inform the HPI of the selected order before beginning the host cycle. This is done by programming the halfword order (HWOB) bit in HPIC. Although HWOB is written at bit 0 in HPIC, its current value is readable at both bit 0 and bit 8 (HWOBSTAT). Thus, the host can determine the current halfword order configuration by checking the least-significant bit of either half of HPIC.

There is one case when the HPI does not require a dual halfword access with HHWIL low for the first halfword and HHWIL high for the second halfword. This is the case when accessing the HPIC register. When accessing HPIC, the state of HHWIL is ignored and the same 16-bit HPIC register is accessed regardless of whether the host performs a single or dual access. For an example timing diagram of this case, see Section 2.7.8.

2.7.7 Performing a Multiplexed Access

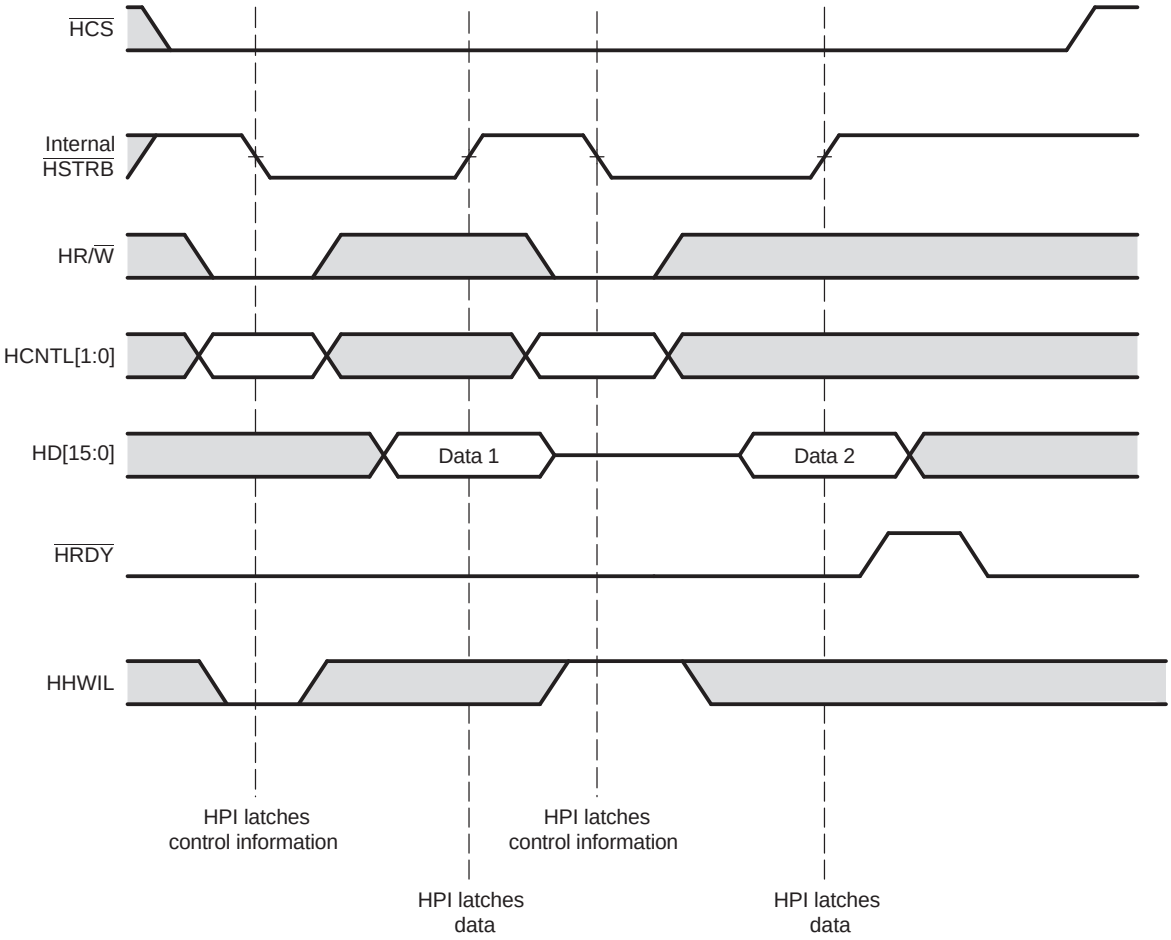
Figure 2 shows an example of signal connections for multiplexed transfers. Figure 4 and Figure 5 show typical HPI signal activity when performing a read and write transfer, respectively. In these cases, the falling edge of internal HSTRB is used to latch the HCNTL[1:0], HR/W, and HHWIL states into the HPI. Internal HSTRB is derived from HCS, HDS1, and HDS2 as described in Section 2.7.4.

Figure 4. Multiplexed-Mode Host Read Cycle



NOTE: 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 2.7.9.

