AN1808
Rev. 1, 8/2005

DSP56300 HI08 Host Port Programming

By Duberly Mazuelos

This application note contains information about programming the HI08 Host Port peripheral of the Freescale DSP56300 DSP family. It supplements the information in the user's manuals. While DSP experience is not required, some experience with embedded applications will help you make trade-offs between available HI08 options during system integration. Each section of this document includes the relevant signals and registers.

CONTENTS

1 HI08 Basics

The HI08 host port is a Freescale DSP56300 family peripheral that provides a byte-wide, full-duplex, double-buffered, parallel port for communication with a host processor. This slave interface supports data transfer between a host and the DSP, as well as commands from a host to the DSP. The 8-bit-wide HI08 data bus supports 8-bit, 16-bit and 24-bit data transfers. Various programmable options provide a glueless connection between the DSP and several industry-standard processors and microcontrollers that are commonly used as system hosts.

Alternatively, the HI08 port pins can provide general-purpose I/O (GPIO), with up to 16 GPIO connections. Both hardware and software reset configure the HI08 port as input GPIO.

2 HI08 Programmer's Model

As **Figure 1** shows, the HI08 peripheral has two register banks:

- Host-side register bank—accessible only to the host
    - DSP-side register bank—accessible only to the DSP core

The separate receive and transmit data paths are double buffered for efficient, high-speed, asynchronous transfers. Note the naming conventions: the host-side transmit data path (host writes) is also the DSP-side receive path, while the host-side receive data path (host reads) is also the DSP-side transmit path.

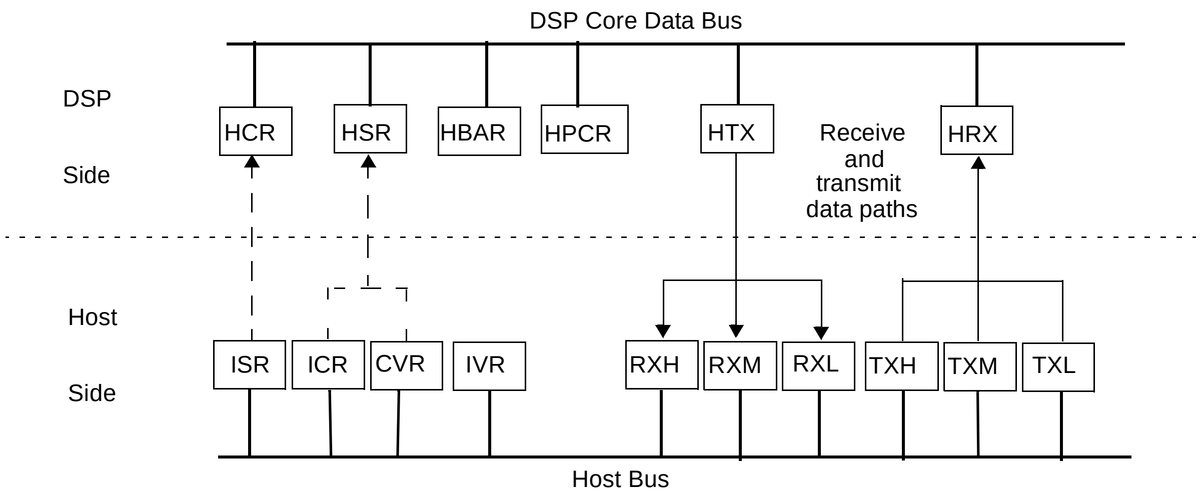

Figure 1. HI08 Programer's Model

2.1 Host-Side Model

To the host, the HI08 appears as eight byte-wide locations mapped in its external address space. The host-side registers can be further partitioned into control registers and data registers, as **Table 1** and **Table 2** show.

Table 1. Host-Side Control Register

Host Address	Control Register
$0	Interface Control register - ICR
$1	Command Vector register - CVR
$2	Interrupt Status register - ISR
$3	Interrupt Vector register - IVR

Table 2. Host-Side Data Register

Host Address	Little Endian Mode	Big Endian Mode
$4	Unused	Unused
$5	Receive/Transmit High - RXH/TXH	Receive/Transmit Low - RXL/TXL
$6	Receive/Transmit Middle - RXM/TXM	Receive/Transmit Middle - RXM/TXM

Table 2. Host-Side Data Register (Continued)

Host Address	Little Endian Mode	Big Endian Mode
$7	Receive/Transmit Low - RXL/TXL	Receive/Transmit High - RXH/TXH

The Receive Data Registers (RXH[M]L) and Transmit Data registers (TXH[M]L) use the same host address. During host writes to these addresses, data is transferred to the Transmit Data registers, while reads are performed from the Receive Data registers.

The Endian mode refers to the method of addressing the host and the way the HI08 transfers data between the host-side data registers and the DSP-side data registers. "Endian Modes" on page 14 has more information on this topic. Regardless of the Endian mode used, when the host writes to the HI08 port, it must write the byte at address $7 last, since this causes the transfer of the 24-bit value from the host-side Transmit Data Registers (TXH[M][L]) to the DSP-side Host Receive Register (HRX). Similarly, when the host reads from the HI08 port, it must read the byte at host address $7 last, since this causes a pending transfer of a 24-bit value from the DSP-side Host Transmit register (HTX) to be transferred to the host-side Receive Data registers (RXH[M][L]).

2.2 DSP-Side Model

To the DSP core, the DSP-side registers appear as six 24-bit registers mapped in internal I/O X memory space; therefore, standard DSP56300 instructions and addressing modes can address these registers. **Table 3** lists control and data registers. Two additional registers are related to the HI08 peripheral when it is used in GPIO mode.

