Application Note

AN2244/D
Rev 1, 2/2002

Interconnecting Two MSC8101ADS Boards Across a 60x-Compatible Bus to the Host Interface

by Renaud Le Friec and Scott Smith

CONTENTS

Within the telecommunications infrastructure, communication between devices is an essential requirement. For example, banks of DSP resource are often used for applications such as voice transcoders within basestations, and these devices must have a mechanism for receiving or transmitting data to or from the outside world. A typical system architecture contains a central controller terminating protocol layers and distributing the payload to one or more DSP banks (or arrays) for further processing (see **Figure 1**). This application note focuses on a subset of the system architecture, describing the interface between an MSC8101 device acting as an integrated communications processor (host MSC8101) and a single MSC8101 device (called the HDI16 MSC8101) acting as a standard digital signal processor (DSP) via its 16-bit host parallel interface (HDI16). The host MSC8101 accesses the HDI16 MSC8101's host interface registers via a memory mapping on its own 60x-compatible bus.

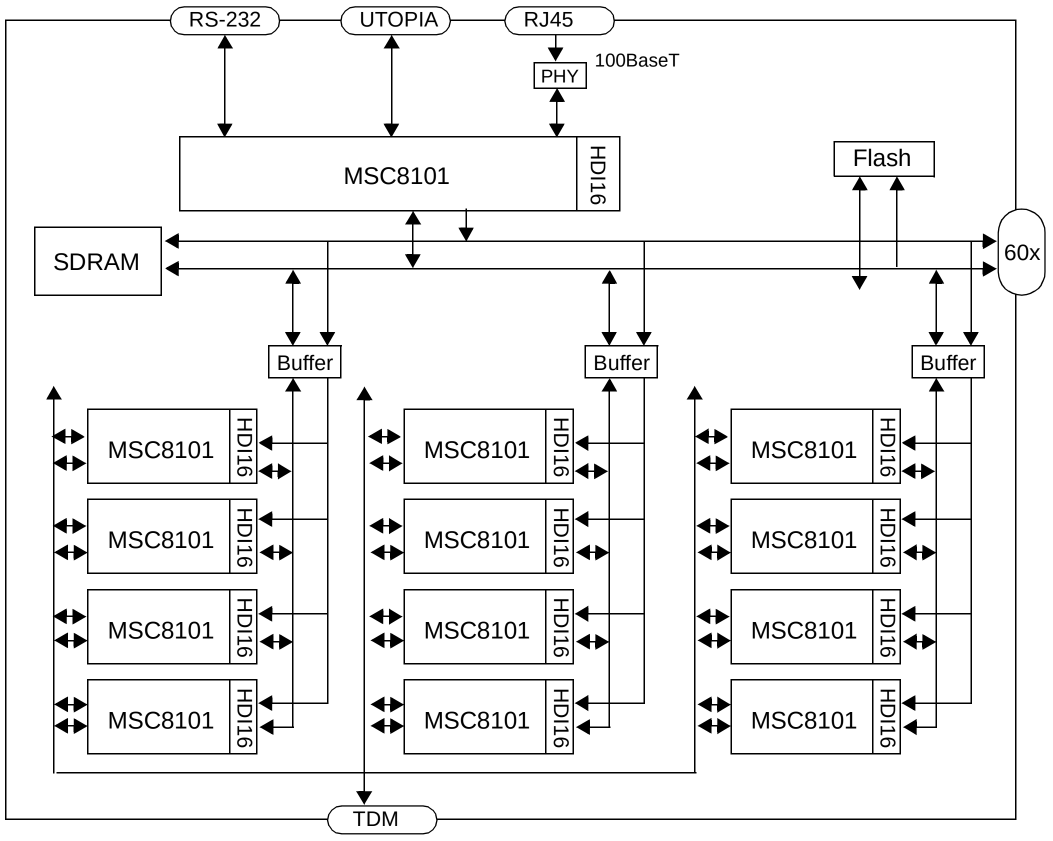


Figure 1. Controller to Multiple-HDI16 DSP Architecture

1 MSC8101 Device Overview

The 16-bit Motorola MSC8101 processor is the first member of the family of DSPs based on the StarCore™ SC140 core. On a single device, this very versatile processor integrates the high-performance SC140 four-ALU (Arithmetic Logic Unit) DSP core with 512 KB of internal memory, a Communications Processor Module (CPM), a 64-bit 60x-compatible bus, a very flexible system integration unit (SIU), and a 16-channel DMA controller. With its four-ALU core, the MSC8101 can execute up to four multiply-accumulate (MAC) operations in a single clock cycle. The MSC8101 CPM is a 32-bit RISC-based communications protocol engine that can network to Time-Division Multiplexed (TDM) highways, Ethernet, and Asynchronous Transfer mode (ATM) backbones. The very large internal memory, 512 KB, reduces the need for external program and data memories. The MSC8101 offers 1200 DSP MIPS or 3000 RISC MIPS performance using an internal 300 MHz clock with a 1.6 V core and independent 3.3 V input/output (I/O). MSC8101 power dissipation is estimated at 0.5 W.

Three key modules of the MSC8101 device that are crucial to the application discussed in this document are as follows:

- *Host interface (HDI16)*. A 16-bit wide, full-duplex, double-buffered parallel port that can connect directly to the data bus of a host processor. The HDI16 supports a variety of buses and gluelessly connects to a number of industry-standard microcomputers, microprocessors, and DSPs. The HDI16 also supports the 8-bit host data bus, which makes it fully compatible with the DSP56300 HDI08 (as viewed by the host side, not from the DSP side). The host bus can operate asynchronously to the SC140 core clock, and the HDI16 registers are divided into two banks. The host register bank is accessible to the external host, and the core register bank is accessible to the SC140 core.
- *Memory controller*. Supports a glueless interface to the external memory and peripheral devices on the external system bus. It also enables interfacing with the internal DSP memory and DSP peripherals residing on the internal local bus. Located on the external system bus, the memory controller controls a maximum of eight memory banks shared by a high-performance SDRAM machine, a general-purpose chip-select machine (GPCM), and two user-programmable machines (UPMs). It supports a glueless interface to synchronous DRAM (SDRAM), SRAM, EPROM, flash EPROM, burstable RAM, regular DRAM devices, extended data output DRAM devices, and other peripherals. Two additional memory banks control access to resources using the local bus.
- *Multi-channel DMA Controller*. Supports up to 16 time-division multiplexed channels and buffer alignment by hardware. The DMA controller connects to both the system bus and the local bus and can function as a bridge between both buses. The DMA controller enables hot swap between channels using time-division multiplexed channels that impose no cost in clock cycles. Sixteen priority levels support synchronous and asynchronous transfers on the bus and give a varying bus bandwidth per channel.

The host MSC8101 device must configure its memory controller to map the HDI16 host-side registers into memory locations within its 60x system bus memory space. The HDI16 MSC8101 device configures the HDI16 directly, treating it as an internal peripheral. After initial set-up of the HDI16 host interface, the communication between the host MSC8101 device and the HDI16 MSC8101 device occurs via DMA transfers, which are automatically triggered by signals generated from the HDI16 host interface. This application note describes how to configure and use the HDI16 host interface, the memory controller, and the DMA Controller.[1]

[1] For details on these MSC8101 modules, consult the following chapters of the *MSC8101 Reference Manual*: Chapter 14 (HDI16), Chapter 10 (memory controller), and Chapter 15 (DMA). For details on programming the HDI16 port and the DMA controller, consult the *MSC8101 User's Guide*.

Also provided are the following:

- The register settings for these modules as used by this application (**Section 7,** *Source Code Files, Software Flow, and Register Settings*).
- The necessary hardware connections between the host MSC8101 device and the HDI16 MSC8101 device.
- The physical connections for interfacing the host and HDI16 MSC8101 devices are described specifically for Motorola's MSC8101 Application Development System (MSC8101ADS) (**Section 6**). The MSC8101ADS board is a general development platform with the MSC8101 device installed in a socket, on-board Flash and SDRAM memory, plus communication transceivers for Fast Ethernet, ATM 155-Uni, E1/T1, RS-232, and a Crystal stereo audio codec. With this system, developers can start developing software or use the schematics as a reference for their own system development. [2]
- Example source code presented in two parts, one for the host MSC8101 device and one for the HDI16 MSC8101 device. These transfers are not optimized. The source code is presented for illustrative purposes, and not for use as an HDI16 driver.

2 System Bus–HDI16 Host Interface

To understand the function of the parallel control interface between the host MSC8101 system bus and the HDI16 MSC8101 host interface, it is important to know the device requirements on each side.

2.1 Host Memory Controller

The sole function of the memory controller SDRAM machine is to enable back-to-back memory read or write operations using page mode, pipelined operation, and bank interleaving for high-performance systems. GPCM-based chip selects interface predominantly to simple asynchronous devices such as ROM, Flash memory, SRAM, and so on. A GPCM-derived chip select ensures a glueless interface to such devices over a range of port sizes (8, 16, and 32-bit) and speed grades. However, as compared to the GPCM, the UPM offers much more flexibility in timing to target a broader range of system devices, and it offers access to more peripherals, making it more suited to our application. Through the UPM-controlled memory interface, software can define the chip selects and control strobes on each bus clock to a 1/4 clock and 1/2 clock granularity, respectively. Developers commonly use this flexibility for user-defined interfaces to application-specific integrated circuits (ASICs) or, as in our implementation example, to DSPs. The UPM-defined interface can be used with any of the host MSC8101 eight chip selects to give a programmable port size and strobe generation matching that required by the HDI16 MSC8101.

