Freescale Semiconductor

Order by
AN1804/D
Rev. 0 , 7/99

DSP56307 Port A-to-HI08 Interface

James Gilbert

Multiple digital signal processors (DSPs) are commonplace in applications such as DSP resource boards and voice transcoder cards. Depending upon the task at hand, several configurations and architectures for this multi-device DSP resource are possible.

Typically, a central microcontroller unit schedules and assigns the DSP workload across multiple devices. However, the DSPs can also perform scheduling. In these cases, the DSPs are organized in a tier, with a group of master DSPs each controlling a subset of slave DSPs. Such implementations require a suitable bus interface between the DSP master and the DSP slave. Devices in the Motorola DSP56300 family meet this requirement in different ways. In DSP56303, DSP56307 and DSP56309, the Port A bus interface on the master DSP can control the HI08 host interface on the slave DSP side.

This document describes the interface between the master DSP56307 Port A expansion port and the slave HI08 host interface of another DSP56307. It details the most straightforward connection and timing of the two interfaces, helping developers get started with their own multi-DSP system designs. Although the implementation applies specifically to the DSP56307, the method is equally applicable to any DSP56300 family member with an HI08 interface.

Contents

1 Hardware Implementation

The system structure used is illustrated in **Figure 1**. Port A on one DSP56307 (DSP1) is essentially a bus master that can control a slave HI08 host interface of the target DSP56307 (DSP2). The signal interconnect is seamless, with no need for any additional glue logic. However, as both the DSP56307 Port A bus and the target HI08 host interface are programmable, it is important to configure each appropriately.

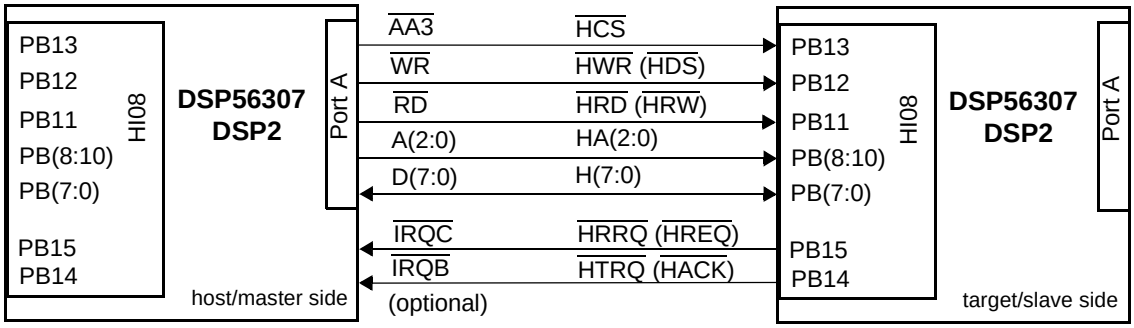

Figure 1. DSP56307 Port A-to-HI08 Host Port Interface

1.1 Port A Interface

The Port A interface is a 24-bit bus interface used for memory and system expansion of the DSP56307. It has several operating modes that allow it to connect to a wide variety of peripherals, both high speed and low speed. For example, it can interface to high-speed static RAMs, Flash and DRAM, as well as to slower support peripherals. The following implementation discussion focuses on the Port A asynchronous SRAM mode.

1.2 HI08 Host Port

The HI08 host port is a byte-wide parallel slave port with various data strobe and multiplexing options. Also, the host port is asynchronous in nature, an attractive feature that simplifies timing constraints and eliminates clock skew with the host bus master. In simple terms, the HI08 host port appears to the master Port A interface as a memory-mapped resource.

All the HI08 host port registers can be accessed with a single Port A Address Attribute line ($\overline{AA3}$), acting as a chip select, and the three least significant address lines. The double-strobe, non-multiplexed mode of the HI08 interface interacts well with these Port A interface signals. On each access to the DSP2 host port registers, the Port A chip select asserts together with the appropriate address and read ($\overline{RD}$) or write ($\overline{WR}$) strobe. The data value is latched in either direction on the negating edge of the Port A controlled strobe ($\overline{RD}$ or $\overline{WR}$). The internal wait state setting in the Port A Bus Control Register (BCR) determines the length and termination of the cycle, so no external termination handshake is required.

1.3 Bus Transfers

To initiate a transfer, the DSP1 Port A master device can simply complete a MOVE instruction to read from or write to the DSP2 target device. However, using only the DSP core instruction sequence to control the bus operations has two repercussions:

- The instruction sequence stalls until the Port A-to-HI08 bus access completes, so valuable DSP1 core instruction execution time is tied up unnecessarily.
- Before further DSP1 Port A bus transactions can begin, DSP1 should poll the appropriate HI08 status flags to confirm that previous read/write bus transfers have been processed.

A more practical approach is to permit the HI08 slave device to request bus cycles only when it is ready to receive them. To support this approach, the HI08 has two possible host request modes:

- Single host request for combined transmit/receive requests ($\overline{\text{HREQ}}$)
- Dual host requests for separate transmit and receive host requests ($\overline{\text{HTRQ}}$ and $\overline{\text{HRRQ}}$)

This implementation uses separate requests from DSP2-to-DSP1 to allow clean separation of the transmit and receive streams. On each DSP2 host request assertion, DSP2 issues an interrupt or DMA service request to DSP1. The Port A interface responds by starting another 24-bit operation for the next 3 byte-wide reads or writes over the bus. If DMA mode is used at both sides of the interface, then the bus transactions can be maintained completely in parallel with the DSP core processing. As such, the DMA host request mechanism ensures best use of the Port A bus bandwidth and minimizes overhead on the DSP core.

In an end system application, the Port A-to-HI08 slave interface can first be used to bootstrap a slave DSP (DSP2) under control of the master DSP (DSP1) and subsequently used to provide an ongoing control/data link. The driver software required for bootstrapping over HI08 is beyond the scope of this document; however, the most appropriate HI08 bootstrap mode would be the HI08 ISA bus mechanism with double strobes enabled.

2 Hardware Timing Details

Careful programming of the DSP56307 on-chip registers can guarantee the timing parameters for both sides of the Port A-to-HI08 host port interface, as **Figure 2** shows. All timings calculations are based on a 4-wait state, 100 MHz DSP56307 Port A interfacing to a 100 MHz DSP56307 HI08 interface, using timing parameters from the DSP56307 technical data sheet. The resulting bus read and write accesses are each seven clocks in duration.

