AN2269
Rev. 3, 4/2006

# Interconnecting MPC8260 and MSC8101 ADS Boards Using DMA Transfers Across a 60x Bus

by Scott Smith and Renaud Le Friec

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 MPC8260 PowerQUICC II™ device acting as an integrated communications processor (host) and a single MSC8101 device (called the HDI16 MSC8101) acting as a standard DSP via its 16-bit host parallel interface (HDI16). The host MPC8260 accesses the HDI16 MSC8101 host interface registers via a memory mapping on its own 60x-compatible bus.

**CONTENTS**


| | | |
|---|---|---|
| **1** | Device Overview | 2 |
| **1.1** | Host MPC8260 Device | 2 |
| **1.2** | MSC8101 Device | 3 |
| **1.3** | Device Inter-Operation | 3 |
| **2** | System Bus–HDI16 Host Interface | 4 |
| **2.1** | Host Memory Controller | 4 |
| **2.2** | HDI16 Host Interface | 4 |
| **2.3** | Physical Interconnections | 6 |
| **2.4** | Host DMA Transfers | 7 |
| **3** | HDI16 Device Configuration, Synchronization, and Set-Up | 8 |
| **3.1** | Device Synchronization | 9 |
| **3.2** | DMA Set-Up | 10 |
| **4** | Host Device Configuration | 11 |
| **4.1** | Host Memory Controller | 12 |
| **4.2** | Host HDI16 Registers | 13 |
| **4.3** | DMA Configuration | 13 |
| **4.4** | Host I/O Ports | 15 |
| **4.5** | Start IDMA Loop | 16 |
| **5** | Physical ADS Settings | 16 |
| **6** | Source Code Files, Software Flow, and Registers | 17 |

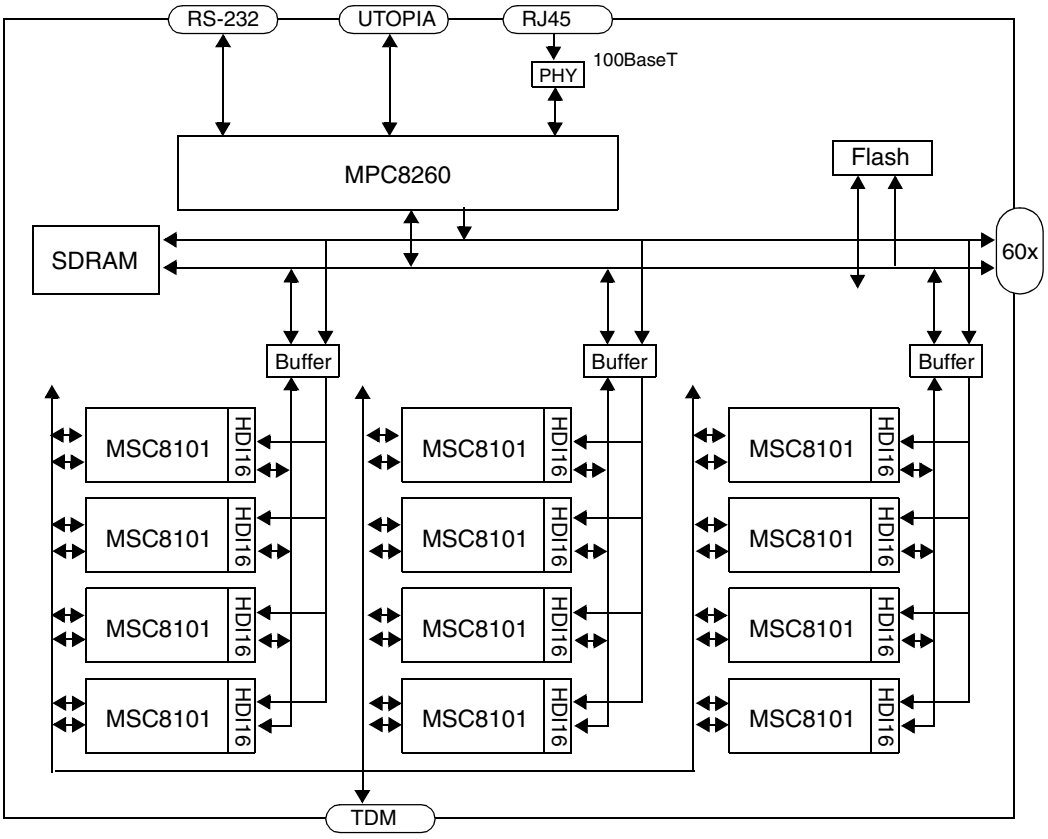



**Figure 1.** Controller to Multiple-HDI16 DSP Architecture

# 1 Device Overview

This section provides a high-level view of the MPC8260 and MSC8101 devices and then describes their inter-operation.

## 1.1 Host MPC8260 Device

The MPC8260 device is a versatile communications processor that integrates a high-performance RISC microprocessor, a very flexible system integration unit (SIU), and many communications peripheral controllers for use in a variety of applications, particularly in communications and networking systems. The core is an embedded variant of the MPC603e microprocessor with 16 KB of instruction cache and 16 KB of data cache and no floating-point unit (FPU). The SIU consists of a flexible memory controller that interfaces to almost any user-defined memory system and many other peripherals making this device a complete system on a chip. The communications processor module (CPM) includes all the peripherals found in the MPC860, with the addition of three high-performance communication channels that support new emerging protocols (for example, 155-Mbps ATM and Fast Ethernet). The MPC8260 has dedicated hardware that can handle up to 256 full-duplex, time-division-multiplexed logical channels. Two key MPC8260 modules that are crucial to the application discussed in this document are as follows:

- *Memory controller*. Controls a maximum of twelve memory banks shared by a high-performance SDRAM machine, a general-purpose chip-select machine (GPCM), and three 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. This flexible memory controller allows the implementation of memory systems with very specific timing requirements.

- *IDMA Emulation*. The Communications Processor Module (CPM) can be configured to provide general-purpose DMA functionality through the SDMA channel. Four general-purpose independent DMA (IDMA) channels are supported. In this special emulation mode, the user can specify any memory-to-memory or peripheral-to/from-memory transfers as if using dedicated DMA hardware.