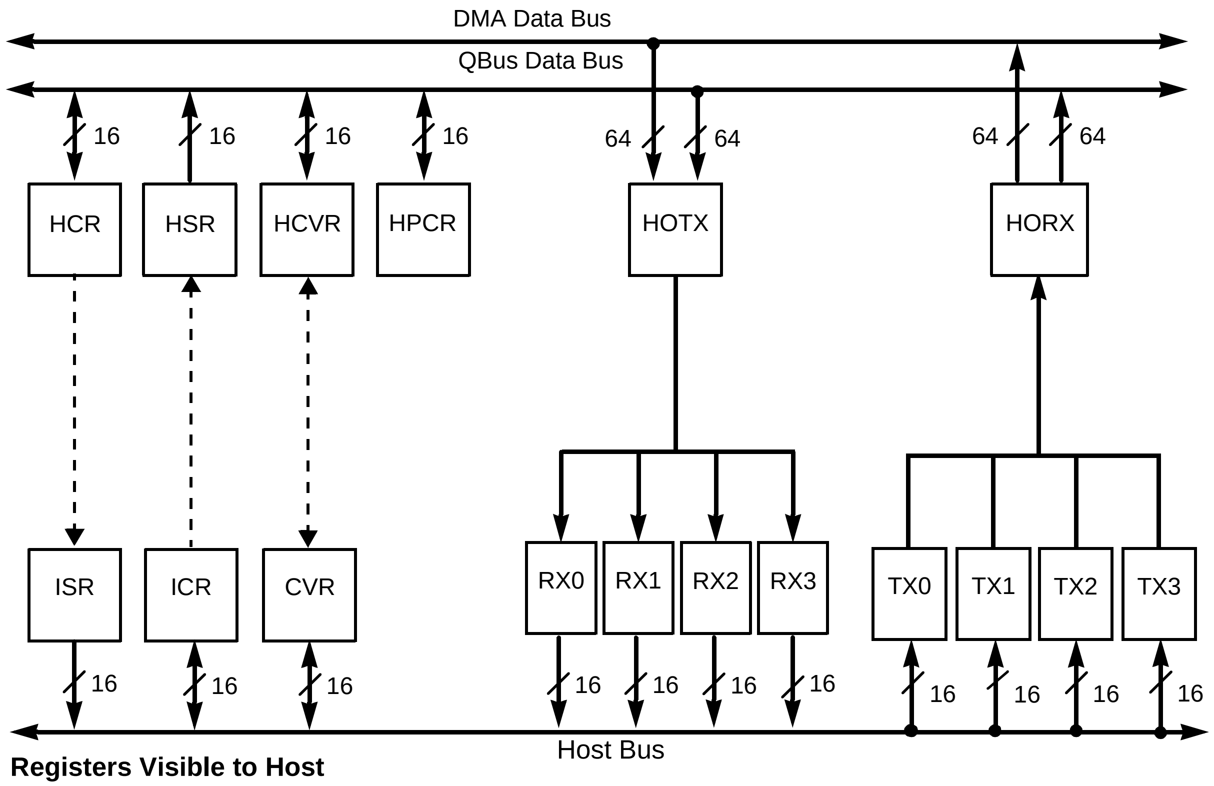
2.2 HDI16 Host Interface

The HDI16 host interface has two sets of 16-bit wide registers, one set visible only within the MSC8101 and the other set visible only to the external host processor. **Figure 2** illustrates the relationship between the two sides. All HDI16 registers are mapped directly onto the MSC8101 QBus, and the transmit and receive FIFOs are mapped onto the DMA data bus so that the DMA controller can access them directly without intervention by the SC140 core. The QBus, which is part of the SC140 extended core interface, is a high-speed pipeline bus with separate address and data phases. It is a single-master bus with the same frequency as the SC140 core. The MSC8101 peripherals, in particular the HDI16 interface, are slaves to the QBus.

[2] For details on the MSC1810ADS board, consult the *MSC8101ADS User's Manual*.

Internally-Visible Registers

Accessed by the SC140 core		
HCR	Host Control Register	0x0000
HSR	Host Status Register	0x0040
HCVR	Host Command Vector Register	0x0060
HPCR	Host Port Control Register	0x0020
HOTX	Host Transmit Data Register	0x0080
HORX	Host Receive Data Register	0x00A0
NOTE1: Both HOTX and HORX are FIFOs with a capacity of four 64-bit words.		



Registers Visible to Host

Accessed by the Host Processor					
ISR	Interface Status Register	0x2			
ICR	Interface Control Register	0x0			
CVR	Command Vector Register	0x1			
RX[0–3]	Receive Data Registers	RX0 0x7	RX1 0x6	RX2 0x5	RX30 x4
TX[0–3]	Transmit Data Registers	TX0 0x7	TX1 0x6	TX2 0x5	TX3 0x4

Figure 2. MSC8101 HDI16 Port Registers

The most important aspect of the HDI16 host interface for our purposes is that it is specified as an asynchronous interface, reducing concerns over clock skew between the HDI16 host interface and the host device buses. Furthermore, since the host MSC8101 accesses all the HDI16 registers in the HDI16 MSC8101 with a single chip select and four address lines, the DSP host interface is in effect an asynchronous memory-mapped region. For the HDI16 host interface in single-strobe mode, the host device asserts a chip select, a single data strobe, and a read/write line to select HDI16 read or write bus operations. The read and write strobes are also used as the data latch control to complete the bus transactions, avoiding the need for any handshake termination signal from the DSP.

Through appropriate mode selection, the HDI16 feature set is fully supported by the host MSC8101 UPM-controlled signals. The UPM-defined interface can be used with any of the host MSC8101's eight chip selects to give the 16-bit port size and strobe generation matching that of the HDI16 MSC8101 device's host interface.

2.3 Physical Interconnections

Table 1 shows the physical interconnections between the host MSC8101ADS UPM-controlled system bus and the host interface port of the HDI16 MSC8101ADS. The connectors specified in this table refer to the corresponding connector names, as defined in the *MSC8101 Application Development System User's Manual*.

Table 1. Physical MSC8101ADS Interconnects

Host MSC8101 Side				HDI16 MSC8101 Side		
P1 Pin No.	P2 Pin No.	System + CPM Edge Connector		P4 = H0 Pin No.	HDI16 Connector	
		Signal Name	Signal Description		Signal Name	Signal Description
C14		EXPD0	60x-Compatible Data Bus	3	HD0	Host Bidirectional Data Port
C15		EXPD1		4	HD1	
C16		EXPD2		5	HD2	
C17		EXPD3		6	HD3	
C18		EXPD4		7	HD4	
C19		EXPD5		8	HD5	
C20		EXPD6		9	HD6	
C21		EXPD7		10	HD7	
C22		EXPD8		11	HD8	
C23		EXPD9		12	HD9	
C24		EXPD10		13	HD10	
C25		EXPD11		14	HD11	
C26		EXPD12		15	HD12	
C27		EXPD13		16	HD13	
C28		EXPD14		17	HD14	
C29		EXPD15		18	HD15	
A10		EXPA25	Address Bus	21	HA0	Host Interface Address Lines
A11		EXPA26		22	HA1	
A14		EXPA29		23	HA2	
A15		EXPA30		24	HA3	
C4		BTOLCS1[1]	Chip Select 6	25	HCS1	Host Chip Select 1
				26	HCS2	Tie this to 3.3V (P4, Pin 33)

Table 1. Physical MSC8101ADS Interconnects (Continued)

Host MSC8101 Side			
P1 Pin No.	P2 Pin No.	System + CPM Edge Connector	
		Signal Name	Signal Description
	D10	DREQ1	DMA Request 1 (PC22)
	D8	DREQ2	DMA Request 2 (PC24)
D10		EXPGPL3	UPM GPL Line 3
D12		EXPGPL5	UPM GPL Line 5
B[1–3] C1, C3, C6, C13 D[1–3] D[6–13] D[16–D32]	C31, C32	0V	Ground

[1] This signal is designated as BTOLSC1 in the *MSC8101ADS User's Manual* but as CS6 in the *MSC8101 Reference Manual*.

HDI16 MSC8101 Side		
P4 = H0 Pin No.	HDI16 Connector	
	Signal Name	Signal Description
27	HRRQ/HACK	Receive Host Request OP
28	HTRQ/HREQ	Transmit Host Request OP
29	HRW	Host Read/Write
30	HDS	Host Data Strobe
1, 2, 19, 20, 35, 36	0V	Ground

Notable features of the connections depicted in **Table 1** are as follows:

- One chip select, CS6 (or BTOLCS1 as it is called here), is used to map memory accesses from the host MSC8101 to the HDI16 MSC8101 HDI16 port and is connected to the HDI16 chip-select line (HCS).
- Two separate General-Purpose Strobe Lines (GPLx) are used in this interface:
  - PGPL3 is programmed to generate the HDI16 read/write line (HRW) which is typically high for a read access and low for write access.
  - PGPL5 is programmed to generate the HDI16 Data strobe (HDS), which must be asserted every 16-bit read or write transaction.
- *Data Lines*. The host MSC8101 device's 60x-compatible bus data lines (EXPDx) directly connect to the HDI16 data lines (HDx).
- *DMA Request/Service Request Lines*. Two separate DMA Request lines (DREQx) connect to the Service Request signals (HRRQ and HTRQ) on the HDI16.
- *Address Lines*. The Address lines between the host MSC8101 and the HDI16 MSC8101 connect to enable burst transfers across the HDI16. The connections are defined as follows:
  - 60x EXPA25 → HA0
  - 60x EXPA26 → HA1
  - 60x EXPA29 → HA2
  - 60x EXPA30 → HA3
- *Ground Lines*. To provide the best ground plane while connecting the two MSC8101ADS boards, it is highly recommended that all grounds be common and connected together.

2.4 Host DMA Transfers

To prevent the host MSC8101 device from dedicating large amounts of SC140 core resource to polling and accessing the HDI16 MSC8101 device continually, the HDI16 peripheral can assert service request interrupts to the host as needed. Two HDI16 request generation modes are available, namely single or double request modes. In Single Request mode, one signal ($\overline{\text{HREQ}}$) is used to request service for both receive and transmit operations, but in Double Request mode, separate requests are possible for transmit and receive (HTRQ and HRRQ). Single Request mode has the advantage that it saves on the number of interrupt inputs consumed on the host processor but the disadvantage of using a single request for both directions. Therefore, either transmit and receive directions implement a known transfer protocol (for example, alternating Tx/Rx), or the host must poll the Interface Status Register (ISR) on each request to determine the direction of data flow.