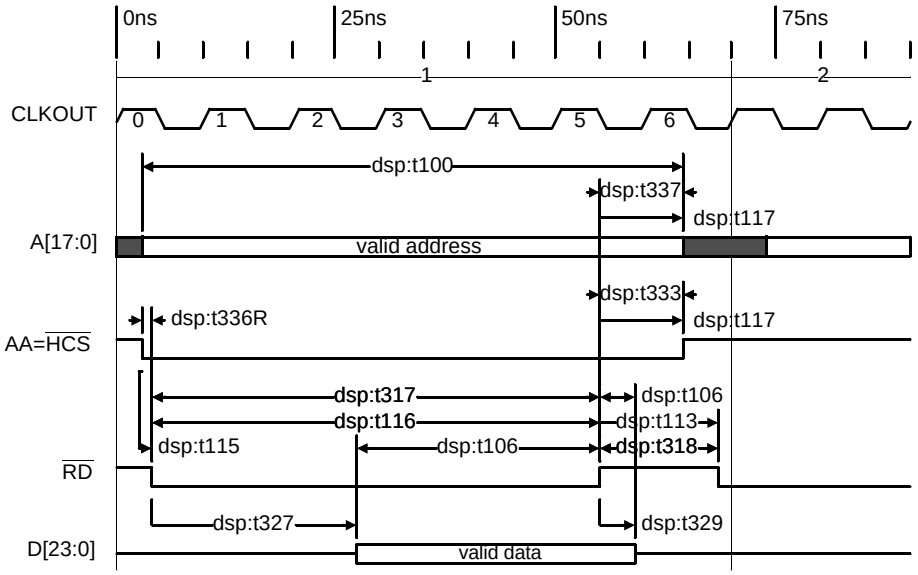

Diagram: 100 MHz DSP56307 Port-A-to HIO8 Read (7 clk).
Assume: 50pF load, BCR=4 wait states
Notes: Assumes short RD strobe negation; i.e., Beware back-back conditions.

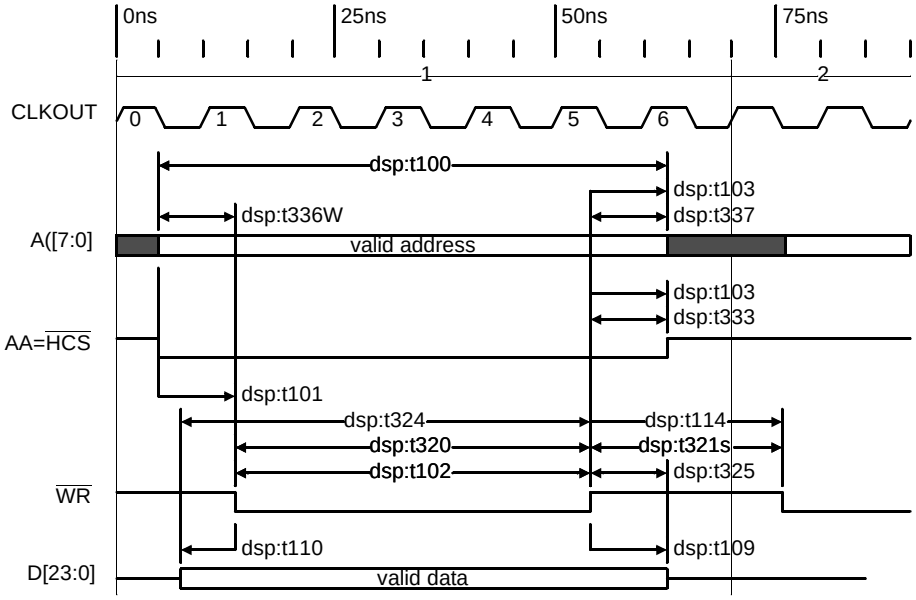

Diagram: 100 MHz DSP56307 Port A-to-HIO8 Write (7 clocks).
Assume: 50pF load, BCR=4 wait states
Notes: Assumes short WR strobe negation. Beware back-back cycle condition

Figure 2. DSP56307 Port A-to-DSP56307 HI08 Host Port Timing

2.1 Strobe Negation Times

Host read and write strobe negation times receive special attention. The primary reason for using at least four wait state accesses is to allow the Port A read strobe negation time (time t113 = 13.5ns at 100 MHz) to meet the HI08 read strobe negation time requirements (time t318 = 9.9ns at 100 MHz), and the Port A write strobe negation (time t114 = 21.5ns at 100 MHz) to meet the HI08 write strobe negation time requirements (time t321 = 16.5ns at 100 MHz). Note that these timings are based on the assumption that there are no back-to-back cycles to the same read or write register. Specifically, the DSP1 Port A firmware should avoid consecutive back-to-back reads of the HI08 Interface Status Registers (ISR), back-to-back reads/writes of the Command Vector Register (CVR) and Interface Control Register (ICR) or back-to-back reads/writes of the last HI08 data register (HI08 address $7).

2.2 Back-to-Back MOVE Accesses

If back-to-back MOVE accesses must be made to any of the aforementioned registers, the read and write strobe negation time between accesses must meet time t318=33.6ns and time t321=33.6ns at 100 MHz. The main cause for consecutive back-to-back MOVE accesses to the same registers over Port A is back-to-back MOVE instructions in the instruction stream.

For example, back-to-back read MOVE instructions to the same HI08 register might be used in a software status polling implementation during HI08 data transfer or during the actual transfer of HI08-to-Port A read data itself. In back-to-back MOVE instructions over Port A, the write strobe is negated for 3.5 clocks (35ns at 100 MHz) and so readily meets minimum time t321. Under worst case theoretical conditions, though, the read strobe does not meet time t318, because it is negated for only 2.5 clocks (25ns at 100 MHz).

In practice, this theoretical limitation should not cause a restriction. When a transfer status bit (e.g., ISR[HTDE] or ISR[HRDF]) is repeatedly polled over the Port A-to-HI08 link, a MOVE must always be followed by a decision instruction (e.g., JCLR or BCC). With this instruction combination, the read strobe negation timings are always met. For back-to-back transfers of HI08 data, a more practical alternative is to use the HI08 host request facility.

2.3 HI08 Transmit/Receive Request

The HI08 host transmit/receive request ($\overline{\text{HTRQ}}$ and $\overline{\text{HRRQ}}$) can be used to indicate a new data read or data write service request to the host (DSP1 in our case). The host responds to the request by generating an interrupt or DMA service to initiate a further Port A-to-HI08 data access on the bus. The HI08 request and DMA/Interrupt service mechanism ensures that successive accesses meet all the HI08 strobe negation timing requirements (t318, t319, t321) for data accesses. Therefore, using this facility places no restrictions on the Port A interface control firmware. The added benefit of using the requests for DMA transactions is that the bus accesses can occur in parallel with the DSP core activity, which sustains overall system performance.