Table 3. DSP-side Data and Control Registers

Type	Register
Control	Host Control Register (HCR) Host Status Register (HSR) Host Port Control Register (HPCR) Host base Address Register (HBAR)
Data	Host Transmit Register (HTX) Host Receive Register (HRX)

3 Transfer Modes

The HI08 operates in one of two host transfer modes, multiplexed or non-multiplexed, depending on the host bus type. Alternatively, the HI08 pins can be used as General Purpose I/O (GPIO). To accommodate these three modes, each HI08 pin serves multiple purposes, as **Table 4**, **Table 5**, and **Table 6** show.

Table 4. Multiplex, Non-multiplex

Pin	Multiplex Mode	Non-multiplex Mode	GPIO Mode
HAD[7–0]	HAD[7–0]	H[7–]0	PB[7–0]
HAS/HA0	HAS	HA0	PB8
HA8/HA1	HA8	HA1	PB9
HA9/HA2	HA9	HA2	PB10
HCS/HA10	HA10	$\overline{\text{HCS}}$/HCS	PB13

The HI08 port can be programmed to use single or dual read/write strobes (signals). In a single-strobe bus, the Host Data Strobe pin (HDS) indicates that valid data is present on the bus, while the Host Read/Write pin (HRW) indicates the type of transaction in process. In dual-strobe mode, the Host Read (HRD) and the Host Write (HWR) lines each indicate data validity and transaction type.

Table 5. Single- and Dual-Strobe Buses

Pin	Single-Strobe Bus	Dual- Strobe Bus	GPIO Mode
HRW/HRD	HRW	$\overline{\text{HRD}}$/HRD	PB11
HDS/HWR	$\overline{\text{HDS}}$/HDS	$\overline{\text{HWR}}$/HWR	PB12

The HI08 can also be programmed to use a single host request line or dual host request lines. Please refer to “Host Requests” on page 12 for information concerning host requests.

Table 6. Single and Dual Host Request Lines

Pin	Single Request	Dual Request	GPIO Mode
HREQ/HTRQ	$\overline{\text{HREQ}}$/HREQ	$\overline{\text{HTRQ}}$/HTRQ	PB14
HACK/HRRQ	$\overline{\text{HACK}}$/HACK	$\overline{\text{HRRQ}}$/HRRQ	PB15

Note that the polarity of the signals can also be programmed to be active high or active low, as required by the host bus. All the options shown here are set by the Host Port Control Register (HPCR).

3.1 Multiplexed Transfer Mode

In multiplexed mode, the HI08 uses the lower eight address lines (address bits [7–0]) to transfer data (data bits [7–0]). **Figure 2** shows a hardware configuration that supports multiplexed transfer mode.

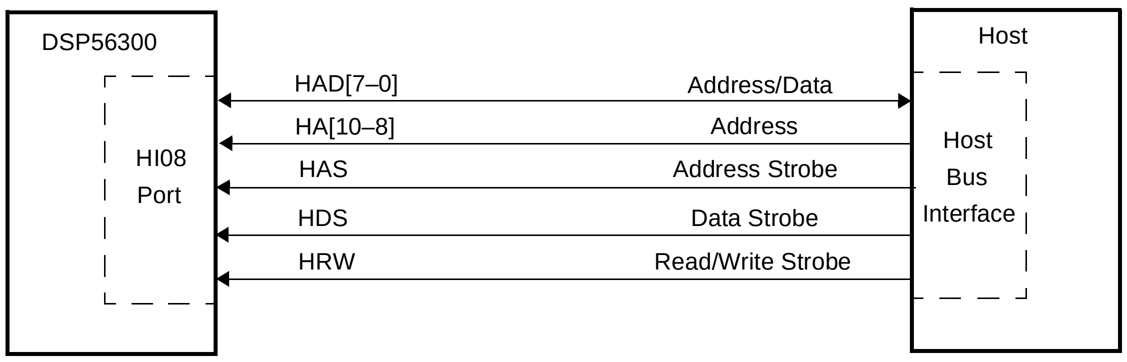

Figure 2. HI08 Hardware Set-Up for Multiplex Mode Transfers

Example 1 lists code that initializes the Host Port Control register (HPCR) to use the HI08 in multiplex mode, as shown in **Figure 2**.

Example 1. Initializing Multiplex Mode

```
INIT_HPCR EQU  $0e0e
               ; [15] = HAP = 0 -> Reserved. Write to 0.
               ; [14] = HRP = 0 -> Reserved. Write to 0.
               ; [13] = HCSP = 0 -> Reserved. Write to 0.
               ; [12] = HDDS = 0 -> Single strobe
               ; [11]  = HMUX = 1 -> Multiplex mode
               ; [10]= HASP = 1 -> HAS active high
               ; [9]  = HDSP = 1 -> HDS active high
               ; [8]  = HROD = 0 -> Reserved. Write to 0.
               ; [7]  = reserved = 0 -> Reserved. Write to 0.
               ; [6]  = HEN = 0 -> Host interface off (GPIO)
               ; [5]  = HAEN = 0 -> Host ack. disabled
               ; [4]  = HREN = 0 -> Host requests disabled
               ; [3]  = HCSEN = 1 -> Enable HA10 in multiplex mode
               ; [2]  = HA9EN = 1 -> Enable HA9 in multiplex mode
               ; [1]  = HA8EN = 1 -> Enable HA8 in multiplex mode
               ; [0]  = HGEN = 0 -> GPIO pins are tri-stated
```

This example code configures the HI08 port to a single-strobe bus, (HPCR[12] = HDDS = 0), where HRW=1 signals a read transaction and HRW=0 signals the write. Active high address (HAS) and data (HDS) strobes are used by setting HPCR[10] = HASP = 1 and HPCR[9] = HDSP = 1, respectively. The example uses all available address lines (HA8EN=1, HA9EN=1, and HCSEN=1).

