TMS320DM36x Digital Media System-on-Chip (DMSoC) Universal Host Port Interface (HPI)

User's Guide

Literature Number: SPRUFI4A
March 2009–Revised August 2009

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 TMS320DM36x Digital Media System-on-Chip (DMSoC).

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 TMS320DM36x Digital Media System-on-Chip (DMSoC). Copies of these documents are available on the internet at www.ti.com.

SPRUFG5 — *TMS320DM365 Digital Media System-on-Chip (DMSoC) ARM Subsystem Reference Guide* This document describes the ARM Subsystem in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The ARM subsystem is designed to give the ARM926EJ-S (ARM9) master control of the device. In general, the ARM is responsible for configuration and control of the device; including the components of the ARM Subsystem, the peripherals, and the external memories.

SPRUFG8 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Video Processing Front End (VPFE) Users Guide* This document describes the Video Processing Front End (VPFE) in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFG9 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Video Processing Back End (VPBE) Users Guide* This document describes the Video Processing Back End (VPBE) in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFH0 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) 64-bit Timer Users Guide* This document describes the operation of the software-programmable 64-bit timers in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFH1 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Serial Peripheral Interface (SPI) Users Guide* This document describes the serial peripheral interface (SPI) in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The SPI is a high-speed synchronous serial input/output port that allows a serial bit stream of programmed length (1 to 16 bits) to be shifted into and out of the device at a programmed bit-transfer rate. The SPI is normally used for communication between the DMSoC and external peripherals. Typical applications include an interface to external I/O or peripheral expansion via devices such as shift registers, display drivers, SPI EPROMs and analog-to-digital converters.

SPRUFH2 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Universal Asynchronous Receiver/Transmitter (UART) Users Guide* This document describes the universal asynchronous receiver/transmitter (UART) peripheral in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The UART peripheral performs serial-to-parallel conversion on data received from a peripheral device, and parallel-to-serial conversion on data received from the CPU.

SPRUFH3 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Inter-Integrated Circuit (I2C) Peripheral Users Guide* This document describes the inter-integrated circuit (I2C) peripheral in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The I2C peripheral provides an interface between the DMSoC and other devices compliant with the I2C-bus specification and connected by way of an I2C-bus.

SPRUFH5 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Multimedia Card (MMC)/Secure Digital (SD) Card Controller Users Guide* This document describes the multimedia card (MMC)/secure digital (SD) card controller in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFH6 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Pulse-Width Modulator (PWM) Users Guide* This document describes the pulse-width modulator (PWM) peripheral in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFH7 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Real-Time Out (RTO) Controller Users Guide* This document describes the Real Time Out (RTO) controller in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFH8 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) General-Purpose Input/Output (GPIO) Users Guide* This document describes the general-purpose input/output (GPIO) peripheral in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The GPIO peripheral provides dedicated general-purpose pins that can be configured as either inputs or outputs.

SPRUFH9 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Universal Serial Bus (USB) Controller Users Guide* This document describes the universal serial bus (USB) controller in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The USB controller supports data throughput rates up to 480 Mbps. It provides a mechanism for data transfer between USB devices and also supports host negotiation.

SPRUFI0 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Enhanced Direct Memory Access (EDMA) Controller Users Guide* This document describes the operation of the enhanced direct memory access (EDMA3) controller in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The EDMA controller's primary purpose is to service user-programmed data transfers between two memory-mapped slave endpoints on the DMSoC.

SPRUFI1 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Asynchronous External Memory Interface (EMIF) Users Guide* This document describes the asynchronous external memory interface (EMIF) in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The EMIF supports a glueless interface to a variety of external devices.

SPRUFI2 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) DDR2/Mobile DDR (DDR2/mDDR) Memory Controller Users Guide* This document describes the DDR2/mDDR memory controller in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The DDR2/mDDR memory controller is used to interface with JESD79D-2A standard compliant DDR2 SDRAM and mobile DDR devices.

SPRUFI3 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Multibuffered Serial Port Interface (McBSP) User's Guide* This document describes the operation of the multibuffered serial host port interface in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The primary audio modes that are supported by the McBSP are the AC97 and IIS modes. In addition to the primary audio modes, the McBSP supports general serial port receive and transmit operation.

SPRUFI4 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Universal Host Port Interface (UHPI) User's Guide* This document describes the operation of the universal host port interface in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFI5 — ***TMS320DM36x Digital Media System-on-Chip (DMSoC) Ethernet Media Access Controller (EMAC) User's Guide*** This document describes the operation of the ethernet media access controller interface in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFI7 — ***TMS320DM36x Digital Media System-on-Chip (DMSoC) Analog to Digital Converter (ADC) User's Guide*** This document describes the operation of the analog to digital conversion in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFI8 — ***TMS320DM36x Digital Media System-on-Chip (DMSoC) Key Scan User's Guide*** This document describes the key scan peripheral in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFI9 — ***TMS320DM36x Digital Media System-on-Chip (DMSoC) Voice Codec User's Guide*** This document describes the voice codec peripheral in the TMS320DM36x Digital Media System-on-Chip (DMSoC). This module can access ADC/DAC data with internal FIFO (Read FIFO/Write FIFO). The CPU communicates to the voice codec module using 32-bit-wide control registers accessible via the internal peripheral bus.

SPRUFJ0 — ***TMS320DM36x Digital Media System-on-Chip (DMSoC) Power Management and Real-Time Clock Subsystem (PRTCSS) User's Guide*** This document provides a functional description of the Power Management and Real-Time Clock Subsystem (PRTCSS) in the TMS320DM36x Digital Media System-on-Chip (DMSoC) and PRTC interface (PRTCIF).

SPRUGG8 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Face Detection User's Guide*This document describes the face detection capabilities for the TMS320DM36x Digital Media System-on-Chip (DMSoC).