For ease of use and efficiency, we use Double Request mode. However, the request signals are not limited to generating interrupts on the host device. They can also control DMA transfers across the interface, which is useful because the SC140 core is not involved in a DMA transfer after initial set-up. The DMA transfers can be configured to attempt to transfer *n*-bytes (the user-configured DMA transfer size) only when the relevant external request signal (DREQx) is present. Because the HTRQ signal indicates one or more free locations within the HDI16 Rx FIFO, connecting it to a DREQ signal guarantees that the host's DMA controller attempts to transfer data to the DSP only when there is a spare location. Similarly, because the HRRQ signal indicates one or more populated locations within the HDI16 Tx FIFO, connecting it to a DREQ signal guarantees that the host DMA controller attempts to transfer data from the DSP only when data is present.

Note: Until RevA of the MSC8101 device is available, the HTRQ and HRRQ signals only indicate the status of 8-byte locations within the HDI16 FIFOs. Therefore, for real-world applications, the DMA transfer size should be set at 8 bytes. However, to show how burst accesses across the interface can be implemented, and because we know the number of bytes being transferred, this application uses the HTRQ and HRRQ signals to control burst transfers.

To facilitate burst transfers, the host's 60x bus address lines are connected to dispose of certain 60x signals:

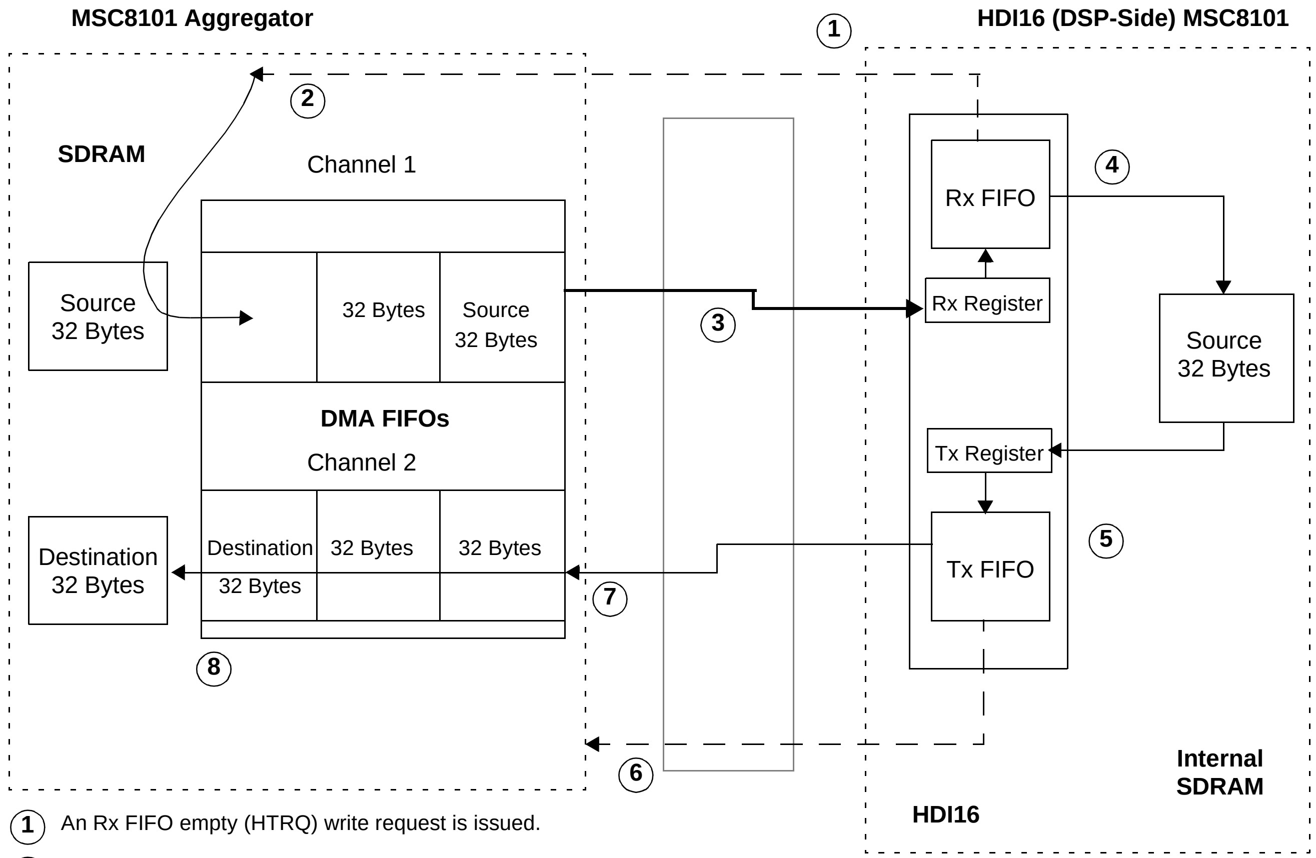
- The host 60x A31 signal is not required because the HDI16 registers are 16-bit word addressed.
- We have also disposed of the 60x A27 and A28 signals so that the host-side transmit and receive registers wrap around the same four 16-bit word addresses.

  For example, the host can obtain the Host Transmit Register 0 on the HDI16 peripheral (address 0x04) by accessing any of the following memory-mapped addresses: 0x20, 0x28, 0x30, or 0x38. This is critical for burst accesses because the source or destination addresses increment after each 16-bit access to the interface for all 16 transactions within that burst. Using the addressing as defined, if the first access is at address 0x20, the last is at address 0x3E but, more importantly, the four host Tx/Rx registers will have been looped around four times.

3 HDI16 Device Configuration, Synchronization, and Set-Up

Figure 3 shows how the host MSC8101 and HDI16 MSC8101 devices interact with each other through their DMA channels. This section describes how to configure the HDI16 slave device to enable this interaction. Section 4, *Host Device Configuration*, on page 10, describes how to configure the host device. The HDI16 MSC8101 is configured to enable the HDI16 host interface for communications with an external host MSC8101 as follows:

1. Configure the memory controller to map the HDI16 registers into memory.
2. Synchronize the HDI16 MSC8101 and host MSC8101 devices.
3. Set up and enable the DMA receive engine.



1. An Rx FIFO empty (HTRQ) write request is issued.
2. DREQ2 assertion: DMA write request triggers transfer.
3. When 32 bytes are in the FIFO and DREQ2 is asserted, the DMA controller transfers data to the 60x system bus.
4. An HDI16 write triggers a DMA transfer to memory.
5. When a 32-byte buffer is full, an interrupt triggers a DMA transfer to a Tx register.
6. A Tx FIFO not empty (HRRQ) DMA read request is issued.
7. DREQ1 assertion: a DMA read request triggers a transfer.
8. When a byte is in the FIFO, the DMA transfers data to SDRAM.

Figure 3. DMA Transfer Interaction Between the Host MSC8101 and the HDI16 MSC8101 Devices

3.1 Device Synchronization

To synchronize the host and HDI16 software communications, we use host flags to monitor the status of the communications. We can access eight HDI16 Host Flags (HF) by polling from the host and HDI16 devices. Either the SC140 core or the host can set or clear these “general-purpose flags” for HDI16-to-host communications. If any of HF[0–7] is set, depending on how the host flags are used, this may indicate an application-specific state within the HDI16 or host requiring intervention by the host processor or the HDI16 processor. The values of HF[0–3] and HF[4–7] are reflected as follows:

- For the HDI16 SC140 core side, in the Host Control (HCR) and Host Status (HSR) registers
- For the host side, in the Interface Control (ICR) and in the Interface Status (ISR) registers

For example, if the HDI16 MSC8101 software modifies these HF values, the host MSC8101 can read the modified values by reading the ISR. HF[0–7] can be used individually or as encoded pairs in a simple HDI16-to-host communication protocol, implemented in both the HDI16 MSC8101 software and host

MSC8101 software. Consequently, the HDI16 HDI16 side acts as a master during the synchronization process. When the HDI16-side software is ready to process commands sent from the host, it sets HF4 to signal to the host that it is awaiting a synchronization acknowledgment. The host acknowledges synchronization by sending back HF0 and HF1. Once this sequence completes, the interface is synchronized.

3.2 DMA Set-Up

Setting up the DMA receive engine involves three main steps:

1. Selecting the mode for HDI16 DMA signals.
2. Initializing the DMA buffers and buffer descriptors.
3. Setting up and operating DMA interrupts.

3.2.1 Mode for HDI16 DMA Signals

The HDI16 can provide DMA signals in two different modes: Single-Request mode (HREQ/HACK) or Double-Request mode (HRRQ/HTRQ). Our application uses Double-Request mode (HRRQ/HTRQ) in order to provide the external host with differentiated receive and transmit direction signals to run DMA data transfers efficiently and transparently. To enable Double-Request mode after power-up, a sequence bit (BCSR0/1) must be set up appropriately in the Board Control and Status Registers of the appropriate ADS board. The BCSRx, which are 32-bit read/write registers, control or monitor most of the hardware options on the ADS. The BCSRx reside on the system bus and are accessible through the MSC8101 memory controller.

3.2.2 DMA Buffers and Buffer Descriptors

Buffer descriptors manipulate buffers and are located in the DMA Channel Parameters RAM (DCPRAM) space.[3] The four parameters attributed to each buffer are:

- *BD_ADDR*. This field holds the buffer address pointer. If the buffer is defined as cyclic, the original address value is restored when the BD_SIZE value reaches zero.
- *BD_SIZE*. This field contains the remaining size of the buffer. This value is decremented by the transfer block size each time the DMA controller issues a transaction, until it reaches zero. When BD_SIZE reaches zero, the original value is restored to the value of BD_BSIZE.
- *BD_ATTR*. This 32-bit parameter describes the attributes of the channel handling this buffer. By setting the appropriate bit in this field, we can program the DMA controller to issue an interrupt when the size reaches zero, meaning that the buffer is terminated/full. This parameter is also used to select the type of transaction to be initiated by the DMA channel.
- *BD_BSIZE*. This field holds the base size of the buffer.