2.4 Summary (Hardware Timing)

If data transfers do not use the HI08 host request mechanism, the DSP1 Port A host should not attempt consecutive data read or write MOVE instructions unless the DSP2 HI08 receive data full or transmit data empty flag is set. Furthermore, both the DMA and the interrupt-based host request methods avoid all Port A-to-HI08 timing restrictions for data transactions, so these are the preferred interface

solutions. Both the DMA and interrupt methods can employ the same hardware implementation (shown in **Figure 1**), in which the DSP2 Host Transmit Request ($\overline{\text{HTRQ}}$) and Host Receive Request ($\overline{\text{HRRQ}}$) are provided to DSP1 interrupt pins. Only the software implementation on DSP1 must change to accommodate what the interrupt pin generates: an interrupt or a DMA request. Example software drivers are listed in **Appendix B** for the DMA-based mode and in **Appendix C** for the interrupt mode. Each appendix includes separate driver code files for the DSP1 Port A side and DSP2 HI08 side.

3 Software Implementation

The interface software moves a block of data in both directions over the 8-bit bus. Separate driver code files are downloaded for the DSP1 Port A side and the DSP2 HI08 side. Each side transmits a data block, starting from a base address of x:$1400, and receives data to a block starting at address x:$1500. Each 24-bit data word is sent using three consecutive byte transfers to fill the 3-byte transmit and receive FIFOs of the HI08 interface.

Note: The x:$1400 and x:$1500 locations point to RAM-only blocks on the DSP56307 and other DSP56300 devices. However, to ensure that this RAM area is available for future devices, refer to the memory map of the specific device.

Two distinct DSP1 Port A master to DSP2 HI08 host port implementations were used to verify the operation:

- DMA-based request servicing (see **Appendix B**).
- Interrupt-based request servicing (see **Appendix C**).

The only distinction between the modes is whether the interrupt source is enabled to generate DMA requests or interrupt handler requests. In either case, the control flow is identical, as **Figure 3** shows.

The appropriate object code files are downloaded into the internal RAM of DSP1 and DSP2. The DSP2 HI08 side starts first, to complete the initialization of the HI08 interface. The DSP2 then waits until it receives a Host Flag 0 (HF0) set indicator from the host side before proceeding. This gives a synchronization point typical of a real system, in which all slave DSP HI08 interfaces are kept pending until enabled by the master host (DSP1). Setting the HF0 host flag enables the DSP1 and DSP2 request methods, and the data transfers continue in either direction until the full data block transfer completes.

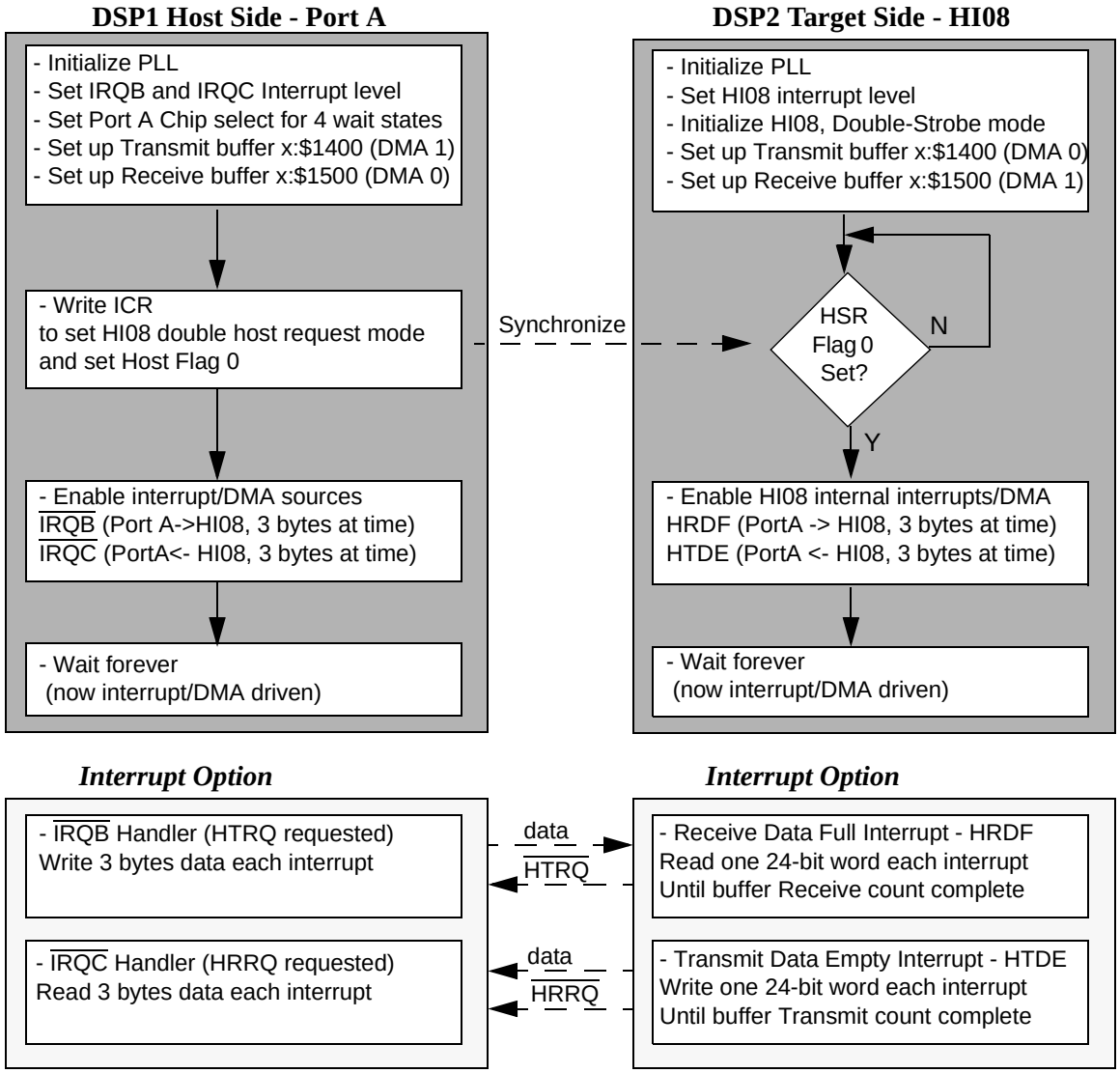

Figure 3. Program Flow

The DMA implementation uses Port A byte-packing to automatically send the 24-bit words as three successive bytes. Because the least significant byte is sent first to store and retrieve data appropriately in memory, Little Endian mode must be selected for the HI08 interface. To maintain the same structure, the interrupt version of the Port A side driver is also transferred in Little Endian byte order. However, in general, either byte ordering mode can be used with the interrupt-based port A method. The only requirement is that the byte order transmitted to HI08 addresses 5, 6, 7 is Little Endian or Big Endian, and that the HI08 port ICR register selection corresponds.