Figure 5. Multiplexed-Mode Host Write Cycle

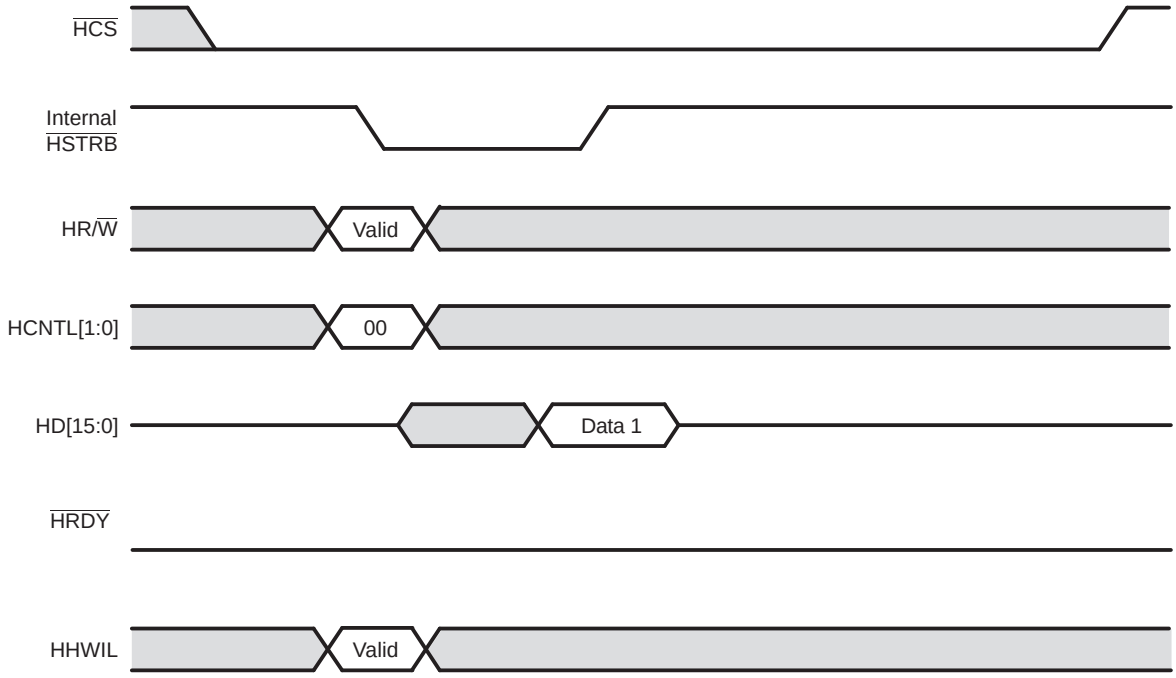


NOTE: Depending on the type of write operation (HPID without autoincrementing, HPIA, HPIC, or HPID with autoincrementing) and the state of the FIFO, transitions on HRDY may or may not occur. For more information, see Section 2.7.9.

2.7.8 Single-Halfword HPIC Cycle

Figure 6 shows the special case (see Section 2.7.6) when the host performs a single-halfword cycle to access the HPIC. The state of HHWIL is ignored and if a dual-halfword access is performed, then the same HPIC register is accessed twice.

Figure 6. Multiplexed-Mode Single-Halfword HPIC Cycle (Read or Write)



2.7.9 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 low) when it has data available for the host. During a write cycle, the HPI is ready (drives HRDY low) when it is ready to latch data from the host. If the HPI is not ready, it can drive HRDY high 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.

When the HPI is not ready to complete the current cycle (HRDY high), 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 low before causing a rising edge on the internal strobe signal (internal HSTRB) to complete the cycle. If internal 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 high 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 2.7.10.

The following sections describe the behavior of HRDY during HPI register accesses. In all cases, the chip select signal, HCS, must be asserted for HRDY to go high

2.7.9.1 HRDY Behavior During Multiplexed-Mode Read Operations

Figure 7 shows an HPIC (HCNTL[1:0] = 00b) or HPIA (HCNTL[1:0] = 10b) read cycle. Neither an HPIC read cycle nor an HPIA read cycle causes HRDY to go high. For this type of access, the state of HHWIL is ignored, so if a dual halfword access is performed, the same register will be accessed twice.

Figure 7. HRDY Behavior During an HPIC or HPIA Read Cycle in the Multiplexed Mode

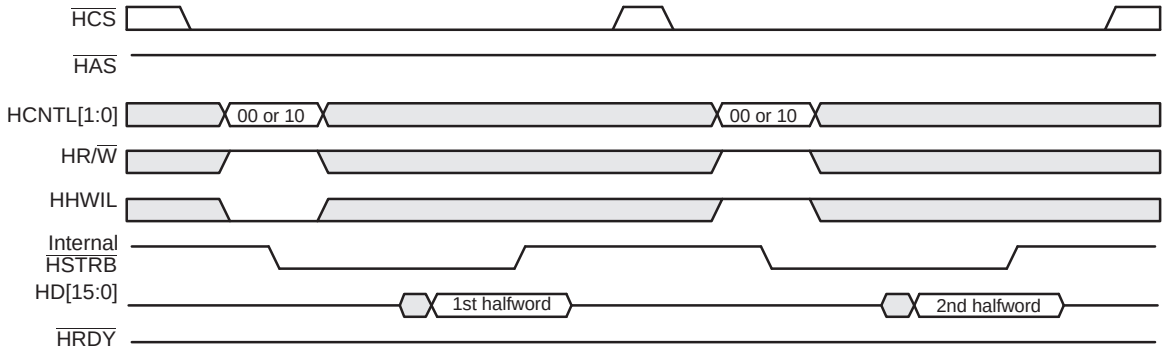


Figure 8 includes an HPID read cycle without autoincrementing. The host writes the memory address during the HPIA (HCNTL[1:0] = 10b) write cycle, and the host reads the data during the HPID (HCNTL[1:0] = 11b) read cycle. HRDY goes high for each HPIA halfword access, but HRDY goes high for only the first halfword access in each HPID read cycle.

Figure 8. HRDY Behavior During a Data Read Operation in the Multiplexed Mode (Case 1: HPIA Write Cycle Followed by Nonautoincrement HPID Read Cycle)