Two different DMA transfer modes can be selected: Flyby mode or Dual-Address mode. In Flyby mode, a single address transaction is used for the transfer, whereas Dual-Access mode requires a complementary channel to be configured. Flyby mode is the mode of choice here because it exactly fulfills the requirements of our application. It is generally used to transfer data between an external peripheral and

[3] For information on buffer descriptors, consult the Overview chapter of the *MSC8101 User's Guide* or the application note entitled *Initializing MSC8101 CPM Parameter RAM and Buffer Descriptors* (AN2147). The format of the buffer descriptors used in DMA transactions is described in the "Programming Model" section of the DMA chapter in the *MSC8101 Reference Manual*.

external memory or an internal peripheral and internal memory. The HDI16 MSC8101treats the HDI16 interface is an internal peripheral, making it easy to transfer data to and from the HDI16 interface and internal memory.

3.2.3 DMA Interrupts

The HDI16 MSC8101's DMA controller services both transmit and receive data registers for data transfers. To set up the DMA interrupt vector appropriately in the vector table, all interrupts must be disabled. The IRQ18 vector is called each time a DMA interrupt is triggered by a "buffer terminated" event. A handler must be attached to this vector to handle the interrupts.

DMA servicing is divided into two parts in the interrupt handler: read/receive and write/transmit. Each routine is assigned to a given channel. In this application, DMA channel 0 and 1 are programmed respectively to perform DMA read and DMA write routines, using DMA Channel Configuration registers (DCHCRx) to configure the connection between a DMA requestor and the corresponding DMA channel. All the channel properties are programmed, including the relevant lines in DCPRAM, before the channel is enabled by asserting the ACTV bit.

The first task of the interrupt routines is to clear pending requests by writing to the Interrupt Pending registers (IPRx). The PIC interrupt pending registers, IPRA and IPRB, are 16-bit read/write registers that the SC140 core uses to monitor pending interrupts and to reset edge-triggered interrupts. The DMA controller reports status and events to the host, and it generates a maskable interrupt for each channel. The maskable interrupt is routed to the PIC according to the setting of the M bit of the relevant channel in the DMA Internal Mask Register (DIMR). By reading the DMA Status Register (DSTR), we can check which channel has requested use of the DMA controller.

When the $\overline{\text{IRQ}}$ servicing routine is called, a special type of DMA transfer starts, depending on the request. For an HDI16 read request, a DMA transfer extracts data for the host receive FIFO and writes to a data buffer in memory. In contrast, if an HDI16 write request is received, a DMA transfer occurs between the data buffer in memory and the HDI16 host transmit FIFO.

4 Host Device Configuration

The operation of the external host is illustrated in its most basic form in the state transition diagram shown in **Figure 4**.

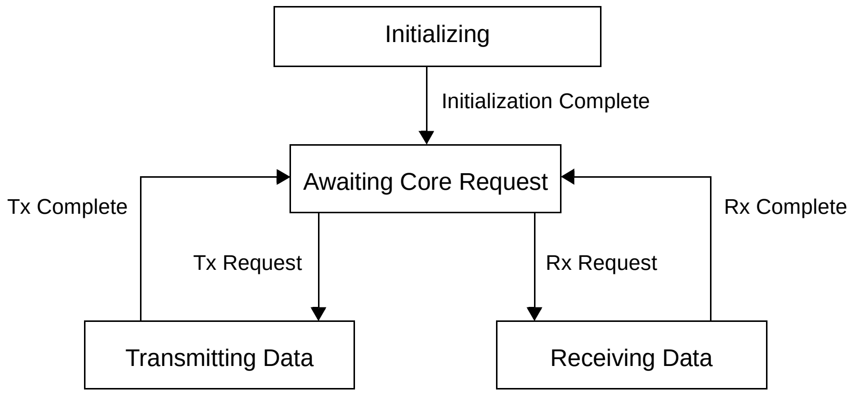


Figure 4. Simple Representation of Host States

To achieve these states, perform the following steps:

1. Configure the host memory controller to enable mapping of the HDI16 registers in memory.
2. Configure the host HDI16 registers, synchronize the host and HDI16 sides, and enable the HDI16 Service Request signals from the host side.
3. Configure the host I/O ports to enable DMA request lines.
4. Configure the host DMA registers and channel buffer descriptors.
5. Configure the host DMA channels so that HDI16 service requests automatically trigger DMA transfers.
6. Configure the common host DMA parameters.
7. Enable host DMA channels.

4.1 Host Memory Controller

The MSC8101 memory controller is used to map each HDI16 host register to a specific memory location. Bank 6 within the memory controller handles these accesses since this bank is free within the current MSC8101ADS memory controller set-up. The Base and Option Registers are programmed to map the HDI16 registers to a base memory address of 0x30000000, select a port size of 16 bits, and enable bursts to the HDI16.

UPM A is selected to issue the required HDI16 signals, so the RAM array for this UPM must be loaded. Where possible, UPM A uses burst transfers across the HDI16, but first a burst DMA transfer size must be selected. Also, the data must be 32-byte aligned in memory. The HDI16 interface timings provided by UPM A are presented in **Section 6**. Within the example code, the host-side HDI16 registers are accessed via a type definition (HPORT) that simply overlays the memory-mapped host register locations.

4.2 Host HDI16 Registers

Once the memory controller is initialized, the HDI16 registers are accessible and configured to meet the needs of this application (see **Figure 2** for a list of these registers). Before initializing any of the HDI16-specific flags, the host must initialize the HDI16 MSC8101 by transmitting its Reset Configuration Word, writing a byte value to each successive reset configuration register.

The host then waits for the HDI16 MSC8101 to set a host flag, which indicates that the HDI16 MSC8101 is up and running. To complete this handshake, the host sets a couple of host flags to notify the HDI16 MSC8101 that it is running, too. At this point in the Application source code there is an `#if-#else` definition to give users the option of testing their physical HDI16 interconnections via a simple polling mechanism, instead of directly using the full-blown DMA transfer version of the code. The polling mechanism operates as a continuous loop composed of the following steps, all performed by the host MSC8101:

1. Wait for the HDI16 transmit buffer empty flag.
2. Write sixteen 16-bit words to the HDI16 Transmit Word Registers (these are processed as 16 separate single-beat transactions, not a burst transfer).
3. Wait for the Receive Data Full flag, which indicates that the HDI16 MSC8101 has data ready.
4. Read sixteen 16-bit words to the HDI16 Receive Word Registers.
5. Compare the data transmitted with the data received and store the number of discrepancies.

Otherwise, DMA mode is selected so the host must configure the HDI16 registers to enable the HRRQ and HTRQ signals, which automate the DMA transfers.

4.3 Host I/O Ports

The registers for the parallel I/O ports are configured, allowing the HRRQ and HTRQ signals to reach the DMA controller via the DMA Request lines (DREQ1 and DREQ2, respectively). The DREQ1 and DREQ2 signals are enabled on Port C using the following registers:[4]

- Port Pin Assignment Register C (PPARC): Bits PC24 and PC22 are set, and all other bits are cleared.
- Port Data Direction Register C (PDIRC): All bits are cleared.
- Port Special Option Register C (PSORC): Bits PC24 and PC22 set, and all other bits are cleared.

For cleanliness, the Port C Open-Drain Register (PODRC) and Port Data Register C (PDATC) are also initialized, with all bits cleared.

4.4 Host DMA Controller

This section describes how to enable the buffer descriptors and configure the DMA channels. The host DMA controller must be configured to transfer data back and forth between the SDRAM and the HDI16. The host MSC8101 treats the HDI16 as external memory on the 60x-compatible system bus, so we must use Dual-Access mode instead of the DMA Flyby mode. We must configure two channels for each complete DMA transfer. These transfers are triggered by the DREQ1 and DREQ2 request signals (see **Figure 5**).

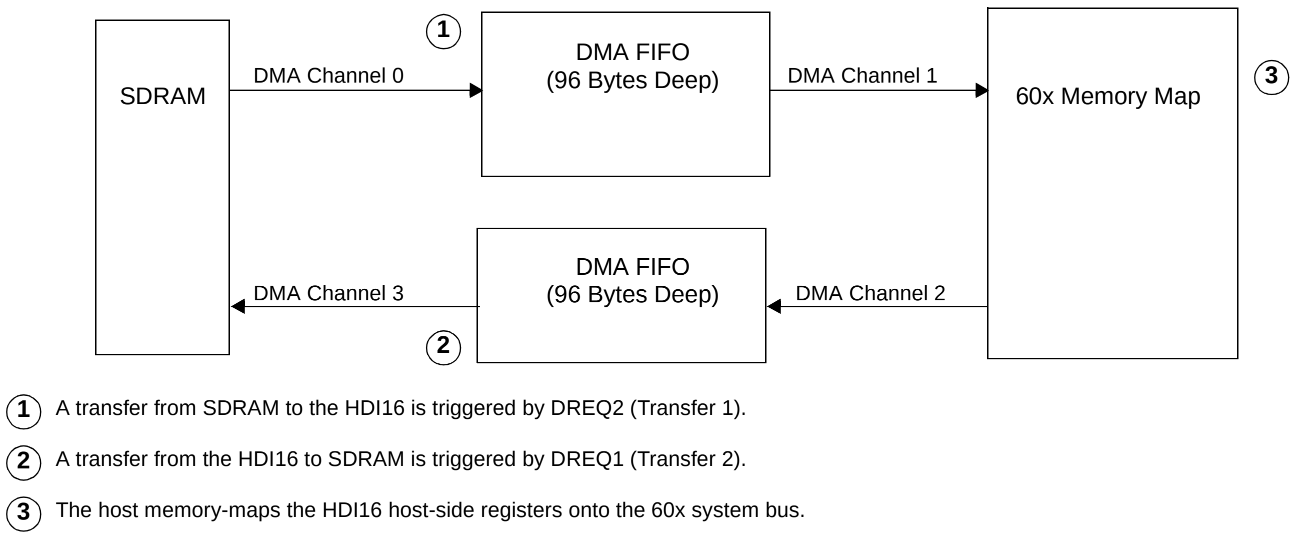

SDRAM
DMA Channel 0
1
DMA FIFO (96 Bytes Deep)
DMA Channel 1
60x Memory Map
3
DMA FIFO (96 Bytes Deep)
DMA Channel 3
DMA Channel 2
2
1 A transfer from SDRAM to the HDI16 is triggered by DREQ2 (Transfer 1).
2 A transfer from the HDI16 to SDRAM is triggered by DREQ1 (Transfer 2).
3 The host memory-maps the HDI16 host-side registers onto the 60x system bus.