Tables 1 through 6 of **Appendix A** list the register settings recommended and used for the DMA mode example. Listings B.1 and B.2 of **Appendix B** contain the code used on the DSP1 bus master (Port A) and DSP2 bus slave (HI08) devices for the DMA mode; Listings C.1 and C.2 of **Appendix C** provide

the equivalent code for the interrupt mode. Although the DMA mode DSP1 Port A driver is intended to be used with the DMA mode DSP2 HI08 driver, each DSP1 Port A driver provided in the listings interplays with any of the DSP2 HI08 side drivers. They are fully interchangeable.

4 Summary

In multi-DSP56307 device configurations, the Port A interface of one DSP can act as a bus controller for HI08 slave interfaces on other DSPs. With suitable 4-wait state programming, the interface is completely glueless. Either an interrupt or DMA implementation is recommended on the Port A side to address all bus interface timing considerations while incurring the least DSP core overhead to maintain data transfers.

A Register Settings

The following tables indicate the register settings used on the DSP port A side and DSP HI08 slave side when both are operating in DMA mode.

Table A-1. DSP56307 Port A Register Settings (DSP1)

Register	Bits	Setting	Meaning
Address Attribute Register 3 =$100CB9	23-12 11-8 7 6 5 4 3 2 1 0	BAC(11:0) =0001 0000 0000 BNC(3:0) = 1100 BPAC = 1 BAM = 0 BYEN = 1 BXEN = 1 BPEN = 1 BAAP = 0 BAT1 = 0 BAT0 = 1	(Example address for 24-bit mode only) Base address = $100 000 (compare $100) Compare all 12 address bits of BAC(11:0) Enable 3-Byte Packing mode for DMA No address multiplexing, A(7:0) Y-memory enabled X-memory enabled Program space enabled AA active low Asynchronous SRAM mode
Bus Control Register =$1F9FFF	23 22 21 20-16 15-13 12-10 9-5 4-0	BRH = 0 BLH = 0 BBS = 0 BDFW(4:0) = 11111 BA3W(2:0) = 100 BA2W(2:0) = 111 BA1W(4:0) = 11111 BA0W(4:0) = 11111	Bus Request Hold = 0 (reset) Bus Lock Hold = 0 (reset) Bus State (read only) = 0 (reset) unassigned external area = 31 wait states AA Bank 3 has 4 wait states (reset) AA Bank 2 has 7 wait states (reset) AA Bank 1 has 31 wait states (reset) AA Bank 0 has 31 wait states
Interface Control Register = $AF	7 6 5 4 3 2 1 0	INIT = 1 - HLEND = 1 HF1 = 0 HF0 = 1 HDRQ = 1 TREQ = 1 RREQ = 1	Initialize the ICR settings below - Little Endian mode Host Flag 1 cleared Host Flag 0 set Double Request mode enabled 00 means polling, 11 means interrupts

Table A-2. DSP56307 Port A DMA Receive Settings for HRRQ Mode (DSP1)

Register	Bits	Setting	Meaning
DMA Control Register 0 (DCR0) =$A812C0	23 22 21-19 18-17 16 15-11 10 9-7 6-4 3-2 1-0	DE=1 DIE = 0 DTM(2:0) = 101 DPR(1:0) = 00 DCON = 0 DRS(4:0) = 00010 D3D = 0 DAM(5:3) = 101 DAM(2:0) = 100 DDS(1:0) = 00 DSS(1:0) = 00	DMA channel enable DMA interrupt disabled DMA Transfer Mode = Mode 5 DMA priority is lowest No DMA continuous mode DMA Request Source is $\overline{\text{IRQC}}$ No DMA 3-Dimensional DMA Destination address post increment +1 DMA Source Address mode - No update DMA Destination space is X-memory DMA Source space is X-memory
DMA Source Register 0 (DSR0)	24-0	100 005	Target HI08 RX address mapped to AA3

Table A-2. DSP56307 Port A DMA Receive Settings for HRRQ Mode (DSP1)

Register	Bits	Setting	Meaning
DMA Destination Reg 0 (DDR0)	24-0	1500	X-memory receive data block address

Table A-3. DSP56307 Port A DMA Transmit settings for HTRQ Mode (DSP1)

Register	Bits	Setting	Meaning
DMA Control Register (DCR1) =$A80A50	23 22 21-19 18-17 16 15-11 10 9-7 6-4 3-2 1-0	DE=1 DIE = 0 DTM(2:0) = 101 DPR(1:0) = 00 DCON = 0 DRS(4:0) = 00001 D3D = 0 DAM(5:3) = 100 DAM(2:0) = 101 DDS(1:0) = 00 DSS(1:0) = 00	DMA channel enable DMA interrupt disabled DMA Transfer Mode = Mode 5 DMA priority is lowest No DMA continuous mode DMA request source is $\overline{\text{IRQB}}$ No DMA 3-dimensional DMA Destination Address mode - No update DMA source address post increment +1 DMA destination space is X-memory DMA source space is X-memory
DMA Source Register 0 (DSR0)	24-0	1400	X-memory transmit data block address
DMA Destination Reg 0 (DDR0)	24-0	100 005	Target HI08 TX address mapped to AA3

Table A-4. DSP56307 HI08 Register Settings (DSP2)

Register	Bits	Setting	Meaning
Host Control Register =$00 DMA mode =$03 Interrupt mode	4 3 2 1 0	HF3 = 0 HF2 = 0 HCIE = 0 HTIE = 0 HRIE = 0	Host Flag 3 cleared Host Flag 2 cleared Host command interrupt enabled Host transmit interrupt disabled Host receive interrupt disabled
Host Status register		Cleared by Initialize command on host side (ICR)	

Table A-4. DSP56307 HI08 Register Settings (DSP2)