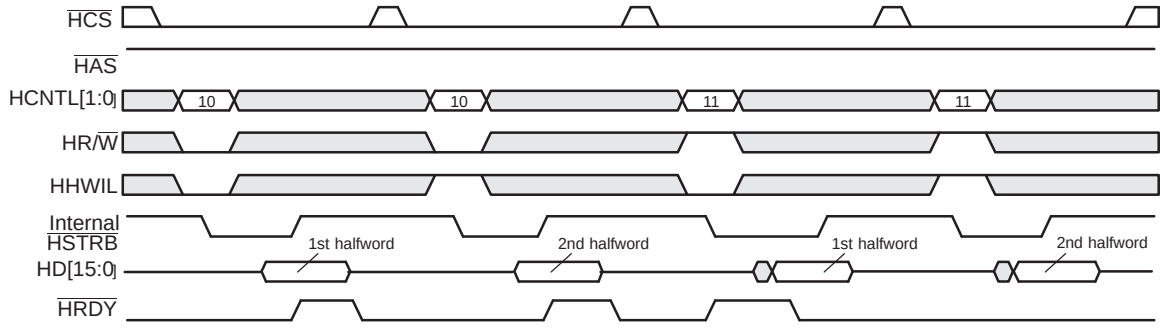
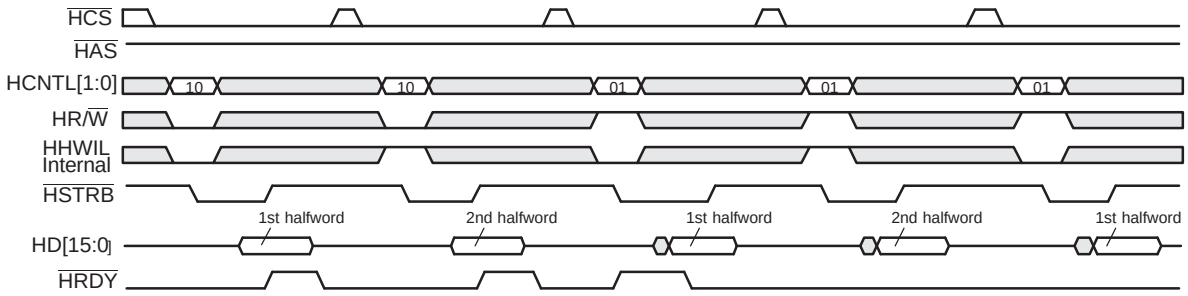


Figure 9 includes an autoincrement HPID read cycle. The host writes the memory address while asserting HCNTL[1:0] = 10b and reads the data while asserting HCNTL[1:0] = 01b. During the first HPID read cycle, HRDY goes high for only the first halfword access, and subsequent HPID read cycles do not cause HRDY to go high.

Figure 9. HRDY Behavior During a Data Read Operation in the Multiplexed Mode (Case 2: HPIA Write Cycle Followed by Autoincrement HPID Read Cycles)



2.7.9.2 HRDY Behavior During Multiplexed-Mode Write Operations

Figure 10 shows an HPIC (HCNTL[1:0] = 00b) write cycle operation. An HPIC write cycle does not cause HRDY to go high and the state of HHWIL is ignored. Firmware is not required to perform a dual access to access HPIC.

Figure 10. HRDY Behavior During an HPIC Write Cycle in the Multiplexed Mode

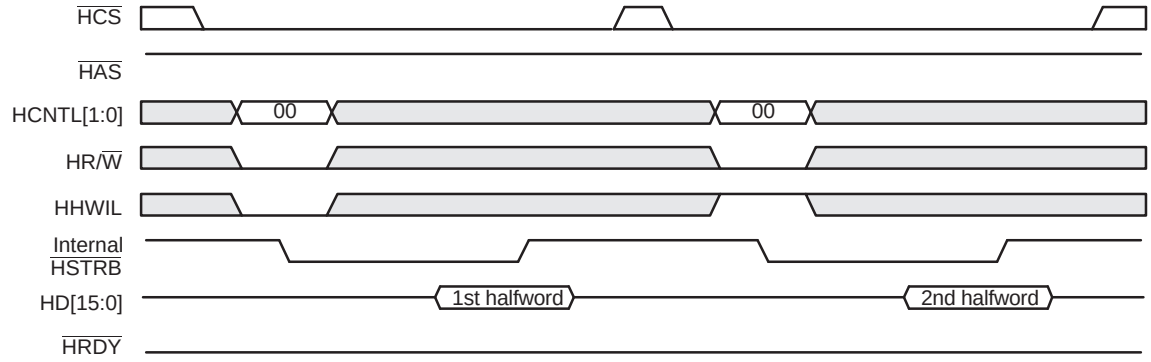


Figure 11 includes a HPID write cycle without autoincrementing. The host writes the memory address while HCNTL[1:0] = 10b and writes the data while HCNTL[1:0] = 11b. During the HPID write cycle, HRDY goes high only for the second halfword access.

Figure 11. HRDY Behavior During a Data Write Operation in the Multiplexed Mode (Case 1: No Autoincrementing)

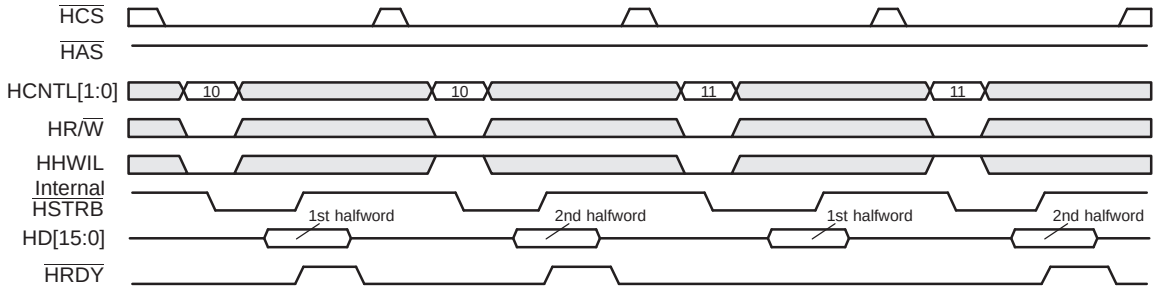


Figure 12 shows autoincrement HPID write cycles when the write FIFO is empty prior to the HPIA write. The host writes the memory address while HCNTL[1:0] = 10b and writes the data while HCNTL[1:0] = 01b. HRDY does not go high during any of the HPID write cycles until the FIFO is full.

