

# TMS320DM646x DMSoC ATA Controller

## User's Guide



Literature Number: SPRUEQ3A

January 2009



<b>Preface</b> .....	<b>6</b>
<b>1 Introduction</b> .....	<b>8</b>
1.1 Purpose of the Peripheral .....	8
1.2 Features .....	8
1.3 Functional Block Diagram.....	9
1.4 Supported Use Cases .....	9
1.5 Industry Standard(s) Compliance.....	10
1.6 Terminology Used in This Document .....	10
<b>2 Architecture</b> .....	<b>10</b>
2.1 Clock Control .....	10
2.2 Signal Descriptions .....	11
2.3 Pin Multiplexing .....	13
2.4 Protocol Description(s).....	13
2.5 General Architecture .....	14
2.6 DMA and PIO Data Transaction Overview .....	19
2.7 Attached Device Reset Considerations.....	23
2.8 Initialization .....	24
2.9 Interrupt Support.....	27
2.10 EDMA Event Support .....	30
2.11 Power Management.....	31
2.12 Emulation Considerations .....	31
<b>3 Use Cases</b> .....	<b>31</b>
3.1 Interfacing to a Standard ATA/ATAPI Device.....	31
3.2 Interfacing to a Standard ATA/ATAPI Device Through a Level-Shifter.....	32
3.3 Interfacing to Compact Flash .....	33
<b>4 Registers</b> .....	<b>35</b>
4.1 Primary IDE Channel DMA Control Register (BMICP) .....	36
4.2 Primary IDE Channel DMA Status Register (BMISP) .....	37
4.3 Primary IDE Channel DMA Descriptor Table Pointer Register (BMIDTP) .....	38
4.4 Primary IDE Channel Timing Register (IDETIMP) .....	39
4.5 IDE Controller Status Register (IDESTAT).....	40
4.6 Ultra-DMA Control Register (UDMACTL) .....	41
4.7 Miscellaneous Control Register (MISCCTL) .....	42
4.8 Task File Register Strobe Timing Register (REGSTB) .....	43
4.9 Task File Register Recovery Timing Register (REGRCVR) .....	44
4.10 Data Register Access PIO Strobe Timing Register (DATSTB).....	45
4.11 Data Register Access PIO Recovery Timing Register (DATRCVR).....	46
4.12 Multiword DMA Strobe Timing Register (DMASTB) .....	47
4.13 Multiword DMA Recovery Timing Register (DMARCVR).....	48
4.14 Ultra-DMA Strobe Timing Register (UDMASTB) .....	49
4.15 Ultra-DMA Ready-to-Pause Timing Register (UDMATRP) .....	50

---

4.16	Ultra-DMA Timing Envelope Register (UDMATENV) .....	51
4.17	Primary IO Ready Timer Configuration Register (IORDYTMP) .....	52
<b>Appendix A</b>	<b>Revision History</b> .....	<b>53</b>

## List of Figures

1	ATA Controller Block Diagram .....	9
2	Physical Region Descriptor (PRD) Table Entry .....	20
3	Primary IDE Channel DMA Control Register (BMICP) .....	36
4	Primary IDE Channel DMA Status Register (BMISP) .....	37
5	Primary IDE Channel DMA Descriptor Table Pointer Register (BMIDTP) .....	38
6	Primary IDE Channel Timing Register (IDETIMP) .....	39
7	IDE Controller Status Register (IDESTAT) .....	40
8	Ultra-DMA Control Register (UDMACTL) .....	41
9	Miscellaneous Control Register (MISCCTL) .....	42
10	Task File Register Strobe Timing Register (REGSTB) .....	43
11	Task File Register Recovery Timing Register (REGRCVR) .....	44
12	Data Register Access PIO Strobe Timing Register (DATSTB) .....	45
13	Data Register Access PIO Recovery Timing Register (DATRCVR) .....	46
14	Multiword DMA Strobe Timing Register (DMASTB) .....	47
15	Multiword DMA Recovery Timing Register (DMARCVR) .....	48
16	Ultra-DMA Strobe Timing Register (UDMASTB) .....	49
17	Ultra-DMA Ready-to-Pause Timing Register (UDMATRP) .....	50
18	Ultra-DMA Timing Envelope Register (UDMATENV) .....	51
19	Primary IO Ready Timer Configuration Register (IORDYTMP) .....	52

## List of Tables

1	Supported ATA Controller Signals .....	11
2	Unsupported ATA Controller Signals .....	13
3	Description of Single Physical Region Descriptor (PRD) Table Entry .....	20
4	ATA Controller Interrupt .....	27
5	Identifying the ATA Controller Interrupt Sources .....	28
6	DMA Driven Interrupt Conditions .....	30
7	ATA/ATAPI Device Interface Connections for a Standard ATA/ATAPI Device .....	31
8	ATA/ATAPI Device Interface Connections for a Standard ATA/ATAPI Device Through a Level-Shifter .....	33
9	ATA/ATAPI Device Interface Connections for a Compact Flash Device .....	34
10	ATA Host Controller Registers .....	35
11	ATA Controller Registers in the Attached Device .....	35
12	Primary IDE Channel DMA Control Register (BMICP) Field Descriptions .....	36
13	Primary IDE Channel DMA Status Register (BMISP) Field Descriptions .....	37
14	Primary IDE Channel DMA Descriptor Table Pointer Register (BMIDTP) Field Descriptions .....	38
15	Primary IDE Channel Timing Register (IDETIMP) Field Descriptions .....	39
16	IDE Controller Status Register (IDESTAT) Field Descriptions .....	40
17	Ultra-DMA Control Register (UDMACTL) Field Descriptions .....	41
18	Miscellaneous Control Register (MISCCTL) Field Descriptions .....	42
19	Task File Register Strobe Timing Register (REGSTB) Field Descriptions .....	43
20	Task File Register Recovery Timing Register (REGRCVR) Field Descriptions .....	44
21	Data Register Access PIO Strobe Timing Register (DATSTB) Field Descriptions .....	45
22	Data Register Access PIO Recovery Timing Register (DATRCVR) Field Descriptions .....	46
23	Multiword DMA Strobe Timing Register (DMASTB) Field Descriptions .....	47
24	Multiword DMA Recovery Timing Register (DMARCVR) Field Descriptions .....	48
25	Ultra-DMA Strobe Timing Register (UDMASTB) Field Descriptions .....	49
26	Ultra-DMA Ready-to-Pause Timing Register (UDMATRP) Field Descriptions .....	50
27	Ultra-DMA Timing Envelope Register (UDMATENV) Field Descriptions .....	51
28	Primary IO Ready Timer Configuration Register (IORDYTMP) Field Descriptions .....	52
A-1	Document Revision History .....	53

## Read This First

---

---

---

### About This Manual

The AT attachment/ATA packet interface (ATA/ATAPI), also known as IDE controller, is the traditional choice of the communication medium between a portable computer (PC) and a hard-disk drive. Ever since its adoption by the industry, it has been the choice of interface between a PC and a common storage medium. This standard interface provides a common attachment interface for system manufacturers, system integrators, software suppliers, and suppliers of intelligent storage devices.

### Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.
- Registers in this document are shown in figures and described in tables.
  - Each register figure shows a rectangle divided into fields that represent the fields of the register. Each field is labeled with its bit name, its beginning and ending bit numbers above, and its read/write properties below. A legend explains the notation used for the properties.
  - Reserved bits in a register figure designate a bit that is used for future device expansion.

### Related Documentation From Texas Instruments

The following documents describe the TMS320DM646x Digital Media System-on-Chip (DMSoC). Copies of these documents are available on the Internet at [www.ti.com](http://www.ti.com). *Tip:* Enter the literature number in the search box provided at [www.ti.com](http://www.ti.com).

The current documentation that describes the DM646x DMSoC, related peripherals, and other technical collateral, is available in the C6000 DSP product folder at: [www.ti.com/c6000](http://www.ti.com/c6000).

**[SPRUEP8](#)** — *TMS320DM646x DMSoC DSP Subsystem Reference Guide*. Describes the digital signal processor (DSP) subsystem in the TMS320DM646x Digital Media System-on-Chip (DMSoC).

**[SPRUEP9](#)** — *TMS320DM646x DMSoC ARM Subsystem Reference Guide*. Describes the ARM subsystem in the TMS320DM646x Digital Media System-on-Chip (DMSoC). The ARM subsystem is designed to give the ARM926EJ-S (ARM9) master control of the device. In general, the ARM is responsible for configuration and control of the device; including the DSP subsystem and a majority of the peripherals and external memories.

**[SPRUEQ0](#)** — *TMS320DM646x DMSoC Peripherals Overview Reference Guide*. Provides an overview and briefly describes the peripherals available on the TMS320DM646x Digital Media System-on-Chip (DMSoC).

**[SPRAA84](#)** — *TMS320C64x to TMS320C64x+ CPU Migration Guide*. Describes migrating from the Texas Instruments TMS320C64x digital signal processor (DSP) to the TMS320C64x+ DSP. The objective of this document is to indicate differences between the two cores. Functionality in the devices that is identical is not included.

**[SPRU732](#)** — *TMS320C64x/C64x+ DSP CPU and Instruction Set Reference Guide*. Describes the CPU architecture, pipeline, instruction set, and interrupts for the TMS320C64x and TMS320C64x+ digital signal processors (DSPs) of the TMS320C6000 DSP family. The C64x/C64x+ DSP generation comprises fixed-point devices in the C6000 DSP platform. The C64x+ DSP is an enhancement of the C64x DSP with added functionality and an expanded instruction set.

[SPRU871](#) — **TMS320C64x+ DSP Megamodule Reference Guide**. Describes the TMS320C64x+ digital signal processor (DSP) megamodule. Included is a discussion on the internal direct memory access (IDMA) controller, the interrupt controller, the power-down controller, memory protection, bandwidth management, and the memory and cache.

# ATA Controller

---

---

---

## 1 Introduction

The AT attachment/ATA packet interface (ATA/ATAPI), also known as IDE controller, is the traditional choice of the communication medium between a portable computer (PC) and a hard-disk drive. Ever since its adoption by the industry, it has been the choice of interface between a PC and a common storage medium. This standard interface provides a common attachment interface for system manufacturers, system integrators, software suppliers, and suppliers of intelligent storage devices.

This document describes the ATA controller on the TMS320DM646x Digital Media System-on-Chip (DMSoC). The ATA controller provides a glueless interface to storage media to be used by video and audio applications for video and audio data storage.

### 1.1 Purpose of the Peripheral

The AT attachment/ATA packet interface (ATA/ATAPI) is an interface that is most commonly used by PCs and portable devices to interface a host processor with data storage or audio devices. The ATA interface debuted in the mid 1980s as an interface between a hard-disk drive and a PC by way of a ribbon cable. Ever since then, other devices, mostly storage, including compact Flash and compact disks have widely adopted the ATA/ATAPI interface, leveraging from its proven capability as the means for connecting to a host processor. These allowed device manufacturers to avoid developing and supporting a proprietary interface that would significantly limit the use of their devices. The ATA/ATAPI interface is popular due to its simplicity, low cost, reliability, compatibility, as well as its wide acceptance and long history of use within the PC industry market.