User's Guide
SPRUFI4A–March 2009–Revised August 2009

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 TMS320DM36x DMSoC 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 DM36x DMSoC 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 ARM CPU and the external host
- HPI address register (HPIA) accessible by both the ARM 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
- 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

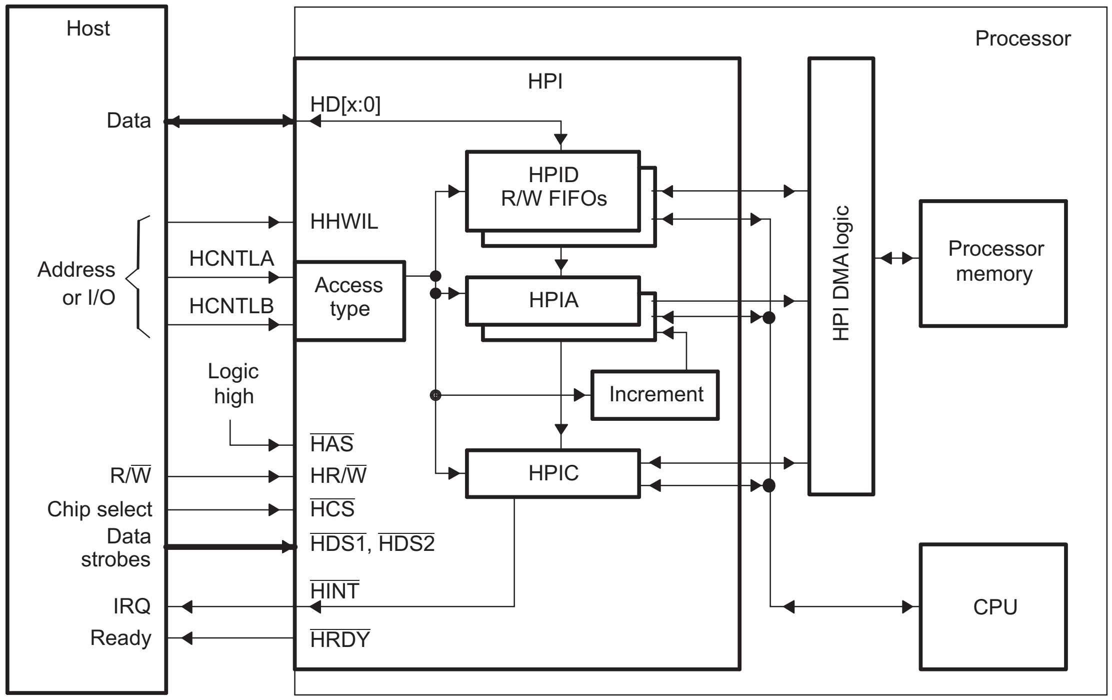
is a high-level block diagram showing how the HPI connects a host (left side of figure) and the processor internal memory (right side of figure). Host activity is asynchronous to the internal processor clock that drives the HPI. The host functions as a master to the HPI. When HPI resources are temporarily busy or unavailable, the HPI communicates this to the host by de-asserting the HPI ready ($\overline{\text{HRDY}}$) 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 in the dual-halfword mode and one 32-bit transfer in the single-fullword mode. 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 2.6.1).

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
processor	The entire digital media system-on-chip
ARM CPU	On-chip ARM processor core
CPU	ARM 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

2 Architecture

2.1 Clock Control

The HPI clock is derived from PLLC1SYSCLK4 . For detailed information on the PLLs and clock distribution on the processor, see the *TMS320DM365 Digital Media System-on-Chip Arm Subsystem Reference guide* (SPRUFG5).

2.2 Memory Map

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

- HPI configuration registers
- DDR2 Memory Controller configuration register file and memory address ranges
- Power and Sleep Controller (PSC) registers
- PLLC1 and PLLC2 registers
- ARM internal memory

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.

Table 1. HPI Pins

Pin	Type	Host Connection	Function
HCNTL[B:A]	I	Address or control pins	**HPI access control inputs**. The HPI latches the logic levels of these pins on the falling edge of internal $\overline{\text{HSTRB}}$ (for details about internal $\overline{\text{HSTRB}}$, see Section 2.6.4). The four binary states of these pins determine the access type of the current transfer (HPIC, HPIA, HPID with and without autoincrementing).
$\overline{\text{HCS}}$	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 HDS (data strobe) signal, but can be connected to an HDS pin for simultaneous select and strobe activity.
HR/$\overline{\text{W}}$	I	R/W strobe pin	**HPI read/write**. On the falling edge of 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.
HHWIL	I	Address or control pins	**Halfword identification line (Only in dual-halfword mode).** 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.
$\overline{\text{HAS}}$	I	None	**Address strobe**. Connect to logic high.
$\overline{\text{HINT}}$	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 $\overline{\text{HINT}}$ 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.
$\overline{\text{HDS1}}$ 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 2.6.4). The direction of the data transfer depends on the logic level of the HR/$\overline{\text{W}}$ 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 $\overline{\text{HDS}}$. During read operations, these pins act as output-enable pins of the host data bus.
HD[x:0]	I/O/Z	Data bus	**HPI data bus**. The HPI data bus carries the address and data to/from the HPI.
$\overline{\text{HRDY}}$	O/Z	Asynchronous ready pin	**HPI-ready signal**. When the HPI drives $\overline{\text{HRDY}}$ low, the host has permission to complete the current host cycle. When the HPI drives $\overline{\text{HRDY}}$ high, the HPI is not ready for the current host cycle to complete.

2.4 *Pin Multiplexing*

HPI is pin multiplexed with Asynchronous EMIF (AEMIF) and General-Purpose Input/Output (GPIO) at the output pin. HPI is available only when bootmode selected is HPI boot mode. In this configuration, DM36x will always act as slave device.

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 Architecture and Operation

2.6.1 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).

2.6.1.1 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[B:A] = 01b, HR/$\overline{\text{W}}$ = 0) updates HPIAR and HPIAW with the same value.
- Both HPI address registers are incremented during autoincrement read/write cycles (HCNTL[B:A] = 10b).
- An HPIA read cycle (HCNTL[B:A] = 01b, HR/$\overline{\text{W}}$ = 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.

2.6.1.2 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[B:A] = 01b) 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.