Figure 12. HRDY Behavior During a Data Write Operation in the Multiplexed Mode (Case 2: Autoincrementing Selected, FIFO Empty Before Write)

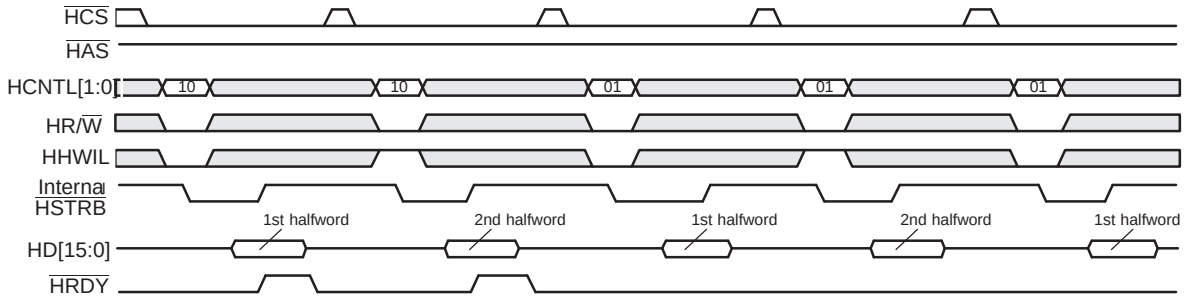
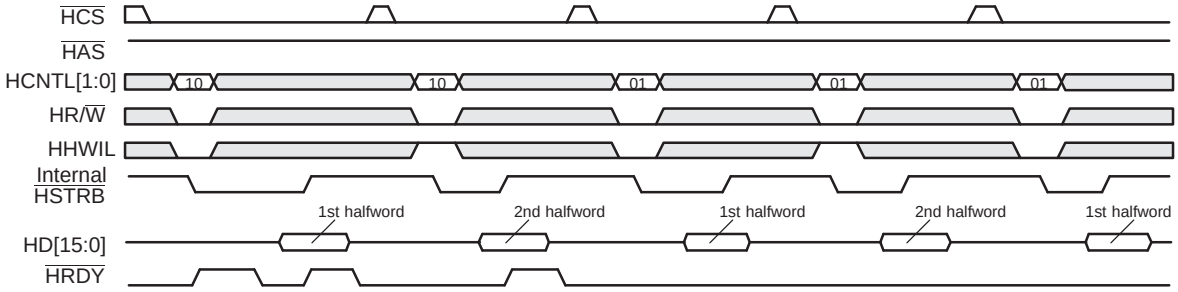


Figure 13 shows a case similar to that of Figure 12. However, in the case of Figure 13, the write FIFO is not empty when the HPIA access is made. HRDY goes high twice for the first halfword access of the HPIA write cycle. The first HRDY high period is due to the nonempty FIFO. The data currently in the FIFO must first be written to the memory. This results in HRDY going high immediately after the falling edge of the data strobe (HSTRB). The second and third HRDY high periods occur for the writes to the HPIA. HRDY remains low for the HPID accesses.

Figure 13. HRDY Behavior During a Data Write Operation in the Multiplexed Mode (Case 3: Autoincrementing Selected, FIFO Not Empty Before Write)

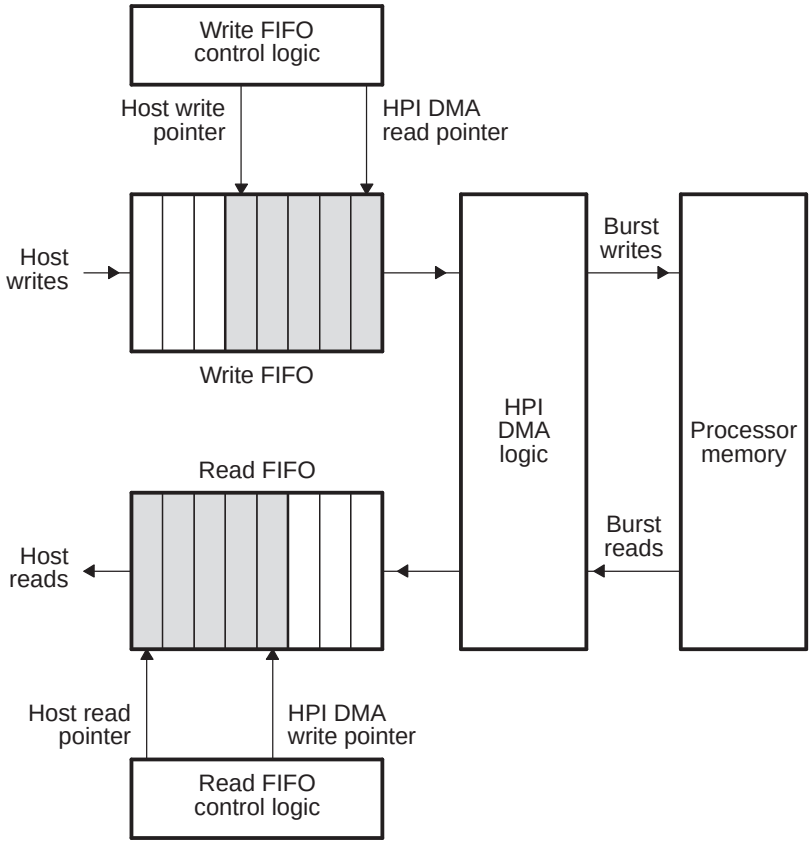


2.7.10 FIFOs and Bursting

The HPI data register (HPID) is a port through which the host accesses two first-in, first-out buffers (FIFOs). As shown in Figure 14, a read FIFO supports host read cycles, and a write FIFO supports host write cycles. Both read and write FIFOs are 8-words deep (each word is 32 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.

Figure 14. FIFOs in the HPI



2.7.10.1 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.

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 data is available in the read FIFO, 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 2.7.10.3).