Register	Bits	Setting	Meaning
Host Port Control Register =$1059	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	HAP = 0 HRP = 0 HCSP = 0 HDDS = 1 HMUX = 0 HASP = 0 HDSP = 0 HROD = 0 - HEN = 1 HAEN = 0 HREN = 1 HCSEN = 1 HA9EN = 0 HA8EN = 0 HGEN = 1	Host acknowledge polarity low ignored HDRQ/HTRQ are active low Host chip select active low Dual data strobes HRD/HWR enabled Non-multiplexed bus Address strobe polarity low ignored HRD/HWR data strobes are active low Host requests driven (not open drain) - Host Port enabled Host Ack ignored when HDRQ=1 HTRQ/HRRQ enabled if HDRQ=1 HCS host chip select enabled Address 9 ignored in non multiplexed Address 8 ignored in non multiplexed Unused host port signals used as GPIO

Table A-5. DSP56307 HI08 DMA Mode Receive settings for internal HI08 HSR[HRDF] interrupt (DSP2)

Register	Bits	Setting	Meaning
DMA Control Register 0 (DCR0) =$889AC0	23 22 21-19 18-17 16 15-11 10 9-7 6-4 3-2 1-0	DE=1 DIE = 0 DTM(2:0) = 001 DPR(1:0) = 00 DCON = 0 DRS(4:0) = 10011 D3D = 0 DAM(5:3) = 101 DAM(2:0) = 100 DDS(1:0) = 00 DSS(1:0) = 00	DMA channel enable DMA interrupt disabled DMA Transfer Mode = Mode 1 DMA priority is lowest No DMA Continuous mode DMA request source is HI08 Rx data full No DMA 3-dimensional DMA destination address post increment +1 DMA source address mode - No update DMA destination space is X-memory DMA source space is X-memory
DMA Source Register 0 (DSR0)	23-0	FFF FC6 (labelled M_HRX)	HI08 host Rx register address (internal)
DMA Destination Reg 0 (DDR0)	23-0	1500	X-memory receive data block address

Table A-6. DSP56307 HI08 DMA Mode Transmit settings for internal HI08 HSR[HTDE] interrupt (DSP2)

Register	Bits	Setting	Meaning
DMA Control Register (DCR1) =$88A250	23 22 21-19 18-17 16 15-11 10 9-7 6-4 3-2 1-0	DE=1 DIE = 0 DTM(2:0) = 001 DPR(1:0) = 00 DCON = 0 DRS(4:0) = 10100 D3D = 0 DAM(5:3) = 100 DAM(2:0) = 101 DDS(1:0) = 00 DSS(1:0) = 00	DMA channel enable DMA interrupt disabled DMA Transfer Mode = Mode 1 DMA priority is lowest No DMA continuous mode DMA request source is HI08 Tx data empty No DMA 3-dimensional DMA destination address mode - No update DMA source address post increment +1 DMA destination space is X-memory DMA source space is X-memory
DMA Source Register 0 (DSR0)	24-0	1400	X-memory transmit data block address
DMA Destination Reg 0 (DDR0)	24-0	FFF FC7 (labelled M_HTX)	HI08 host Tx register address (internal)

B DMA Mode Code Listings

Example B-1. DSP1 Port A Side DMA Based Driver

```
;------------------------------------------------------------------------------
;File:       porta_DMA.asm
;Version:    V0.1
;Eng:        Jim Gilbert
;Date:       6 Aug 98
;
;Purpose: Port A side of Port A to Hi08 Test
;      18 bytes transferred in each direction, DMA driven
;      Port A Tx 3 bytes at a time to Hi08 (HTRQ* -> IRQB* initiated)
;      Port A Rx 3 bytes at a time from Hi08 (HRRQ* -> IRQC* initiated)
;
;Copyright (C) 1998 Motorola
;      Wireless Infrastructure Systems Division
;      Networking & Computing Systems Group,
;      Semiconductor Products Sector
;------------------------------------------------------------------------------
      opt    CC,MEX
      page   140
             nolist
             include        "ioequ.asm"
             list

             section        porta_data
;------------------------------------------------------------------------------
;Test Pattern Section
;------------------------------------------------------------------------------
;External Definitions
              xdef   TXADDR              ; Transmit data base address
              org    x:$1400             ; 6 words of Transmit test data
TXADDR        dc            $202122,$232425,$262728
              dc            $292A2B,$2C2D2E,$2F3031
              endsec

              section  porta
;------------------------------------------------------------------------------
;Code Section
;------------------------------------------------------------------------------
;External Definitions
      xdef    START                        ;Program start address
      xref    TXADDR                       ;Transmit data start address

;------------------------------------------------------------------------------
; Local defines
;------------------------------------------------------------------------------
START           EQU    $100
INIT_PCTL       EQU          $040007       ; PLL ON, XTLD OFF and mul x8
INIT_DCR0       EQU          $2812C0       ; Rx DMA channel config
INIT_DCR1       EQU          $280A50       ; Tx DMA channel config
HI08T_ICR       EQU          $100000       ; target DSP HI08 ICR register
HI08T_ISR       EQU          $100002       ; target DSP HI08 ISR register
HI08T_IO        EQU          $100005       ; target DSP HI08 data I/O base
;TXADDR         EQU          $1400         ; Transmit source address
RXADDR          EQU          $1500         ; Receive buffer target address
TXCNT           EQU          $6            ; Transmit Count (6words=18bytes)
RXCNT           EQU          $6            ; Receive count (6words=18bytes)

;------------------------------------------------------------------------------
; Interrupt table
;------------------------------------------------------------------------------
                org          p:I_RESET     ;Hardware reset
                jmp          START
                org          p:I_IRQB      ; IRQB should go to DMA 1 for Tx
                jsr          error         ;
                org          p:I_IRQC      ; IRQC should go to DMA 0 for Rx
                jsr          error         ;

;------------------------------------------------------------------------------
```

```
;Local Routine
;-------------------------------------------------------------------------------
       org    p:START
       ori    #$3,mr                       ;Mask interrupts
       movep  #INIT_PCTL,x:<<M_PCTL        ;Initialise PLL
       movep  #$1F9FFF,x:<<M_BCR  ;4 wait states for AA3 (Chip select to HI08)
       movep  #$100CB9,x:<<M_AAR3 ;AA3 setup for $F00000 Base. Use to access HI08 Target

;-------------------------------------------------------------------------------
;Set up the receive buffer pointers
;-------------------------------------------------------------------------------
       movep  #HI08T_IO,x:<<M_DSR0     ;receive source location
       movep  #RXADDR,x:<<M_DDR0       ;receive target register
       movep  #INIT_DCR0,x:<<M_DCR0    ;configure DMA CH0 for Rx, don't enable

;-------------------------------------------------------------------------------
;Set up the transmit buffer pointers
;-------------------------------------------------------------------------------
       movep  #TXADDR,x:<<M_DSR1       ;transmit source location
       movep  #HI08T_IO,x:<<M_DDR1     ;transmit target register
       movep  #INIT_DCR1,x:<<M_DCR1    ;configure DMA CH1 for Tx, don't enable

;-------------------------------------------------------------------------------
;Enable HI08 Request mode
;-------------------------------------------------------------------------------
       move   #$AF,a0              ;initialise DSP target HI08 ICR register
       move   a0,x:HI08T_ICR       ;bits 0-1=11 HTRQ and HRRQ interrupts enabled
                                   ;bit 2 = 1 Double request mode enabled
                                   ;bit 3 = 1 Flag 0 set as indicator to slave HI08 HSR
                                   ;bit 4 = 0 Flag 1 clear "
                                   ;bit 5 = 1 for little endian
                                   ;bit 7 = 1 for INIT HI08 command.
       move   x:HI08T_ICR,a1       ;read back to verify!

;-------------------------------------------------------------------------------
; Enable interrupts.
;-------------------------------------------------------------------------------
       nop                         ; allow HI08 side to enable Double req
       nop                         ; before enabling IRQB, IRQC interrupts
       nop
       bset   #23,x:<<M_DCR0       ;Enable DMA receive (PortA <- HI08)
       bset   #23,x:<<M_DCR1       ;Enable DMA transmit (PortA -> HI08)
theend jmp    *                    ;wait forever. Now interrupt driven
error  jmp    *
       endsec

;-------------------------------------------------------------------------------
;       Copyright (C) 1998 Motorola
;-------------------------------------------------------------------------------
```