Figure 5. Host-Side DMA Channel Configuration

Because four DMA channels must be enabled to handle the two complete transfer mechanisms, four buffer descriptors are configured as follows, one for each channel:[5]

- Transfer 1, Channel 0
  - *Attributes*. Read Channel, Cyclic Address, Burst Transfer Size, Flush FIFO
  - *Source Address*. Transmit Test Pattern array within SDRAM
  - *Size and Base Size*. 32 bytes (one burst)

[4] The I/O port registers are described in detail in the “Programming Model” section in the Parallel I/O Ports chapter of the *MSC8101 Reference Manual*.

[5] The format of these buffer descriptors is described in detail in the “Programming Model” section of the DMA chapter in the *MSC8101 Reference Manual*.

- Transfer 1, Channel 1
  - *Attributes*. Write Channel, NO_INC, Cyclic Address, Burst Transfer Size, Flush FIFO
  - *Source Address*. Host Transmit Word Registers of HDI16
  - *Size and Base Size*. 32 bytes (one burst)
- Transfer 2, Channel 2
  - *Attributes*. Read Channel, NO_INC, Cyclic Address, Burst Transfer Size, Flush FIFO
  - *Source Address*. Host Receive Word Registers of HDI16
  - *Size and Base Size*. 32 bytes (one burst)
- Transfer 2, Channel 3
  - *Attributes*. Write Channel, Cyclic Address, Burst Transfer Size, Flush FIFO
  - *Source Address*. Received Test Pattern array within SDRAM
  - *Size and Base Size*. 32 bytes (one burst)

Channels 0 through 3 specify a cyclic address so that when a transfer completes, the pointer jumps back to the base address, and the buffer is executed again. Channels 1 and 2 must also specify that they require no incrementing of the address (NO_INC) to stop the automatic incrementing of the data address by eight bytes once an eight-byte segment of the burst is transferred. The four 16-bit HDI16 Data Registers can transfer only 8 bytes of a 32-byte burst at any one time, but after each 8-byte segment we still want to point to the original address of these registers.

Since there are four DMA channels, four DMA Configuration Registers must be initialized. The content of each register gives the following configurations for data transfers, all of which occur on the 60x-compatible system bus:

- *Transfer 1, Channel 0:* Internal Requestor. Buffer Descriptor 0 is used for this channel; the channel priority level is 6.
- *Transfer 1, Channel 1:* DREQ2 starts the transfer. DREQ2 is level triggered and active low. Buffer Descriptor 1 is used for this channel; the channel priority level is 5.
- *Transfer 2, Channel 2:* DREQ1 starts the transfer. DREQ1 is level triggered and active low. Buffer Descriptor 2 is used for this channel; the channel priority level is 3.
- *Transfer 2, Channel 3:* Internal Requestor. Buffer Descriptor 3 is used for this channel; the channel priority level is 4.

The DMA channels that transfer the data across the 60x:HDI16 interface (channel 1 for Tx and channel 2 for Rx) are controlled by DREQ input signals. These signals connect to the HTRQ and HRRQ signals output via the HDI16 host interface. A transfer of *n*-bytes (where *n* is the DMA transfer size specified within the buffer descriptor parameters for a specific channel) proceeds only when the relevant DREQ signal is active.

Channels 0 and 3 use the Internal Requestor trigger to initiate a transfer. Each pair of DMA channels (0 and 1, 2 and 3, and so on) share a DMA FIFO, and the internal requestor triggers a DMA transfer depending on the state of this FIFO. For example, here DMA Channel 0 is used to populate the DMA FIFO by retrieving data from memory. Whenever there is enough free space within the FIFO to load *n*-bytes (where *n* is the DMA transfer size), this DMA channel triggers. Similarly, DMA Channel 3 triggers whenever there is data within the FIFO that must be saved in memory.

4.5 Common Host Parameters

Common host parameters are not needed in our application, so both internal and external DMA interrupts are masked by clearing all bits in the DMA Internal Mask Register (DIMR) and DMA External Mask Register (DEMR). For cleanliness, the DMA Status Register (DSTR) and the DMA

Transfer Error Address Status Register (DTEAR) are cleared by writing a value of one (1) to all the bits, and the DMA Pin Configuration Register (DPCR) bits are cleared because we do not require it.

4.6 Host DMA Channels

Each DMA channel is enabled by setting the ACTV bit in the configuration register associated with that channel. Here, the activation of the DMA channels is enclosed in an infinite loop so we can monitor and test the results of the transfer. The loop consists of the following basic steps:

1. Activate Transfer 1 (Host Transmit) DMA Channels 0 and 1.
2. Activate Transfer 2 (Host Receive) DMA Channels 2 and 3.
3. Wait for the DMA transfers to complete.
4. Compare the data received with that transmitted and record any discrepancies.
5. Increment the contents of the first transmit test pattern location so that the same data is not sent repeatedly. This is the basic operation of the host HDI16 DMA transfer mechanism.

5 Hardware Timings

The host MSC8101 UPM-controlled bus and the HDI16 MSC8101 host interface are both programmable. Careful programming of the host MSC8101 chip select registers and UPM can accommodate the HDI16 MSC8101 host port timings. The timings in **Table 1** are based on an implementation with a 40 MHz host MSC8101 60x-compatible bus-to-200 MHz HDI16 MSC8101. On any bus access the critical timing for both reads and writes is typically the data latch point. For the UPM-based read access, the host MSC8101 has the flexibility to latch data on a rising or falling CLKOUT edge. The falling CLKOUT edge is used here to latch the HDI16 data into the host MSC8101 as soon as possible. After the data is latched, an appropriate HDI16 host interface data hold time is ensured before the data strobe (DS) and Chip Select (CS1) are negated. For the UPM-based write access, the critical action is enveloping the DS assertion with CS asserted to ensure a proper write data hold time after latching by the HDI16 host port. Special attention is given to both the host read and write access strobe (DS) negation times (HDS assertion).

The HDI16 MSC8101 specifies some restrictions for consecutive register access, which results in a hold-off negation time for the read and write access strobes. Rather than restrict the firmware to avoid consecutive bus accesses to host port registers, the negation hold-off times are accommodated in the UPM hardware interface settings. Additional clocks are built into the end of UPM-based cycle, giving appropriate time before the next bus cycle starts.

For an application with a 40 MHz host MSC8101 60x-compatible bus and a 200 MHz HDI16 MSC8101, the host port read and write accesses are both five 40 MHz clocks. The timings are based on a buffered connection between the host MSC8101ADS and the HDI16 MSC8101ADS host port. The timings can be readily adapted to allow external decode logic to be added to support chip selects for a larger number of DSP HDI16 host ports.

Figure 6 and **Figure 7** represent the various signals at both the host side (60x) and the DSP side (HDI16) of the interface.

Note: For best viewing results, view **Figure 6** and **Figure 7** online using a zoom factor of 300 percent.