## 1.2 MSC8101 Device

The 16-bit Freescale 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 CPM, a 64-bit 60x-compatible bus, an SIU, and a 16-channel DMA controller. With its four-ALU core, the MSC8101 executes 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. Two key MSC8101 modules 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.
- *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/synchronous transfers on the bus and give a varying bus bandwidth per channel.

## 1.3 Device Inter-Operation

The host MPC8260 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 MPC8260 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 MPC8260 memory controller and IDMA module as well as the MSC8101 HDI16 host interface and DMA controller.[1]

1. For details on these MPC8260 modules, consult the *MPC8260 PowerQUICC II User's Manual* For details on these MSC8101 modules, consult the *MSC8101 Reference Manual* For details on programming the HDI16 port and the DMA controller, consult the *MSC8101 User's Guide*.

# 2 System Bus–HDI16 Host Interface

To understand the function of the parallel control interface between the host MPC8260 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 SDRAM machine provides an interface to synchronous DRAMs, using SDRAM pipelining, bank interleaving, and back-to-back page mode to achieve the highest performance. The GPCM provides interfacing for simpler, lower-performance memory resources and memory-mapped devices. The GPCM has inherently lower performance because it does not support bursting. GPCM-controlled banks are used primarily for boot loading and access to low-performance memory-mapped peripherals. The UPM supports address multiplexing of the external bus, refresh timers, and generation of programmable control signals for row and column address strobes to allow for a glueless interface to DRAMs, burstable SRAMs, and almost any kind of peripheral. The refresh timers allow refresh cycles to be initiated. The UPM can generate different timing patterns for the control signals that govern a memory device. These patterns define how the external control signals behave during a read, write, burst-read, or burst- write access request. Refresh timers are also available to generate user-defined refresh cycles periodically. We selected the UPM as the most suitable interface for the application discussed in this document.

## 2.2 HDI16 Host Interface

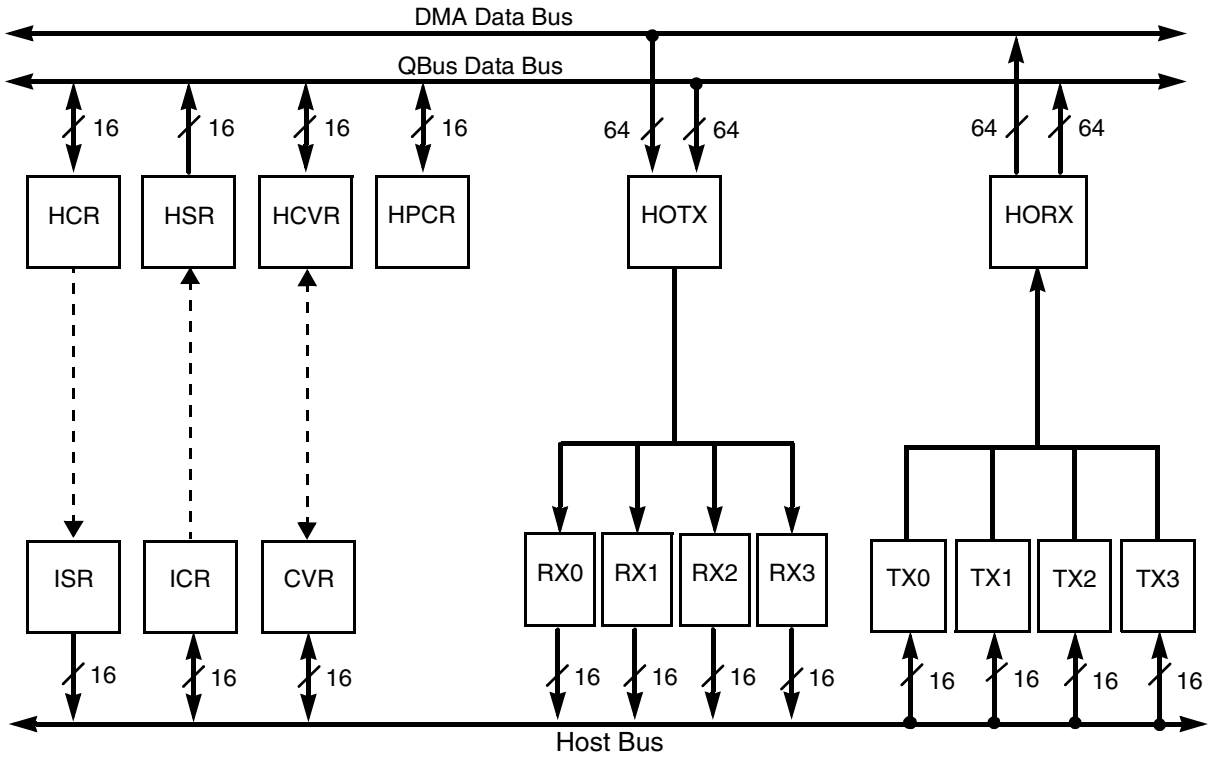
The HDI16 host interface has two sets of 16-bit wide registers, one set visible only within the MSC8101 and the other set visible only to the external host processor (see **Figure 2**). All HDI16 registers are mapped directly onto the MSC8101 QBus, and the transmit and receive FIFOs are mapped onto the 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.

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 MPC8260 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 MPC8260 UPM-controlled signals. The UPM-defined interface can be used with any of the host MPC8260's twelve chip selects to give the 16-bit port size and strobe generation matching that of the HDI16 MSC8101 device's host interface.

### 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 | RX30x 4 |
| TX[0–3] | Transmit Data Registers | TX0 0x7 | TX1 0x6 | TX2 0x5 | TX3 0x4 |

**Figure 2.** MSC8101 HDI16 Port Registers

## 2.3 Physical Interconnections

**Table 1** shows the physical interconnections between the host MPC8260ADS 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 MPC8260 and MSC8101 Application Development System (ADS) user's manual.