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 nonautoincrement 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 HPIC 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 wants to minimize a stall condition on the interface. The host can initiate prefetching by writing 1 to the FETCH bit and later perform a read. The host can make use of the time it takes to load the read FIFO with read data, during which the HPI was not ready, by using the CPU to service other tasks.

Both types of continuous or burst reads described in the previous paragraphs begin with a write to the HPI address 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.

Note: 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 processor 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.

2.7.10.2 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 processor 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 processor 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-word 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 programmable and is configured by writing to the TIMOUT bits in the HPI configuration register (HPI_CTL). Note that you should only modify the TIMOUT bits once during device initialization. When modifying the TIMOUT bits, you must ensure that the HPI FIFOs are empty and there are no on-going HPI transactions.

In an actual system environment, the TIMOUT value should generally be set considerably larger than the typical time expected for the host to fill half of the FIFO. If the TIMOUT value is set less than this, HPI throughput performance can be significantly reduced due to excessive FIFO flushing that will result in more frequent not-ready conditions on the host interface, and decrease the overall effectiveness of the FIFO.

A practical worst-case value would be to set the timeout to the maximum latency that can be tolerated after the last word is written from the host before the HPI initiates a transfer to memory. Note that calculation of the value for worst-case latency should take into account the time required to perform the data transfer to memory, so that the overall system latency requirements are met.

Note: 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 processor memory. As soon as the host activates a write cycle without autoincrementing, bursting activity ceases until the occurrence of an autoincrement write cycle. A nonautoincrement write cycle always should be preceded by the initialization of HPIAW or by another nonautoincrement access, so that the write FIFO is flushed beforehand.

2.7.10.3 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, any read or write flush condition causes 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 a write to the HPIA register.
- 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.

2.7.10.4 FIFO Behavior When a Hardware Reset Occurs

A hardware reset (RESET pin driven low) 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 reset occurs, the HRDY signal is asserted (driven low), allowing the host to complete the cycle. When the cycle is complete, HRDY is deasserted (driven high). 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 the reset condition is true, and the host is idle (internal HSTRB is held high), the FIFOs are held in reset, and host transactions are held off with an inactive HRDY signal.

2.8 Reset Considerations

The HPI has two reset sources: software reset and hardware reset.

2.8.1 Software Reset Considerations

The HPI is not affected by a software reset issued by the emulator.

2.8.2 Hardware Reset Considerations

When the entire processor is reset with the RESET pin:

- If the internal strobe signal, internal $\overline{\text{HSTRB}}$, is high (host is inactive), $\overline{\text{HRDY}}$ is driven low and remains low until the reset condition is over.
- If internal $\overline{\text{HSTRB}}$ is high (host cycle is active), $\overline{\text{HRDY}}$ is driven low, allowing the host to complete the cycle. When internal $\overline{\text{HSTRB}}$ goes high (cycle is complete), $\overline{\text{HRDY}}$ is driven high and remains high 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 (see Section 3).
- The read and write FIFOs and the associated FIFO logic are reset (this includes a flush of the FIFOs).
- Host-to-CPU and CPU-to-host interrupts are cleared.

2.9 Initialization

The following steps are required to configure the HPI after a hardware reset:

1. Perform the necessary device pin multiplexing setup (see the device-specific data manual).
2. Program the VDD3P3V_PWDN register to power up the IO pins for the HPI (see the device-specific data manual).
3. Choose the desired time-out value for write operations by configuring the TIMOUT field in HPI_CTL. Note that you should only modify the TIMOUT bits once during device initialization. When modifying the TIMOUT bits, you must ensure that the HPI FIFOs are empty and there are no on-going HPI transactions.
4. Choose how HPIAR and HPIAW will be controlled by configuring the DUALHPIA bit in HPIC.
5. Choose how halfword ordering will be handled by configuring the HWOB bit in HPIC.
6. Choose how the HPI will respond to emulation suspend events by configuring the FREE and SOFT bits in PWREMU_MGMT.
7. Choose the desired initial addresses and write the addresses to HPIAW and HPIAR, appropriately.

The HPI is now ready to perform data transactions.

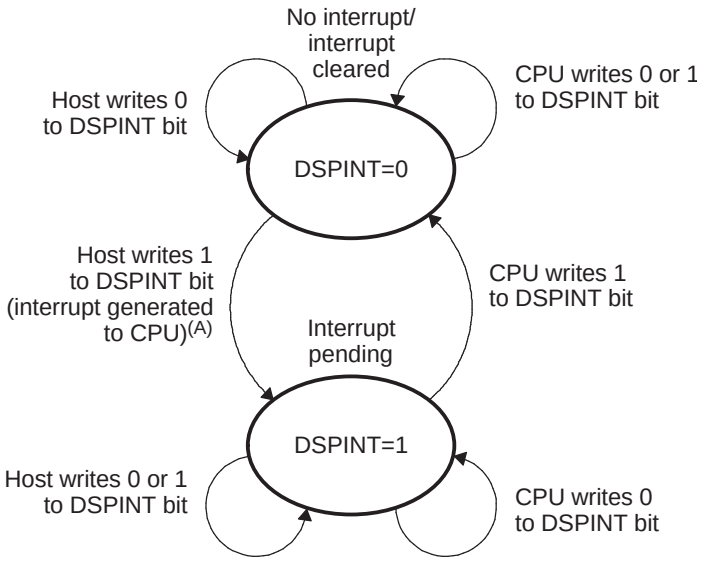
2.10 Interrupt Support

The host can interrupt the CPU via the DSPINT bit in HPIC, as described in Section 2.10.1. The CPU can send an interrupt to the host by using the HINT bit in HPIC, as described in Section 2.10.2.

2.10.1 DSPINT Bit: Host-to-CPU Interrupts

The DSPINT bit in HPIC allows the host to send an interrupt request to the CPU. The use of the DSPINT bit is summarized in Figure 15.

Figure 15. Host-to-CPU Interrupt State Diagram



A 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 in HPIC.