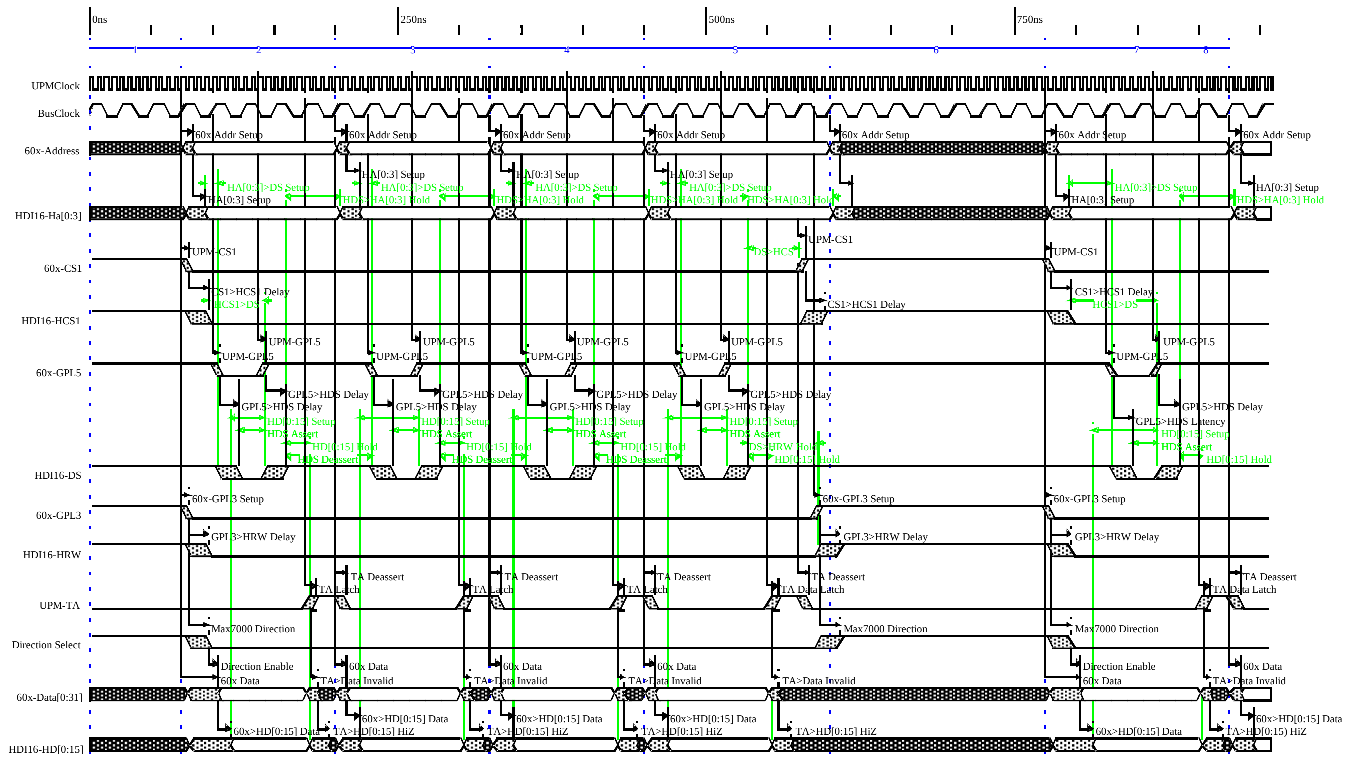


Figure 6. Host MSC8101 60x-to-HDI16 MSC8101 Host Port Write Timing

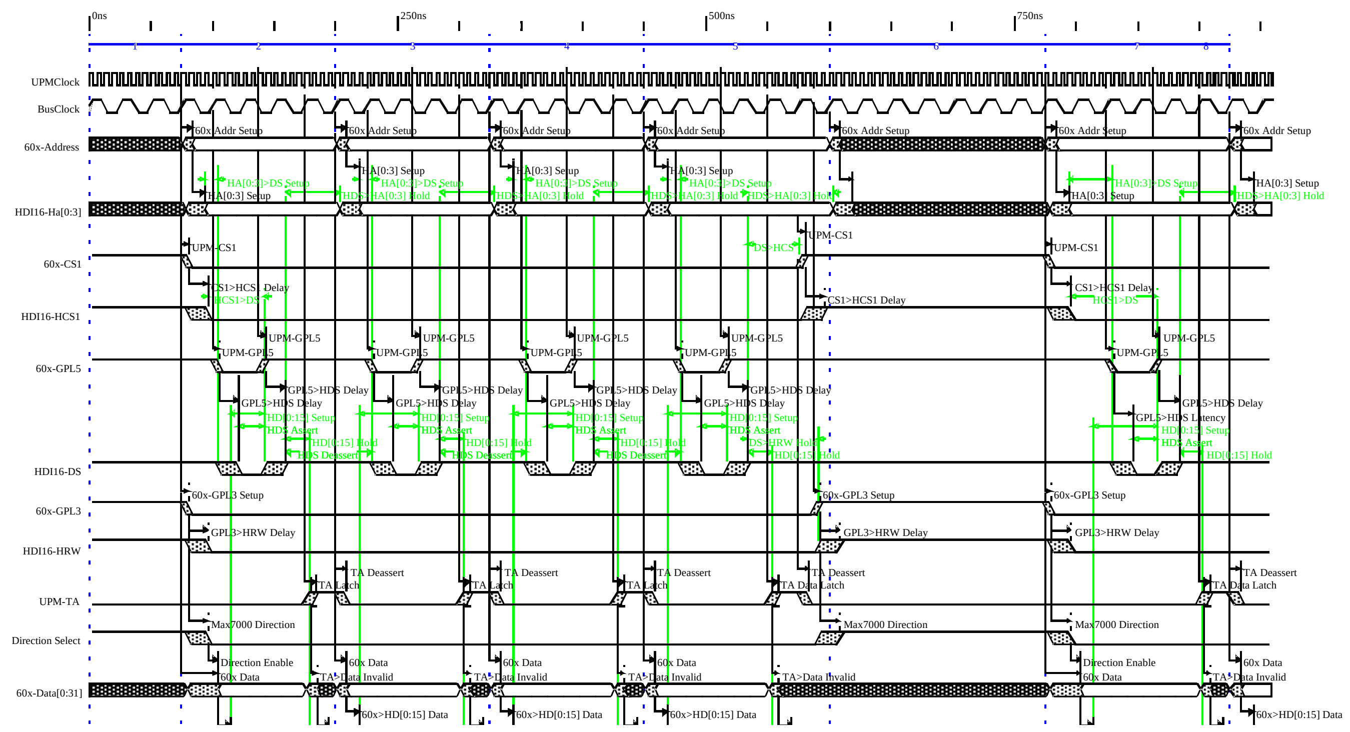


Figure 7. Host MSC8101 60x-to-HDI16 MSC8101 Host Port Read Timing

Table 2 defines the hexadecimal values loaded into the UPM RAM Array to obtain these timing signals.

Table 2. UPM RAM Array Values

Cycle Type	Single Read	Burst Read	Single Write	Burst Write	Refresh	Exception
UPM Offset	**0**	**0x8**	**0x18**	**0x20**	**0x30**	**0x3c**
offset + 0	0x0FFFFC00	0x0FFFFC80	0x0FFF3C00	0x0FFF3C80	0xFFFFFFFF	0xFFFFFFFF
offset + 1	0x0FFFF000	0x0FFFF000	0x0FFF3000	0x0FFF3000	0xFFFFFFFF	0xFFFFFFFF
offset + 2	0x0FFFF000	0x0FFFF000	0x0FFF3400	0x0FFF3400	0xFFFFFFFF	0xFFFFFFFF
offset + 3	0x0FFFF000	0x0FFFF000	0x0FFF3C00	0x0FFF3C00	0xFFFFFFFF	0xFFFFFFFF
offset + 4	0xFFFFFC05	0x0FFFFC8C	0xFFFF3C05	0x0FFF3C8C	0xFFFFFFFF	
offset + 5	0xFFFFFFFF	0xFFFFFC01	0xFFFFFFFF	0xFFFFFC01	0xFFFFFFFF	
offset + 6	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	
offset + 7	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	
offset + 8		0xFFFFFFFF		0xFFFFFFFF	0xFFFFFFFF	
offset + 9		0xFFFFFFFF		0xFFFFFFFF	0xFFFFFFFF	
offset + a		0xFFFFFFFF		0xFFFFFFFF	0xFFFFFFFF	
offset + b		0xFFFFFFFF		0xFFFFFFFF	0xFFFFFFFF	
offset + c		0xFFFFFFFF		0xFFFFFFFF		
offset + d		0xFFFFFFFF		0xFFFFFFFF		
offset + e		0xFFFFFFFF		0xFFFFFFFF		
offset + f		0xFFFFFFFF		0xFFFFFFFF		

Note: Within the UPM RAM array definition in the `upminit.c` source file are many `#if 0-#else-#endif` statements that can be changed to `#if 1` to select faster HDI16 transfer UPM patterns. These faster patterns have been successfully tested.

6 Physical MSC8101ADS Settings

Table 3 defines the default switch configuration for the HDI16 MSC8101 platform.[6]

Table 3. HDI16 MSC8101ADS Switch Settings

Switch	Settings used	Comments
SW1 – HOST	On-on-on–on	
SW2 – PPC_CTRL	On-on-on-on-on-on-on-on	
SW5	32-bit	Mandatory to use the HDI16
SW6	32-bit	Mandatory to use the HDI16
SW9	On-on-on-off-on-off-on-on	MODCLK #40, HDI16 enabled by SW9/8 being on.
SW10	On-on-on-on	
SW11 – S/W_OPT	On-on-on-on	

Table 4 defines the default Switch configuration for the host MSC8101 platform.

Table 4. Host MSC8101ADS Switch Settings

Switch	Settings used	Comments
SW1 – HOST	On-on-on–on	
SW2 – PPC_CTRL	On-on-on-on-on-on-on-on	
SW5	64 bit	
SW6	64 bit	
SW9	On-on-on-off-on-off-off-off	MODCLK #40, Load Reset Config word from Altera Device, disabled HDI16.
SW10	On-on-on-on	
SW11 – S/W_OPT	On-on-on-on	

Our application uses a 16.384 MHz CLKIN crystal on both ADS platforms. With the SW9 settings defined previously, MODCLK 40 is selected, yielding a host 60x bus clock speed of ~40 MHz on the host MSC8101 device and a DSP core speed of ~200 MHz on the HDI16 MSC8101 device.

7 Source Code Files, Software Flow, and Register Settings

This application was developed using the Metrowerks® CodeWarrior® for StarCore, production release 1.0. The CodeWarrior IDE provides a set of tools for developing software using a GUI. The Metrowerks project file references an `8101_Initialization.cfg` file with which it can configure SDRAM and so on via the debugger. The location of this file can be modified to suit your own implementation of Metrowerks. **Table 5** defines both the host-side and DSP-side source code project files associated with this application note.

Table 5. Source Code Project Files

Module	Filename	Description
HDI16 MSC8101	8101.mcp	Metrowerks Project File
	main.c	Main program
	hi16.c	HDI16 and DMA procedures
	MmapQ001.h	8101 Register Memory Mapping
	reg8101.h	8101 Registers
	hi16.h	HDI16 and DMA procedure and variable declarations
	type8101.h	Specific variables types used in this application
Host MSC8101	HDI16DMAHost.mcp	Metrowerks project file
	hdi16.c	Initializes HDI16, initializes HDI16 MSC8101, sets up transmit test pattern and received test pattern destination address, and passes control to DMA routine.
	upminit.c	The functions within are used to memory map the HDI16 (Host Interface) for the host MSC8101 by initializing UPM A.

[6] For details on how to power up and set up the MSC8101ADS board, consult the *MSC8101ADS User's Manual*. One chapter of this manual presents installation instructions for the MSC8101ADS board.

Table 5. Source Code Project Files (Continued)

Module	Filename	Description
Host MSC8101 Cont.	dma.c	Exercise the DMA functionality using both Peripheral -> Memory and Memory -> Peripheral in Dual Address Mode.
	msc8101.h	MSC8101 Register Memory Mapping
	types.h	Specific variables types used in this application
	hdi16.h	Includes common header files, defines HDI16 memory map and card_initialize (UPM set-up) function prototype.
	hdi16masks.h	Defines bit masks for HDI16 Host Registers and a timeout count value for “wait” loops.
	dma.h	Header file which defines the do_dma function prototype.
	dmatest.h	Defines DMA buffer, 8101ads and IO Port specific constants.

Figure 8 and **Figure 9** detail the software flow of both the HDI16 MSC8101 and host MSC8101 programs.