**Table 1.** ADS Physical Interconnects

| Host MPC8260 Side | | | | HDI16 MSC8101 Side | | |
|---|---|---|---|---|---|---|
| P16 Pin No. | P4 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) |
| | D31 | IDMA2 | DREQ–IDMA Request 2 (PC1) | 27 | HRRQ/HACK | Receive Host Request OP |
| | D32 | IDMA1 | DREQ–IDMA Request 1 (PC0) | 28 | HTRQ/HREQ | Transmit Host Request OP |
| D10 | | EXPGPL3 | UPM GPL Line 3 | 29 | HRW | Host Read/Write |
| D12 | | EXPGPL5 | UPM GPL Line 5 | 30 | HDS | Host Data Strobe |
| B[1–3] C1, C3, C6, C13 D[1–3] D[6, 13] D[16–D32] | C31, C32 | 0V | Ground | 1, 2, 19, 20, 35, 36 | 0V | Ground |

**Table 1.** ADS Physical Interconnects (Continued)

| Host MPC8260 Side | | | | HDI16 MSC8101 Side | | |
|---|---|---|---|---|---|---|
| P16 Pin No. | P4 Pin No. | System + CPM Edge Connector | | P4 = H0 Pin No. | HDI16 Connector | |
| | | Signal Name | Signal Description | | Signal Name | Signal Description |
| | | | | | | |

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

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 MPC8260 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 MPC8260 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 (IDMAx:DREQ) connect to the Service Request signals (HRRQ and HTRQ) on the HDI16.
- *Address Lines*. The Address lines between the host MPC8260 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 ADS boards, it is highly recommended that all grounds be common and connected together.

## 2.4 Host DMA Transfers

To prevent the host MPC8260 device from dedicating large amounts of 603e 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 603e 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 (IDMAx:DREQ) 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.

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

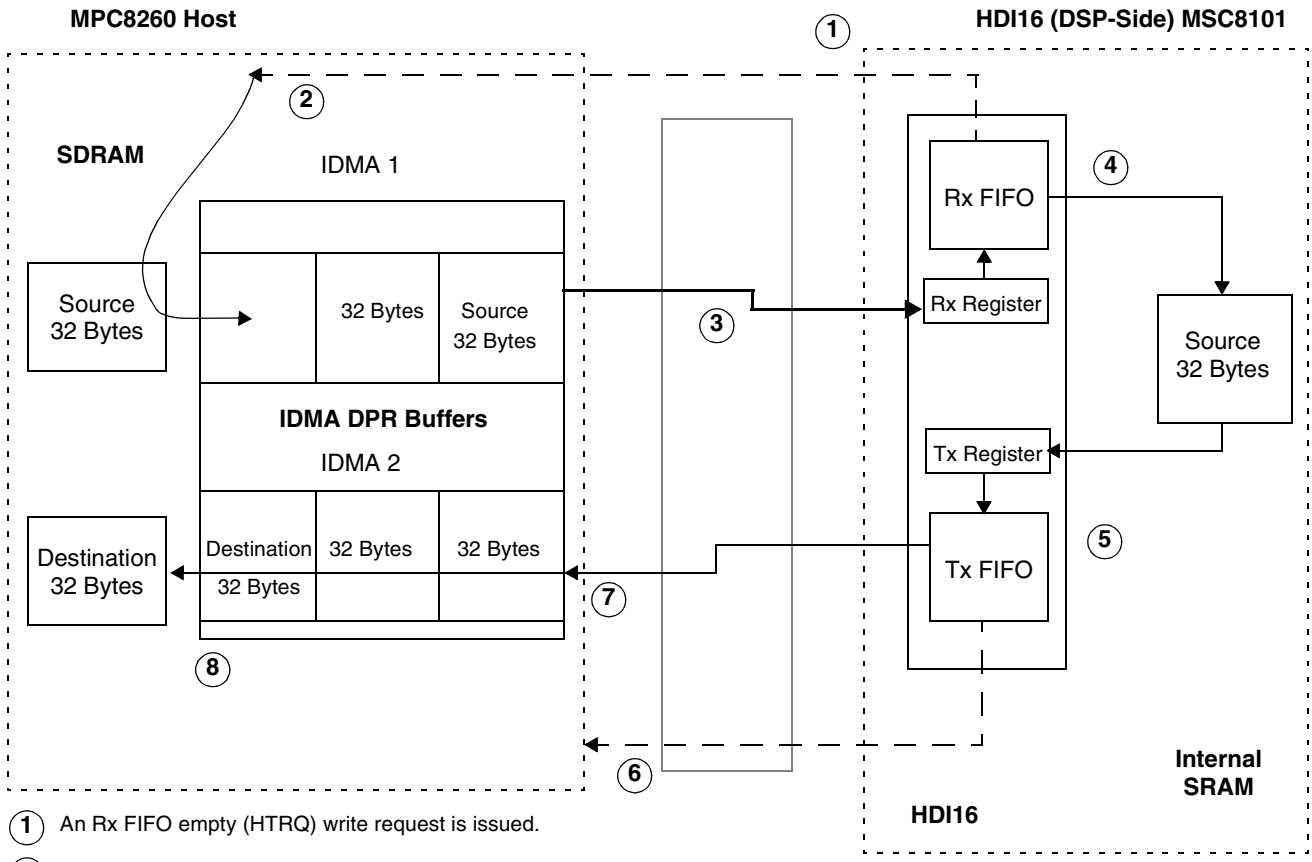
- The host 60x A31 signal is not required because the HDI16 registers are 16-bit word addressed.
- We have also disposed of the 60x A27 and A28 signals so that the host-side transmit and receive registers wrap around the same four 16-bit word addresses.

  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 MPC8260 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 11, describes how to configure the host device. The HDI16 MSC8101 is configured to enable the HDI16 host interface for communications with an external host MPC8260 as follows:

1. Configure the memory controller to map the HDI16 registers into memory.
2. Synchronize the HDI16 MSC8101 and host MPC8260 devices.
3. Set up and enable the DMA receive engine.