Note that the HPCR[HEN] bit is initially cleared, which configures the HI08 port in GPIO mode. To assure proper operation, the HPCR bits HAP, HRP, HCSP, HDDS, HMUX, HASP, HDSP, HROD, HAEN, HREN, HCSEN, HA9EN, and HA8EN should not be set when HEN is set or simultaneously with setting HEN. Similarly, bits HAP, HRP, HCSP, HDDS, HMUX, HASP, HDSP, HROD, HAEN and HREN should be changed only if HEN is cleared. The code listed in **Example 2**, which meets these requirements, initializes and enables the HI08 port from the DSP core side

Example 2. Initializing the HI08 Port from the DSP Core Side

```
movep   #INIT_HPCR,x:M_HPCR    ; init HI08 HPCR register
bset    #M_HEN,x:M_HPCR        ; enable_HI08
```

The host should initialize the Interface Control register (ICR) to establish the Endian mode and enable requests, if necessary. (“Endian Modes” on page 14 contains further information.)

In multiplexed mode, two factors determine HI08 port register selection:

- Address on the host bus
- Internal chip select logic, represented in **Figure 3**.

The DSP compares the Host Base Address Register (HBAR) to address bits [10–3] to define the address where the HI08 will appear on the host memory map. The DSP core must initialize HBAR prior to any data transfers. Note that the HI08 port latches only the lower 8 address bits [HAD7–0] so that these signal lines can then be used to transfer data. For proper device selection, the upper address bits (HA[10-8]) must remain asserted during the data transaction.

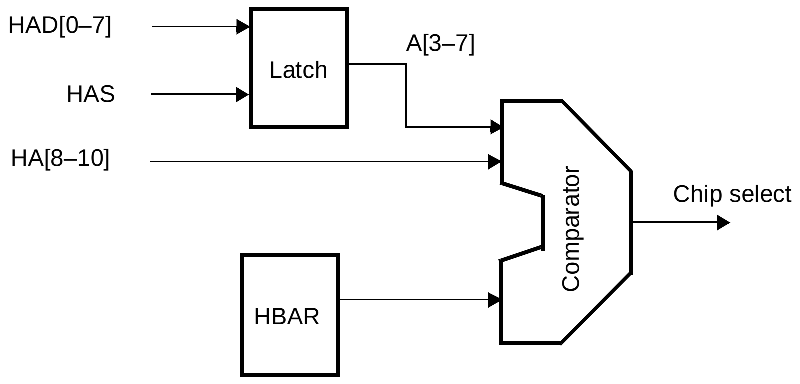

Figure 3. Multiplex Mode Self Chip Select

The DSP performs the following steps when it initializes the HI08 port in multiplex mode:

1. Initializes Host Port Control register (HPCR). Make sure HPCR[HEN] = 0.
2. Initialize Host Base Address register (HBAR).
3. Enable the HI08 by setting HPCR[HEN] = 1.

The host should initialize the Interface Control Register (ICR) to indicate the endian mode to be used.

3.2 Non-Multiplexed Transfer Mode

In Non-Multiplexed mode, the HI08 port appears to the host as 8-bit wide SRAM. Thus, the minimum hardware setup necessary for using the HI08 port in non-multiplexed mode is a chip select, three address lines to access the eight HI08 eight register locations, eight data lines and one or two read/write strobes. **Figure 4** on page 6 shows a simple hardware setup that supports Non-Multiplex Transfer mode.

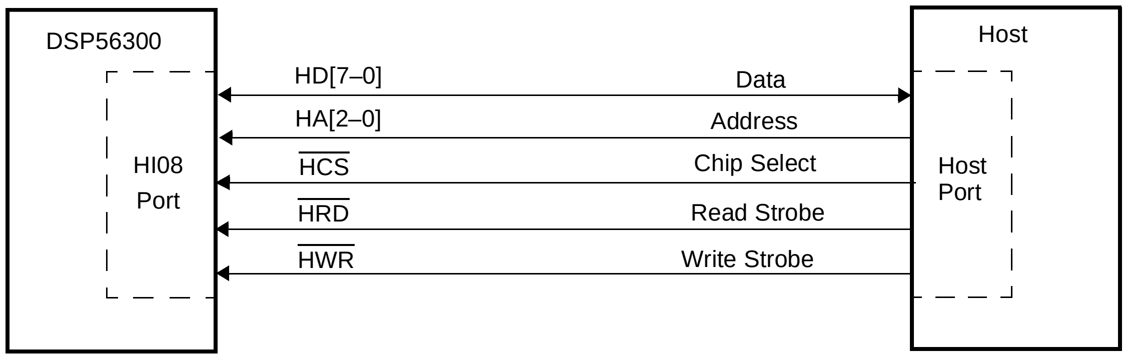

Figure 4. HI08 Hardware Setup for Non-Multiplex Mode Transfers

The equate listed in **Table 3** defines the initial values of the Host Port Control Register (HPCR) that are necessary to set up the HI08 for the Non-Multiplex mode represented in **Figure 4**.