The DM646x DMSoC supports an onboard ATA/ATAPI host controller module (IDE controller) allowing it to exploit access to a vast majority of available data storage and audio devices. The onboard IDE host controller performs PIO, multiword, and ultra-DMA transactions with ATA and ATAPI compliant devices. Hard-disk drive, compact disk (CD), compact Flash (CF), and DVD are members of ATA/ATAPI-compliant devices that the IDE host controller is destined to interface with. This allows applications like streaming media and digital still cameras the means for easy access to commonly used external storage devices.

The content described here assumes the knowledge of ATA/ATAPI-6 and compact Flash v2.0 specifications and should be used in conjunction with these documents.

### 1.2 Features

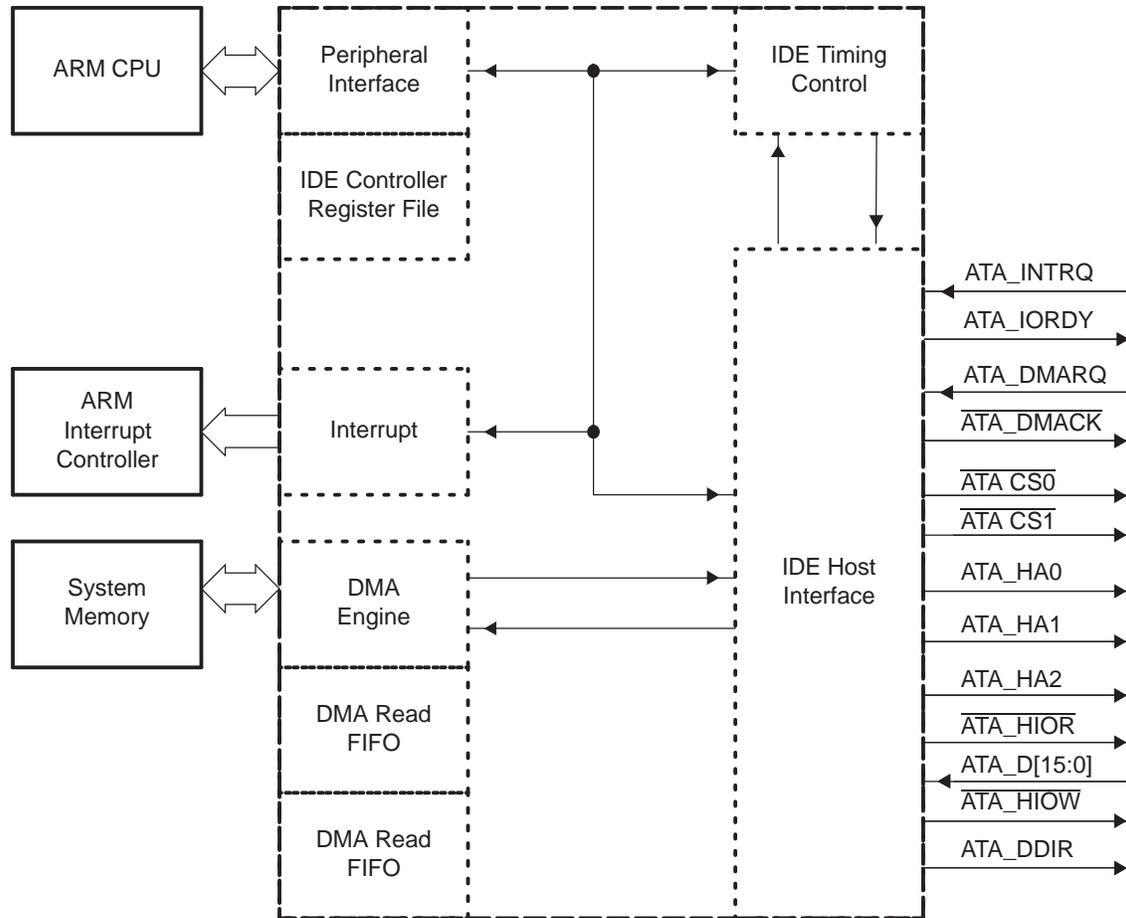
The IDE host controller logic supports PIO, multiword DMA and ultra-DMA (ultra-ATA) modes. The ATA controller has the following features:

- Single channel capable for connecting up to two ATA/ATAPI devices
- Supports interface to compact Flash (CF) configured in True-IDE mode
- Supports PIO modes 0, 1, 2, 3, and 4
- Supports multiword DMA modes 0, 1, and 2
- Supports ultra-DMA modes 0, 1, 2, 3, 4, and 5
- Full scatter gather DMA capability
- Programmable timing parameters provide support of any multiple ATA timing options mode at any processor clock frequency

### 1.3 Functional Block Diagram

The ATA controller is shown in [Figure 1](#).

**Figure 1. ATA Controller Block Diagram**



### 1.4 Supported Use Cases

The IDE controller is commonly used to interface to ATA and ATAPI devices. Devices like hard disk drives are ATA devices. Devices like CDs, compact flash, and DVDs are ATAPI devices. Both ATA and ATAPI devices use the same identical interface and differ partly in protocol (specifically the way command is delivered to the attached device). This difference is transparent to the IDE host controller and meaningful to the driver/firmware that is running on the host supporting the attached device. For ATA devices, all commands and commands parameters are register-driven while ATAPI devices make use of both register-driven and packet-driven commands. Consult the ATA/ATAPI-6 specification for more information.

For information on how to interface to a standard ATA/ATAPI device, see [Section 3.1](#).

For information on how to interface to a standard ATA/ATAPI device with a higher I/O voltage requirement using a level-shifter, see [Section 3.2](#).

For information on how to interface to a compact Flash (CF) device, see [Section 3.3](#).

## 1.5 Industry Standard(s) Compliance

The IDE controller supports the ATA/ATAPI-6 and compact Flash v2.0 specifications. The specific modes of operations (PIO, multiword, and ultra-DMA) depend on the frequency at which the IDE controller operates. For this reason, not all modes supported by the specification and device may be exercised if the clocking frequency to the IDE controller is low. The actual mode supported (subset of the ATA/ATAPI standard) by the device is directly tied to the IDE controller clock sourcing. For more information on the clocking provided to the peripheral, see [Section 2.1](#). In addition to possible reduction of mode support due to low frequency clocking, the processor does not pin out all the signals available in the ATA/ATAPI specification. Additional signals (necessary to support wide range of devices) that are not present within the ATA/ATAPI standard specification are also available. See [Section 2.2](#) for more information on the signals supported on this peripheral.

## 1.6 Terminology Used in This Document

The following is a brief explanation of some terms used in this document:

Term	Meaning
ATA/ATAPI	AT attachment/ATA packet interface
ATA controller	ATA/ATAPI controller peripheral
CF	Compact Flash
device	External ATA/ATAPI device attached to the IDE controller
DMA	DMA within the ATA controller, not the processor EDMA system
host	Processor IDE controller (this peripheral), unless otherwise specified
IDE controller	Synonymous with ATA controller
processor	DM646x Digital Media Processor

## 2 Architecture

This section discusses the architecture of the ATA controller.

### 2.1 Clock Control

The IDE controller uses a single programmable clock for its operation. This clock needs to be reconfigured prior to enabling access to the attached ATA/ATAPI device. The maximum clock frequency supplied to the ATA controller is dependent upon the clock frequency of the processor. See the device specific data manual for more information on the processor voltage and speed characteristics.

SYSClk4 (PLL0 output frequency divided by 6) is the source of the clock to the ATA peripheral. To ensure proper operation, the operating frequency of the ATA controller clock should be chosen in such a way that it is at least twice as fast as the ATA data strobe frequency in order to achieve expected throughput.

## 2.2 Signal Descriptions

### 2.2.1 ATA/ATAPI Interface Signals Supported by the ATA Controller

Table 1 describes the signals supported by the IDE controller interface.

**Table 1. Supported ATA Controller Signals**

Terminal Name	Direction from IDE Controller	Description
$\overline{\text{ATA\_CS0}}$ $\overline{\text{ATA\_CS1}}$	Output	<p>Chip Select Signals 0 and 1</p> <p>These are the chip select signals from the host used to select the Command Block or Control Block registers. <math>\overline{\text{ATA\_CS0}}</math> is asserted during Command Block register accesses. <math>\overline{\text{ATA\_CS1}}</math> is asserted during Control Block register accesses. When <math>\overline{\text{ATA\_DMACK}}</math> is asserted, <math>\overline{\text{ATA\_CS0}}</math> and <math>\overline{\text{ATA\_CS1}}</math> are both asserted and transfers will be 16 bits wide.</p>
$\overline{\text{ATA\_HA0}}$ $\overline{\text{ATA\_HA1}}$ $\overline{\text{ATA\_HA2}}$	Output	<p>Device Address Bits[2:0]</p> <p>This is a 3-bit binary coded address asserted by the host to access a register or data port in the attached device.</p>
$\overline{\text{ATA\_D}}[15:0]$	Input/Output	<p>Host Read/Write Data Bus</p> <p>This is a 16-bit bi-directional data interface between the host and the device. Data transfers are 16-bits wide except for CF devices that implement 8-bit data transfers. In this case, <math>\overline{\text{ATA\_D}}[7:0]</math> will be used for 8-bit register transfers.</p>
$\overline{\text{ATA\_DMARQ}}$	Input	<p>Device DMA Request</p> <p>The device will assert this signal, used for DMA data transfers between host and device, when the device is ready to transfer data to or from the host in any of the DMA modes. For multiword transfers, the direction of data transfer is controlled by <math>\overline{\text{ATA\_HIOR}}</math> and <math>\overline{\text{ATA\_HIOW}}</math>. This signal is used in a handshake manner with <math>\overline{\text{ATA\_DMACK}}</math>, that is, the device will wait until the host asserts <math>\overline{\text{ATA\_DMACK}}</math> before negating <math>\overline{\text{ATA\_DMARQ}}</math>, and re-asserting <math>\overline{\text{ATA\_DMARQ}}</math> if there is more data to transfer. For multiword DMA transfers, the <math>\overline{\text{ATA\_DMARQ}}/\overline{\text{ATA\_DMACK}}</math> handshake is used to provide flow control during the transfer. For ultra-DMA, the <math>\overline{\text{ATA\_DMARQ}}/\overline{\text{ATA\_DMACK}}</math> handshake is used to indicate when the function of interface signals changes.</p> <p>This signal will be released when the device is not selected.</p>
$\overline{\text{ATA\_DMACK}}$	Output	<p>Host DMA Acknowledge</p> <p>This signal is used by the host in response to <math>\overline{\text{ATA\_DMARQ}}</math> to initiate DMA transfers. For multiword DMA transfers, the <math>\overline{\text{ATA\_DMARQ}}/\overline{\text{ATA\_DMACK}}</math> handshake is used to provide flow control during the transfer. For ultra-DMA, the <math>\overline{\text{ATA\_DMARQ}}/\overline{\text{ATA\_DMACK}}</math> handshake is used to indicate when the function of interface signals changes.</p> <p>When <math>\overline{\text{ATA\_DMACK}}</math> is asserted, <math>\overline{\text{ATA\_CS0}}</math> and <math>\overline{\text{ATA\_CS1}}</math> is not asserted and transfers are 16 bits wide.</p>
$\overline{\text{ATA\_IORDY}}$	Input	<p>Device I/O ready during PIO transaction DMA ready during ultra-DMA write DMA strobe during ultra-DMA read</p> <p><math>\overline{\text{ATA\_IORDY}}</math> is negated to extend the host transfer cycle of any host register access (PIO 8-bit) or PIO data access when the device is not ready to respond to a transfer request. If the device requires that the host transfer cycle time be extended, the device will assert the <math>\overline{\text{ATA\_IORDY}}</math> signal. Devices that support PIO modes 3 and above are required by the ATA/ATAPI specification to support <math>\overline{\text{ATA\_IORDY}}</math>.</p> <p>For ultra-DMA data-write transaction, this signal is a flow control signal for data-out bursts. This signal is asserted by the device to indicate to the host that the device is ready to receive Ultra DMA data-out bursts. The device may negate <math>\overline{\text{ATA\_IORDY}}</math> to pause an Ultra DMA data-out burst.</p> <p>For ultra-DMA data-read transaction, this signal is a data-in strobe signal from the device for data-in burst. Both the rising and falling edges of <math>\overline{\text{ATA\_IORDY}}</math> latch the data from <math>\overline{\text{ATA\_D}}[15:0]</math> into the host. The device may stop generating <math>\overline{\text{ATA\_IORDY}}</math> edges to pause an ultra DMA data-in burst.</p> <p>This signal is released when the device is not selected.</p>

