List of Figures

1 HPI Block Diagram 8
2 Example of Host-Processor Signal Connections 12
3 HPI Strobe and Select Logic 14
4 Multiplexed-Mode Host Read Cycle 16
5 Multiplexed-Mode Host Write Cycle 17
6 Multiplexed-Mode Single-Halfword HPIC Cycle (Read or Write) 18
7 $\overline{\text{HRDY}}$ Behavior During an HPIC or HPIA Read Cycle in the Multiplexed Mode 19
8 $\overline{\text{HRDY}}$ Behavior During a Data Read Operation in the Multiplexed Mode (Case 1: HPIA Write Cycle Followed by Nonautoincrement HPID Read Cycle) 19
9 $\overline{\text{HRDY}}$ Behavior During a Data Read Operation in the Multiplexed Mode (Case 2: HPIA Write Cycle Followed by Autoincrement HPID Read Cycles) 19
10 $\overline{\text{HRDY}}$ Behavior During an HPIC Write Cycle in the Multiplexed Mode 20
11 $\overline{\text{HRDY}}$ Behavior During a Data Write Operation in the Multiplexed Mode (Case 1: No Autoincrementing) 20
12 $\overline{\text{HRDY}}$ Behavior During a Data Write Operation in the Multiplexed Mode (Case 2: Autoincrementing Selected, FIFO Empty Before Write) 20
13 $\overline{\text{HRDY}}$ Behavior During a Data Write Operation in the Multiplexed Mode (Case 3: Autoincrementing Selected, FIFO Not Empty Before Write) 21
14 FIFOs in the HPI 22
15 Host-to-CPU Interrupt State Diagram 26
16 CPU-to-Host Interrupt State Diagram 27
17 Peripheral Identification Register (PID) 29
18 Power and Emulation Management Register (PWREMU_MGMT) 30
19 Host Port Interface Control Register (HPIC)–Owner Access Permissions 31
20 Host Port Interface Control Register (HPIC)–Non-Owner Access Permissions 31
21 Host Port Interface Write Address Register (HPIAW) 33
22 Host Port Interface Read Address Register (HPIAR) 34
23 HPI Configuration Register (HPI_CTL) 35

List of Tables

1 HPI Pins 10
2 Options for Connecting Host and HPI Data Strobe Pins 14
3 Access Types Selectable With the HCNTL Signals 15
4 Cycle Types Selectable With the HCNTL and HR/$\overline{\text{W}}$ Signals 15
5 HPI Registers Relative to Base Address 01C6 7800h 28
6 HPI Registers Relative to Base Address 01C4 0000h 28
7 Peripheral Identification Register (PID) Field Descriptions 29
8 Power and Emulation Management Register (PWREMU_MGMT) Field Descriptions 30
9 Host Port Interface Control Register (HPIC) Field Descriptions 32
10 Host Port Interface Write Address Register (HPIAW) Field Descriptions 33
11 Host Port Interface Read Address Register (HPIAR) Field Descriptions 34
12 HPI Configuration Register (HPI_CTL) Field Descriptions 35
A-1 Document Revision History 36

Read This First

About This Manual

This document describes the features and operation of the host port interface (HPI) in the TMS320DM643x Digital Media Processor (DMP).

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 TMS320DM643x Digital Media Processor (DMP). Copies of these documents are available on the Internet at [redacted]. *Tip:* Enter the literature number in the search box provided at [redacted].

The current documentation that describes the DM643x DMP, related peripherals, and other technical collateral, is available in the C6000 DSP product folder at: [redacted]/c6000.

SPRU978 — ***TMS320DM643x DMP DSP Subsystem Reference Guide.*** Describes the digital signal processor (DSP) subsystem in the TMS320DM643x Digital Media Processor (DMP).

SPRU983 — ***TMS320DM643x DMP Peripherals Overview Reference Guide.*** Provides an overview and briefly describes the peripherals available on the TMS320DM643x Digital Media Processor (DMP).

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.

User's Guide
SPRU998C–February 2007

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 TMS320DM643x DMP 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 DM643x DMP 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