Example 3. Initializing the Non-Multiplex Mode

```
INIT_HPCR EQU   $1008
                ; [15]  = HAP = 0 => Reserved. Write to 0.
                ; [14]  = HRP = 0 -> Reserved. Write to 0.
                ; [13]  = HCSP = 0 -> CS active low
                ; [12]  = HDDS = 1 -> Dual strobe
                ; [11]  = HMUX = 0 -> Non-multiplexed mode
                ; [10] = HASP = 0 -> Reserved. Write to 0 in non-mux
                ; [9]  = HDSP = 0 -> HRD & HWR active low
                ; [8]  = HROD = 0 -> Reserved. Write to 0.
                ; [7]  = reserved = 0
                ; [6]  = HEN = 0 -> Host interface disabled
                ; [5]  = HAEN = 0 -> Host ack. disabled
                ; [4]  = HREN = 0 -> Host request disabled
                ; [3]  = HCSEN = 1 -> CS pin enabled
                ; [2]  = HA9EN = 0 -> Reserved. Write to 0 in non-mux mode
                ; [1]  = HA8EN = 0 -> Reserved. Write to 0 in non-mux mode
                ; [0]  = HGEN = 0 -> Reserved. Write to 0 in non-mux mode
```

This initialization configures the HI08 port bus transactions to use dual, active low, read (HRD) and write (HWR) strobes by setting HPCR[12] = HDDS = 1 and HPCR[9] = HDSP = 0. Active low Chip Select (HCS) is programmed with HPCR[13] = HCSP = 0. Notice that the HPCR[HEN] bit is initially cleared, which configures the HI08 port in GPIO mode. The same code that initializes the Multiplex Transfer mode (**Example 2** on page 5) can also initialize and enable the HI08 port.

The following list summarizes the steps to be followed by the DSP to initialize the HI08 port in non-multiplex mode:

1. Initializes Host Port Control Register (HPCR). Make sure HPCR[HEN] = 0.
2. Enable the HI08 by setting HPCR[HEN] = 1.

The host should initialize the Interface Control register (ICR) to indicate the endian mode to be used.

3.3 General-Purpose I/O (GPIO)

To program the HI08 port to provide GPIO, the Host Enable bit (HEN) in the HPCR is cleared to deactivate the HI08, and the Host GPIO Port Enable bit (HGEN) bit is set to enable the pins configured as GPIO.

The Host Data Direction register (HDDR) controls the direction of the data flow for each of the HI08 pins configured as GPIO. The Host Data register (HDR) holds the data values of pins configured as GPIO. These registers, like the HPCR, are mapped to X memory I/O space and are accessible to the core using conventional move and bit operations. The host cannot access HDDR and HDR.

Not all pins need to be configured as GPIO to use this feature. Even when the HI08 functions as the host interface, the unused signals can be configured as GPIO (if, for example, an application is not using the host request pins). The same ability to configure unused signals for GPIO also applies to HA10, HA9 and HA8 in multiplex mode.

4 Handshaking Protocols

The HI08 is a slave-only device, which means that the host is the master of all bus transfers. In host-to-DSP transfers, the host writes data to the Transmit Byte Registers (TXH[M][L]). In DSP-to-host transfers, the host reads data from the Receive Byte Registers (RXH[M][L]). However, the DSP side has access only to the Host Receive Data Register (HRX) and the Host Transmit Data Register (HTX). When available, data automatically moves between the host-side data registers and the DSP-side data registers. This double-buffered mechanism allows for fast data transfers, but creates a “pipeline” that can stall communication (if the pipeline is either full or empty) or cause erroneous data transfers (overwriting new data or reading old data twice). The HI08 port has several handshaking mechanisms to counter any of these potential buffering problems.

For example, a host writing several pieces of data to the HI08 port should first determine whether any data previously written to the Transmit Byte Registers (TXH[M][L]) has successfully transferred to the DSP side. A handshaking protocol makes this possible. If the host-side Transmit Byte Registers (TXH[M][L]) are empty, the host writes the data to these registers. The transfer to the DSP-side Host Receive Data Register (HRX) occurs only if HRX is empty (that is, the DSP has read it). The DSP core then uses an appropriate handshaking protocol to move data from the HRX to the receiving buffer or register. If the handshaking protocol were not used, the host might overwrite data not yet transferred to the DSP side, or the DSP might receive stale data.

A similar situation occurs when the host performs multiple reads from the HI08 port Receive Byte registers (RXH[M][L]). The DSP side uses an appropriate handshaking protocol to determine whether any data previously written to the Host Transmit Register (HTX) has been successfully transferred to the host-side registers. If HTX is empty, the DSP writes the data to this register. Data is transferred to the host-side Receive Byte Registers (RXH[M][L]) only if they are empty (that is, that host has read them). The host can then use any of the handshaking protocols available to determine if more data is available to be read.

The DSP56300 family HI08 port offers the following handshaking protocols for data transfers with the host:

- Software polling
- Interrupts
- Core DMA access
- Host requests

As the following sections discuss, several factors determine which protocol to use, including:

- The amount of data to be transferred
- The timing requirements for the transfer
- The availability of resources such as processing bandwidth and DMA channels

Recall that the transfers described here occur between the host and the DSP asynchronously. Each transfers data at its own pace. However, using an appropriate handshaking protocol allows data to be transferred at optimum rates.

4.1 Software Polling

Software polling is the simplest handshaking protocol, but it demands the most core processing power. In software polling, the host or the DSP core uses status bits to test the data registers and determine whether they are empty or full. However, while polling these status bits, the DSP core cannot perform other processing.