**Table 1. Supported ATA Controller Signals (continued)**

Terminal Name	Direction from IDE Controller	Description
ATA_HIOR	Output	<p>PIO Read Transaction Indicator DMA Ready during Ultra-DMA Read DMA Data Strobe during Ultra-DMA Write</p> <p>ATA_HIOR is the strobe signal used by the host to read device registers or data. Data is transferred on the negation of this signal.</p> <p>When performing ultra-DMA data read transaction, this signal is used by the host for flow control during data-in bursts. This signal is asserted by the host to indicate to the device that the host is ready to receive ultra-DMA data-in bursts. The host may negate ATA_HIOR to pause an ultra DMA data-in burst.</p> <p>When performing ultra-DMA data write transaction, the host uses the <math>\overline{\text{ATA\_HIOR}}</math> to strobe the ultra-DMA data-out burst.</p> <p>Both the rising and falling edges of <math>\overline{\text{ATA\_HIOR}}</math> latch the data from ATA_D[15:0] into the device. The host may stop generating ATA_HIOR edges to pause an ultra DMA data-out burst.</p>
ATA_HIOW	Output	<p>PIO Write Transaction Indicator Stop Ultra-DMA Data Read/Write Bursts</p> <p>ATA_HIOW is the strobe signal used by the host to write device registers (PIO 8-bit) or data (PIO 16-bit). Data is transferred on the negation of this signal.</p> <p>The host will negate <math>\overline{\text{ATA\_HIOW}}</math> prior to initiation of an ultra DMA burst. The host will negate <math>\overline{\text{ATA\_HIOW}}</math> before data is transferred in an ultra-DMA burst. Assertion of ATA_HIOW by the host during an ultra-DMA burst signals the termination of the ultra DMA burst.</p>
ATA_INTRQ	Input	<p>Attached Device Interrupt Request</p> <p>This signal is used by the selected device to interrupt the host system when interrupt pending is set. When the nIEN bit is cleared to zero and the device is selected, INTRQ is enabled. When the nIEN bit is set to one or the device is not selected, the INTRQ signal is disabled.</p> <p>When asserted, this signal should be negated by the device within 400 ns of the negation of ATA_HIOR that reads the Status register to clear interrupt pending. When asserted, this signal should be negated by the device within 400 ns of the negation of ATA_HIOW that writes the Command register to clear interrupt pending.</p> <p>When the device is selected by writing to the Device register while interrupt pending is set, ATA_INTRQ should be asserted within 400 ns of the negation of <math>\overline{\text{ATA\_HIOW}}</math> that writes the device register. When the device is deselected by writing to the device register while interrupt pending is set, ATA_INTRQ should be released within 400 ns of the negation of <math>\overline{\text{ATA\_HIOW}}</math> that writes the device register.</p> <p>This signal shares a function with an EMIF signal. For simplicity, in the remainder of this document this signal will be referred to as INTRQ.</p>
ATA_DDIR	Output	<p>External Level Shifter Direction Indicator</p> <p>This signal is used when a 3.3 V or 5.0 V tolerant ATA/ATAPI devices are used to interface with the IDE controller. This signal indicates the direction of the current data transfer.</p> <p>Although this signal provides direction control, it is the responsibility of the user to enable the shifter properly. The processor does not provide an enable control for the level shifter.</p>

### 2.2.2 ATA/ATAPI/Compact Flash Specification Signals Not Supported by This Peripheral

The processor supports all the signals needed to realize the supported modes of operation. [Table 2](#) lists some of the signals that are present within the ATA/ATAPI specification that are not available. In addition, the processor supports additional signals to interface to devices that are 3.3 V and 5.0 V tolerant devices.

**Table 2. Unsupported ATA Controller Signals**

<b>Name</b>	<b>Description</b>
RESET	This signal is used by the IDE controller to perform a hardware reset on the attached ATA/ATAPI device. A GPIO signal can be used for this purpose due to lack of a dedicated ATA reset signal on the processor.
Cable Select (CSEL)	The processor host IDE controller requires the use of the proper type cable based on the maximum mode of operation in which the external device is intended to operate. Specifically, if the external device supports UDMA mode greater than mode2, then it is expected that an 80-conductor ribbon cable be used to connect the IDE controller with the ATA device. The use of a 40-conductor ribbon cable with higher UDMA mode of operation would result in reduced throughput due to the introduction of error from signals propagating on adjacent conductors. This signal is not necessary because an 80-conductor cable must be used with the DM646x.
Cable ID (CBLID)	The Cable ID signal is used by attached devices in order to configure themselves as master or slave without the need for a jumper configuration setting. The host does not need to have a dedicated signal to drive this signal. This task is achieved by tying the appropriate pin (on the header on the processor side) low external to the processor.
True IDE Mode Selector (ATA_SEL)	This signal needs to be driven low prior to power-up in order to configure a Compact Flash in True IDE mode. Since any Compact Flash mounted to work with the processor is only useful in True IDE mode, the state of this signal is tied to ground by the header. Note that CF devices use a 50 pin header that is different from the standard ATA/ATAPI header.

### 2.3 Pin Multiplexing

On the DM646x DMSoC, extensive pin multiplexing is used to accommodate the largest number of peripheral functions in the smallest possible package. Pin multiplexing is controlled using a combination of hardware configuration at device reset and software programmable register settings. Refer to the device-specific data manual to determine how pin multiplexing affects the ATA.

### 2.4 Protocol Description(s)

The IDE controller supports the protocols defined in the ATA/ATAPI-6 and compact Flash 2.0 specifications.

## 2.5 General Architecture

### 2.5.1 Programmable Timing Registers

The IDE controller implements several programmable timing registers that allow users to reprogram the key IDE interface signal timings for the four transfer types: 8-bit task file registers accesses, 16-bit PIO data accesses, multiword DMA transfers, and ultra-DMA transfers. This allows the host controller to effectively be reconfigured to support a wide range of input clock frequencies and any target interface for all transfer types.

The default state for the programmable timing registers logic is disabled; that is, the TIMORIDE bit in the miscellaneous control register (MISCCTL) is 0. The programmable timing registers should be enabled by setting the TIMORIDE bit to 1 for proper operation. The IDE controller may not be able to operate correctly if the programmable timing registers are not enabled.

Four sets of programming timing registers exist to support the four types of IDE transfers. In addition, the MISCCTL upper bits allow for a common control over the write data hold time for three of the four transfers: task file writes, PIO writes, and multiword DMA writes. The write data hold time in MISCCTL must satisfy the worst case requirement of these three modes.

- The task file register strobe timing register (REGSTB) and the task file register recovery timing register (REGRCVR) control the timing parameters for 8-bit accesses of the task file registers.
- The data register PIO strobe timing register (DATSTB) and the data register PIO recovery timing register (DATRCVR) control the timing parameters for PIO data accesses.
- The DMA strobe timing register (DMASTB) and the DMA recovery timing register (DMARCVR) control timings for multiword DMA accesses.
- The ultra DMA strobe timing register (UDMASTB), the ultra DMA ready-to-stop timing register (UDMATRP), and the ultra DMA timing envelope register (UDMATENV) control the timing of ultra DMA accesses.

The timing override registers are programmed with a value indicating the number of clock cycles (minus 1 cycle) that the IDE controller will wait to meet a particular timing parameter. You identify the minimum or maximum value for a timing parameter from the IDE specification, determine the IDE clock frequency, and then calculate the number of clock cycles necessary to meet that timing parameter. For example, if the required IDE controller clock frequency is 99 MHz (10.10 ns period), then to meet a minimum IDE interface timing requirement of 25 ns, 3 clock cycles are required (3 cycles  $\times$  10.10 ns = 30.3 ns). This means that the corresponding timing register that controls the parameter would be programmed with a 2 (3 clock cycles minus 1 cycle). If the IDE controller clock frequency is 50 MHz (20 ns) and a minimum timing requirement of 55 ns is to be met, this would require at least 3 clock cycles (3 cycles  $\times$  20 ns = 60 ns); therefore, the timing register should be programmed with a value of 2.

Note that programming the HWNHLD $n$ P bits in MISCCTL controls the write data hold times used for task file registers writes, PIO data writes, and multiword DMA data writes. Since a single timing programmable parameter is used for all three types of write transfers, the value programmed here will be the largest among the three transfers: PIO-8 bit, PIO-16-bit, and multiword DMA.

### 2.5.1.1 Programming 8-bit Task File Timing Registers

The REGSTB, REGRCVR, and MISCCTL are used in reprogramming the timings for 8-bit task file register accesses. The required IDE timing parameters for 8-bit task file register accesses are defined in the ATA/ATAPI-6 specification.

The REGSTB and REGRCVR can be programmed to match the parameters  $t_0$  (cycle time),  $t_2$  (strobe time), and  $t_{2i}$  (recovery time). The HWNHLD bits in MISCCTL are used to program the hold time for the write data ( $t_4$  parameter).