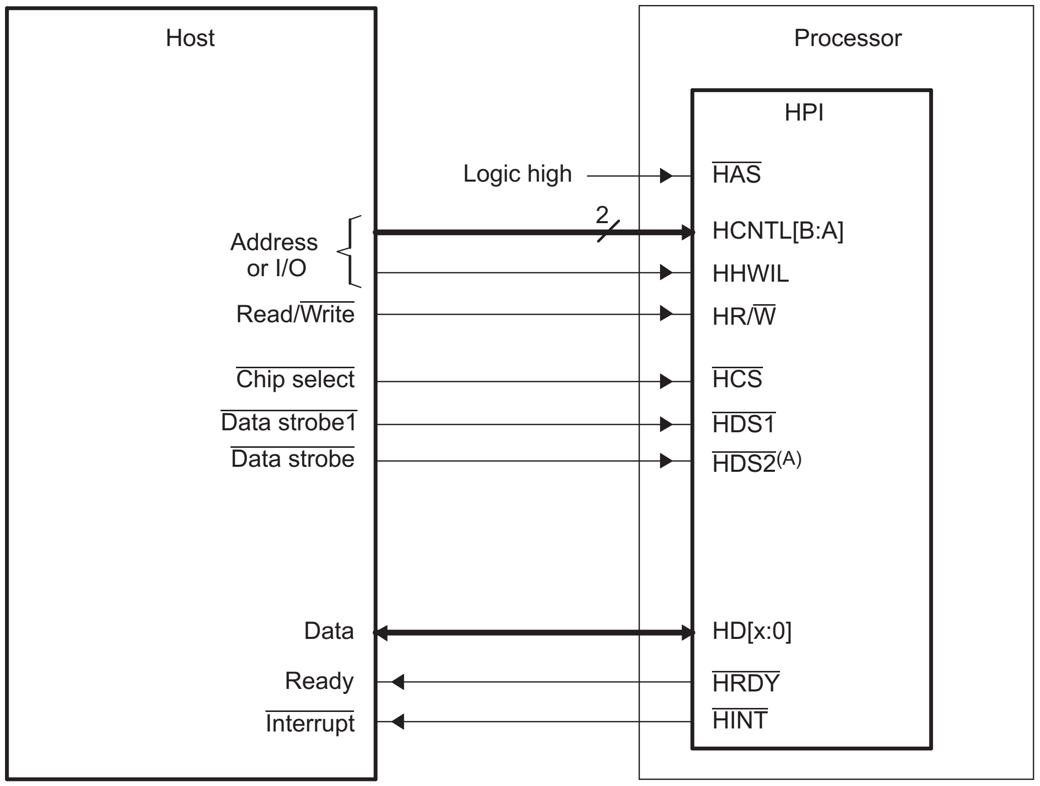
NOTE: The HPIASEL bit does not affect the HPI DMA logic. Regardless of the value of HPIASEL, the HPI DMA logic uses HPIAR when reading from memory and HPIAW when writing to memory.

- A host HPID access with autoincrementing (HCNTL[B:A] = 10b) 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.

2.6.2 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 2.6.4

2.6.3 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) in dual-halfword mode and the HPIA-related bits (DUALHPIA and HPIASEL) in both dual-halfword and single-fullword modes. 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 2.6.1.
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 HCNTLB and HCNTLA signals to indicate which register is to be accessed. The host must also drive the appropriate level on the HR/$\overline{\text{W}}$ 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 de-asserting the HPI-ready ($\overline{\text{HRDY}}$) output signal.

When performing an access, the HPI first latches the levels on HCNTL[B:A], HR/$\overline{\text{W}}$, and other control signals. This latching can occur on the falling edge of the internal strobe signal (for details, see Section 2.6.4). After the control information is latched, the HPI initiates an access based on the control signals.

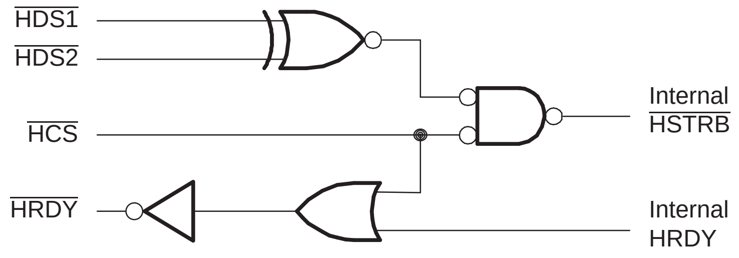
If the host wants to read data from processor resources (see Section 2.2), the HPI DMA logic reads the resource address from HPIAR and retrieves the data from the addressed memory location. When the data has been placed in HPID, the HPI drives the data onto its HD bus. The $\overline{\text{HRDY}}$ signal informs the host whether the data on the HD bus is valid ($\overline{\text{HRDY}}$ low) or not valid yet ($\overline{\text{HRDY}}$ high). 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 processor resources (see Section 2.2), the operation is similar. After the host determines that the HPI is ready to latch the data ($\overline{\text{HRDY}}$ is low), 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.

2.6.4 $\overline{\text{HDS2}}$, $\overline{\text{HDS1}}$, and $\overline{\text{HCS}}$: Data Strobing and Chip Selection

As shown in Figure 3, the strobing logic is a function of three key inputs: the chip select pin ($\overline{\text{HCS}}$) 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 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 $\overline{\text{HDS}}$ pins.

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

NOTE: The $\overline{\text{HCS}}$ input and one $\overline{\text{HDS}}$ 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 $\overline{\text{HRDY}}$ is gated by $\overline{\text{HCS}}$ as previously described.

It is not recommended to tie both $\overline{\text{HDS1}}$ and $\overline{\text{HDS2}}$ to static logic levels and use $\overline{\text{HCS}}$ 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 $\overline{\text{HDS1}}$ and the other to $\overline{\text{HDS2}}$ [1]. Since such a host might not provide a R/$\overline{\text{W}}$ line, take care to satisfy HR/$\overline{\text{W}}$ 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 $\overline{\text{HDS1}}$ and the other to $\overline{\text{HDS2}}$ [1]. Since such a host might not provide a R/$\overline{\text{W}}$ line, take care to satisfy HR/$\overline{\text{W}}$ 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 $\overline{\text{HDS1}}$ or $\overline{\text{HDS2}}$, and connect the other pin to logic-level 1.
Host has one active-high strobe pin	Connect the strobe pin to $\overline{\text{HDS1}}$ or $\overline{\text{HDS2}}$, and connect the other strobe pin to logic-level 0.