On the DSP-side, two situations are possible. In transfers from the DSP to the host (host reads), the DSP core must determine if the HTX is empty (and thus available for more data) or full (waiting for the host to read the data from its side). In this case, the DSP core polls the Host Transmit Data Empty bit in the Host Status register (HSR[1][HTDE]). If HSR[1][HTDE]=0, data has not been transferred from HTX to the host-side RXH[M][L] data registers. If HSR[1][HTDE]=1, the HTX is empty and the DSP core can write more data to it. The DSP56300 assembly code in **Example 4** implements the polling and data transfer from a transmit data buffer in Y memory (to which TBUFF_PTR points) to the HTX register:

Example 4. Implementing Polling and Data Transfer to HTX

```
jclr    #1,x:M_HSR,*                ; loop if HSR[1]:HTDE=0
move    y:(TBUFF_PTR)+,x:M_HTX      ; move data to HTX
```

Similarly, in transfers from the host to the DSP (host writes), the DSP side should determine if the HRX is full (and thus needs to be moved to a memory buffer or a register) or empty (no data available from the host). The DSP core polls the Host Receive Data Full bit in the Host Status Register (HSR[0][HRDF]). If HSR[0][HRDF]=0, data has not been transferred from the host-side TXH[M][L] data registers to the DSP-side HRX. If HSR[0][HRDF]=1, the HTX is full and the DSP core can read data from it. The following DSP56300 assembly code in **Example 5** implements the polling and data transfer from the HRX register to a receive data buffer in Y memory (to which RBUFF_PTR points):

Example 5. Implementing Polling and Data Transfer from HTX

```
jclr    #0,x:M_HSR,*                  ; loop if HSR[0]:HRDF=0
movep   x:M_HRX,y:(RBUFF_PTR)+        ; move data to buffer
```

A similar mechanism is available on the host side. When data is transferred to the DSP (host writes), the host can poll the Transmit Data Empty bit in the Interface Status Register (ISR[1][TXDE]). If ISR[1]=TXDE=1, the host can write data to the TXL:M:H data registers. Otherwise, it must wait until the data currently in these registers is transferred to the DSP-side data registers. Finally, in transfers from the DSP to the host (host reads) the host can poll the Receive Data Full bit in the Interface Status register (ISR[0][RXDF]). When set, ISR[0][RXDF] indicates that data is available in the RXL[M][H] data registers.

4.1.1 Host Flags

The HI08 control registers have four general-purpose flags for communication between the host and the DSP:

- DSP side: The HSR Host Flag bits (HCR[4–3]=HF[3–2]) can pass application-specific information to the host. (The host-side ISR Host Flag bits (ISR[4–3]=HF[3–2]) reflect the status of HF[3–2].)
- Host side: The ICR Host Flag bits (ICR[4–3]=HF[1–0]) can pass application-specific information to the DSP. (The DSP-side HSR Host Flag bits (HSR[4–3]=HF[1–0]) reflect the status of HF[1–0].)

4.2 Interrupts

Software interrupts allow the core to perform other processing tasks while waiting for HI08 resources to become available. An enabled interrupt automatically occurs when these resources become available for the data transfer. The interrupt routine then transfers the data between the host and the DSP.

Table 7 lists the three HI08 interrupt sources (and their masking bits) in the Host Control Register (HCR):

Table 7. HI08 Core Interrupt Sources and Masking Bits

Interrupt Source	Masking Bit
Receive Data Full interrupts	Host Receive Interrupt Enable bit (HCR[0]=HRIE)
Transmit Data Empty interrupts	Host Transmit Interrupt Enable bit (HCR[1]=HTIE)
Host Command interrupts	Host Command Interrupt Enable bit (HCR[2]=HCIE)

Figure 5 depicts how these bits operate. The DSP uses Receive Data Full and Transmit Data Empty interrupts to move data to or from the HTX and HRX data registers. The host uses Host Command interrupts to force execution of a DSP interrupt routine (as **"Host Commands" on page 13** explains).

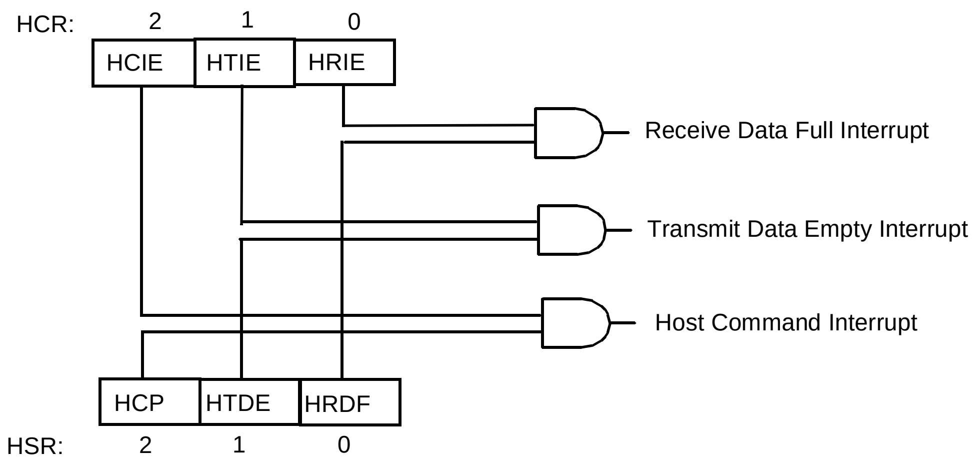

Figure 5. HI08 DSP Core Interrupt Operation

The BSET and BCLR instructions easily enable and disable the HI08 interrupts, as **Example 6** shows:

Example 6. Enabling the Host Receive Interrupt

```
bset    #M_HRIE,x:M_HCR; enable host receive interrupt
```

The DSP56300 family allows two types of interrupts, short and long. Short interrupts are two words long and fit at the corresponding exception vector location, typically between $02 and $100 in program memory. The code in **Example 7** is a short interrupt service routine to receive data from the HRX data register and place it in a buffer in Y memory pointed to by r0. Note that this code, like most short interrupt routines, assumes that the required resources (data register x1 and pointer r0 in this example) are not used anywhere else in the system.