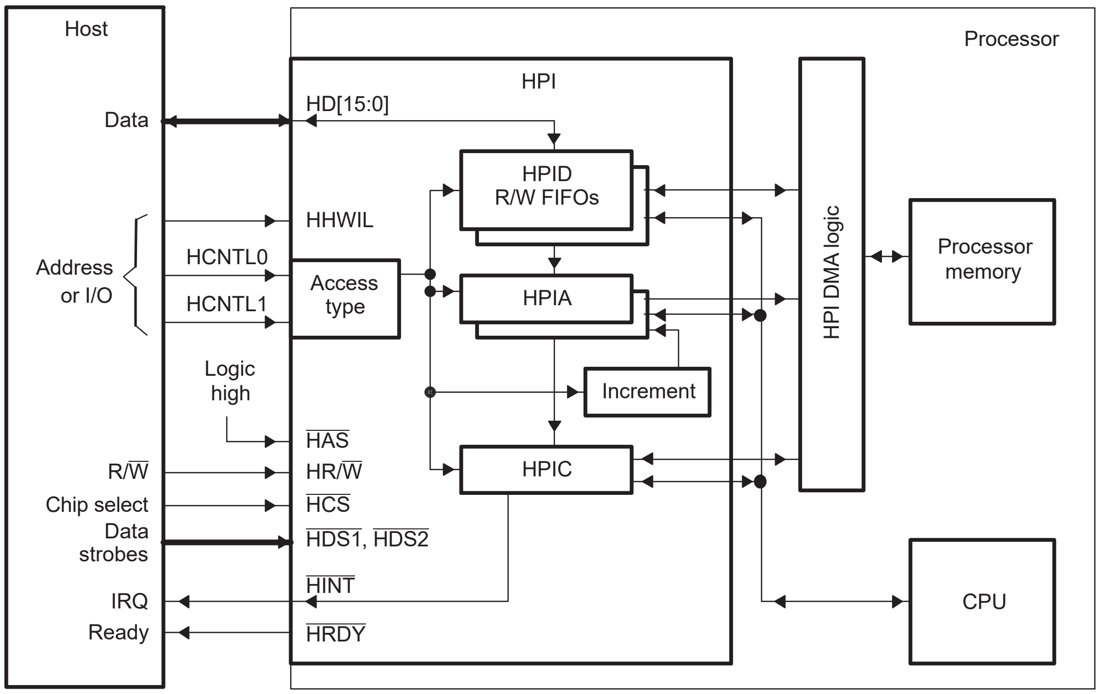
Figure 1 is a high-level block diagram showing how the HPI connects a host (left side of figure) and the processor internal memory (right side of figure). Host activity is asynchronous to the internal processor clock that drives the HPI. The host functions as a master to the HPI. When HPI resources are temporarily busy or unavailable, the HPI communicates this to the host by deasserting the HPI ready ($\overline{\text{HRDY}}$) 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. 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 SYSCLK3, which is the PLL1 clock divided by 6. For detailed information on the PLLs and clock distribution on the processor, see the *TMS320DM643x DMP DSP Subsystem Reference Guide* (SPRU978).

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 DM643x DMP 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 DM643x DMP 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 $\overline{\text{HSTRB}}$ (for details about internal $\overline{\text{HSTRB}}$, 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).
$\overline{\text{HCS}}$	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**. 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 2.7.4). 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[15:0]	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.

2.7 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 DM643x DMP.

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/$\overline{\text{W}}$ = 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/$\overline{\text{W}}$ = 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.

2.7.1.2 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.

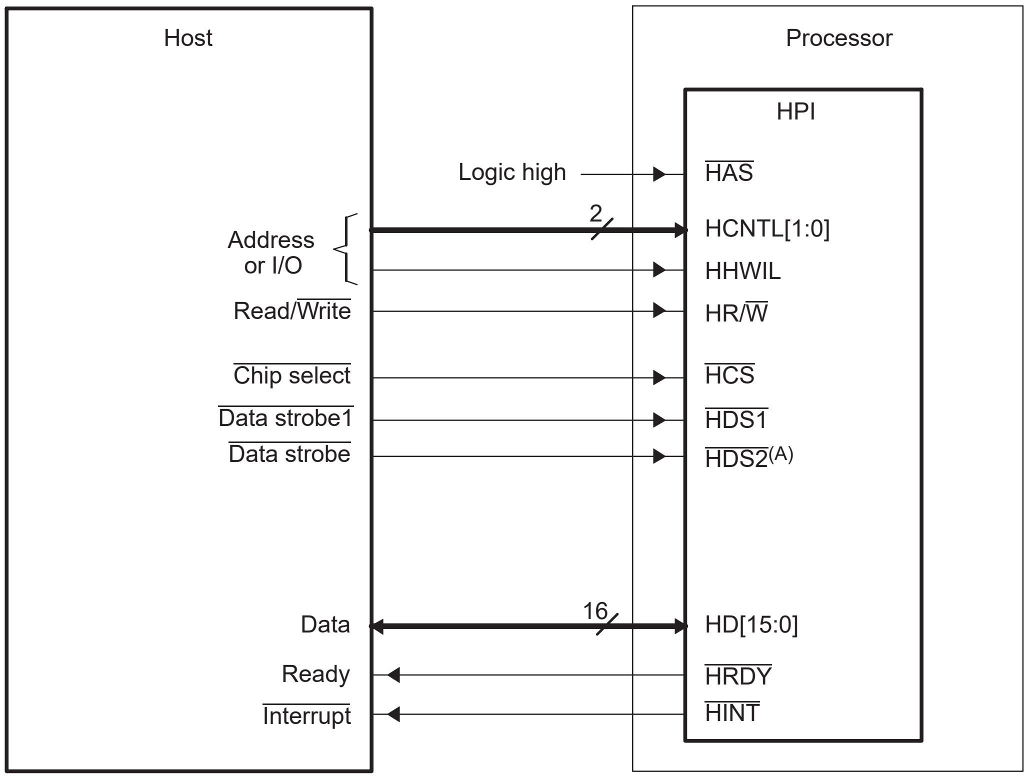
Note: The HPIASEL bit does not affect the HPI DMA logic. Regardless of the value of HPIASEL, the HPI DMA logic uses HPIAR when reading from memory and HPIAW when writing to memory.

- 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/$\overline{\text{W}}$ 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 deasserting the HPI-ready ($\overline{\text{HRDY}}$) output signal.

When performing an access, the HPI first latches the levels on HCNTL[1:0], 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 2.7.4). 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 $\overline{\text{HRDY}}$ 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.