[1] The HR/$\overline{\text{W}}$ signal could be driven by a host address line in this case.

2.6.5 HCNTL[B:A] and HR/$\overline{\text{W}}$: Indicating the Cycle Type

The cycle type consists of:

- The access type that the host selects by driving the appropriate levels on the HCNTL[B:A] pins of the HPI. Table 3 describes the four available access types.
- The transfer direction that the host selects with the HR/$\overline{\text{W}}$ pin. The host must drive the HR/$\overline{\text{W}}$ 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 $\overline{\text{HSTRB}}$.

Table 3. Access Types Selectable With the HCNTL Signals

HCNTLB	HCNTLA	Access Type
0	0	**HPIC access.** The host requests to access the HPI control register (HPIC).
0	1	**HPIA access.** The host requests to access the appropriate HPI address register (HPIAR and/or HPIAW).
1	0	**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	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 4. Cycle Types Selectable With the HCNTL and HR/$\overline{\text{W}}$ Signals

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

2.6.6 HPI Dual-Halfword Access

In dual-halfword mode, each host cycle consists of two consecutive halfword transfers. For each transfer, the host must specify the cycle type with HCNTL [B:A] and HR/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.

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 a dual access.

2.6.6.1 HPI Dual-Halfword Read

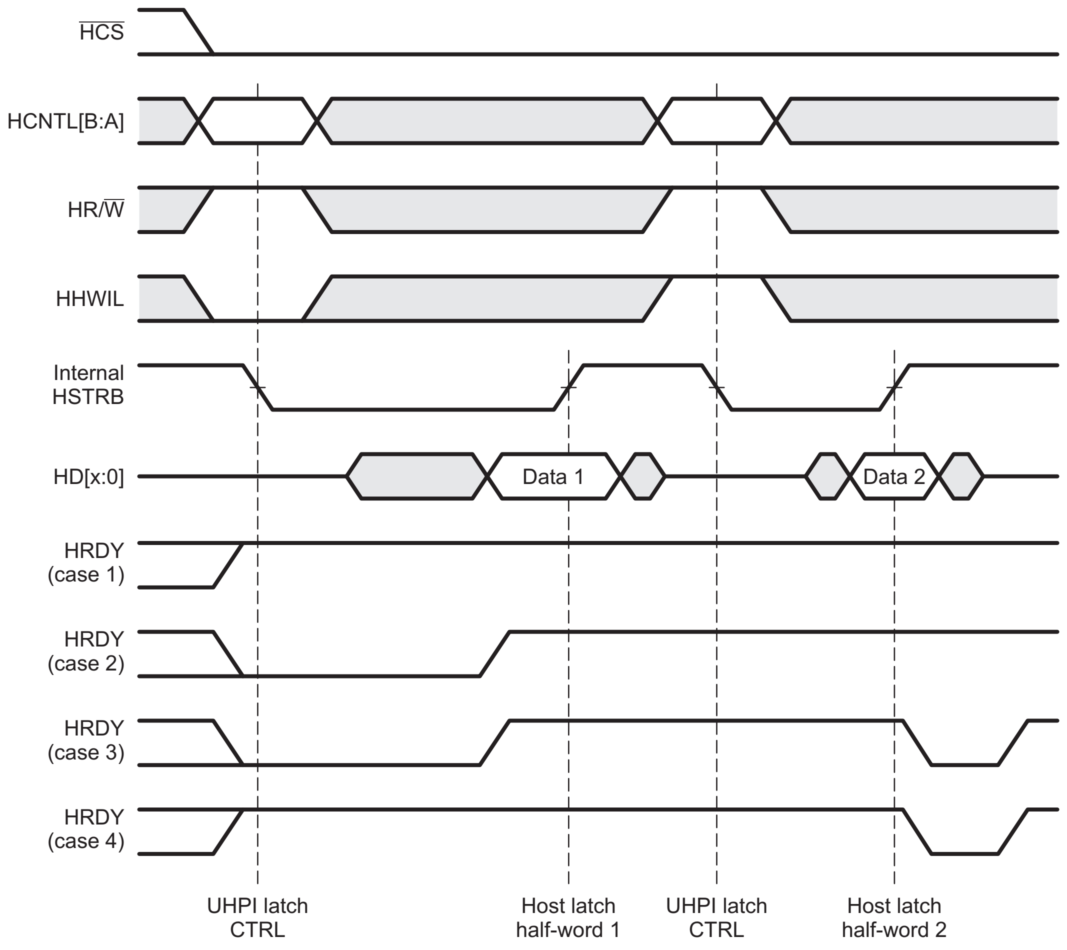
Figure 4 illustrates a HPI read cycle in dual-halfword mode.

First, the HCNTL[B:A], HR/W and HHWIL levels are latched at the falling edge of the internal HSTRB. Then the HPI interface initiates a read of the HPIA, the HPIC, or CPU memory (via the Vbus master and CPU DMA). If the read is from CPU memory, the address for the memory read is the value located in the HPIA and XHPIA registers. When the read data has been retrieved, the HPI drives the data onto the HD[x:0] bus. When the HD[x:0] bus is valid, the host should then latch this data by causing the internal HSTRB to transition from low to high. This can be achieved using /HDS1, /HDS2, and /HCS. This process is repeated for the second halfword of the access.

By de-asserting the HRDY signal, the HPI can insert wait states to indicate to the host that the read data will not be valid for a number of CPU clock cycles. 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 number of wait states required may be related to the amount of concurrent DMA activity to CPU memory. The host can begin a new cycle by latching new control information even if the HRDY signal is de-asserted inactive low. Once an external host cycle has been initiated, however, the host must not cause an internal HSTRB rising edge until HRDY is asserted active high. If an internal HSTRB rising edge is caused by the external host while HRDY is de-asserted, the cycle will be terminated with invalid data being returned (read cycle) or written to memory (write cycle). Accesses to the HPIC and HPIA registers should never cause HRDY to be de-asserted with the exception of the following conditions:

- A reset condition was active just before and during the access (Vbus reset or HPI_RST).
- An HPIA access occurs before the Vbus side address register has been updated from a previous HPIA access.