1. An Rx FIFO empty (HTRQ) write request is issued.
2. IDMA1:DREQ assertion: DMA write request triggers transfer.
3. When 8 bytes are in the DPR buffer and HTRQ is asserted, the IDMA 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 the Tx FIFO.
6. A Tx FIFO not empty (HRRQ) DMA read request is issued.
7. IDMA2:DREQ assertion triggers a transfer.
8. When 32 bytes are in the DPR buffer, the IDMA transfers data to SDRAM.

**Figure 3.** DMA Transfer Interaction Between the Host MPC8260 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 MPC8260 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 MPC8260 software. Consequently, the HDI16 MSC8101 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.[2] 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.

2. 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*.

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 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 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**.

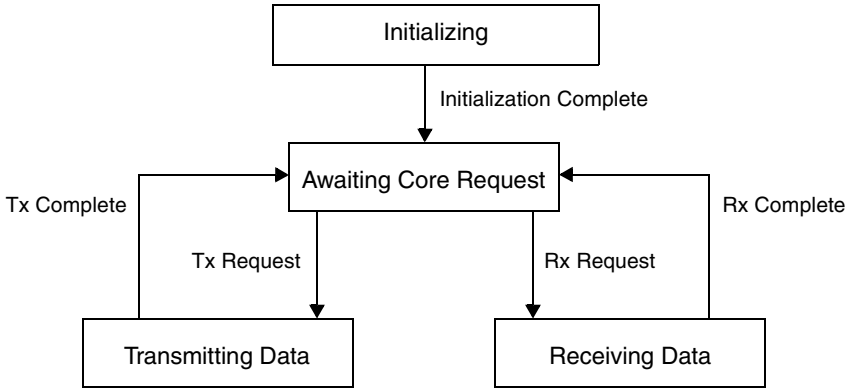



**Figure 4.** Simple Representation of Host States

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 IDMA registers and channel buffer descriptors.
4. Configure the host I/O ports to enable DMA request lines.

## 4.1 Host Memory Controller

The MPC8260 memory controller maps 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 MPC8260ADS 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. 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. **Table 2** defines the hexadecimal values loaded into the UPM RAM Array to obtain the required HDI16 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** | 0xFFFFFC00 | 0x0FFFFC80 | 0x8FFF2C00 | 0x0FFF3880 | 0xFFFFFFFF | 0xFFFFFFFF |
| **offset + 1** | 0x0FFFFC00 | 0x0FFFF000 | 0x0FFF3C00 | 0x0FFF3000 | 0xFFFFFFFF | 0xFFFFFFFF |
| **offset + 2** | 0x0FFFF000 | 0x0FFFF000 | 0x0FFF3000 | 0x0FFF3000 | 0xFFFFFFFF | 0xFFFFFFFF |
| **offset + 3** | 0x0FFFF000 | 0x0FFFF008 | 0x0FFF3404 | 0x0FFF3000 | 0xFFFFFFFF | 0xFFFFFFFF |
| **offset + 4** | 0x0FFFF000 | 0x0FFFFC84 | 0x3FFF3C01 | 0x0FFF348C | 0xFFFFFFFF | |
| **offset + 5** | 0x0FFFF004 | 0x0FFFFC01 | 0xFFFFFFFF | 0xFFFFFC01 | 0xFFFFFFFF | |
| **offset + 6** | 0x0FFFF000 | 0xFFFFFFFF | 0xFFFFFFFF | 0xFFFFFFFF | 0xFFFFFFFF | |

**Table 2.** UPM RAM Array Values (Continued)

| Cycle Type | Single Read | Burst Read | Single Write | Burst Write | Refresh | Exception |
|---|---|---|---|---|---|---|
| **UPM Offset** | **0** | **0x8** | **0x18** | **0x20** | **0x30** | **0x3C** |
| **offset + 7** | 0x3FFFFC01 | 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 | | |

## 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 MPC8260:

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 as 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 DMA Configuration

There are two IDMA channels, one that transfers data from SDRAM to the host interface and another that transfers data from the host interface to SDRAM (IDMA1 and IDMA2, respectively).

### 4.3.1 Buffer Descriptors

The buffer descriptor (BD) associated with IDMA Channel 1 (host Tx) is configured with the following parameters:

- *Valid*. The BD contains valid data for the transfer to be processed.
- *Wrap*. This is the last BD in the BD table, wrap back to the BD at the base of the table once the current buffer is processed. Our application uses only one BD, so it wraps back to itself.
- *Interrupt*. Once the buffer is processed, set a bit in the IDSR register (which we could use to generate a core interrupt, if desired).
- *Last*. This is the last buffer in the chain; terminate the current IDMA transfer after it is processed.
- *Continuous mode*. Do not clear the Valid bit after this buffer is processed so that the data within it is still valid when we issue the START_IDMA command again.
- Destination byte ordering is Big Endian.
- Destination address is on the 60x-compatible bus.
- Source byte ordering is Big Endian.
- Source address is on the local bus.
- Data length is 32-bytes (one burst).
- *Source buffer pointer*. Points to 32-byte aligned SDRAM location where the transmit pattern is stored.
- *Destination Buffer Pointer*. Points to 32-byte aligned location of the HDI16 Tx 0 Host Register.

The BD associated with IDMA Channel 2 (Host Rx) is configured with the following parameters:

- *Valid*. The BD contains valid data for transfer to be processed.
- *Wrap*. This is the last BD in the BD table; wrap back to the BD at the base of the table once the current one is processed. Our application uses only one BD, so it wraps back to itself.
- *Interrupt*. Once the buffer is processed, set a bit in the IDSR register (which we can use to generate a core interrupt, if desired).
- *Last*. This is the last buffer in the chain; terminate the current IDMA transfer after it is processed.
- *Continuous mode*. Do not clear the Valid bit after this buffer is processed so that the data within it is still valid when we issue the START_IDMA command again.
- Destination byte ordering is Big Endian.
- Destination address is on the local bus.
- Source byte ordering is Big Endian.
- Source address is on the 60x bus.
- Data length is 32 bytes (one burst).
- *Source buffer pointer*. Points to 32-byte aligned location of the HDI16 Rx 0 Host Register.
- *Destination buffer pointer*. Points to 32-byte aligned location in SDRAM at which to store the received pattern.