The REGSTB directly controls the number of clock cycles that the  $\overline{\text{ATA\_HIOR}}$  and  $\overline{\text{ATA\_HIOW}}$  strobes will be asserted during 8-bit task file accesses. This corresponds to the strobe width timing parameter,  $t_2$ .

The REGRCVR defines the number of clock cycles for the recovery time between task file accesses. This corresponds to recovery timing parameter,  $t_{2i}$ .

The sum of both parameters ( $t_2 + t_{2i}$ ) must be equal or greater than the cycle time,  $t_0$ . The HWNHLD $n$ P bits in MISCCTL allow control over the number of clock cycles for the write data hold time during task file writes. With knowledge of the IDE controller clock frequency, you can program the appropriate number of clock cycles to match the timing requirements. Note that the timing registers must be programmed with a value one less than the desired number of cycles, so a value of 0 specifies 1 clock cycle, a value of 1 specifies 2 clock cycles, etc.

[Example 1](#) and [Example 2](#) illustrate how the 8-bit task file timing registers can be programmed.

#### **Example 1. 8-Bit Task File Timing Registers Programming for Mode 0**

Programming task file accesses for mode 0 operation using an IDE controller clock frequency of 66 MHz (15 ns period).

For mode 0 operation,  $t_2$  requires a minimum of 290 ns, this translates to a minimum of 20 clock cycles (20 cycles  $\times$  15 ns = 300 ns). There is no requirement for  $t_{2i}$  in mode 0 operation, but  $t_0$  requires a minimum of 600 ns, or 40 clock cycles. This means REGSTB and REGRCVR can be programmed to any combination equaling 40 (or more) clock cycles, with REGSTB specifying at least 20 clock cycles.

Sample programming values are REGSTB = 13h (20 clock cycles) and REGRCVR = 13h (20 clock cycles), or REGSTB = 1Fh (32 clock cycles) and REGRCVR = 7h (8 clock cycles). In addition, the minimum write data hold time specified is 30 ns, so the HWNHLD $n$ P bits in MISCCTL should be programmed to a value of at least 2 clock cycles. A sample value is HWNHLD $n$ P = 2h (3 clock cycles) providing enough hold time and margin.

#### **Example 2. 8-Bit Task File Timing Registers Programming for Mode 4**

Programming task file accesses for mode 4 operation using an IDE controller clock frequency of 99 MHz (10.10 ns period).

For mode 4 operation,  $t_2$  requires a minimum of 70 ns, this translates to a minimum of 7 clock cycles (7 cycles  $\times$  10.10 ns = 70.70 ns).  $t_{2i}$  is defined to be at least 25 ns, this translates to a minimum of 3 clock cycles (3 cycles  $\times$  10.10 ns = 30.30 ns). The minimum cycle time  $t_0$  is 120 ns, or 12 clock cycles. This means REGSTB and REGRCVR can be programmed to any combination equaling 12 (or more) clock cycles, with REGSTB specifying at least 7 cycles and REGRCVR specifying at least 3 cycles.

Sample programming values are REGSTB = 7h (8 clock cycles) and REGRCVR = 3h (4 clock cycles), or REGSTB = 8h (9 clock cycles) and REGRCVR = 2h (3 clock cycles). In addition, the minimum write data hold time specified is 10 ns, so the HWNHLD $n$ P bits in MISCCTL should be programmed to a value of at least 1 clock cycle. A sample value is HWNHLD $n$ P = 1h (2 clock cycles) providing enough hold time and margin.

### 2.5.1.2 Programming Data Register Timing Register Access

The DATSTB and DATRCVR are used in reprogramming the timings for 16-bit PIO data accesses. The required IDE timing parameters for 8-bit task file register accesses are defined in the ATA/ATAPI-6 specification.

The DATSTB and DATRCVR can be programmed to match the parameters  $t_0$  (cycle time),  $t_2$  (strobe time), and  $t_{2i}$  (recovery time). The HWNHLD $n$ P bits in MISCCTL are used to program the hold time for the write data ( $t_4$  parameter).

The DATSTB directly controls the number of clock cycles that the  $\overline{\text{ATA\_HIOR}}$  and  $\overline{\text{ATA\_HIOW}}$  strobes will be asserted during the PIO data access. This corresponds to the strobe width timing parameter,  $t_2$ .

The DATRCVR defines the number of clock cycles for the recovery (de-assert) time for the PIO data access. This corresponds to recovery timing parameter,  $t_{2i}$ .

The sum of both parameters ( $t_2 + t_{2i}$ ) must be equal or greater than the cycle time,  $t_0$ . The HWNHLD $n$ P bits in MISCCTL allow control over the number of clock cycles for the write data hold time during PIO data writes. With knowledge of the IDE controller clock frequency, you can program the appropriate number of clock cycles to match the timing requirements. Note that the timing registers must be programmed with a value one less than the desired number of cycles, so a value of 0 specifies 1 clock cycle, a value of 1 specifies 2 clock cycles, etc.

[Example 3](#) and [Example 4](#) illustrate how the data register access timing registers can be programmed.

#### **Example 3. Data Register Timing Registers Programming for Mode 2**

Programming PIO data accesses for mode 2 operation using an IDE controller clock frequency of 33 MHz (30 ns period).

For mode 2 operation,  $t_2$  requires a minimum of 100 ns, this translates to a minimum of 4 clock cycles (4 cycles  $\times$  30 ns = 120 ns). There is no requirement for  $t_{2i}$  in mode 2 operation, but the minimum requirement for  $t_0$  is 240 ns, or 8 clock cycles. This means DATSTB and DATRCVR can be programmed to any combination equaling 8 (or more) clock cycles, with DATSTB specifying at least 4 clock cycles.

Sample programming values are DATSTB = 3h (4 clock cycles) and DATRCVR = 3h (4 clock cycles), or REGSTB = 5h (6 clock cycles) and DATRCVR = 1h (2 clock cycles). In addition, the minimum write data hold time specified is 15 ns, so the HWNHLD $n$ P bits in MISCCTL should be programmed to a value of at least 1 clock cycle. A sample value is HWNHLD $n$ P = 0h (0 clock cycles) providing enough hold time and margin.

#### **Example 4. Data Register Timing Registers Programming for Mode 3**

Programming PIO data accesses for mode 3 operation using an IDE controller clock frequency of 99 MHz (10.10 ns period).

For mode 3 operation,  $t_2$  requires a minimum of 80 ns, this translates to a minimum of 8 clock cycles (8 cycles  $\times$  10.10 ns = 80.80 ns). The minimum requirement on  $t_{2i}$  is defined to be 70 ns, this translates to a minimum of 7 clock cycles (7 cycles  $\times$  10.10 ns = 70.70 ns). The minimum cycle time  $t_0$  is 180 ns, or 18 clock cycles. This means that DATSTB and DATRCVR can be programmed to any combination equaling 18 (or more) clock cycles, with DATSTB specifying at least 8 cycles and DATRCVR specifying at least 7 cycles.

Sample programming values are DATSTB = 9h (10 clock cycles) and DATRCVR = 7h (8 clock cycles), or DATSTB = Ah (11 clock cycles) and DATRCVR = 6h (7 clock cycles). In addition, the minimum write data hold time specified is 10 ns, so the HWNHLD $n$ P bits in MISCCTL should be programmed to a value of at least 1 clock cycle. A sample value is HWNHLD $n$ P = 0h (1 clock cycle) providing enough hold time and margin.

### 2.5.1.3 Programming Multiword DMA Register Accesses

The DMASTB and DMARCVR are used in reprogramming the timings for multiword DMA transfers. The required IDE timing parameters for multiword DMA transfers are defined in the ATA/ATAPI-6 specification.

The DMASTB and DMARCVR can be programmed to match the parameters  $t_0$  (cycle time),  $t_D$  (strobe time), and  $t_{KW}$  (recovery time for DMA write). The HWNHLDnP bits in MISCCTL are used to program the hold time for the write data ( $t_H$  parameter).

The DMASTB directly controls the number of clock cycles that the  $\overline{\text{ATA\_HIO\!R}}$  and  $\overline{\text{ATA\_HIO\!W}}$  strobes will be asserted during multiword DMA transfers. This corresponds to the strobe width timing parameter,  $t_D$ .

The DMARCVR defines the number of clock cycles for the recovery time for multiword DMA transfers. This corresponds to recovery timing parameters,  $t_{KR}$  and  $t_{KW}$ .

The sum of both parameters must be equal or greater than the cycle time,  $t_0$ . The HWNHLDnP bits in MISCCTL allow control over the number of clock cycles for the write data hold time during multiword DMA data writes. With knowledge of the system clock frequency, you can program the appropriate number of clock cycles to match the timing requirements. Note that the timing registers must be programmed with a value one less than the desired number of cycles, so a value of 0 specifies 1 clock cycle, a value of 1 specifies 2 clock cycles, etc.

[Example 5](#) and [Example 6](#) illustrate how the timing of the multiword DMA registers can be programmed.

#### **Example 5. Multiword DMA Register Access Programming for Mode 0**

Programming multiword DMA transfers for mode 0 operation using an IDE controller clock frequency of 66 MHz (15 ns period).

For mode 0 operation,  $t_D$  requires a minimum of 215 ns, this translates to a minimum of 15 clock cycles (15 cycles  $\times$  15 ns = 225 ns). The minimum requirement on  $t_{KW}$  is 215 ns (for writes), this translates to a minimum of 15 clock cycles (15 cycles  $\times$  15 ns = 225 ns). The minimum requirement for  $t_0$  is 480 ns, or 32 clock cycles. This means DMASTB and DMARCVR can be programmed to any combination equaling 32 (or more) clock cycles, with both DMASTB and DMARCVR specifying at least 15 clock cycles.

Sample programming values are DMASTB = Fh (16 clock cycles) and DMARCVR = Fh (16 clock cycles), or DMASTB = 10h (17 clock cycles) and DMARCVR = Eh (15 clock cycles). In addition, the minimum write data hold time specified is 20 ns, so the HWNHLDnP bits in MISCCTL should be programmed to a value of at least 2 clock cycles. A sample value is HWNHLDnP = 2h (3 clock cycles) providing enough hold time and margin and also covering the PIO and task file mode 0 timings.

#### **Example 6. Multiword DMA Register Access Programming for Mode 2**

Programming multiword DMA transfers for mode 2 operation using an IDE controller clock frequency of 99 MHz (10.10 ns period).

For mode 2 operation,  $t_D$  requires a minimum of 70 ns, this translates to a minimum of 7 clock cycles (7 cycles  $\times$  10.10 ns = 70.70 ns). The minimum requirement on  $t_{KW}$  is 25 ns, this translates to a minimum of 3 clock cycles (3 cycles  $\times$  10.10 ns = 30.30 ns). The minimum cycle time  $t_0$  is 120 ns, or 12 clock cycles. This means that DMASTB and DMARCVR can be programmed to any combination equaling 12 (or more) clock cycles, with DMASTB specifying at least 7 cycles and DMARCVR specifying at least 3 cycles.