Example 7. Moving Data from Register to Buffer (Interrupt Routine)

```
org     P:$60
movep   x:M_HRX,y:(r0)+        ; HI08 Receive Data Full interrupt
```

Similarly, a short Transmit Data Empty interrupt service routine could look like the code shown in **Example 8**. This short interrupt moves data from a transmit buffer pointed to by r5 to the HI08 HTX register.

Example 8. Moving Data from Buffer to Register (Interrupt Routine)

```
org     P:$62
move    y:(r5)+,x:M_HTX; HI08 Transmit Data Empty interrupt
```

Long interrupts are exceptions. They require more than two instructions to service and are reached by a *jump to subroutine* instruction (*jsr*) at the exception vector location where a short interrupt would be located. A long interrupt routine can perform more sophisticated functions, such as saving the state of ALU and AGU registers and disabling other interrupts, if necessary. The exception routine may also need to determine whether the current data is the last data to be transferred, since this is a good place to decide when to disable the interrupt.

4.3 Core DMA Access

The DSP56300 family Direct Memory Access (DMA) controller permits transfers between internal or external memory and I/O without any intervention by the core. A DMA channel can be set up to transfer data to or from the HTX and HRX data registers. This frees the core to use its processing power on functions other than polling or interrupt routines for the HI08. This may well be the best method to use for data transfers, but requires that one of the six DMA channels be available. Two HI08 DMA sources are possible, as **Table 8** shows

Table 8. DMA Request Sources

Requesting Device	DCRx[15–11]=DRS[4–0]
Host Receive Data Full (HRDF=1)	10011
Host Transmit Data Empty (HTDE=1)	10100

For example, in a DSP-to-host transfer (host reads) using DMA channel 1, the data resides in a transmit buffer starting at address TBUFF_START and containing TBUFF_SIZE data. **Example 9** on page 11 shows these parameters and the initial values of the channel 1 DMA Control Register (DCR1).

Example 9. DCR1: Parameters and Initial Values

```
TBUFF_START    EQU    $001000        ; Transmit buffer start address
TBUFF_SIZE     EQU    $20            ; Transmit buffer contains 32 datums
INIT_DCR1      EQU    $88a251        ;Initialization of DCR1
        ; [23] = DE = 1 -> DMA enabled
        ; [22] = DIE = 0 -> DMA interrupts disabled
        ; [21:19] = DTM2:DTM0 = 001 = by request, DE=0 when done, words
        ; [18:17] = DPR1:DPR0 = 00 -> DMA0 priority level 0
        ; [16] = DCON = 0 -> continuous mode off
        ; [15:11] = DRS4:DRS0 = 10100 -> request on HTDE=1
        ; [10] = D3D = 0 -> non-3D mode
        ; [9:7] = DAM5:DAM3 = 100 -> destination is linear access, no update
        ; [6:4] = DAM2:DAM0 = 101 -> source is linear access, post +1
        ; [3:2] = DDS1:DDS0 = 00 -> X memory DMA0 destination
        ; [1:0] = DSS1:DSS0 = 01 -> Y memory DMA0 source
```

The code in **Example 10** makes it easy to program DMA channel 1. Notice that the initial values of DCR1 set up the DMA request to occur on a Host Transmit Data Empty condition. The DMA transfers TBUFF_SIZE data from TBUFF_PTR (post-incremented by 1 by the DMA controller) to the HI08 HTX data register.

Example 10. Programming DCR1

```
        bclr    #M_D1L0,x:M_IPRC       ; disable DMA1 interrupts
        bclr    #M_D1L1,x:M_IPRC
        movep   #TBUFF_START,x:M_DSR1  ; DMA1 source is transmit buffer
        movep   #M_HTX,x:M_DDR1        ; DMA1 destination is HTX
        movep   #TBUFF_SIZE-1,x:M_DCO1 ; DMA1 count is the full buffer
        movep   #INIT_DCR1,x:M_DCR1    ; init. DMA1 control register
```

DMA transfers do not access the host bus. Using an appropriate polling mechanism, the host must determine when data is available in the host-side receive data registers.

Refer to the *DSP56300 Family Manual* to learn about DMA access in the Freescale 56300 family.

4.4 Host Requests

The host requests protocol provides a set of signal lines that allow the DSP side to request data transfers from the host. The request signal lines from the DSP normally connect to the host's interrupt request pins (IRQx) and indicate to the host when the DSP HI08 port requires service. The DSP side can be configured to use either a single request line for both receive and transmit requests, called Host Request (HREQ), or two signal lines, a Host Transmit Request (HTRQ) and a Host Receive Request (HRRQ), one for each type of transfer.

Host requests must be enabled on both the DSP side and the host side:

- DSP side: Set the HPCR Host Request Enable bit (HPCR[4]=HREN).
- Host side:
  - — To configure the HI08 to use a single request line (HREQ), clear the ICR Double Host Request bit (ICR[2]=HDRQ).
  - — To configure the HI08 to use both transmit and request lines, set the ICR[2]=HDRQ bit.

Further, to enable receive and transmit requests, the ICR Receive Request Enable bit (ICR[0]=RREQ) and the ICR Transmit Request Enable bit (ICR[1]=TREQ) must be set on the host side.

4.4.1 Host Request Operation

With host requests enabled, the host request pins operate as **Figure 6** shows. The host can then test these ICR bits to determine the interrupt's source.