These BD settings enable 32-byte burst and 64-bit transfers to be performed continuously by issuing the START_IDMA command associated with a particular IDMA channel, without updating the BD parameters.

### 4.3.2 DPR Parameters

IDMA Channel 1 (host Tx) DPR parameters are configured as follows:

- *Dual Address mode*. The data read from the source address is stored in DPR and then transferred from DPR to the destination; that is, it is a two-step process. The interim storage location in DPR is specified by DPR_BUF.
- *Set SINC*. Source address pointer is incremented by the number of bytes transferred in the source read transaction.
- *Set DINC*: Destination address pointer is incremented by the number of bytes transferred in the source read transaction.
- *ERM*: Trigger IDMA upon DREQ assertion
- *S/D*: Read from memory, write to memory (the HDI16 appears as memory to the host).
- *SS_MAX*: Maximum transfer size is 32 bytes.
- *STS*: Source transfer size is 32 bytes.
- *DTS*: Destination transfer size is 8 bytes.

IDMA Channel 2 (host Rx) DPR parameters are configured as follows:

- *Dual Address mode*. The data read from the source address is stored in DPR and then transferred from DPR to the destination in a two-step process. DPR_BUF specifies the interim storage location in DPR.
- *Set SINC*: Increments the source address pointer by the number of bytes transferred in the source read transaction.
- *Set DINC*. Destination address pointer is incremented by the number of bytes transferred in the source read transaction.
- *ERM*. Trigger IDMA upon DREQ assertion
- *S/D*. Read from memory/write to memory (the HDI16 interface appears as memory to the host).
- *SS_MAX*. Maximum transfer size is 32 bytes.
- *STS*. Source transfer size is 8 bytes.
- *DTS*. Destination transfer size is 32 bytes.

## 4.4 Host I/O Ports

The registers for the parallel I/O ports are configured to allow the HRRQ and HTRQ signals to reach the DMA controller via the DMA Request lines, IDMA1:DREQ and IDMA2:DREQ, respectively, which are enabled on Port C using the following registers:[3]

- Port Pin Assignment Register C (PPARC): bits PC0 and PC1 are set, and all other bits are cleared.
- Port Data Direction Register C (PDIRC): all bits are cleared.
- Port Special Option Register C (PSORC): all 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.

3. For details on these registers, consult the Parallel I/O Ports chapter of the *MSC8101 Reference Manual*.

## 4.5 Start IDMA Loop

Once the memory controller, IDMA BDs, DPR parameters, and IO ports are initialized, the system enters an infinite *while* loop that constantly performs the following steps:

1. Issue the IDMA Channel 1 (host 32-byte Tx) start command.
2. Wait for IDMA Channel 1 to complete by checking the IDMA1 BD Complete (BC) bit.
3. Clear the IDMA1 BD Complete bit.
4. Issue the IDMA Channel 2 (host 32-byte Rx) start command.
5. Wait for IDMA Channel 2 to complete by checking the IDMA2 BD Complete (BC) bit.
6. Clear the IDMA2 BD Complete bit.
7. Compare the 32-byte test pattern transmitted by the host to the MSC8101 with the one received, and record any discrepancies.
8. Increment the first location of the transmit test pattern so that the exact same pattern is not sent every time.

# 5 Physical ADS Settings

**Table 3** defines the default switch configuration for the HDI16 MSC8101 platform.[4]

**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 | |

Our application uses a 16.384 MHz CLKIN crystal on the MSC8101 ADS platform. With the SW9 settings defined previously, MODCLK 40 is selected, yielding a host 60x bus clock speed of ~40 MHz and a DSP core speed of ~200 MHz. Our application uses a 66 MHz CLKIN crystal on the MPC8260ADS platform. **Table 4** defines the default switch configuration for the host MPC8260ADS.

**Table 4.** Host MPC8260ADS Switch Settings

| Switch | Settings | Comments |
|---|---|---|
| DS1 | Off–on–off–on–off–off–on–off | |
| DS2 | On–on–on–on | |
| DS3 | On–on–on | |

4. 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 4.** Host MPC8260ADS Switch Settings

| Switch | Settings | Comments |
|---|---|---|
| Jumpers | Factory defaults | |

# 6 Source Code Files, Software Flow, and Registers

This application was developed using the CodeWarrior™ for StarCore, production release 1.0. The CodeWarrior IDE provides a set of tools for developing software using a GUI. The 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. **Table 5** defines both the host-side and DSP-side source code project files associated with this application note. **Figure 5** details the software flow of the HDI16 MSC8101 program, and **Figure 6** details the software flow of the host MPC8260 program.

**Table 5.** Source Code Project Files

| Module | Filename | Description |
|---|---|---|
| HDI16 MSC8101 | 8101.mcp | 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 MPC8260 | VADS8260_MT.reg | VisionClick configuration file to initialize the SDRAM and BCSR memory banks on the MPC8260ADS. |
| | hdi16.c | Initializes the HDI16, initializes the core DSP, sets up the test transmit pattern and the received test pattern destination address, and passes control to the DMA routine. |
| | upminit.c | Contains functions to map the memory of the HDI16 host interface for the HDI16 MSC8101 device by initializing UPM A. |
| | dma.c | Exercises the DMA functionality using memory-to-memory in Dual-Access mode. |
| | Mpc8260.h | Handles the MPC8260 register memory-mapping. |
| | hdi16.h | Includes common header files, defines the HDI16 memory map, and defines the card_initialize (UPM set-up) function prototype. |
| | hdi16masks.h | Defines bit masks for the HDI16 host registers and a timeout count value for *wait* loops. |
| | dma.h | Defines the do_dma function prototype. |
| | dmatest.h | Defines the DMA buffer, MSC8101ADS, and I/O port-specific constants. |
| | for_diab/sbc82xx.lk | Diab data link editor command file. |
| | for_diab/makefile | Diab makefile. Paths require changing to accommodate the user's environment. |