Figure 4. Host Read in Dual-Halfword Mode



HRDY behavior will vary depending on the cycle type and previous conditions. There are four possible cases of HRDY behavior. In every case, HRDY may be initially de-asserted due to prefetch or memory write activity caused by the previous cycle. HRDY behavior is illustrated in Figure 4 and described below:

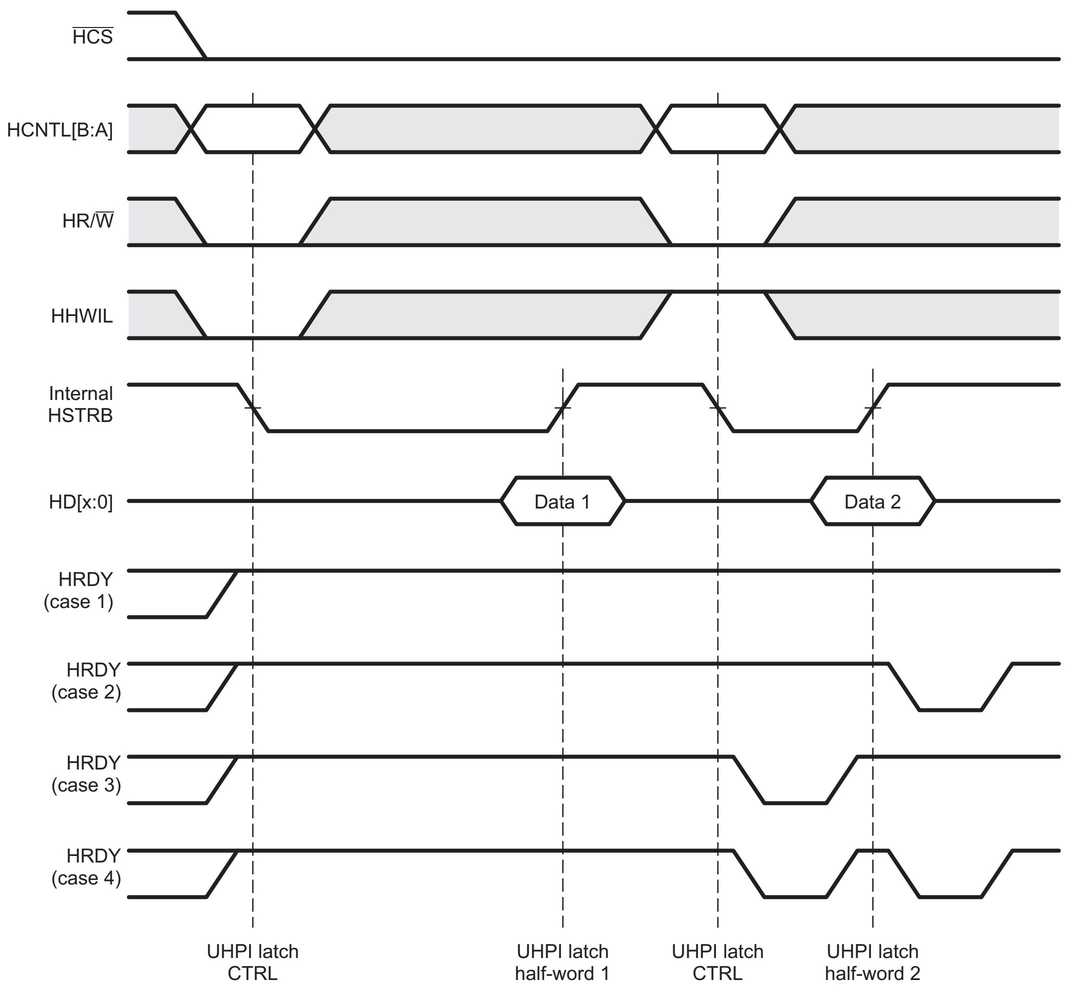
- Case 1: HPIC/HPIA read or HPID read with auto-increment and data was previously prefetched into the read FIFO.
- Case 2: HPID read with no auto-increment or with auto-increment and read FIFO was initially empty.
- Case 3: HPID read with auto-increment, read FIFO was initially empty and was empty after the cycle (may never happen).
- Case 4: HPID read with auto-increment and data was previously prefetched but the read FIFO became empty as a result of the cycle.

2.6.6.2 HPI Dual-Halfword Write

The duel-halfword write cycle is similar to the dual-halfword read cycle (see Section 2.6.6.1) except that during HPID access, CPU memory access can't be initiated until after the second rising edge of internal HSTRB. The first rising edge of internal HSTRB causes the first halfword of the write to be latched into the HPI. The second rising edge of internal HSTRB causes the second halfword of the write to be latched into the HPI.

Figure 5 illustrates a HPI write cycle in dual-halfword mode.

Figure 5. Host Write in Dual-Halfword Mode



HRDY behavior will vary depending on the cycle type and previous conditions. There are four possible cases of HRDY behavior. In all cases, HRDY may be initially de-asserted due to prefetch or memory write activity caused by the previous cycle. HRDY behavior is illustrated in Figure 5 and described below:

- Case 1: HPIC/HPIA write or HPID auto-increment write with space available in write FIFO.
- Case 2: HPID write without auto-increment or with auto-increment and write FIFO became full as a result of the cycle.
- Case 3: HPID write with auto-increment and write FIFO was initially full but is not full after the cycle.
- Case 4: HPID write without auto-increment and write FIFO was initially not empty, or HPID write with auto-increment and write FIFO was initially full and became full again as a result of the host cycle.