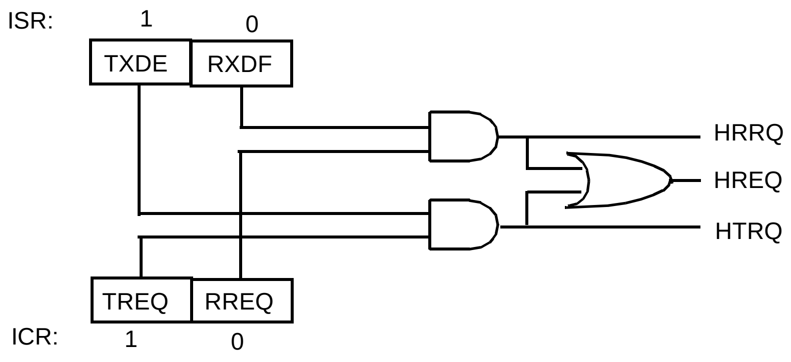

Figure 6. HI08 Host Request Operation

Table 9 shows how the HREQ pin operates with a single request line.

Table 9. HREQ Pin Operation In Single Request Mode (ICR[2] = HDRQ = 0)

ICR[1]=TREQ	ICR[0]=RREQ	HREQ Pin
0	0	No interrupts
0	1	RXDF request enabled
1	0	TXDE Request enabled
1	1	RXDF and TXDE request enabled

Table 10 shows how the transmit request (HTRQ) and receive request (HRRQ) lines operate with dual host requests enabled.

Table 10. HTRQ and HRRQ Pin Operation In Double Request Mode (ICR[2]=HDRQ=1)

ICR[1]=TREQ	ICR[0]=RREQ	HTRQ Pin	HRRQ Pin
0	0	No interrupts	No interrupts
0	1	No interrupts	RXDF request enabled
1	0	TXDE Request enabled	No interrupts
1	1	TXDE Request enabled	RXDF request enabled

To clear the interrupt request, the host must read or write the appropriate HI08 host-side data register. Two alternative ways to deassert the host request pins are to mask (clear) the host request enable bit or to reset the DSP.

4.4.2 The Interrupt Vector Register

The IVR is an 8-bit read/write register that typically contains the interrupt vector number used with the MC680xx host processor to service the DSP. In this case, if the host request signal is asserted to notify the Freescale MC68000 family host of an HI08 request, the host processor asserts the Host Acknowledge signal (HACK) as part of its interrupt acknowledge signal. When HREQ and HACK are simultaneously asserted, the contents of the IVR are placed on the host data bus. The host then uses this information as the vector for the interrupt service routine.

5 Host Commands

This innovative feature of the HI08 host interface allows the host to issue any of 128 DSP interrupt routines as command requests for the DSP to execute. For example, the host may issue a host command that sets up and enables a DMA transfer. The DSP56300 family processors with an HI08 port have reserved interrupt vector addresses for application-specific host command interrupts (in the DSP56303 these correspond to addresses $64 through $FF). Note that this flexibility is independent of the data transfer mechanisms in the HI08. It enables the host processor to read or write DSP registers or memory locations, perform control status or debugging operations and start DMA transfers, among other functions.

Note that when the DSP enters the Stop mode, it electrically disconnects the HI08 pins, and, thus, disables the HI08 until the core leaves Stop mode. While the HI08 configuration remains unchanged in Stop mode, the DSP core cannot be restarted via the HI08 interface. DO NOT issue a STOP via a host command unless you provide some other mechanism for exiting this mode.

Setting the HCR[2]=HCIE bit on the DSP side enables host command interrupts (refer to “Interrupts” on page 9). The host then issues host commands via the Command Vector register (CVR). Setting the Host Command (CVR[7]=HC=1) requests the command interrupt and writing the seven Host Vector bits (CVR[6–0]=HV6–HV0) selects the vector address. When DSP core recognizes the host command interrupt, the address of the interrupt taken is 2 x HV. This allows the host to force execution of any interrupt handler (e.g. SSI, SCI, IRQx, etc.). The host can write the HC and HV bits simultaneously.

6 Endian Modes

The Host Little Endian bit in the host-side Interface Control Register (ICR[5]=HLEND) allows the host to access the HI08 data registers in Big Endian or Little Endian mode. In Little Endian mode (HLEND=1), a host transfer occurs as **Figure 7** shows.