When the host sets the DSPINT bit, the HPI generates an interrupt pulse to 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.

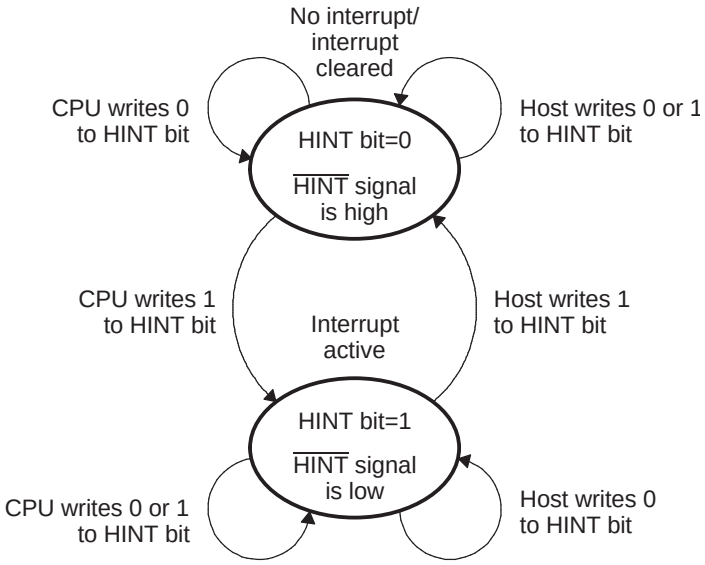
Writes of 0 have no effect. A hardware reset immediately clears DSPINT and thus clears an active host-to-CPU interrupt.

Note: When the HPIC is owned by the processor (not the host), the host is allowed to write into the interrupt fields in the HPIC register.

2.10.2 HINT Bit: CPU-to-Host Interrupts

The HINT bit in HPIC allows the CPU to send an interrupt request to the host. The use of the HINT bit is summarized in Figure 16.

Figure 16. 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 this, 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.

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

Note: When the HPIC is owned by the processor (not the host), the host is allowed to write into the interrupt fields in the HPIC register.

2.10.3 Interrupt Multiplexing

The HPI has a single interrupt source to the DSP CPU. This interrupt source is not multiplexed with any other interrupt source on the CPU.

2.11 EDMA Event Support

The HPI does not provide synchronization events to the EDMA system. Memory accesses from the HPI are handled automatically, independent of the EDMA controller. The HPI controller has its own dedicated DMA and its operation and configuration are transparent.

2.12 Power Management

The HPI peripheral can be placed in reduced-power modes to conserve power during periods of low activity. The power management of the HPI peripheral is controlled by the processor Power and Sleep Controller (PSC). The PSC acts as a master controller for power management for all of the peripherals on the device. For detailed information on power management procedures using the PSC, see the *TMS320C642x DSP Power and Sleep Controller (PSC) User's Guide* (SPRUEN8).

2.13 Emulation Considerations

The FREE and SOFT bits in the power and emulation management register (PWREMU_MGMT) 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). 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, $\overline{\text{HRDY}}$ is driven high during a host cycle that cannot be completed due to the write FIFO being full or the read FIFO being empty. If this occurs, $\overline{\text{HRDY}}$ continues to be driven high, 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.

3 Registers

Table 5 and Table 6 list the memory-mapped registers for the HPI. See the device-specific data manual for the memory addresses of these registers.

Table 5. HPI Registers Relative to Base Address 01C6 7800h

Offset	Acronym	Register Description	Section
0h	PID	Peripheral Identification Register	Section 3.1
4h	PWREMU_MGMT	Power and Emulation Management Register	Section 3.2
30h	HPIC	Host Port Interface Control Register	Section 3.3
34h	HPIAW	Host Port Interface Write Address Register	Section 3.4
38h	HPIAR	Host Port Interface Read Address Register	Section 3.5

Table 6. HPI Registers Relative to Base Address 01C4 0000h

Offset	Acronym	Register Description	Section
30h	HPI_CTL	HPI Configuration Register	Section 3.6

3.1 Peripheral Identification Register (PID)

The peripheral identification register (PID) contains identification data (class, revision, and type) for the peripheral. PID is shown in Figure 17 and described in Table 7.

Figure 17. Peripheral Identification Register (PID)

31	24	23	16
Reserved		TID	
R-0		R-1	

15	8	7	0
CID		PREV	
R-2h		R-0	

LEGEND: R = Read only; -*n* = value after reset

Table 7. Peripheral Identification Register (PID) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reserved
23-16	TID	0-Fh	Identifies type of peripheral.
		1	HPI module
15-8	CID	0-Fh	Identifies class of peripheral.
		2h	Host port.
7-0	PREV	0-Fh	Identifies revision of peripheral.
		0	Current revision of peripheral.

3.2 Power and Emulation Management Register (PWREMU_MGMT)

The power and emulation management register (PWREMU_MGMT) determines the emulation mode of the HPI. PWREMU_MGMT is shown in Figure 18 and described in Table 8.

Figure 18. Power and Emulation Management Register (PWREMU_MGMT)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		SOFT	FREE
R-0		R/W-0	R/W-0

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

Table 8. Power and Emulation Management Register (PWREMU_MGMT) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	SOFT		Determines emulation mode functionality of the HPI. When the FREE bit is cleared to 0, the SOFT bit selects the HPI mode.
		0	Upon emulation suspend, the HPI operation is not affected.
		1	In response to an emulation suspend event, the HPI logic halts after the current HPI transaction is completed.
0	FREE		Free run emulation control. Determines emulation mode functionality of the HPI. When the FREE bit is cleared to 0, the SOFT bit selects the HPI mode.
		0	The SOFT bit selects the HPI mode.
		1	The HPI runs free regardless of the SOFT bit.