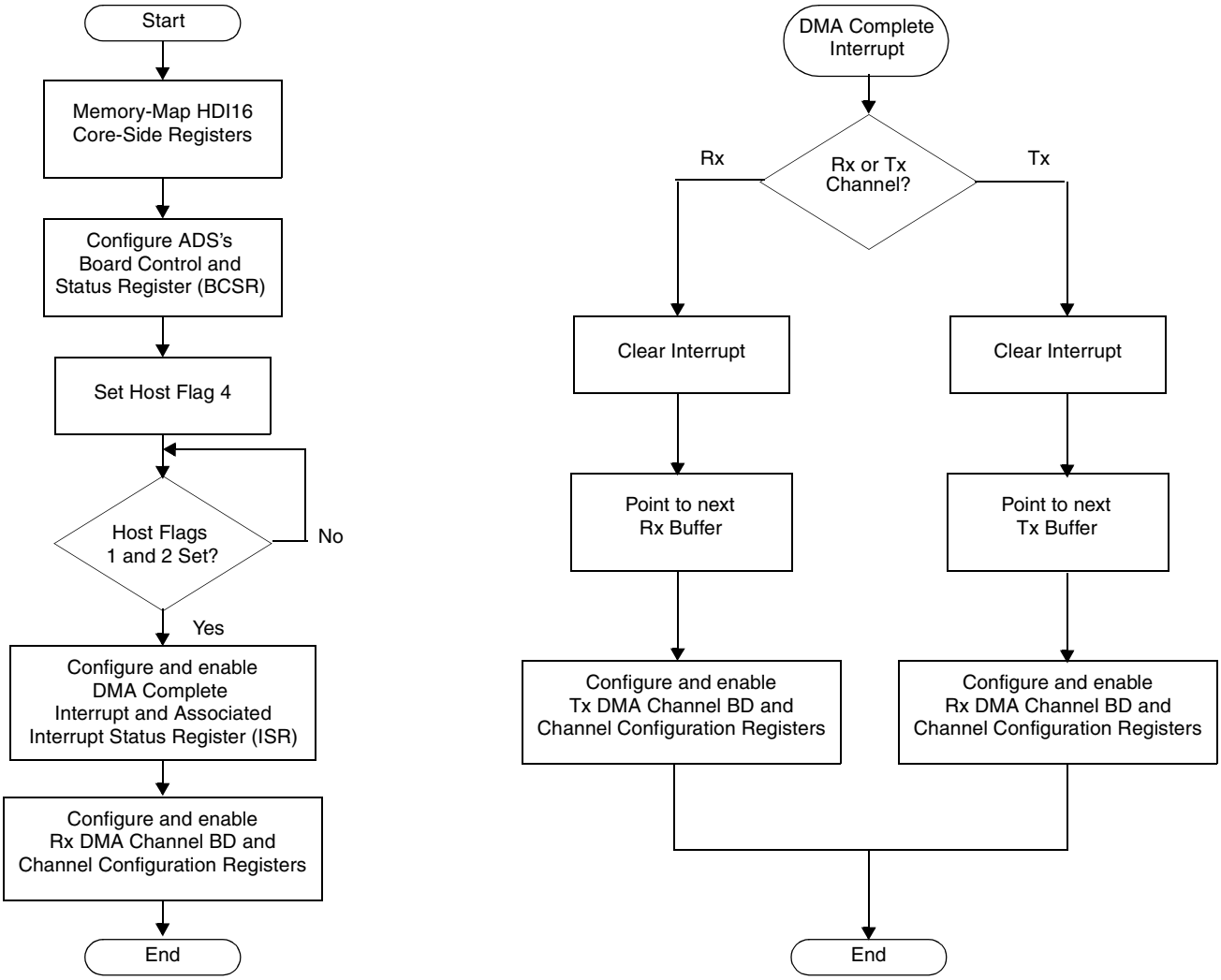



**Figure 5.** HDI16 MSC8101 Software Flow

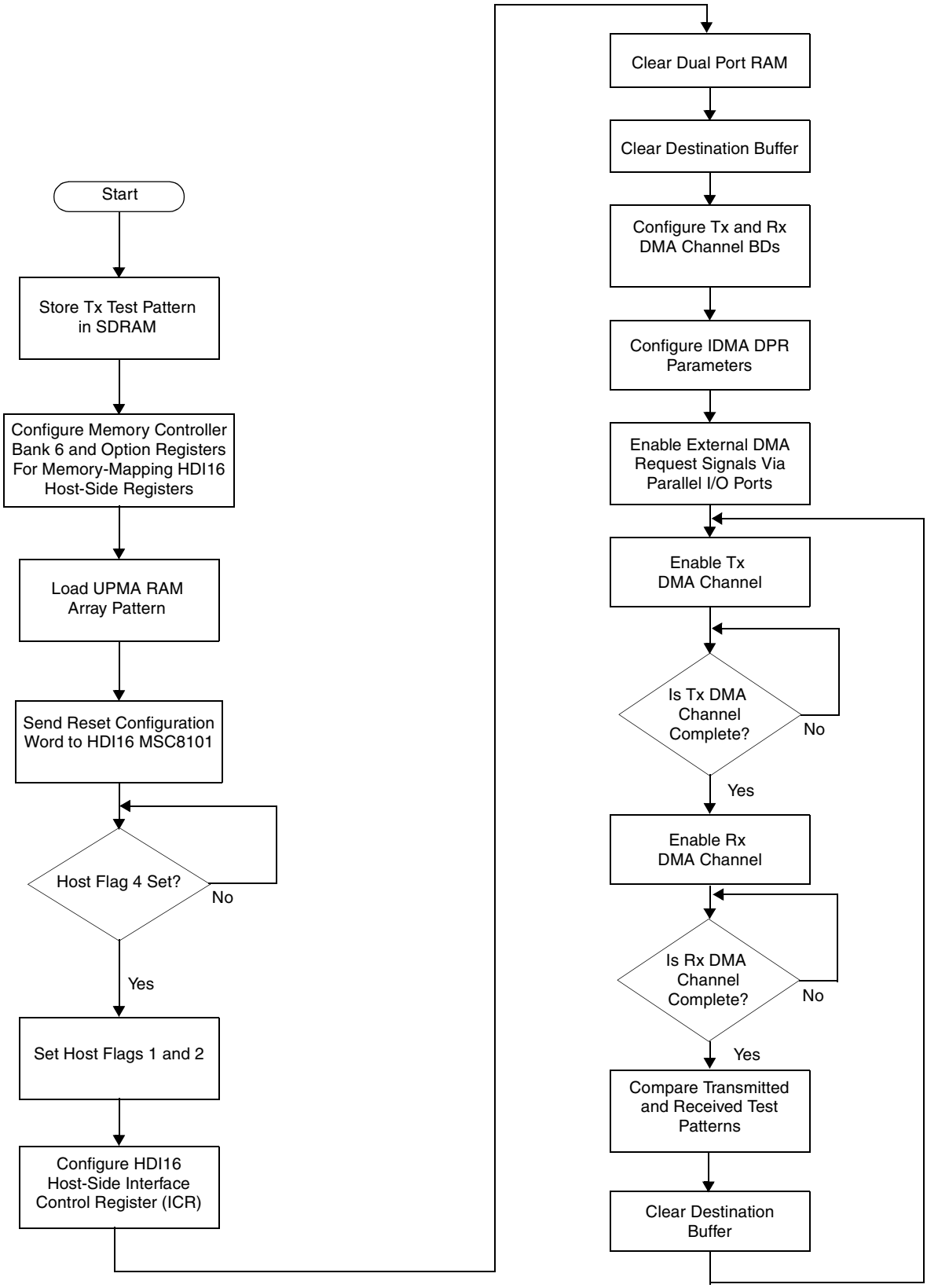



**Figure 6.** Host MPC8260 Software Flow

**Table 6** and **Table 7** present the register settings for the HDI16 MSC8101 device and the host MPC8260 device. Unless indicated otherwise in **Table 6**, 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)<br>*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) (0x4080)<br>"Core-Side Programming Model" in the chapter on the Host Interface (HDI16) | 1<br>8 | 1<br>1 | HTRQ and HRRQ active high. Enable HDI16 peripheral. |
| HDI16 Host Control Register (HCR) (0x8000 \| 0x0000)<br>"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)<br>"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)<br>"Interrupts Programming Model" in the chapter on the Interrupt Scheme | 4<br>5–7 | 1<br>011 | IRQ18 – Level triggered<br>Interrupt Priority Level 3 |
| DMA Channel 0 – BD_ADDR (0x02000000 + Message Offset)<br>"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)<br>"DMA Programming Model" in the chapter on DMA | | | Size of the Rx buffer |
| DMA Channel 0 – BD_ATTR (0x80000000)<br>"DMA Programming Model" in the chapter on DMA | 0<br>22–24<br>27 | 1<br>000<br>0 | Interrupt on completion<br>64-bit maximum transfer size<br>Write transaction |
| DMA Channel 0 – BD_BSIZE (0)<br>"DMA Programming Model" in the chapter on DMA | | | Base size of the buffer |
| DMA Channel 0 – DCHCR (0x80004005)<br>Section on the "DMA Programming Model" in the chapter on DMA | 0<br>1<br>10–15<br>17<br>19–23<br>28–31 | 1<br>0<br>000000<br>1<br>00000<br>0101 | Channel is enabled<br>Local bus<br>Buffer Descriptor 0<br>Flyby mode<br>HDI16 read request<br>Priority 5 |