Example B-2. DSP2 HI08 Side DMA Based Driver

```
;------------------------------------------------------------------------------
;File:       hi08_dma.asm
;Version:    V0.1
;Eng:        Jim Gilbert
;Date:       24 Aug 98
;
;Purpose: HI08 side of Port A to Hi08 Test
;      18 bytes transferred in each direction, DMA driven
;      HSR_HRDF DMA interrupt receives 3 bytes at a time when Rx buffer full
;      HSR_TXDE DMA interrupt transmits 3 bytes at a time when Tx buffer empty
;
;Copyright (C) 1998 Motorola
;      Wireless Infrastructure Systems Division
;      Networking & Computing Systems Group,
;      Semiconductor Products Sector
;------------------------------------------------------------------------------
      opt    CC,MEX
      page   140
             nolist
             include        "ioequ.asm"
             list

             section        hi08_data
;------------------------------------------------------------------------------
;Test Pattern Section
;------------------------------------------------------------------------------
;External Definitions
              xdef    TXADDR        ; Transmit data base address
              org     x:$1400       ; 6 words of Transmit test data
TXADDR        dc            $000102,$030405,$060708
              dc            $090A0B,$0C0D0E,$0F1011
      endsec

      section                 hi08
;------------------------------------------------------------------------------
;Code Section
;------------------------------------------------------------------------------
;External Definitions
             xdef    START          ;Program start address
             xref    TXADDR         ;Transmit data start address

;------------------------------------------------------------------------------
;Constants Variable Definitions
;------------------------------------------------------------------------------
START        EQU    $100
INIT_PCTL    EQU    $040007       ; PLL ON, XTLD OFF and mul x8.
INIT_HCR     EQU    $0000         ; No flags, Tx/Rx interrupts disabled
INIT_HPCR    EQU    $1059         ; Double strobe, Double host request, No Ack. HI08 EN
INIT_DCR0    EQU    $089AC0            ; receive DMA channel config, mode 1
INIT_DCR1    EQU    $08A250            ; transmit DMA channel config, mode 1
;TXADDR      EQU    $1400              ; Transmit source address
RXADDR       EQU    $1500              ; Receive buffer target address
TXCNT        EQU    $6                 ; Transmit count = 6 words
RXCNT        EQU    $6                 ; Receive count = 6 words

;------------------------------------------------------------------------------
;Interrupt table
;------------------------------------------------------------------------------
             org    p:I_RESET          ; Hardware reset
             jmp    START
             org    p:I_HRDF           ; host receive data full not used
             jsr    error              ;
             org    p:I_HTDE           ; host transmit data empty not used
             jsr    error              ;
             org    p:I_HC             ; host command not used
             jsr    error              ;

;------------------------------------------------------------------------------
;Local Routine
;------------------------------------------------------------------------------
```

```
            org    p:START
            ori    #$3,mr                ;Mask interrupts
            movep  #INIT_PCTL,x:M_PCTL   ;Initialise PLL
            movep  #INIT_HCR,x:<<M_HCR   ;Initialise HI08
            movep  #INIT_HPCR,x:<<M_HPCR ;Enable Hi08 in double strobe & request mode

;-------------------------------------------------------------------------
;Set up the receive buffer pointers
;-------------------------------------------------------------------------
            movep  #M_HRX,x:<<M_DSR0     ;receive source register (HI08 Rx buffer)
            movep  #RXADDR,x:<<M_DDR0    ;receive target buffer base
            movep  #RXCNT-1,x:<<M_DCO0   ;configure DMA CH0 receive count
            movep  #INIT_DCR0,x:<<M_DCR0 ;configure DMA CH0 for Rx, don't enable

;-------------------------------------------------------------------------
;Set up the transmit buffer pointers
;-------------------------------------------------------------------------
            movep  #TXADDR,x:<<M_DSR1    ;transmit source buffer base
            movep  #M_HTX,x:<<M_DDR1     ;transmit target register (HI08 Tx buffer)
            movep  #TXCNT-1,x:<<M_DCO1   ;configure DMA CH0 transmit count
            movep  #INIT_DCR1,x:<<M_DCR1 ;configure DMA CH1 for Tx, don't enable

;-------------------------------------------------------------------------
;Ensure ICR initialisation by host before enabling DMA interrupts.
;-------------------------------------------------------------------------
pol_ICR     jclr   #3,x:<<M_HSR,*     ;Await Flag 0 set by host master before proceeding

;-------------------------------------------------------------------------
;Enable DMA Interrupts.
;-------------------------------------------------------------------------
            bset         #23,x:<<M_DCR0      ;Enable DMA receive (HI08 <- PORTA)
            bset         #23,x:<<M_DCR1      ;Enable DMA transmit (HI08 -> PORTA)
theend      jmp          *                   ;wait forever. Now DMA driven
error       jmp          *
            endsec

;-------------------------------------------------------------------------
;       Copyright (C) 1998 Motorola
;-------------------------------------------------------------------------
```