3.3 Host Port Interface Control Register (HPIC)

The host port interface control register (HPIC) stores configuration and control information for the HPI. As shown in Figure 19 and Figure 20 and described in Table 9, the owner and non-owner do not have the same access permissions. The owner of HPIC has full read/write access; the non-owner of HPIC has primarily read-only access, but the exception is that the non-owner can write 1 to the HINT bit to generate an interrupt to the host.

Figure 19. Host Port Interface Control Register (HPIC) - Owner Access Permissions

31	16
Reserved	
R-0	

15	12	11	10	9	8
Reserved		HPIASEL	Reserved	DUALHPIA	HWOBSTAT
R-0		R/W-0	R/W-0	R/W-0	R-0

7	6	5	4	3	2	1	0
Reserved	Reserved		FETCH	Reserved	HINT	DSPINT	HWOB
R-0	R/W-0		R/W-0	R-1	R/W-1 (Host) R/W1C-0 (CPU)	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing 0 has no effect);; *-n* = value after reset

Figure 20. Host Port Interface Control Register (HPIC) - Non-Owner Access Permissions

31	16
Reserved	
R-0	

15	12	11	10	9	8
Reserved		HPIASEL	Reserved	DUALHPIA	HWOBSTAT
R-0		R-0	R-0	R-0	R-0

7	6	5	4	3	2	1	0
Reserved	Reserved		FETCH	Reserved	HINT	DSPINT	HWOB
R/W-0	R-0		R-0	R-1	R/W-1 (Host) R/W1C-0 (CPU)	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing 0 has no effect);; *-n* = value after reset

Table 9. Host Port Interface Control Register (HPIC) Field Descriptions

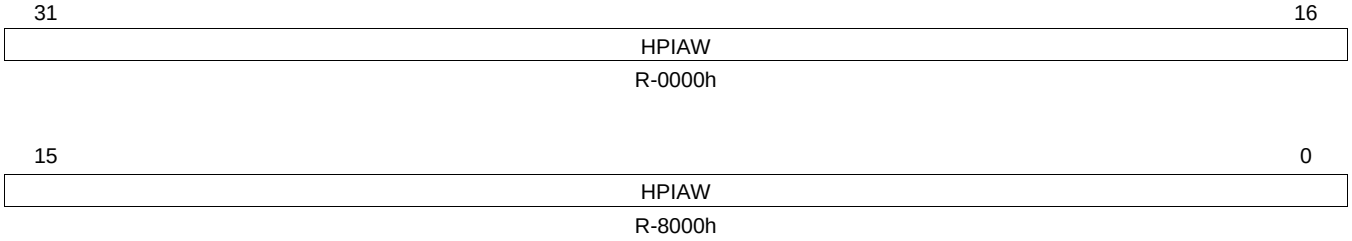
Bit	Field	Value	Description
31-12	Reserved	0	Reserved
11	HPIASEL		HPI address register select bit. When DUALHPIA = 1, the HPIASEL bit is used to select the HPI address register to be accessed.
		0	Selects the HPI write address register (HPIAW).
		1	Selects the HPI read address register (HPIAR).
10	Reserved	0	Reserved. Always write 0 to this bit.
9	DUALHPIA		Dual HPIA mode configuration bit. The CPU can access both HPI address registers separately, regardless of the DUALHPIA setting. (Regardless of this bit, dual HPIA mode is implied when the CPU has ownership of the HPI address registers).
		0	The two HPI address registers (HPIAW and HPIAR) operate as a single HPI address register in terms of host accesses.
		1	Dual HPIA mode operation is enabled.
8	HWOBSTAT		HWOB status. The value of the HWOB bit is also stored in this bit position. A write to the HWOB bit also updates HWOBSTAT.
		0	HWOB bit is logic 0.
		1	HWOB bit is logic 1.
7-5	Reserved	0	Reserved. Always write 0 to this bit.
4	FETCH		Host data fetch request bit. Only the host may write to FETCH. When a host writes a 1 to FETCH, a request is posted in the HPI to prefetch data into the read FIFO. Host and CPU reads of FETCH return a 0.
3	Reserved	1	Reserved
2	HINT		Processor-to-host interrupt. The CPU writes a 1 to HINT to generate a host interrupt. HINT has an inverted logic level to the $\overline{\text{HINT}}$ pin. The host must write a 1 to HINT to clear the $\overline{\text{HINT}}$ pin; writing a 0 to HINT by the host or processor has no effect.
		0	No effect.
		1	A CPU write generates a host interrupt ($\overline{\text{HINT}}$ signal goes low). A host write sets the $\overline{\text{HINT}}$ signal high (clears the interrupt).
1	DSPINT		Host-to-processor interrupt. The host writes a 1 to DSPINT to generate a processor interrupt; writing a 0 to DSPINT by the host or processor has no effect.
		0	No effect.
		1	A host write generates a processor interrupt.
0	HWOB		Halfword ordering bit. HWOB affects both data and address transfers. HWOB must be initialized before the first data or address register access.
		0	First halfword is most significant.
		1	First halfword is least significant.

3.4 Host Port Interface Write Address Register (HPIAW)

The HPI contains two 32-bit address registers: one for read operations (HPIAR) and one for write operations (HPIAW). The host port interface write address register (HPIAW) is shown in Figure 21 and described in Table 10. The HPI can be configured such that HPIAR and HPIAW act as a single 32-bit HPIA (single-HPIA mode) or as two separate 32-bit HPIAs (dual-HPIA mode) from the perspective of the host. For details about these HPIA modes, see Section 2.7.1.

Note that the addresses loaded into the HPI address registers must be byte addresses, and must be 32-bit word aligned (with the least-significant two bits equal to zero), for use in addressing memory space within the C642x DSP.