Sample programming values are DMASTB = 7h (8 clock cycles) and DMARCVR = 3h (4 clock cycles), or DMASTB = 8h (9 clock cycles) and DMARCVR = 2h (3 clock cycles). In addition, the minimum write data hold time specified is 10 ns, so the HWNHLDnP bits in MISCCTL should be programmed to a value of at least 1 clock cycle. A sample value is HWNHLDnP = 2h (3 clock cycles) providing enough hold time and margin and also covering the PIO and task file mode 2 timings.

### 2.5.1.4 Programming Ultra-DMA Register Accesses

The UDMASTB, UDMATRP, and UDMATENV are used in reprogramming the timings for ultra-DMA transfers. The required IDE timings parameters for ultra-DMA transfers are defined in the ATA/ATAPI-6 specification.

The UDMASTB, UDMATRP, and UDMATENV can be programmed to match the parameters  $t_{CYC}$  (cycle time),  $t_{2CYCTYP}$  (two cycle time),  $t_{RP}$  (ready-to-pause), and  $t_{ENV}$  (time envelope).

The UDMASTB directly controls the number of clock cycles for the ultra-DMA strobe during ultra-DMA transfers. This corresponds to the strobe width timing parameter,  $t_{CYC}$ , which is ( $t_{2CYCTYP}$ ).

The UDMATRP defines the number of clock cycles for the ready-to-pause timing parameter,  $t_{RP}$ .

The UDMATENV indicates the number of clock cycles for the timing envelope timing parameter,  $t_{ENV}$ .

With knowledge of the IDE controller clock frequency, you can program the appropriate number of clock cycles to match the timing requirements. Note that the timing registers must be programmed with a value one less than the desired number of clock cycles, so a value of 0 specifies 1 clock cycle, a value of 1 specifies 2 clock cycles, etc.

[Example 7](#) and [Example 8](#) illustrate how the timing of the ultra-DMA registers can be programmed.

#### **Example 7. Ultra-DMA Register Access Programming for Mode 5**

Programming ultra-DMA transfers for mode 5 operation using an IDE controller clock frequency of 99 MHz (10.10 ns period).

For mode 5 operation,  $t_{2CYC}$  is 40 ns ( $t_{CYC} = 20$  ns), this translates to a minimum of 2 clock cycles (2 cycles  $\times$  10.10 ns = 20.20 ns) per UDMA cycle. The requirement on  $t_{RP}$  is 85 ns, this translates to a minimum of 9 clock cycles (9 cycles  $\times$  10 ns = 90.90 ns).  $t_{ENV}$  has a minimum value of 20 ns and a maximum value of 50 ns, so this needs to be 2 to 5 clock cycles (2  $\times$  10 ns = 20.20 ns to 5  $\times$  10 ns = 50.50 ns).

Sample programming values are UDMASTB = 1h (2 clock cycles), UDMATRP = 8h (9 clock cycles), and UDMATENV = 2h (3 clock cycles).

#### **Example 8. Ultra-DMA Register Access Programming for Mode 4**

Programming ultra-DMA transfers for mode 4 operation using an IDE controller clock frequency of 66 MHz (15 ns period).

For mode 4 operation,  $t_{2CYC}$  is 60 ns ( $t_{CYC} = 30$  ns), this translates to a minimum of 2 clock cycles (2 cycles  $\times$  15 ns = 30 ns) per UDMA cycle. The requirement on  $t_{RP}$  is 100 ns, this translates to a minimum of 7 clock cycles (7 cycles  $\times$  15 ns = 105 ns).  $t_{ENV}$  has a minimum value of 20 ns and a maximum value of 55 ns, so this needs to be 2 to 4 clock cycles (2 cycles  $\times$  15 ns = 30 ns to 4 cycles  $\times$  15 ns = 60 ns).

Sample programming values are UDMASTB = 1h (2 clock cycles), UDMATRP = 6h (7 clock cycles), and UDMATENV = 2h (3 clock cycles).

## 2.6 DMA and PIO Data Transaction Overview

The IDE controller supports a dedicated DMA controller in order to handle DMA transfers between host memory and attached ATA/ATAPI device. This occurs for both multiword and ultra-DMA transfers during DMA writes to and reads from the device. The DMA controller includes logic to manage the physical region descriptors (PRDs) that describe the DMA transfers, and controls a set of 32-bit wide FIFOs (256-byte read FIFO and 256-byte write FIFO resident within the IDE controller) to temporarily store data for DMA transfers.

When pre-fetching and post-writing is enabled, the dedicated DMA controller also controls PIO sector writes and reads even though this is a PIO transaction. The internal FIFO is reused for these PIO transfers to store the pre-fetch/post-write data. During PIO reads, the sector data is pre-fetched into the FIFO and then read out of the FIFO by firmware. During PIO write, the PIO write sector data is stored in the FIFO and then eventually written out through the IDE interface to the device.

### 2.6.1 DMA Controller and FIFO Operation

The DMA controller is the primary initiator for DMA transfers for both multiword and ultra-DMA read and write transfers. For any ATA DMA writes to the device, the DMA controller will initiate a read transfer from host memory to fill up its internal FIFO and then transfer the data out of the FIFO to the ATA device. For ATA DMA read transfers from the device to host/system memory, the DMA controller will initiate a read transfer and read data from the ATA device into the internal FIFO and then transfer the data from the FIFO into host memory. The DMA controller makes use of the physical region descriptors to identify buffer locations and sizes of data (in bytes) to be transferred for its data transaction.

The difference between multiword DMA and ultra-DMA transfers is mostly on the signaling part of the transaction. For multiword DMA transactions, the host IDE controller is responsible for strobing data. In addition, a single word is transferred per strobe cycle. For ultra-DMA data transactions, data strobing is handled by the entity that is sending the data. If the data transaction is a write to a disk, then the host IDE controller is responsible for strobing the data. However, if the data transaction is a read from a disk, then the attached device is responsible for strobing the data. In addition, during ultra-DMA transactions, one word (2 bytes/word) is transferred for every edge of the strobe within a single strobe cycle increasing the throughput.

#### 2.6.1.1 Physical Region Descriptors (PRDs)

The physical region descriptors (PRDs) define the physical memory regions that data is transferred from (when moving data from host memory to the ATA device) or transferred to (when moving data from the ATA device to host memory). The DMA controller contains the logic and control for handling the physical region descriptors. When a DMA transfer is started, the DMA controller will issue a two-word burst to read the first descriptor from memory. It then decodes and stores the byte transfer count and the source or target memory address, and then starts the DMA operation using the descriptor information. Once a PRD has been completely processed by the DMA controller and the entire DMA transfer has completed, the next PRD in the table will be fetched and processed. If all PRD entries have been processed, an interrupt is generated.

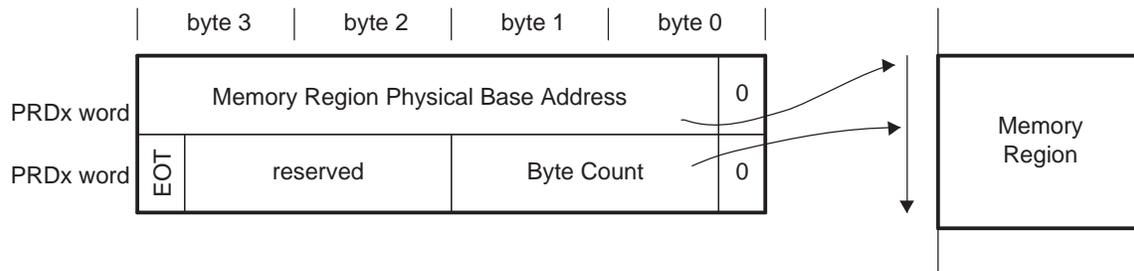
The DMA interface is used to control data transfers to and from host memory and the FIFO internal to the IDE controller block. The DMA engine contains the bus master IDE control registers. These registers are the primary IDE channel DMA control register (BMICP), the primary IDE channel DMA status register (BMISP), and the primary IDE channel DMA descriptor pointer register (BMIDTP).

The DMA engine interfaces to the DMA interface to the processor on one end and the IDE controller interface on the other. The IDE controller block strobes data into and out of the FIFO in the DMA engine block, based on the setting of the read/write direction bit (DMADIR bit in BMICP).

The BMIDTP is the pointer to a physical region descriptor table in system memory. This descriptor table contains physical region descriptors, which describe areas of memory that are involved in the data transfer. The descriptor table must be aligned on a 4-byte boundary and the table cannot cross a 64 KB boundary in memory.

The physical memory region to be transferred is described by a physical region descriptor (PRD). Each PRD entry is 8 bytes in length (Figure 2 and Table 3). The first 4 bytes specify the byte address of a physical memory region; the next two bytes specify the count of the region in bytes (64K byte limit per region). A value of zero in these two bytes indicates 64K. Bit 7 of the last byte indicates the end of the table (EOT) when set to 1.

**Figure 2. Physical Region Descriptor (PRD) Table Entry**



**Table 3. Description of Single Physical Region Descriptor (PRD) Table Entry**

PRDx Word	Bit	Description
0	31-0	Memory region physical base address for PRDx (Must be 2-bytes aligned, even).
1	31	End of descriptor table marker (last PRD marker).
1	30-16	Reserved
1	15-0	Byte count (must be even). A value of 0 implies 64 Kbytes size.

DMA activity is started when the host software sets the DMA start/stop bit (DMASTART bit in BMICP). The DMA engine state machine starts fetching the first entry in the PRD table. Once the PRD is fetched, the DMA engine state machine starts moving data between system memory and the FIFO. If the end of table (EOT) bit is 0, a new PRD table entry is fetched when the current physical region data transfer is complete. Data transfer continues until the last block of data is moved and the EOT bit is set to 1.

By linking multiple PRDs where each PRD identifies a buffer within the reach of the dedicated ATA DMA, the IDE controller can perform a single transaction where data is read from (or written to) multiple (PRD entries) buffers allowing the realization of the scatter gather capability.

When all data transfers for the DMA command submitted to the device are complete, the ATA/ATAPI device asserts INTRQ (if enabled). The DMA channel waits until the last data words have been moved to/from system memory before setting the interrupt bit (and passing the interrupt to the host interrupt handler) and clearing the bus master IDE active bit (IDEACT bit in BMISP).