C Interrupt Mode Code Listings

Example C-1. DSP1 Port A-Side Interrupt-Based Driver

```
;------------------------------------------------------------------------------
;   File:     porta_3bl.asm
;   Version:  V0.1
;   Eng:      Jim Gilbert
;   Date:     6 Aug 98
;
;   Purpose:  Port A side of Port A to HI08 Test
;             18 bytes transferred in each direction, IRQ driven
;             Port A Tx 3 bytes at a time to Hi08 (HTRQ* -> IRQB* initiated)
;             Port A Rx 3 bytes at a time from Hi08 (HRRQ* -> IRQC* initiated)
;             Little endian mode
;
;   Copyright (C) 1998 Motorola
;             Wireless Infrastructure Systems Division
;             Networking & Computing Systems Group,
;             Semiconductor Products Sector
;------------------------------------------------------------------------------
        opt      CC,MEX
        page     140
                         nolist
                         include      "ioequ.asm"
                         list

                         section      porta_data
;------------------------------------------------------------------------------
;Test Pattern Section
;------------------------------------------------------------------------------
;External Definitions
             xdef   TXADDR                ; Transmit data base address
             org    x:$1400               ; 6 words of Transmit test data
TXADDR    dc                       $202122,$232425,$262728
          dc                       $292A2B,$2C2D2E,$2F3031
          endsec

          section   porta
;------------------------------------------------------------------------------
;Code Section
;------------------------------------------------------------------------------
;External Definitions
        xdef START               ;Program start address
        xref TXADDR              ;Transmit data start address

;------------------------------------------------------------------------------
;Local defines
;------------------------------------------------------------------------------
START         EQU    $100
INIT_PCTL     EQU    $040007           ; PLL ON, XTLD OFF and mul x8
HI08T_ICR     EQU    $100000           ; target DSP HI08 ICR register
HI08T_ISR     EQU    $100002           ; target DSP HI08 ISR register
HI08T_IO      EQU    $100005           ; target DSP HI08 data I/O base
;TXADDR       EQU    $1400             ; Transmit source address
RXADDR        EQU    $1500             ; Receive buffer target address
TXCNT         EQU    $6                ; Transmit Count (6words=18bytes)
RXCNT         EQU    $6                ; Receive count (6words=18bytes)

;------------------------------------------------------------------------------
;Interrupt table
;------------------------------------------------------------------------------
             org    p:I_RESET          ;Hardware reset
             jmp    START
             org    p:I_IRQA           ;
             jsr    error              ;
             org    p:I_IRQB           ; handles PortA transmits to HI08
             jsr    irqb_int           ;
             org    p:I_IRQC           ; handles PortA receives from HI08
             jsr    irqc_int           ;
```

```
            org    p:I_IRQD          ;
            jsr    error             ;

;-------------------------------------------------------------------------
;Local Routine
;-------------------------------------------------------------------------
      org         p:START
      ori         #$3,mr                ;Mask interrupts
      movep       #INIT_PCTL,x:<<M_PCTL ;Initialise PLL
      movep       #$1F9FFF,x:<<M_BCR    ;4 wait states for AA3 (Chip select to HI08)
      movep       #$100CB9,x:<<M_AAR3   ;AA3 setup for $F00000 Base. Use to access HI08
                                        ;Target
      movep       #$0001B0,x:<<M_IPRC   ;Set IRQB, IRQC int to priority 2, edge
                                        ;sensitive
                                          ;bit 4-3 = 10 for IRQB level 2
                                          ;bit   5 = 1  for IRQB edge
                                          ;bit 7-6 = 10 for IRQC level 2
                                          ;bit   8 = 1  for IRQC edge

;-------------------------------------------------------------------------
;Set up the receive buffer pointers
;-------------------------------------------------------------------------
      move        #RXADDR,r5            ;Rx block destination base pointer

;-------------------------------------------------------------------------
;Set up the transmit buffer pointers
;-------------------------------------------------------------------------
      move        #TXADDR,r2            ;Tx Block Source base pointer

;-------------------------------------------------------------------------
;Enable HI08 Request mode
;-------------------------------------------------------------------------
      move   #$AF,a0             ;initialise DSP target HI08 ICR register
      move   a0,x:HI08T_ICR      ;bits 0-1=11 HTRQ and HRRQ interrupts enabled
                                 ;bit 2 = 1 Double request mode enabled
                                 ;bit 3 = 1 Flag 0 set as indicator to slave HI08 HSR
                                 ;bit 4 = 0 Flag 1 clear "
                                 ;bit 5 = 1 for little endian
                                 ;bit 7 = 1 for INIT HI08 command.
      move   x:HI08T_ICR,a1      ;read back to verify!

;-------------------------------------------------------------------------
;Enable interrupts.
;-------------------------------------------------------------------------
         nop                     ; allow HI08 side to enable Double req
         nop                     ; before enabling IRQB, IRQC interrupts
         nop
         andi    #$FC,mr         ;Unmask interrupts
theend   jmp     *               ;wait forever. Now interrupt driven


;-------------------------------------------------------------------------
; Interrupt Handlers
;-------------------------------------------------------------------------
; Transmit Port A to Hi08 3 bytes at a time (HTRQ* -> IRQB* initiated)
irqb_int
      clr         a                     ;
      move        r2,a0                 ;
;     cmp         #(TXADDR+TXCNT),a     ; if not end Tx block
                                        ; This don't work. z bit not set!!
      clr         b
      move        #TXADDR+TXCNT,b0      ;
      cmp         b,a                   ; if not end Rx block
      jne         txcont                ; continue transmit & interrupts
      bra irqb_e

txcont
      move        #HI08T_IO,r6          ; Set pointer to memory mapped Target HI08
      clr         a
      clr         b
      move        x:(r2)+,b0            ; prepare next word (3 bytes) from Tx buffer
      do          #3,txloop
```

```
       move        b0,x:(r6)+           ; transmit next byte data to HI08
       asr         #8,b,b               ; move along to next byte
txloop
       nop
irqb_e
       nop
       rti

; Receive Port A from Hi08 3 bytes at a time (HRRQ* -> IRQC* initiated)
irqc_int
       move        #HI08T_IO,r1         ; Set pointer to memory mapped Target HI08
       clr         a                    ;
       clr         b                    ;
       do          #3,rxloop
       move        x:(r1)+,a            ; receive next byte data from HI08
       and         #$0000FF,a
       move        a1,x0
       or          x0,b                 ; build up b with each byte in turn
       asr         #8,b,b               ; move along to next byte
       nop
rxloop
       move        b0,x:(r5)+           ; store full word (3 bytes) in Rx buffer
       nop
       rti
error  jmp         *                    ; loop forever.
       endsec
;-------------------------------------------------------------------------------
;       Copyright (C) 1998 Motorola
;-------------------------------------------------------------------------------
```