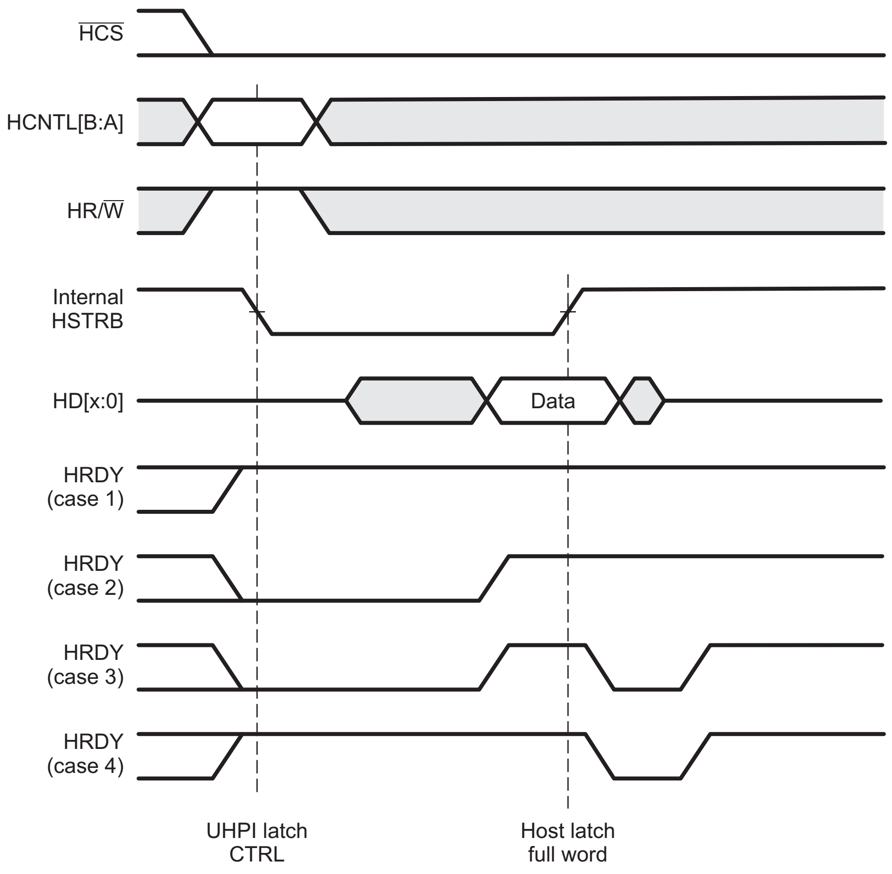
2.6.7 HPI Single-Fullword Access

2.6.7.1 HPI Single-Fullword Read

The single-fullword read cycle is similar to the dual-halfword read cycle (see Section 2.6.6.1) except the entire word is read in a single access so the HHWIL bit is not used.

Figure 6 illustrates a HPI read cycle in single-fullword mode.

Figure 6. Host Read in Single-Fullword Mode



HRDY behavior will vary depending on the cycle type and previous conditions. The are four possible cases of HRDY behavior. In all cases, HRDY may be initially de-asserted due to prefetch or memory write activity caused by the previous cycle. HRDY behavior is illustrated in Figure 6 and described below:

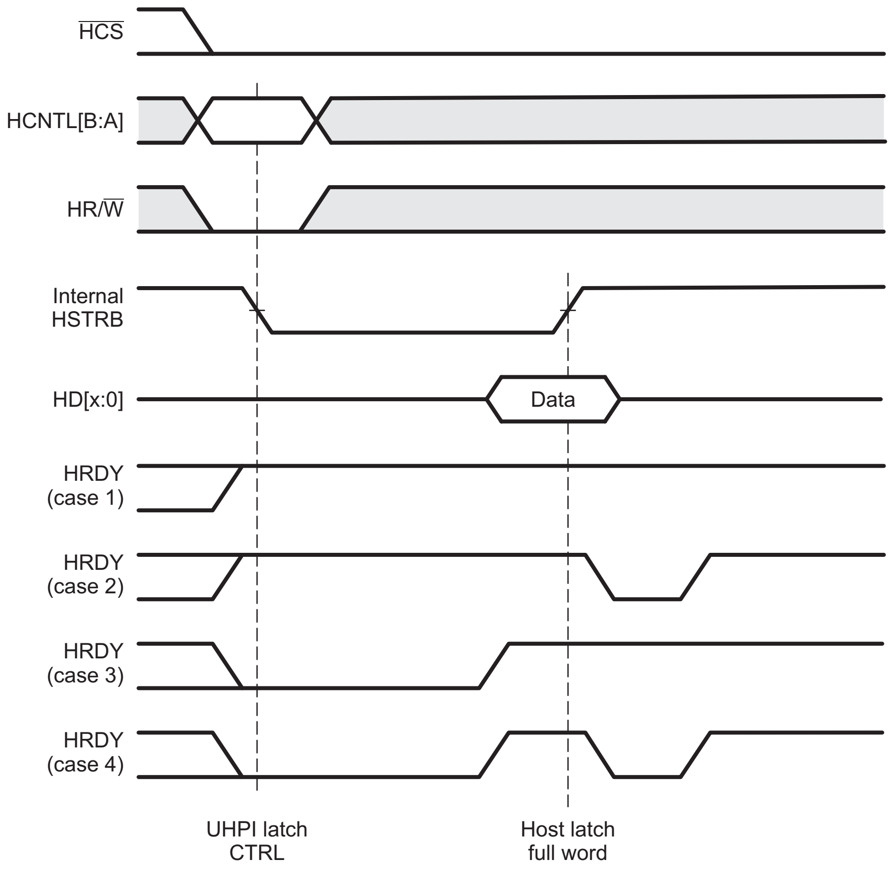
- Case 1: HPIC/HPIA read or HPID read with auto-increment and data was previously prefetched into the read FIFO.
- Case 2: HPID read with no auto-increment or with auto-increment and read FIFO was initially empty.
- Case 3: HPID read with auto-increment, read FIFO was initially empty, and read FIFO was still empty after the cycle (may never happen).
- Case 4: HPID read with auto-increment and data was previously prefetched but the read FIFO became empty as a result of the cycle.

2.6.7.2 HPI Single-Fullword Write

The single-fullword write cycle is similar to the dual-halfword write cycle (see Section 2.6.6.2) except the entire word is read in a single access so the HHWIL bit is not used.

Figure 7 illustrates a HPI write cycle in single-fullword mode.

Figure 7. Host Write in Single-Fullword Mode



HRDY behavior will vary depending on the cycle types and previous conditions. The are four possible cases of HRDY behavior. In all cases, HRDY may be initially de-asserted due to prefetch or memory write activity caused by the previous cycle. HRDY behavior is illustrated in Figure 7 and described below:.