### 2.6.1.2 DMA Driven Disk Write Transfer Operation

To perform an ATA DMA write operation to an attached ATA/ATAPI device, a physical descriptor region is programmed by firmware indicating a system memory address to transfer data from and the number of bytes to transfer. The BMIDTP register is initialized with the start address of the first physical region descriptor. The DMA write command is then issued to the drive, and the DMA transfer starts when firmware programs the DMASTART bit in the primary IDE channel DMA control register (BMICP). When the DMASTART bit is set, the DMA controller issues a two-word (8 bytes) request to read the first physical region descriptor (PRD) entry from memory. This information is stored and processed by the DMA controller to indicate where in memory to start reading data from and also the size of data (in bytes) to transfer to the device. The DMA controller will then read out the data from host/system memory in bursts starting at the specified transfer address. Data read from the system memory is stored in the FIFO. The DMA controller will issue consecutive read bursts until it fills the FIFO. The transfer will start up again when the IDE controller starts emptying out the FIFO and space is available in the FIFO. When the DMA finishes processing the current PRD, based on the state of the EOT field of the current PRD, it will either fetch the next PRD or wait until the FIFO is fully empty to generate an interrupt (and also set the INTRSTAT bit in the primary IDE channel DMA status register, BMISP) and also clear the IDEACT bit in BMISP indicating the end of the transfer. The total data size entered within the PRD tables for a given DMA transaction should be equal to the data size value entered via command. The combination of status bit settings by the DMA controller and the attached device will allow you to identify the user entry condition.

### 2.6.1.3 DMA Driven Disk Read Transfer Operation

To perform an ATA DMA read operation from an attached ATA/ATAPI device, a physical descriptor region is programmed by firmware indicating a system memory address to transfer the disk data to and the number of bytes to transfer. The BMIDTP register is initialized with the start address of the first physical region descriptor. The DMA read command is then issued to the drive, and the DMA transfer is started when firmware programs the DMASTART bit in the primary IDE channel DMA control register (BMICP). When the DMASTART bit is set, the DMA controller first issues a two-word (8 bytes) request to read the first physical region descriptor (PRD) entry from memory. This information is stored and processed by the DMA controller to indicate where in memory to store the data read from the disk and also the size (in bytes) of data transfer. The IDE controller will facilitate the reception of data from the attached device and store read data within the read FIFO. This occurs for both multiword and ultra-DMA writes. The IDE interface control logic continues reading words from the ATA device and storing them into the FIFO. The DMA controller routes read data to the system/host memory based on the PRD entries. This continues until the size entered in the PRD is exhausted or the FIFO becomes empty. When enough data to fill all the sizes noted within the PRD entries has been stored at the host/system memory, the DMA generates an interrupt (also sets the INTRSTAT bit in the primary IDE channel DMA status register, BMISP) and also clears the IDEACT bit in BMISP indicating the end of the transfer. The total data size entered within the PRD tables for a given DMA transaction should be equal to the data size value entered via command. The combination of status bit settings by the DMA controller and the attached device will allow you to identify the user entry condition.

### 2.6.1.4 Miscellaneous Cases

#### 2.6.1.4.1 Multiword and Ultra-DMA Abort

After any type of DMA transfer has been initiated (multiword or ultra-DMA read or write), firmware is allowed to prematurely abort the transfer. A DMA transfer is initiated when a PRD has been set up and the DMASTART bit in the primary IDE channel DMA control register (BMICP) has been set to 1. After this time, firmware is allowed to prematurely terminate the transfer if it has not completed yet by writing a 0 to the DMASTART bit. When a DMA abort is detected by the IDE DMA controller, the DMA state machine will immediately abort the current transfer and return to an idle state. The IDE interface state machine will finish its current word transfer and terminate the transfer. The FIFO logic will be reset, so any words currently in the FIFO are lost. Also, any data transferred from memory to disk (during disk writes) or from the disk to memory (during disk reads) should be considered invalid, as it will not be clear which words were actually transferred.

#### 2.6.1.4.2 Ultra-DMA Pause

Pauses in the data flow occur whenever one side of the transfer is not keeping up with the data flow.

For disk read transfers where data is being read from the disk and transferred to system memory, this occurs when the IDE controller is filling up the FIFO (with read data from the attached device) faster than the on-board DMA can write data in to host/system memory. When this situation occurs, the IDE controller pauses the transfer by deasserting the ATA\_HIOR line. After ATA\_HIOR is deasserted, the host may receive up to three additional words from the device before transfers are fully halted. Once read data has been moved out of the FIFO to host/system memory and there is more space available in the FIFO, the IDE controller ends the pause and starts the UDMA transfer again.

For disk write transfers where data is being read from host/system memory and written to the disk, the IDE controller only pauses the write transfer if there is no more data to be read out of the FIFO and send through the ATA interface to the drive. This occurs when the host/system memory bus throughput is less than that of the ATA interface.

The attached ATA/ATAPI device can also pause the write transfer by deasserting the ATA\_IORDY signal. When this occurs, the IDE controller simply stops reading data out of the FIFO and pauses. When the attached device is ready to resume the write transfer it will reassert the ATA\_IORDY, which allows the IDE controller to start reading data out of the FIFO.

#### 2.6.1.4.3 Ultra-DMA Terminate

The IDE controller can terminate an ultra-DMA read or write transfer at any time. This can occur when firmware tries to interrupt an ultra-DMA transfer with a PIO read of the status register. When this situation occurs, the IDE controller will issue an ultra-DMA terminate by deasserting the ATA\_DMACK signal. The IDE controller will then issue the PIO read to the device, complete the PIO read, and then start up the ultra-DMA transfer again. DMA activity on the host side is transparent to this intermediate task and data will continue to be transferred until the appropriate FIFO thresholds are reached. Once the IDE controller completes the PIO read it will continue with its ultra-DMA transfer.

Termination of an ultra-DMA transfer also occurs when firmware decides to abort the entire DMA transfer (as described in [Section 2.6.1.4.1](#)). When this occurs, the host controller terminates the request by deasserting the ATA\_DMACK signal. In addition, the IDE controller is reset back to an idle state.

#### 2.6.1.5 PIO Data Transaction

For PIO data transaction, the CPU is responsible to continually perform access to the data register in order to perform PIO disk read or write transaction. The firmware is also responsible to keep track of the size of the data to be transferred. One disadvantage of performing PIO data transaction is low throughput. The host CPU is much faster than the maximum attainable interface speed (PIO mode 4), which forces the host CPU to halt for a significant amount of time for each single word transfer.

### 2.6.1.6 PIO Pre-Fetch/Post-Write Feature

The IDE host controller has a pre-fetch/post-write feature that would alleviate, when enabled, the wait time incurred to the host CPU by the slower IDE interface by allowing the CPU to perform burst read/write PIO data accesses. It does so by making use of the DMA controller logic and its associated FIFO for storing PIO read/write data. This is transparent to the CPU and the firmware (user). All the firmware has to do is enable the pre-fetch/post-write feature, by setting the PREPOST0/1 bits in the primary IDE channel timing register (IDETIMP) prior to performing PIO data-in/data-out access. For PIO disk write operation, write data is intercepted by the DMA controller and sent to the write FIFO that eventually gets driven out to the attached device. For PIO disk read transactions, data read is stored in the read FIFO and the CPU can perform burst read access to the data register once enough data from the attached device has been read onto the read FIFO.

## 2.7 Attached Device Reset Considerations

### 2.7.1 Attached Device Hardware Reset Considerations

ATA/ATAPI specification-compliant devices include a hardware signal that can be used by a host to perform hardware reset. The processor does not have a dedicated signal for this purpose. If this capability is desired, use of a processor GPIO signal to reset the attached device is recommended.

### 2.7.2 Attached Device Software Reset Considerations

The IDE controller can perform software reset in one of two ways:

1. By writing to bit 2 (SRST field) of the device control register
2. By invoking a DEVICE RESET command (applicable to ATAPI device only)

The host can perform software reset to attached devices by writing to bit 2 (SRST bit) of the device control register. Note that both devices would respond to the write, regardless of which device is selected.

An ATAPI device (packet device) allows the use of the additional command DEVICE RESET to perform a reset to a selected device. This command is available to a packet device only and is not supported by ATA devices.

### 2.7.3 Peripheral Hardware Reset Considerations

When a hardware reset occurs on the processor, all of the peripheral registers are reset to their default values (as shown in [Section 4](#)) and the IDE controller interface is disabled.

### 2.7.4 Peripheral Software Reset Considerations

The ATA controller peripheral can be reset by the processor Power and Sleep Controller (PSC) by two methods:

- Disabling and enabling the ATA controller peripheral
- Performing a synchronized reset

For more detailed information on the use of the processor Power and Sleep Controller (PSC), see the *TMS320DM646x DMSoC ARM Subsystem Reference Guide* ([SPRUEP9](#)).

## 2.8 Initialization

Proper host initialization needs to take place prior to accessing attached ATA/ATAPI device. From the host side, the programmable timing registers are required to be programmed appropriately for the type of control (PIO 8-bit) and data (PIO, multiword, or ultra-DMA) transaction to be used. It is important that the programming of the registers on the host side matches the device side mode of operation. For this reason, it is recommended that the firmware initialize the host IDE controller registers for PIO mode0 (for both PIO 8-bit and PIO 16-bit) and interrogate the device for its current settings and its capability. After determining the attached current settings and capability, the firmware can reconfigure the attached device with a new configuration or reconfigure its own programmable timing registers to match the device's current capability.

### 2.8.1 Host Initialization

The correct values needed to program the ATA register file are dependent upon the clock frequency used by the IDE controller. This frequency is programmable and the firmware needs to be aware of this frequency prior to configuring the IDE controller registers. The IDE controller registers can be initialized in multiple ways. Initialization examples are provided in the following sections.

#### 2.8.1.1 PIO 8-Bit and PIO 16-Bit Initialization

PIO 8-bit transactions are always used for control transactions and PIO-16 bit transactions are used for data transactions. The IDE controller should be programmed with the lowest mode, which is mode 0 for both control and data, since the device-operating mode is not known and the lowest mode is supported by all other higher modes.

#### **CAUTION**

The value programmed in MISCCTL is the largest of the PIO 8-bit, PIO-16-bit, and multiword DMA write hold time values, since this timing parameter is used by the three transfers.

1. Identify the IDE controller clock frequency.
2. Initialize the programmable timing registers for PIO 8-bit mode 0 timing:
  - Task file register strobe timing register (REGSTB)
  - Task file register recovery timing register (REGRCVR)
  - Miscellaneous control register (MISCCTL)
3. Initialize the programmable timing registers for PIO 16-bit mode 0 timing:
  - Data register PIO strobe timing register (DATSTB)
  - Data register PIO recovery timing register (DATRCVR)
  - Miscellaneous control register (MISCCTL)
4. Enable the use of the programmable timing registers (over-ride timing):
  - Miscellaneous control register (MISCCTL)
5. Enable access to attached devices:
  - Primary IDE channel timing register (IDETIMP)

### 2.8.1.2 Multiword DMA Initialization

It is important that the attached device capability is identified prior to invoking any type of DMA activity. In addition, prior to using a DMA transaction, it is the responsibility of the firmware to configure the mode for the type of DMA transaction to be used (in this case it is multiword DMA, not ultra-DMA). When this step is performed, the attached device uses the selected DMA signaling when a data transaction that utilizes the DMA is invoked. The host has a register field that is programmed to select the type of DMA to use. The attached devices do not have this selection capability. When a DMA mode transaction is requested, devices use the DMA mode type programmed most recently. In this section, the initialization steps configure the use for multiword DMA mode

#### CAUTION

The value programmed in MISCCTL is the largest of the PIO 8-bit, PIO-16-bit, and multiword DMA write hold time values, since this timing parameter is used by the three transfers.