Example C-2. DSP2 HI08-Side Interrupt-Based Driver

```
;-------------------------------------------------------------------------------
;File:    hi08_3b.asm
;Version: V0.1
;Eng:     Jim Gilbert
;Date:    6 Aug 98
;
;Purpose: HI08 side of Port A to Hi08 Test
;      18 bytes transferred in each direction, IRQ driven
;      HSR_HRDF interrupt receives 3 bytes at a time when Rx buffer full
;      HSR_TXDE interrupt transmits 3 bytes at a time when Tx buffer empty
;
;Copyright (C) 1998 Motorola
;      Wireless Infrastructure Systems Division
;      Networking & Computing Systems Group,
;      Semiconductor Products Sector
;-------------------------------------------------------------------------------
       opt    CC,MEX
       page   140
              nolist
              include        "ioequ.asm"
              list
              section        hi08_data
;-------------------------------------------------------------------------------
;Test Pattern Section
;-------------------------------------------------------------------------------
;External Definitions
                xdef   TXADDR              ; Transmit data base address
                org    x:$1400             ; 6 words of Transmit test data
TXADDR          dc            $000102,$030405,$060708
                dc            $090A0B,$0C0D0E,$0F1011
                endsec

                section       hi08
;-------------------------------------------------------------------------------
;Code Section
;-------------------------------------------------------------------------------
;External Definitions
        xdef  START                          ;Program start address
       xref   TXADDR                         ;Transmit data start address
```

```
;------------------------------------------------------------------------------
;Constants Variable Definitions
;------------------------------------------------------------------------------
START         EQU  $100
INIT_PCTL     EQU           $040007     ; PLL ON, XTLD OFF and mul x8.
INIT_HCR      EQU           $0003       ; No flags, Tx/Rx interrupts enabled
INIT_HPCR     EQU           $1059       ; Double strobe, Double host request,No Ack,HI08
                                        ; on
;TXADDR       EQU           $1400          ; Transmit source address
RXADDR        EQU           $1500          ; Receive buffer target address
TXCNT         EQU           $6             ; Transmit count = 6 words
RXCNT         EQU           $6             ; Receive count = 6 words

;------------------------------------------------------------------------------
;Interrupt table
;------------------------------------------------------------------------------
              org           p:I_RESET      ;Hardware reset
              jmp           START
              org           p:I_HRDF       ; host receive data full
              sr            rxf_int        ;
              org           p:I_HTDE       ; host transmit data empty
              jsr           txe_int        ;
              org           p:I_HC         ; host command
              jsr           error          ;

;------------------------------------------------------------------------------
;Local Routine
;------------------------------------------------------------------------------
          org       p:START
          ori       #$3,mr                  ;Mask interrupts
          movep     #INIT_PCTL,x:M_PCTL     ;Initialise PLL
          movep     #$000002,x:<<M_IPRP     ;Set HI08 int to priority 2
          movep     #INIT_HCR,x:M_HCR       ;Initialise HI08
          movep     #INIT_HPCR,x:M_HPCR     ;Enable Hi08 in double strobe & request mode

;------------------------------------------------------------------------------
;Set up the receive buffer pointers
;------------------------------------------------------------------------------
          move      #RXADDR,r5              ;Rx block destination base pointer

;------------------------------------------------------------------------------
;Set up the transmit buffer pointers
;------------------------------------------------------------------------------
          move      #TXADDR,r2              ;Tx Block Source base pointer

;------------------------------------------------------------------------------
;Ensure ICR initialisation by host before enabling interrupts.
;------------------------------------------------------------------------------
pol_ICR
       movep       x:<<M_HSR,a1            ;check out value in register
       jclr        #3,x:<<M_HSR,pol_ICR    ;Await Flag 0 set by host master before
                                           ;proceeding

;------------------------------------------------------------------------------
;Enable Interrupts
;------------------------------------------------------------------------------
       andi        #$FC,mr                 ;Unmask interrupts
wait   jmp         *                       ;wait forever. Now interrupt driven

;------------------------------------------------------------------------------
;Interrupt Handlers
;------------------------------------------------------------------------------
;Receive from HI08 host 1 24-bit word at a time
rxf_int
       clr         a                    ;
       move        r5,a0                ;
;      cmp         #RXADDR+RXCNT-1,a    ; if not end Rx block
                                        ; This don't work. z bit not set!!
       clr         b
       move        #RXADDR+RXCNT-1,b0   ;
       cmp         b,a                  ; if not end Rx block
       jne         rxcont               ; continue receive interrupts
```

```
       bclr        #0,x:<<M_HCR          ; last so Disable host receive (HTRQ) interrupts
       nop
rxcont
       movep       x:<<M_HRX,a0          ; receive next byte data from HI08
       move        a0,x:(r5)+            ;
       nop
       rti

;Transmit to Hi08 host 1 24-bit word at a time
txe_int
       clr         a                     ;
       move        r2,a0                 ;
;      cmp         #TXADDR+TXCNT,a       ; if not end Tx block
                                         ; This don't work. z bit not set!
       clr         b
       move        #TXADDR+TXCNT,b0      ;
       cmp         b,a                   ; if not end Tx block
       jne         txcont                ; continue transmit & interrupts
       bclr        #1,x:<<M_HCR          ; last so Disable transmit (HRRQ) interrupts
       bra         irqc_e
txcont
       move        x:(r2)+,a0            ; transmit next byte data from buffer to HI08
       movep       a0,x:<<M_HTX          ;
irqc_e
       nop
       rti

error  jmp         *                     ; loop forever.
       endsec

;------------------------------------------------------------------------------
;       Copyright (C) 1998 Motorola
;------------------------------------------------------------------------------
```

NOTES:

NOTES:

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 Muenchen, 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
support.asia@freescale.com

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

freescale™
semiconductor