| DMA Channel 1 – BD_ADDR (0x02000000 + Message Offset)<br>"DMA Programming Model" in the chapter on DMA | | | Internal SRAM address where the next Tx buffer is to be stored |

**Table 6.** HDI16 MSC8101 Register Settings (Continued)

| Register | Bits | Setting | Description |
|---|---|---|---|
| DMA Channel 1 – BD_SIZE (0x20)<br>"DMA Programming Model" in the chapter on DMA | | | Size of the Tx buffer |
| DMA Channel 1 – BD_ATTR (0x80000010)<br>"DMA Programming Model" in the chapter on DMA | 0<br>22–24<br>27 | 1<br>000<br>1 | Interrupt on completion<br>64-bit maximum transfer size<br>Read transaction |
| DMA Channel 1 – BD_BSIZE (0)<br>"DMA Programming Model" in the chapter on DMA | | | Base size of the buffer |
| DMA Channel 1 – DCHCR (0x80014105)<br>"DMA Programming Model" in the chapter on DMA | 0<br>1<br>10–15<br>17<br>19–23<br>28–31 | 1<br>0<br>000001<br>1<br>00001<br>0101 | Channel is enabled<br>Local bus<br>Buffer Descriptor 1<br>Flyby mode<br>HDI16 write request<br>Priority 5 |
| DMA Internal Mask Register (DIMR) (0 0x80000000 0x40000000)<br>"DMA Programming Model" in the chapter on DMA | 0<br>1 | 0 ǀ 1<br>0 ǀ 1 | Enable or disable channel 0 interrupts<br>Enable or disable channel 1 interrupts |

Unless indicated otherwise, all registers in **Table 7** are described in detail in the *MPC8260 PowerQUICC II User's Manual*, so most of the chapter/section references in the first column of the table are to this manual. For details on the IDMA registers, consult chapter 18 of the *MPC8260 PowerQUICC II User's Manual* (SDMA Channels and IDMA Emulation).