1. Identify the IDE controller clock frequency.
2. Initialize the programmable timing registers for PIO 8-bit mode 0 timing:
  - Task file register strobe timing register (REGSTB)
  - Task file register recovery timing register (REGRCVR)
  - Miscellaneous control register (MISCCTL)
3. Initialize the programmable timing registers for PIO 16-bit mode 0 timing:
  - Data register PIO strobe timing register (DATSTB)
  - Data register PIO recovery timing register (DATRCVR)
  - Miscellaneous control register (MISCCTL)
4. Enable the use of the programmable timing registers (over-ride timing):
  - Miscellaneous control register (MISCCTL)
5. Enable access to attached devices:
  - Primary IDE channel timing register (IDETIMP)
6. Invoke IDENTIFY DEVICE command on to the attached device.
7. Read device configuration data.
8. Identify if multiword DMA is supported. If so, select mode value to configure and continue.
9. If it is necessary to change the device configuration or to configure the device to use multiword DMA transactions on the next DMA command posting, perform the SET FEATURES command.
  - This command may need to be performed more than once: once for PIO mode selection and once for multiword DMA mode selection. If modes other than PIO mode 0 are selected, then it will be necessary to reconfigure the PIO 8-bit and PIO 16-bit for the appropriate mode.
10. Configure the multiword DMA programmable timing registers for the multiword DMA mode that matches the attached device operation:
  - DMA strobe timing register (DMASTB)
  - DMA recovery timing register (DMARCVR)
  - Miscellaneous control register (MISCCTL)
11. Select multiword DMA mode instead of ultra-DMA:
  - Ultra DMA control register (UDMACTL)
12. Configure PRD table entry address
  - Primary IDE channel DMA descriptor pointer register (BMIDTP)
13. Configure PRD table entries.

You are now ready to perform PIO 8-bit, PIO 16-bit, and multiword DMA access.

### 2.8.1.3 Ultra-DMA Initialization

It is important that the attached device capability is identified prior to invoking any type of DMA activity. In addition, prior to using a DMA transaction, it is the responsibility of the firmware to configure the mode for the type of DMA transaction to be used. When this step is performed, the attached device uses the selected DMA signaling when a data transaction that utilizes DMA is invoked. The host has a register field that is programmed to select the type of DMA to use. The attached devices do not have this selection capability. When a DMA mode transaction is requested, devices use the DMA mode type programmed most recently. In this section, the initialization steps configure the use for ultra-DMA mode.

1. Identify the IDE controller clock frequency.
2. Initialize the programmable timing registers for PIO 8-bit mode 0 timing:
  - Task file register strobe timing register (REGSTB)
  - Task file register recovery timing register (REGRCVR)
  - Miscellaneous control register (MISCCTL)
3. Initialize the programmable timing registers for PIO 16-bit mode 0 timing:
  - Data register PIO strobe timing register (DATSTB)
  - Data register PIO recovery timing register (DATRCVR)
  - Miscellaneous control register (MISCCTL)
4. Enable the use of the programmable timing registers (over-ride timing):
  - Miscellaneous control register (MISCCTL)
5. Enable access to attached devices:
  - Primary IDE channel timing register (IDETIMP)
6. Invoke IDENTIFY DEVICE command on to the attached device.
7. Read device configuration data.
8. Identify if ultra-DMA is supported. If so, select mode value to configure and continue.
9. If it is necessary to change the device configuration or to configure the device to use ultra-DMA transactions on the next DMA command posting, perform the SET FEATURES command.
  - This command may need to be performed more than once: once for PIO mode selection and once for ultra-DMA mode selection. If modes other than PIO mode 0 are selected, then it will be necessary to reconfigure the PIO 8-bit and PIO 16-bit for the appropriate mode.
10. Configure the ultra-DMA programmable timing registers for the ultra-DMA mode that matches the attached device operation:
  - Ultra DMA strobe timing register (UDMASTB)
  - Ultra DMA ready-to-stop timing register (UDMATRP)
  - Ultra DMA timing envelope register (UDMATENV)
11. Select ultra-DMA mode instead of multiword DMA:
  - Ultra DMA control register (UDMACTL)
12. Configure PRD table entry address
  - Primary IDE channel DMA descriptor pointer register (BMIDTP)
13. Configure PRD table entries

You are now ready to perform PIO 8-bit, PIO 16-bit, and ultra-DMA access.

## 2.8.2 Attached Device Initialization

After power-on reset, the attached ATA/ATAPI device comes up in a default mode that might not be known by the host firmware. However, as long as it is a functional device, it should be ready to communicate with the firmware. It is recommended that the host firmware configures the host IDE controller to generate PIO mode 0 timing (since this is the lowest mode and a subset of all higher modes) for communicating with the attached device for its initial communication with the attached device. After the firmware configures the host controller to operate in PIO mode 0, select the appropriate IDE controller clock frequency, enable the programmable timing registers, and enable the device register access. Start by invoking the IDENTIFY DEVICE command to retrieve device information. Based on the data retrieved, the firmware reconfigures the attached device and then the IDE controller with the identical fastest PIO and DMA modes supported. The firmware uses the SET FEATURES command to configure the attached device. Consult the ATA/ATAPI specification along with the particular device specification for more information.

## 2.9 Interrupt Support

The ATA controller has a single interrupt source (Table 4) to the ARM and/or the DSP. This interrupt source is not multiplexed with any other interrupt source on either CPU. For more information on the DSP interrupt controller (DSPINTC) and the ARM interrupt controller (AINTC), see the *TMS320DM646x DMSoC DSP Subsystem Reference Guide* ([SPRUEP8](#)) and the *TMS320DM646x DMSoC ARM Subsystem Reference Guide* ([SPRUEP9](#)).

**Table 4. ATA Controller Interrupt**

Acronym	Interrupt Number	
	AINTC	DSPINTC
IDEINT	22	90

The IDE host DMA controller is ultimately responsible for dispatching interrupts to the ARM interrupt controller that is used to interrupt the system processor for various conditions. There are three sources for interrupt generation:

- PIO transaction
- DMA transaction
- ATA\_IORDY (IORDY) timer timeout

### 2.9.1 Interrupt Events and Sources

Three interrupt events and two interrupt status bit fields (INTRSTAT and IORDYINT) are available to record interrupt existence. A single vector within the ARM CPU interrupt vector space exists and is used as the ATA interrupt entry to handle all the three events. The associated interrupt service routine is responsible for identifying the interrupt source and providing service accordingly.

The ATA interrupt source could be the ATA\_IORDY (IORDY) timer, the attached device, or the DMA controller. Regardless of the source of the interrupt, the DMA controller is eventually responsible for dispatching the interrupt to the CPU.

Upon receiving an interrupt, the interrupt service routine (ISR) identifies the source by examining the state of the INTRSTAT and IORDYINT bits. If the IORDYINT bit is set and an interrupt is received, then the source of the interrupt is the ATA\_IORDY (IORDY) timer. However, if an interrupt is received and none of the two bits have been set, then the source of the interrupt is a PIO transaction. If an interrupt is received and the INTRSTAT bit is set, then the interrupt occurred due to a successful DMA transaction. A summary of these states is shown in [Table 5](#).

When any of the interrupt status bits is set, the ISR is responsible for clearing the interrupt flag by writing a 1 to the bit. If this is not done, the ISR may have difficulty identifying the source of the interrupt correctly when the next interrupt is received.

**Table 5. Identifying the ATA Controller Interrupt Sources**

Interrupt Source	INTRSAT Bit	IORDYINT Bit
PIO transaction	0	0
DMA transaction	1	1
IORDY timeout	0	1

### 2.9.1.1 ATA Interrupt Source: ATA\_IORDY (IORDY) Timer

The ATA\_IORDY (IORDY) timer interrupt is generated by the IDE controller logic when the IDE controller ATA\_IORDY (IORDY) timer is programmed with a non-zero value and the device fails to deassert the ATA\_IORDY signal before the timer runs out when performing PIO transactions.

If the IORDY (ATA\_IORDY) timer register is written with a non-zero value, the timer starts counting when the attached device asserts the ATA\_IORDY signal. If the attached device fails to deassert the ATA\_IORDY signal before the timer runs out, an interrupt is generated. This allows the host controller not to wait an indefinite amount of time in the event that the external device hangs and never comes back to complete its transaction.

When the interrupt is generated, the IORDYINT bit is set in the primary IDE channel DMA status register (BMISP). The interrupt can be cleared by writing a 1 to the IORDYINT bit. This will deassert the interrupt and also clear the IORDYINT status bit. This feature can be used with any of the PIO transactions (control or data).

### 2.9.1.2 Acknowledging IORDY Timer Interrupt

When an IORDY timer is enabled and the attached device did not de-assert the IORDY signal prior to the IORDY timer counting down to zero, an IORDY timer interrupt would be generated. At this time, a bit that signifies the IORDY timer interrupt (IORDYINT) would be set within BMISP. At the same time, the interrupt is routed on to the CPU. After the CPU ISR determines that the cause of the interrupt (by reading BMISP), it clears the IORDY timer interrupt by writing a 1 into the IORDYINT bit. The firmware can also re-initialize the primary IO ready timer configuration register (IORDYTMP) with the timer count (based on the IDE controller clock) value if future interrupt is to be expected.

---

**Note:** This feature and operation is transparent to the attached device and no function is required to be performed on the device side.

---

### 2.9.1.3 ATA Interrupt Source: Attached Device

For some non-data and PIO data transactions to an attached ATA device, the attached device will assert the INTRQ signal when a user has chosen to be interrupted by clearing bit 1 of the device control register within the attached device prior to posting the transaction. For non-data commands, the device would assert the INTRQ signal after the completion of the command (for example, the SET FEATURES command). For commands that require a data-in or data-out transaction, an interrupt would be generated based on the direction of the data flow. For data-in transactions, the attached device generates the interrupt by asserting the INTRQ signal when the device is ready to transfer the data. For data-out transactions, the attached device generates the interrupt by asserting the INTRQ signal when the device is finished receiving the data. The timeline and the number of interrupts generated depends on the type of command issued. For PIO Read/Write Sector, a single interrupt is generated for each sector transferred. For writes, an interrupt is generated after each sector is transferred. If the number of sectors transferred per transaction is more than one, then more than one interrupt would be generated. For the Read/Write Block command, a single interrupt is generated for each transfer prior to the start of data transfer for data-in transactions and after the end of the transfer for data-out transactions. If the data-in and data-out transaction is PIO driven, none of the interrupt status bits (INTRSTAT and IORDYINT) would be set. However, the received interrupt would be passed to the CPU by the DMA controller. The ISR is responsible for deasserting the ATA\_IORDY (IORDY) interrupt by reading the status/command register of the device.