Figure 21. Host Port Interface Write Address Register (HPIAW)



LEGEND: R = Read only; -*n* = value after reset

Table 10. Host Port Interface Write Address Register (HPIAW) Field Descriptions

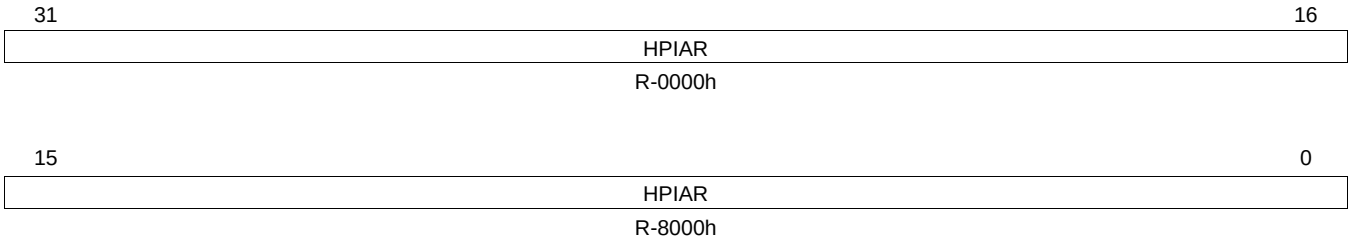
Bit	Field	Value	Description
31-0	HPIAW	0-FFFF FFFFh	Host port interface write address.

3.5 Host Port Interface Read Address Register (HPIAR)

The HPI contains two 32-bit address registers: one for read operations (HPIAR) and one for write operations (HPIAW). The host port interface read address register (HPIAR) is shown in Figure 22 and described in Table 11. The HPI can be configured such that HPIAR and HPIAW act as a single 32-bit HPIA (single-HPIA mode) or as two separate 32-bit HPIAs (dual-HPIA mode) from the perspective of the host. For details about these HPIA modes, see Section 2.7.1.

Note that the addresses loaded into the HPI address registers must be byte addresses, and must be 32-bit word aligned (with the least-significant two bits equal to zero), for use in addressing memory space within the C642x DSP.

Figure 22. Host Port Interface Read Address Register (HPIAR)



LEGEND: R = Read only; -*n* = value after reset

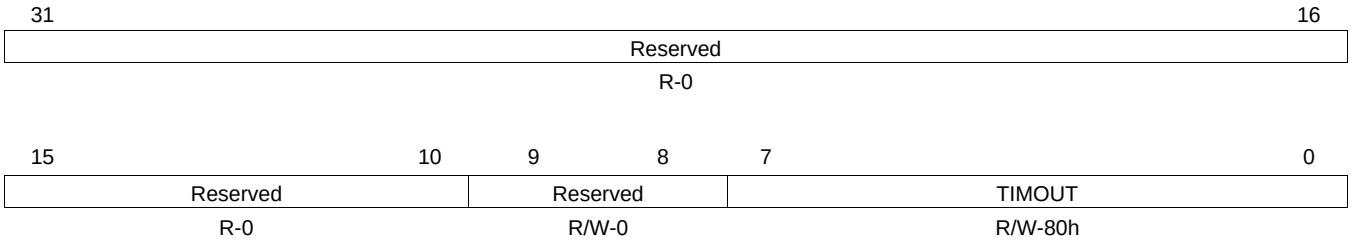
Table 11. Host Port Interface Read Address Register (HPIAR) Field Descriptions

Bit	Field	Value	Description
31-0	HPIAR	0-FFFF FFFFh	Host port interface read address.

3.6 HPI Configuration Register (HPI_CTL)

The HPI configuration register (HPI_CTL) is a 32-bit register used for configuring the write timeout FIFO value. You should only modify the TIMOUT bits once during device initialization. When modifying the TIMOUT bits, you must ensure that the HPI FIFOs are empty and there are no on-going HPI transactions. The HPI_CTL is shown in Figure 23 and described in Table 12.

Figure 23. HPI Configuration Register (HPI_CTL)



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

Table 12. HPI Configuration Register (HPI_CTL) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved.
9-8	Reserved	0	Reserved. Always write 0 to these bits.
7-0	TIMOUT	0-FFh	Write FIFO timeout value in clock cycles. When the host performs autoincrement writes into the CPU space, the CPU DMA moves data from the write FIFO in bursts of 4 words at a time. During the instance where the words to be written within the write FIFO are less than 4 words, the DMA will wait for a duration of clock cycles programmed in TIMOUT prior to the DMA moving the FIFO data to the CPU buffer (perform a flush). You should only modify the TIMOUT bits once during device initialization. When modifying the TIMOUT bits, you must ensure that the HPI FIFOs are empty and there are no on-going HPI transactions. See Section 2.7.10.2 for additional information regarding the timeout function.

Appendix A Revision History

Table A-1 lists the changes made since the previous version of this document.

Table A-1. Document Revision History

Reference	Additions/Modifications/Deletions
Section 2.7.2	Host-HPI Signal Connections: Updated Figure 2, Example of Host-Processor Signal Connections
Section 2.7.9	Hardware Handshaking Using the HPI-Ready ($\overline{\text{HRDY}}$) Signal: Updated paragraphs
Section 2.7.9.1	$\overline{\text{HRDY}}$ Behavior During Multiplexed-Mode Read Operations: Updated paragraphs
Section 2.7.9.2	$\overline{\text{HRDY}}$ Behavior During Multiplexed-Mode Write Operations: Updated paragraphs
Section 2.7.10.4	Changed section title to **FIFO Behavior When a Hardware Reset Occurs**: Deleted software reset references
Section 2.8.2	Hardware Reset Considerations: Updated second list item
Section 2.13	Emulation Considerations: Updated second list item
Section 3.3	Host Port Interface Control Register (HPIC): Updated paragraph

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 TI 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. Information of third parties may be subject to additional restrictions.

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.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

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
Clocks and Timers	www.ti.com/clocks	Digital Control	www.ti.com/digitalcontrol
Interface	interface.ti.com	Medical	www.ti.com/medical
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
RFID	www.ti-rfid.com	Telephony	www.ti.com/telephony
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

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