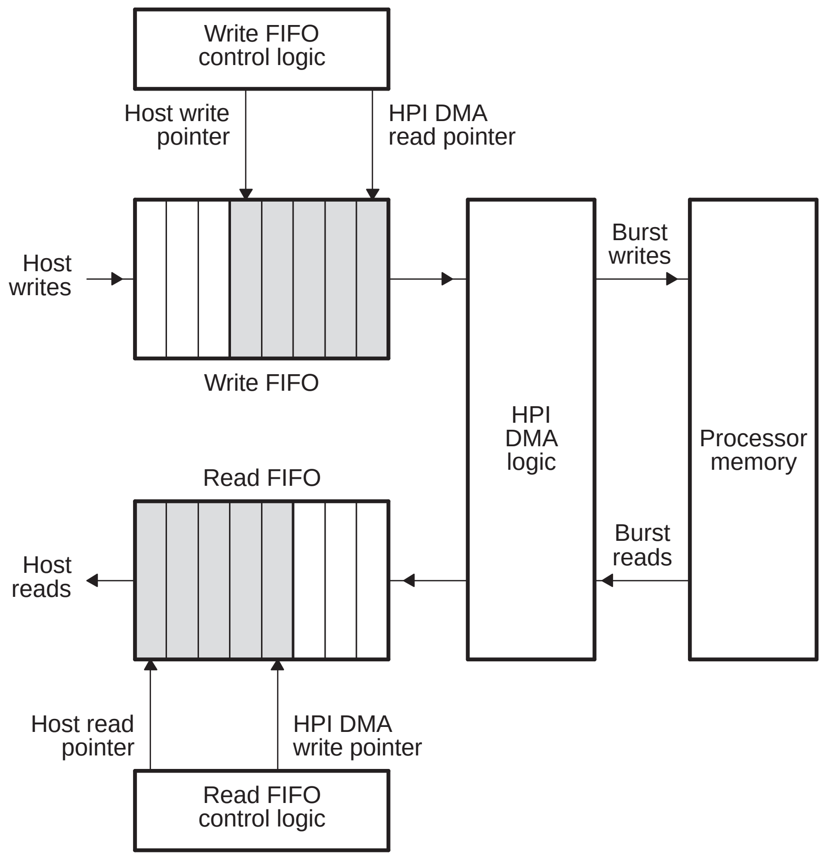
- Case 1: HPIC or HPIA write or HPID write with auto-increment and space was available in write FIFO.
- Case 2: HPID write without auto-increment or with auto-increment and write FIFO becomes full as a result of the cycle.
- Case 3: HPID auto-increment write with write FIFO initially full or an HPIA write while the Vbus side address register is still being updated from a previous HPIA write.
- Case 4: HPID write without auto-increment and write FIFO is initially not empty or HPID write with auto-increment and write FIFO was initially full and became full again as a result of the cycle.

2.6.8 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 8, 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 $\overline{\text{HRDY}}$ signal is de-asserted less often when there are multiple reads or writes to consecutive addresses.

Figure 8. FIFOs in the HPI



2.6.8.1 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 de-assertion of the $\overline{\text{HRDY}}$ 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 2.6.8.3).

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.

2.6.8.2 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 de-asserting $\overline{\text{HRDY}}$ 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 bit in the HPI Control Register (HPICTL).

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.

2.6.8.3 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 de-assertion of $\overline{\text{HRDY}}$ 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.

2.6.8.4 FIFO Behavior When a Hardware Reset or Software Reset Occurs

A hardware reset (RESET pin driven low) or an HPI software reset (HPI_RST = 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 $\overline{\text{HRDY}}$ signal is asserted (driven low), allowing the host to complete the cycle. When the cycle is complete, $\overline{\text{HRDY}}$ is de-asserted (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 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 $\overline{\text{HRDY}}$ signal.

2.7 Reset Considerations

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

2.7.1 Software Reset Considerations

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

2.7.2 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 low (host cycle is active), $\overline{\text{HRDY}}$ is driven high, allowing the host to complete the cycle. When internal $\overline{\text{HSTRB}}$ goes high (cycle is complete), $\overline{\text{HRDY}}$ 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 (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.

2.8 Initialization

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

1. Choose whether the host or the CPU will be the owner of the host port interface control register (HPIC) by configuring the CTLMODE bit of the HPI control register (HPICTL) in the System Module.
2. Choose whether the host or the CPU will be the owner of the host port interface address register (HPIA) by configuring the ADDMODE bit in HPICTL.
3. Choose the desired time-out value for write operations by configuring the TIMOUT field in HPICTL.
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 the power and emulation management register (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.

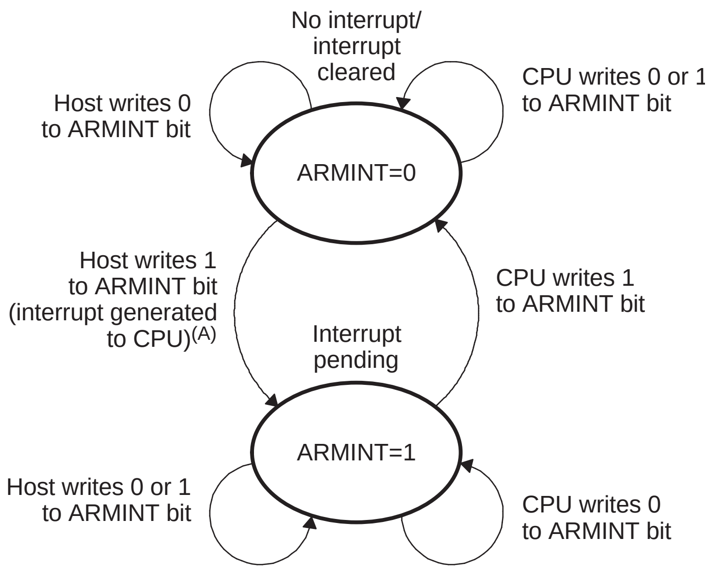
2.9 Interrupt Support

The host can interrupt the CPU via the ARMINT bit in HPIC, as described in Section 2.9.1 . The CPU can send an interrupt to the host by using the HINT bit in HPIC, as described in Section 2.9.2.