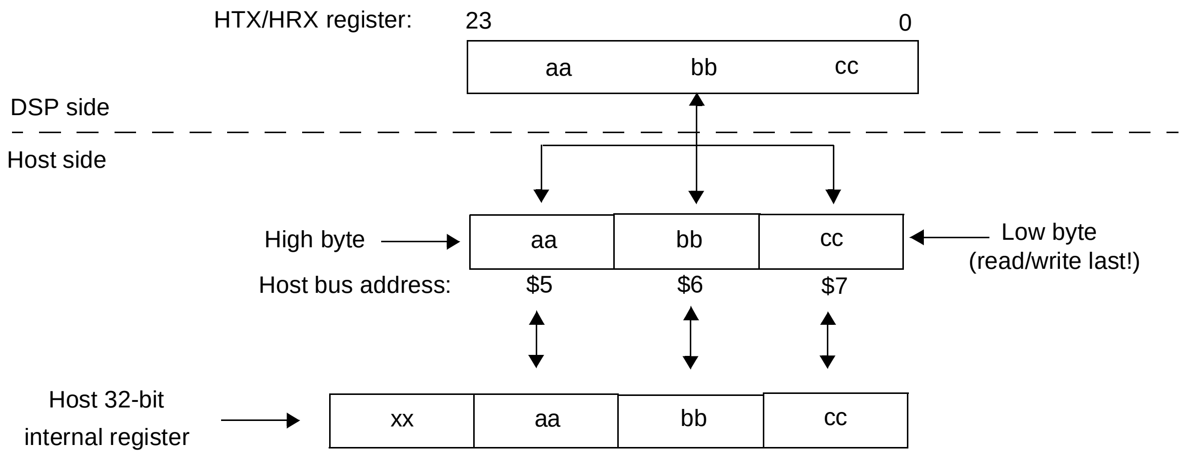

Figure 7. HI08 Read and Write Operations in Little-Endian Mode

In some cases, a host may transfer data one byte at a time. Thus, to transfer 24 bits of data requires three store (or load) byte operations. Also, to cause the transfer of data to the DSP-side HRX, the data byte at host bus address $7 must be written last. In other cases, the host bus controller may be sophisticated enough to allow the host to transfer all bytes in a single operation (instruction). For example, in the Power PC MPC860 processor, the General Purpose Controller Module (GPCM) in the memory controller can be programmed such that the host can execute a single read (load word, *ldw*) or write (store word, *stw*) instruction to HI08 port and cause four byte transfers to occur on the host bus. For example, the 32-bit data transfer operation shown in **Figure 7** writes byte data xx to HI08 address $4, byte aa to address $5, byte bb to address $6 and byte cc to address $7 (assuming that the 24 bits of data are contained in the lower 24 bits of the host's 32-bit data register, as shown).

A similar operation occurs when the HI08 is initialized in Big Endian mode by clearing the Host Little Endian bit (ICR[5]=HLEND), as in **Figure 8** on page 14 depicts. Note that the HI08 hardware properly transfers the data between the DSP-side and the host-side data registers.

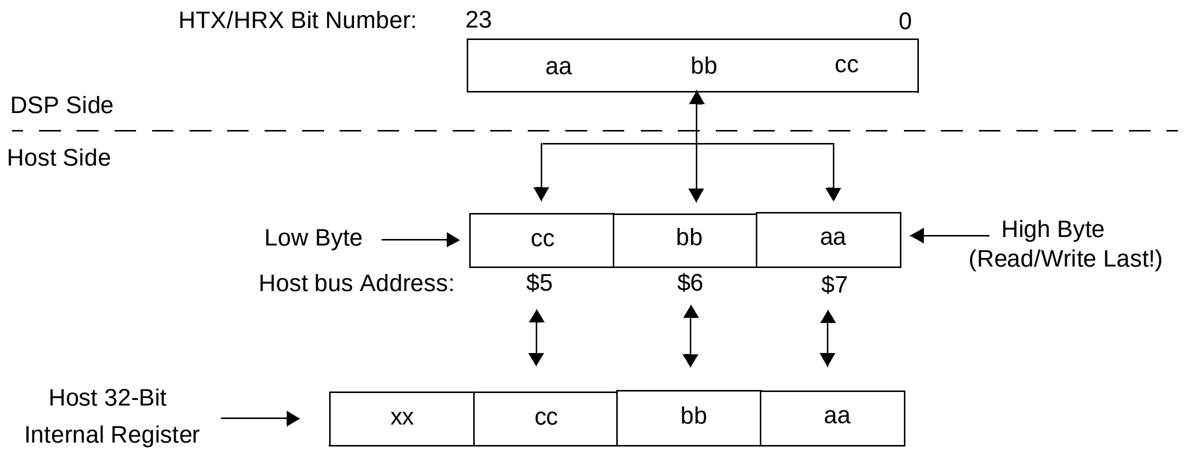

Figure 8. HI08 Read and Write Operations in Big-Endian Mode

7 Boot-up Using the HI08 Host Port

The DSP56300 core has eight bootstrap operating modes to start up after reset. As the processor exits the Reset state, it loads the values at the external mode pins MODA/IRQA, MODB/IRQB, MODC/IRQC and MODD/IRQD into the Chip Operating Mode bits (MA, MB, MC and MD) of the Operating Mode register (OMR). These bits determine the bootstrap operating mode. Modes C, D, E and F use the HI08 host port to bootstrap the application code to the DSP. **Table 11** describes these modes for the DSP56307.

Table 11. DSP56307 Bootstrap Operating Modes

Mode	MODD	MODC	MODB	MODA	HI08 Bootstrap Description
C	1	1	0	0	ISA/DSP5630x mode
D	1	1	0	1	HC11 non-multiplexed bus mode
E	1	1	1	0	8051 multiplexed bus mode
F	1	1	1	1	MC68302 bus mode

The bootstrap program is factory-programmed into an internal 192-word by 24-bit bootstrap ROM at locations \$FF0000 to \$FF00BF of P memory. This program can load program RAM segments from the HI08 host port.

When using any of the modes shown in **Table 11**, the core begins to execute the bootstrap program and configure the HI08 based on the OMR Mode bits. The bootstrap program then expects the following data sequence when downloading the user program from the HI08:

1. Three bytes (least significant byte first) indicating the number of 24-bit program words to be loaded
2. Three bytes (least significant byte first) indicating the 24-bit starting address in P memory to load the user’s program
3. The user’s program (three bytes, least significant byte first, for each program word)

When the bootstrap program completes loading the specified number of words, it jumps to the specified starting address and executes the loaded program.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations not listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GMBH
Technical Information Center
Schatzbogen 7
81829 München, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T. Hong Kong
+800 2666 8080

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Order No.: AN1808
Rev. 1
8/2005