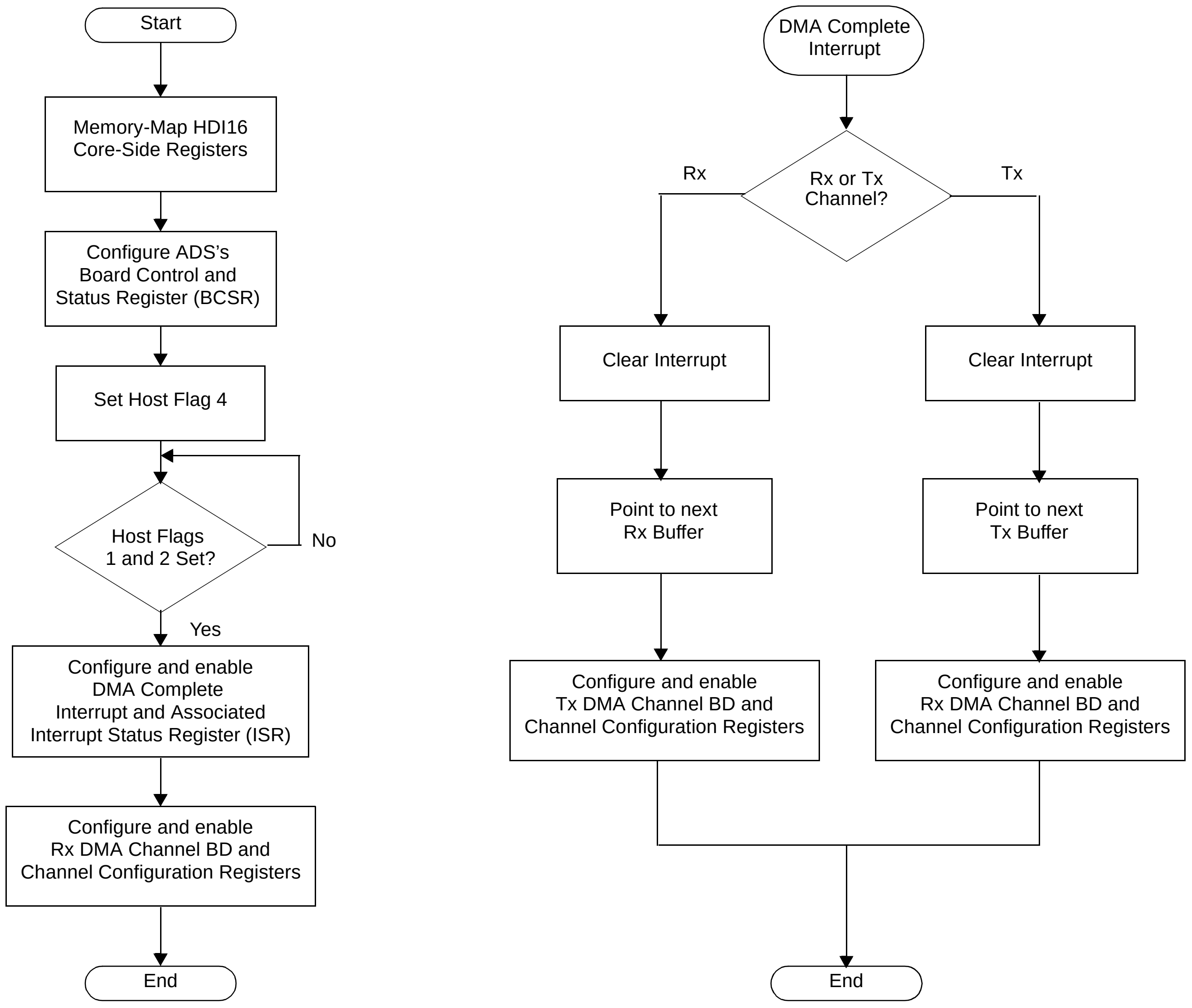

Start
Memory-Map HDI16 Core-Side Registers
Configure ADS's Board Control and Status Register (BCSR)
Set Host Flag 4
Host Flags 1 and 2 Set?
No
Yes
Configure and enable DMA Complete Interrupt and Associated Interrupt Status Register (ISR)
Configure and enable Rx DMA Channel BD and Channel Configuration Registers
End
DMA Complete Interrupt
Rx or Tx Channel?
Rx
Tx
Clear Interrupt
Clear Interrupt
Point to next Rx Buffer
Point to next Tx Buffer
Configure and enable Tx DMA Channel BD and Channel Configuration Registers
Configure and enable Rx DMA Channel BD and Channel Configuration Registers
End

Figure 8. HDI16 MSC8101 Software Flow

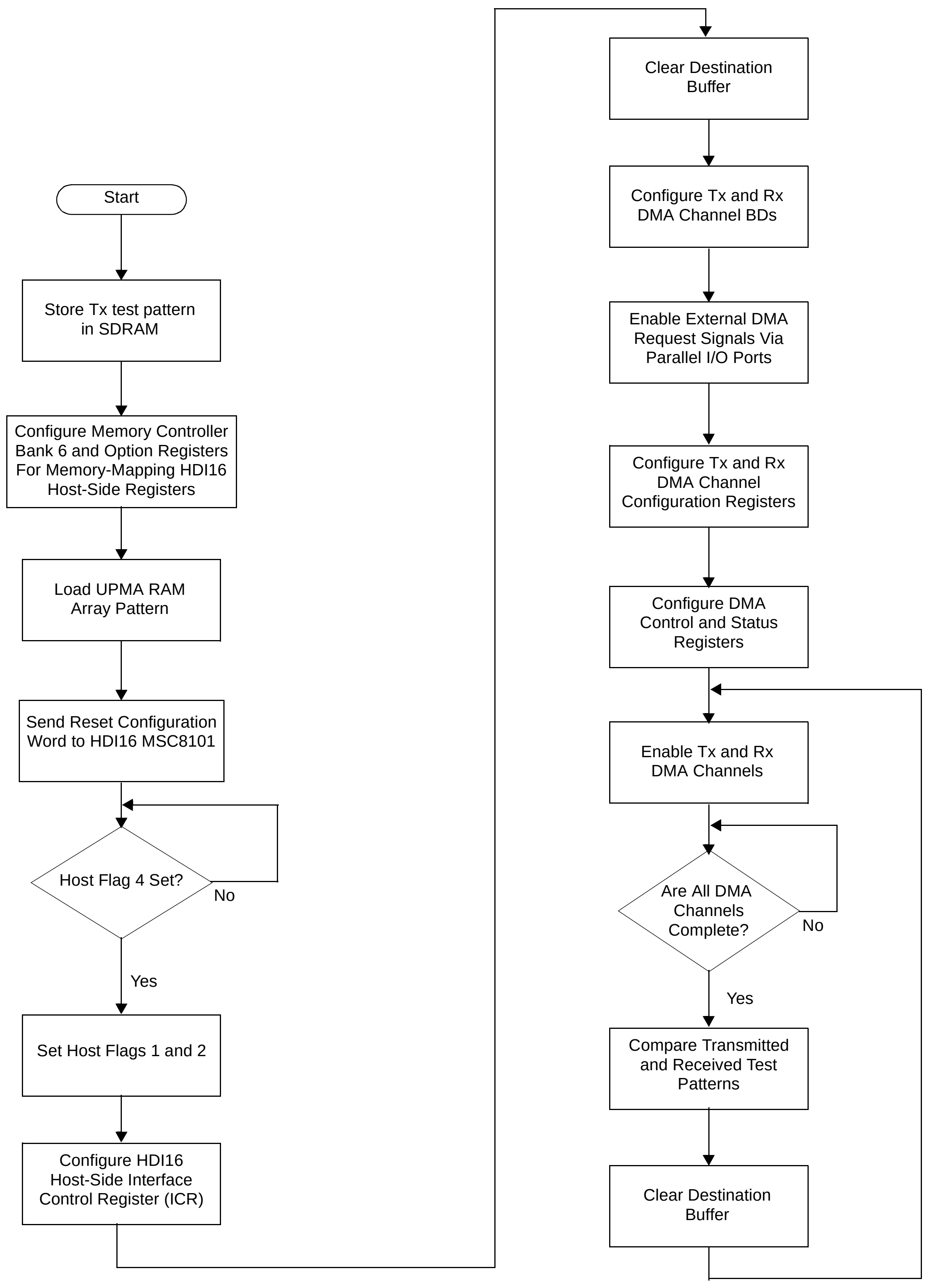

Start
Store Tx test pattern in SDRAM
Configure Memory Controller Bank 6 and Option Registers For Memory-Mapping HDI16 Host-Side Registers
Load UPMA RAM Array Pattern
Send Reset Configuration Word to HDI16 MSC8101
Host Flag 4 Set?
No
Yes
Set Host Flags 1 and 2
Configure HDI16 Host-Side Interface Control Register (ICR)
Clear Destination Buffer
Configure Tx and Rx DMA Channel BDs
Enable External DMA Request Signals Via Parallel I/O Ports
Configure Tx and Rx DMA Channel Configuration Registers
Configure DMA Control and Status Registers
Enable Tx and Rx DMA Channels
Are All DMA Channels Complete?
No
Yes
Compare Transmitted and Received Test Patterns
Clear Destination Buffer

Figure 9. Host MSC8101 Software Flow

Table 6 and **Table 7** present the register settings for the HDI16 MSC8101 device and the host MSC8101 device. Unless indicated otherwise, all registers are described in detail in the *MSC8101 Reference Manual*, so most of the chapter/section references in the first column of the table are to this manual.