**Table 7.** Host MPC8260 Register Settings

| Register | Bits | Setting | Description |
|---|---|---|---|
| Memory Controller Option Register 6 (OR6) (0XFFF00000)<br>Chapter 10, Memory Controller | 0–16<br><br><br>23 | 11111111<br>11110000<br>0<br>0 | The corresponding address bits are used in the comparison with address pins.<br><br>The banks support bursts. |
| Memory Controller Base Register 6 (BR6) (0x30001081)<br>Chapter 10, Memory Controller | 0–16<br><br><br>19–20<br>24–26<br>31 | 00110000<br>00000000<br>0<br>10<br>100<br>1 | Bank address<br><br><br>16-bit port size<br>UPM A machine select<br>Valid bank |
| Machine A Mode Register (MAMR) (0 ǀ 0x10000000)<br>Chapter 10, Memory Controller | 2–3 | 00 ǀ 01 | Normal operation ǀ Write to array |
| HDI16 Interface Control Register (ICR) (0x0607)<br>*MSC8101 Reference Manual,* "External Host-Side Programming Model" in the chapter on the Host Interface (HDI16) | 5<br>6<br>13<br>14<br>15 | 0 ǀ 1<br>0 ǀ 1<br>1<br>1<br>1 | Host Flag 0 on or off<br>Host Flag 1 on or off<br>Select HTRQ and HRRQ signals<br>Enable HTRQ generation<br>Enable HRRQ generation |

**Table 7.** Host MPC8260 Register Settings

| Register | Bits | Setting | Description |
|---|---|---|---|
| Port C – Port Pin Assignment Register (PPARC)<br>(0xC000000)<br>Chapter 35, Parallel I/O Ports | 0<br>1 | 1<br>1 | IDMA1: DREQ<br>IDMA2: DREQ |
| Port C – Port Special Option Register (PSORC) (0x00000000)<br>Chapter 35, Parallel I/O Ports | 0<br>1 | 0<br>0 | IDMA1: DREQ<br>IDMA2: DREQ |
| Port C – Port Data Direction Register (PDIRC) (0x00000000)<br>Chapter 35, Parallel I/O Ports | 0<br>1 | 0<br>0 | IDMA1: DREQ<br>IDMA2: DREQ |
| RISC Controller Configuration Register (RCCR)<br>(0x00C00000)<br>Chapter 13, Communications Processor Module Overview | 8<br>9 | 1<br>1 | DREQ1 is level sensitive<br>DREQ2 is level sensitive |
| IDMA 1 Buffer Descriptor – Attributes (0xBA081200) | 0<br>2<br>3<br>4<br>6<br>12–13<br>15<br>19–20<br>22 | 1<br>1<br>1<br>1<br>1<br>10<br>0<br>10<br>1 | BD Valid<br>Last BD in the table<br>Generate interrupt<br>Last buffer of a chain<br>Continuous mode<br>Big Endian Dest Byte Ordering<br>Destination lies on 60x bus<br>Big Endian Dest Byte Ordering<br>Source lies on local bus |
| IDMA 1 Buffer Descriptor – Data Length (0x0020) | | | Size of the buffer to be transmitted |
| IDMA 1 Buffer Descriptor – Source Data Buffer Pointer (0x4000000) | | | Address of source buffer in SDRAM. |
| IDMA 1 Buffer Descriptor – Destination Data Buffer Pointer (0x30000020) | | | Address of memory-mapped HDI16 data registers. |
| IDMA 1 Parameter RAM – IBASE (0x0000) | | | Offset from DPR Base for the set of IDMA 1 buffer descriptors. |
| IDMA 1 Parameter RAM – IBDPTR (0x0000) | | | Offset for the first BD to be processed. |
| IDMA 1 Parameter RAM – DCM (0x0029) | 10<br>12<br>14–15 | 1<br>1<br>01 | Source increment address<br>Enable external request mode<br>Read from memory, write to peripheral |
| IDMA 1 Parameter RAM – DPR_BUF (0x0800) | | | IDMA 1 transfer buffer base address |
| IDMA 1 Parameter RAM – SS_MAX (0x0020) | | | Maximum transfer size of 32 bytes |
| IDMA 1 Parameter RAM – STS (0x0020) | | | Maximum source (SDRAM) transfer size of 32 bytes |
| IDMA 1 Parameter RAM – DTS (0x0008) | | | Maximum destination (HDI16 register mapping) transfer size of 8 bytes |

**Table 7.** Host MPC8260 Register Settings

| Register | Bits | Setting | Description |
|---|---|---|---|
| IDMA 2 Buffer Descriptor – Attributes (0xBA091000) | 0<br>2<br>3<br>4<br>6<br>12–13<br>15<br>19–20<br>22 | 1<br>1<br>1<br>1<br>1<br>10<br>0<br>10<br>1 | BD valid<br>Last BD in the table<br>Generate interrupt<br>Last buffer of a chain<br>Continuous mode<br>Big Endian Dest Byte Ordering<br>Destination lies on local bus<br>Big Endian Dest Byte Ordering<br>Source lies on 60x bus |
| IDMA 2 Buffer Descriptor – Data Length (0x0020) | | | Size of the buffer to be transmitted. |
| IDMA 2 Buffer Descriptor – Source Data Buffer Pointer (0x3000020) | | | Address of memory-mapped HDI16 data registers |
| IDMA 2 Buffer Descriptor – Destination Data Buffer Pointer (0x4000020) | | | Address of destination buffer in SDRAM |
| IDMA 2 Parameter RAM – IBASE (0x1000) | | | Offset from DPR base for the set of IDMA 2 buffer descriptors |
| IDMA 2 Parameter RAM – IBDPTR (0x1000) | | | Offset for the first BD to be processed |
| IDMA 2 Parameter RAM – DCM (0x001a) | 11<br>12<br>14–15 | 1<br>1<br>10 | Source increment address<br>Enable external request mode<br>Read from peripheral, write to memory |
| IDMA 2 Parameter RAM – DPR_BUF (0x1800) | | | IDMA 2 transfer buffer base address |
| IDMA 2 Parameter RAM – SS_MAX (0x0020) | | | Maximum transfer size of 32 bytes |
| IDMA 2 Parameter RAM – STS (0x0008) | | | Maximum source (HDI16 register mapping) transfer size of 8 bytes |
| IDMA 2 Parameter RAM – DTS (0x0020) | | | Maximum destination (SDRAM) transfer size of 32 bytes |

## *How to Reach Us:*

**Home Page:**
www.freescale.com

**E-mail:**
support@freescale.com

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

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

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

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

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



Document Order No.: AN2269
Rev. 3
4/2006

freescale™
semiconductor