2.9.1 ARMINT Bit: Host-to-CPU Interrupts

The ARMINT bit in HPIC allows the host to send an interrupt request to the CPU. The use of the ARMINT bit is summarized in Figure 9.

Figure 9. Host-to-CPU Interrupt State Diagram


No interrupt/ interrupt cleared
Host writes 0 to ARMINT bit
CPU writes 0 or 1 to ARMINT bit
ARMINT=0
Host writes 1 to ARMINT bit (interrupt generated to CPU)(A)
CPU writes 1 to ARMINT bit
Interrupt pending
ARMINT=1
Host writes 0 or 1 to ARMINT bit
CPU writes 0 to ARMINT bit

A When the ARMINT 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 (ARMINT = 0).

To interrupt the CPU, the host must:

1. Drive both HCNTLB and HCNTLA low to request a write to HPIC.
2. Write 1 to the ARMINT bit in HPIC.

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

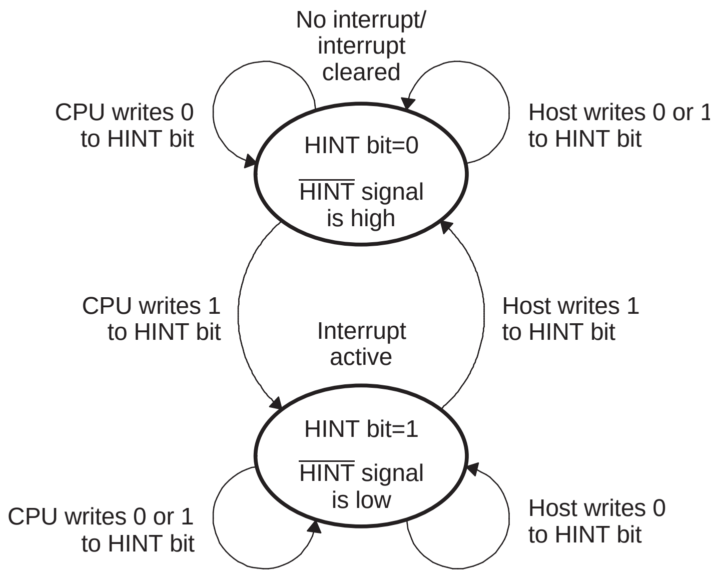
Writes of 0 have no effect. A hardware reset immediately clears ARMINT and thus clears an active host-to-CPU interrupt.

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

2.9.2 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 10.

Figure 10. CPU-to-Host Interrupt State Diagram


No interrupt/ interrupt cleared
CPU writes 0 to HINT bit
Host writes 0 or 1 to HINT bit
HINT bit=0
$\overline{\text{HINT}}$ signal is high
CPU writes 1 to HINT bit
Interrupt active
Host writes 1 to HINT bit
HINT bit=1
$\overline{\text{HINT}}$ signal is low
CPU writes 0 or 1 to HINT bit
Host writes 0 to HINT bit

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 onto the interrupt fields in the HPIC register.

2.9.3 Interrupt Multiplexing

The HPI has a single interrupt source (Table 5) to the ARM CPU . This interrupt source is multiplexed with Asynchronous EMIF interrupt source.

Table 5. HPI Interrupt

ARM Event	Acronym	Source
30	HPIINT	HPI

2.10 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 it's own dedicated DMA and it's operation and configuration are transparent.

2.11 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 *TMS320DM365 Digital Media System-on-Chip (DMSoC)* (SPRUFG5).

2.12 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 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, $\overline{\text{HRDY}}$ 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.

3 Registers

Table 6 and Table 7 lists the memory-mapped registers for the HPI. See the device-specific data manual for the memory addresses of these registers.

Table 6. HPI Registers

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 7. HPI Register in System Module

Offset	Acronym	Register Description	Section
30h	HPICTL	HPI Control 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 11 and described in Table 8.

Figure 11. Peripheral Identification Register (PID)

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

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

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

Table 8. 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.
		2	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 12 and described in Table 9.

Figure 12. 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 9. 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 13 and Figure 14 and described in Table 10, 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 exceptions are:

- the non-owner can write 1 to the HINT bit to generate an interrupt to the host.
- the non-owner can write 1 to HPI_RST bit to cause an HPI software reset.

Figure 13. 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	ARMINT	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 14. 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	ARMINT	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 10. 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	ARMINT		Host-to-processor interrupt. The host writes a 1 to ARMINT to generate a processor interrupt; writing a 0 to ARMINT 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 15 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 2.6.1.

Figure 15. Host Port Interface Write Address Register (HPIAW)

31	16
HPIAW	
R-0000h	

15	0
HPIAW	
R-8000h	

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

Table 11. 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 16 and described in Table 12. 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 2.6.1.

Figure 16. Host Port Interface Read Address Register (HPIAR)

31	16
HPIAR	
R-0000h	

15	0
HPIAR	
R-8000h	

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

Table 12. 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 Control Register (HPICTL)

The HPI control register (HPICTL) in the System Module controls write access to the HPI control and address registers and determines the host time-out value. It also determines operation of the HPI wrapper. HPICTL is not reset by a soft reset so that the HPI data width will remain correctly configured. The HPICTL is shown in Figure 17 and described in Table 13.

Figure 17. HPI Control Register (HPICTL)

31–16
Reserved
R-0

15–10	9	8	7–0
Reserved	CTLMODE	ADDMODE	TIMEOUT
R-0	R/W-0	R/W-0	R/W-80h

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

Table 13. HPI Control Register (HPICTL) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved		Any writes to these bit(s) must always have a value of 0.
9	CTLMODE		HPIC register write access.
		0	Host.
		1	Processor (if ADDMODE = 1)
8	ADDMODE		HPIA register write access.
		0	Host
		1	Processor
7-0	TIMOUT	0-FFh	Host burst write time-out value.

Appendix A Revision History

This document has been revised to include the following technical change(s).

Table 14. Document Revision History

Reference	Additions/Modifications/Deletions
Global	All instances of HCNTL0 were changed to HCNTLA; all instances of HCNTL1 were changed to HCNTLB; all instances of HCNTL[1:0] were changed to HCNTL[B:A].