Table 6. HDI16 MSC8101 Register Settings

Register	Bits	Setting	Description
MSC8101ADS BCSR (0x40000000) *MSC8101 Application Development System User's Manual*, section on "Board Control and Status Registers" in the chapter on Support Information	BCSR0/1	1	Select Host Request mode
HDI16 Host Port Control Register (HPCR) (0x0081) "Core-Side Programming Model" in the chapter on the Host Interface (HDI16)	8 15	1 1	Enable HDI16 peripheral. Host address 0x4 is used as host DMA transfer acknowledge input.
HDI16 Host Control Register (HCR) (0x8000 | 0x0000) "Core-Side Programming Model" in the chapter on the Host Interface (HDI16)	15	1 | 0	Host Flag 4 on or off
PIC Interrupt Pending Register B (IPRB) (0x0004) "Interrupts Programming Model" in the chapter on the Interrupt Scheme	13	1	Indicates that an IRQ18 interrupt is pending. A write clears the interrupt pending.
Edge/Level-Triggered Interrupt Priority Register E (ELIRE) (0x0300) "Interrupts Programming Model" in the chapter on the Interrupt Scheme	4 5–7	1 011	IRQ18 – Level triggered Interrupt Priority Level 3
DMA Channel 0 – BD_ADDR (0x02000000 + Message Offset) "DMA Programming Model" in the chapter on DMA			Internal SRAM address where the next Rx buffer is to be stored
DMA Channel 0 – BD_SIZE (0x20) "DMA Programming Model" in the chapter on DMA			Size of the Rx buffer
DMA Channel 0 – BD_ATTR (0x80000000) "DMA Programming Model" in the chapter on DMA	0 22–24 27	1 000 0	Interrupt on completion 64-bit maximum transfer size Write transaction
DMA Channel 0 – BD_BSIZE (0) "DMA Programming Model" in the chapter on DMA			Base size of the buffer
DMA Channel 0 – DCHCR (0x80004005) Section on the "DMA Programming Model" in the chapter on DMA	0 1 10–15 17 19–23 28–31	1 0 000000 1 00000 0101	Channel is enabled Local bus Buffer Descriptor 0 Flyby mode HDI16 read request Priority 5
DMA Channel 1 – BD_ADDR (0x02000000 + Message Offset) "DMA Programming Model" in the chapter on DMA			Internal SRAM address where the next Tx buffer is to be stored
DMA Channel 1 – BD_SIZE (0x20) "DMA Programming Model" in the chapter on DMA			Size of the Tx buffer
DMA Channel 1 – BD_ATTR (0x80000010) "DMA Programming Model" in the chapter on DMA	0 22–24 27	1 000 1	Interrupt on completion 64-bit maximum transfer size Read transaction
DMA Channel 1 – BD_BSIZE (0) "DMA Programming Model" in the chapter on DMA			Base size of the buffer

Table 6. HDI16 MSC8101 Register Settings (Continued)

Register	Bits	Setting	Description
DMA Channel 1 – DCHCR (0x80014105) “DMA Programming Model” in the chapter on DMA	0 1 10–15 17 19–23 28–31	1 0 000001 1 00001 0101	Channel is enabled Local bus Buffer Descriptor 1 Flyby mode HDI16 write request Priority 5
DMA Internal Mask Register (DIMR) (0 0x80000000 0x40000000) “DMA Programming Model” in the chapter on DMA	0 1	0 | 1 0 | 1	Enable or disable channel 0 interrupts Enable or disable channel 1 interrupts

Table 7. Host MSC8101 Register Settings

Register	Bits	Setting	Description
Memory Controller Option Register 6 (OR6) (0XFFF00000) “Memory Controller Programming Model” in the chapter on the Memory Controller	0–16 23	11111111 11110000 0 0	The corresponding address bits are used in the comparison with address pins. The banks support bursts.
Memory Controller Base Register 6 (BR6) (0x30001081) “Memory Controller Programming Model” in the chapter on the Memory Controller	0–16 19–20 24–26 31	00110000 00000000 0 10 100 1	Bank address 16-bit port size UPM A machine select Valid bank
Machine A Mode Register (MAMR) (0 | 0x10000000) “Memory Controller Programming Model” in the chapter on the Memory Controller	2–3	00 | 01	Normal operation | Write to array
HDI16 Interface Control Register (ICR) (0x0607) “External Host-Side Programming Model” in the chapter on the Host Interface (HDI16)	5 6 13 14 15	0 | 1 0 | 1 1 1 1	Host Flag 0 on or off Host Flag 1 on or off Select HTRQ and HRRQ signals Enable HTRQ generation Enable HRRQ generation
Port C – Port Pin Assignment Register (PPARC) (0x00000280) “Parallel I/O Ports Programming Model” in the chapter on Parallel I/O Ports	22 24	1 1	DMA DREQ 1 DMA DREQ 2
Port C – Port Special Option Register (PSORC) (0x00000280) “Parallel I/O Ports Programming Model” in the chapter on Parallel I/O Ports	22 24	1 1	DMA DREQ 1 DMA DREQ 2
Port C – Port Data Direction Register (PDIRC) (0x00000000) “Parallel I/O Ports Programming Model” in the chapter on Parallel I/O Ports	22 24	0 0	Input Input
DMA Channel 0 – BD_ADDR (0x20000000) “DMA Programming Model” in the chapter on DMA			Address of Tx buffer
DMA Channel 0 – BD_SIZE (0x20) “DMA Programming Model” in the chapter on DMA			Size of the Tx buffer

Table 7. Host MSC8101 Register Settings (Continued)

Register	Bits	Setting	Description
DMA Channel 0 – BD_ATTR (0x40000230) “DMA Programming Model” in the chapter on DMA	0 1 2 4 22–24 26 27	0 1 0 0 100 1 1	No interrupt Cyclic address No continuous buffer Increment address 32-byte maximum transfer size Flush FIFO Read transaction
DMA Channel 0 – BD_BSIZE (0x20) “DMA Programming Model” in the chapter on DMA			Size of the Tx buffer
DMA Channel 0 – DCHCR (0x40C00046 | 0xC0C00046) “DMA Programming Model” in the chapter on DMA	0 1 10–15 17 25 28–31	0 | 1 1 000000 0 1 0110	Channel enabled or disabled 60x-compatible bus Buffer Descriptor 0 Dual access transaction Internal requestor Priority 6
DMA Channel 1 – BD_ADDR (0x30000020) “DMA Programming Model” in the chapter on DMA			Address of memory-mapped HDI16 host-side data registers
DMA Channel 1 – BD_SIZE (0x20) “DMA Programming Model” in the chapter on DMA			Size of the Tx buffer
DMA Channel 1 – BD_ATTR (0x48000220) “DMA Programming Model” in the chapter on DMA	0 1 2 4 22–24 26 27	0 1 0 1 100 1 0	No interrupt Cyclic address No continuous buffer No address Increment 32-byte maximum transfer size Flush FIFO Write transaction
DMA Channel 1 – BD_BSIZE (0x20) “DMA Programming Model” in the chapter on DMA			Size of the Tx buffer
DMA Channel 1 – DCHCR (0x40C10905 | 0xC0C10905) “DMA Programming Model” in the chapter on DMA	0 1 8 9 10–15 17 19–23 25 28–31	0 | 1 1 1 1 000001 0 01001 0 0101	Channel enabled or disabled 60x-compatible bus DREQ level triggered DREQ active low Buffer Descriptor 1 Dual access transaction External request DREQ 2 No internal requestor Priority 5
DMA Channel 2 – BD_ADDR (0x30000020) “DMA Programming Model” in the chapter on DMA			Address of memory-mapped HDI16 host-side data registers
DMA Channel 2 – BD_SIZE (0x20) “DMA Programming Model” in the chapter on DMA			Size of the Rx buffer
DMA Channel 2 – BD_ATTR (0x48000230) “DMA Programming Model” in the chapter on DMA	0 1 2 4 22–24 26 27	0 1 0 1 100 1 1	No interrupt Cyclic address No continuous buffer No address increment 32-byte maximum transfer size Flush FIFO Read transaction

Table 7. Host MSC8101 Register Settings (Continued)

Register	Bits	Setting	Description
DMA Channel 2 – BD_BSIZE (0x20) Section 15.6.1 of [1]			Size of the Rx buffer
DMA Channel 2 – DCHCR (0x40C20803 | 0xC0C20803) “DMA Programming Model” in the chapter on DMA	0 1 8 9 10-15 17 19-23 25 28-31	0 | 1 1 1 1 000010 0 01000 0 0011	Channel enabled or disabled 60x-compatible bus DREQ level triggered DREQ active low Buffer Descriptor 2 Dual access transaction External request DREQ 1 No internal requestor Priority 3
DMA Channel 3 – BD_ADDR (0x20000000 + Buffer Size) “DMA Programming Model” in the chapter on DMA			Address of Rx buffer
DMA Channel 3 – BD_SIZE (0x20) “DMA Programming Model” in the chapter on DMA			Size of the Rx buffer
DMA Channel 3 – BD_ATTR (0x40000220) “DMA Programming Model” in the chapter on DMA	0 1 2 4 22–24 26 27	0 1 0 0 100 1 0	No interrupt Cyclic address No continuous buffer Increment address 32-byte maximum transfer size Flush FIFO Write transaction
DMA Channel 3 – BD_BSIZE (0x20) “DMA Programming Model” in the chapter on DMA			Size of the Rx buffer
DMA Channel 3 – DCHCR (0x40030044 | 0xC0030044) “DMA Programming Model” in the chapter on DMA	0 1 10–15 17 25 28–31	0 | 1 1 000011 0 1 0100	Channel enabled or disabled 60x-compatible bus Buffer Descriptor 3 Dual access transaction I Internal requestor Priority 4
DMA Internal Mask Register (DIMR) (0) “DMA Programming Model” in the chapter on DMA			Disable all DMA complete interrupts

HOW TO REACH US:

USA/EUROPE/LOCATIONS NOT LISTED:

Motorola Literature Distribution;
P.O. Box 5405, Denver, Colorado 80217
1-303-675-2140 or 1-800-441-2447

JAPAN:

Motorola Japan Ltd.; SPS, Technical Information Center,
3-20-1, Minami-Azabu Minato-ku, Tokyo 106-8573 Japan
81-3-3440-3569

ASIA/PACIFIC:

Motorola Semiconductors H.K. Ltd.; Silicon Harbour
Centre, 2 Dai King Street, Tai Po Industrial Estate,
Tai Po, N.T., Hong Kong
852-26668334

TECHNICAL INFORMATION CENTER:

1-800-521-6274

HOME PAGE:

http://motorola.com/semiconductors

MOTOROLA

AN2244/D