#### 2.9.1.4 Acknowledging Attached Device Interrupt

When a device is ready or finished with a data transaction or when a device finishes executing certain commands, it generates an interrupt to the host in order to indicate the completion status. The device communicates its status by asserting the ATA\_INTRQ signal. The device asserts the ATA\_INTRQ signal only if the host has enabled interrupt generation by clearing the nIEN of the Device Control Register prior to invoking a command that generates an interrupt.

Once a device asserts the ATA\_INTRQ signal, the signal remains asserted until the host acknowledges the interrupt. The host acknowledges the interrupt by reading the Status Register. Upon reading the Status Register, the device would de-assert (stops driving) the ATA\_INTRQ signal.

---

**Note:** The firmware should refrain from reading the Status Register to get the status of the device when it is expecting an interrupt from the attached device since this might cause the interrupt to be cleared if the interrupt arrived while the Status Register read is taking place causing the firmware to miss an interrupt. The firmware can read the alternate Status Register and avoid the possibility of in advertent interrupt acknowledgment.

---

#### 2.9.1.5 ATA Interrupt Source: DMA Controller

For any type of DMA transfer (multiword or ultra-DMA reads and writes), the interrupt is asserted once the DMA transfer has completed. For DMA data-in transactions (data reads from the attached ATA device to host/system memory), the DMA interrupt will be generated once the last data word that has been read from the attached ATA device is written to the host/system memory. For DMA data-out transactions (data writes to the attached ATA device from host/system memory), the interrupt is issued once the last data word has been read from system memory and written to the ATA device. When the interrupt signal is asserted, the INTRSTAT bit in the primary IDE channel DMA status register (BMISP) is set to a 1. The interrupt can be cleared by writing a 1 to the INTRSTAT bit. This deasserts the interrupt and clears the INTRSTAT bit. Note that an interrupt may not be generated if the data length described by the PRD table does not match the length specified by the ATA/ATAPI command. The state of the INTRSTAT and IDEACT bits in the BMISP register are used to identify the state and condition of the DMA controller (see [Section 4.2](#) for more information). When the attached device is configured to generate interrupts, by clearing bit 1 of the device control register, the attached device generates an interrupt at the end of the DMA transaction and asserts the INTRQ signal.

Note that even though both the DMA and attached devices are capable of generating interrupts for the same transaction (this happens during DMA read/write transactions), the final interrupt generated to the CPU is a single interrupt. There are only two bits that identify the three sources. In this case, both the ATA controller DMA as well as the attached ATA device have generated an interrupt. However, only a single interrupt would be received and the ISR would treat the interrupt as a DMA interrupt only. It is the duty of the firmware to read the Status/Command register (in the attached device) in order to deassert (clear) the ATA\_IORDY (IORDY) interrupt. This is the normal case when the PRD size entered for the DMA controller is equal to the command size entered to the attached device.

There are times when the user/firmware violates this normal condition by entering different values (that is, the PRD size entered is not equal to the command size entered). Even though this condition should be avoided, other status bits can be used to identify the status and condition of the transaction. Note that the interrupt is not always generated when the equality of size entered is violated. [Table 6](#) summarizes the possible interrupt conditions for a DMA driven data transaction. The column Interrupt Source displays the source of the interrupt during a DMA driven data transaction; it is either the attached device or the DMA controller.

**Table 6. DMA Driven Interrupt Conditions**

INTRSTAT Bit	IDEACT Bit	Interrupt Source		Description
		Device	DMA (Host)	
0	1	No	No	DMA transfer is in progress. Too early to generate an interrupt.
1	0	Yes	Yes	The IDE device generated an interrupt and the physical region descriptors have been exhausted. This is normal completion where the size of the physical memory regions is equal to the IDE device transfer size. The ARM CPU will receive an interrupt.
1	1	Yes	No	DMA Size > CMD Size: The IDE device generated an interrupt. The controller has not reached the end of the physical memory regions. This is a valid completion case when the size of the physical memory regions is larger than the IDE device transfer size. The ARM CPU will receive an interrupt even though the DMA is not done.
0	0	No	No	DMA Size < CMD Size. If the DMAERROR bit in BMISP is 1, there was a problem transferring data to/from memory. If the DMAERROR bit is 0, the descriptor table specified a smaller buffer size than the programmed IDE transfer size. The ARM CPU will not receive an interrupt.

**Note:** When performing a DMA transaction, the attached device should always be configured to generate an interrupt. If this step is omitted, no interrupt will be generated.

### 2.9.2 Acknowledging Interrupt for DMA Transaction

After a successful DMA transaction, the device generates an interrupt to notify the host by driving the ATA\_INTRQ signal. The Host DMA logic is eventually responsible for passing the interrupt to the host CPU. In addition to passing the interrupt to the host, the DMA logic also sets interrupt bit within the BMISP register.

The host CPU firmware is then responsible for acknowledging and clearing the interrupt received. For a DMA interrupt, the following three tasks (in the order shown) should be carried out to properly acknowledge and clear the DMA interrupt bit field:

- Read Device Status Register:** If the ATA\_INTRQ signal is driven by the device, reading the status register is a form of acknowledgment to the device and the device will stop driving the ATA\_INTRQ as soon as the read to the Status Register is done.
- Clear the DMA Start/Stop bit:** This bit is set to start a DMA transaction. When a DMA transaction is done, this bit stays set even though the DMA is done. In order to continue with a future DMA transaction, this bit must be cleared prior to initiating a new transaction.
- Clear the DMA interrupt status bit within the BMISP register:** This bit is set by the DMA to signify the type of interrupt (DMA in this case) received. It needs to be cleared for identifying the next/future DMA interrupts.

The order that the above three operations can be done is any combination, except steps 3, 2, 1. You must not sequence your DMA acknowledge operations as steps 3, 2, 1, and if you do, then it is most likely that you will be receiving additional false interrupt. The DMA logic would falsely think that an interrupt from PIO transaction is received, following the DMA transaction, when in fact there was none.

### 2.9.3 Interrupt Multiplexing

On the DM646x DMSoC, the ATA/ATAPI controller interrupt is not multiplexed with any other interrupt source.

## 2.10 EDMA Event Support

The ATA/ATAPI controller peripheral does not utilize the processor EDMA system. The controller has its own dedicated DMA system.

## 2.11 Power Management

The ATA/ATAPI controller can be placed in reduced power modes to conserve power during periods of low activity. The power management of the peripheral is controlled by the processor Power and Sleep Controller (PSC). The PSC acts as a master controller for power management of all of the peripherals on the processor. For detailed information on power management procedures using the PSC, see the *TMS320DM646x DMSoC ARM Subsystem Reference Guide* ([SPRUEP9](#)).

## 2.12 Emulation Considerations

The ATA/ATAPI controller is not affected by emulation suspend events such as breakpoints and halts and continues to run when these events occur.

## 3 Use Cases

The following sections describe how to interface the ATA controller.

### 3.1 Interfacing to a Standard ATA/ATAPI Device

#### 3.1.1 Hardware Interface to a Standard ATA/ATAPI Device

The traditional 40-pin or upgraded 80-pin conductor cable is commonly used to interface with hard-disk drive type storage. The correct ribbon cable size to use mostly depends on the speed the application requires. If ultra-DMA modes 2 and greater are to be used, the 80-conductor ribbon cable must be used; otherwise, there will be problems with data integrity due to data corruption resulting from noise (cross-talk). [Table 7](#) shows a mapping of the processor ATA/ATAPI controller signals with those on a generic hard-disk drive.

**Table 7. ATA/ATAPI Device Interface Connections for a Standard ATA/ATAPI Device**

Processor ATA/ATAPI Controller Signal	Attached ATA/ATAPI Device Signal	Description
ATA_CS[1:0]	CS[1:0]	Chip select 0 and 1, used by host to <u>select command</u> block and control block registers. During DMA transfers, both CS0 and CS1 are negated.
ATA_HA0 ATA_HA1 ATA_HA2	DA[2:0]	DA[2:0] specifies register or data port address.
ATA_D[15:0]	D[15:0]	D[15:0] is a 16-bit bi-directional Host Data Bus.
ATA_DMARQ	DMARQ	ATA_DMARQ is driven by the device to request DMA transfer.
ATA_DMACK	DMACK	ATA_DMACK is driven by the host to acknowledge DMA transfer.
ATA_IORDY	IORDY/DDMARDY/ STROBE	IORDY is negated by device to extend PIO transfer. DDMARDY flow control for ultra-DMA data-out (write) transfer. DSTROBE is data strobe signal for ultra-DMA data-in transfer.
ATA_HIOR	DIOR/HDMARDY/ HSTROBE <sup>(1)</sup>	DIOR is PIO transaction Read Strobe. HDMARDY flow control for ultra-DMA data-in (read) transfer. HSTROBE is data strobe signal for ultra-DMA data-out transfer.
ATA_HIOW	DIOW/STOP <sup>(1)</sup>	DIOR is PIO transaction Write Strobe. STOP is used to signal ultra-DMA burst transfer termination.
ATA_INTRQ	INTRQ <sup>(1)</sup>	Interrupt Request used by device to extend host PIO data.
GPIO signal used as a RESET	RESET	Reset used by host to reset the device.
ATA_DDIR	N/A	This signal is meaningful if the device to be connected to is higher (greater than 1.8 V) voltage tolerant.

<sup>(1)</sup> The ATA/ATAPI specification applies different signal names to this signal depending on the mode of operation.

### 3.1.2 Software Configuration for Interfacing to a Standard ATA/ATAPI Device

The following is a recommendation of how the firmware performs a host and device initialization. The initialization steps are:

1. Identify IDE controller clock frequency.
2. Initialize programmable timing registers for PIO mode 0 register access based on the clock frequency.
3. Initialize programmable timing registers for PIO mode 0 data access based on the clock frequency.
4. Enable programmable timing logic.
5. Enable IDE register access.
6. Perform IDENTIFY FEATURE command to determine the device capability.
7. Configure device with modes supported by both host and device.
8. Configure host with modes supported by both host and device.

Both the ATA controller and device are now ready to perform data transaction. For a detailed description of the software configuration of both the ATA controller and the external device, see [Section 2.8](#).

## 3.2 Interfacing to a Standard ATA/ATAPI Device Through a Level-Shifter

### 3.2.1 Interfacing to 5.0 V ATA/ATAPI Devices

The IDE controller provides glueless interface to 3.3 V tolerant ATA/ATAPI devices. If you need to interface with devices that are 5.0 V tolerant, an additional logic/buffer (level shifter) is required external to the processor. This is to drive the signal sourced by the processor at a level acceptable to the external device I/O buffer, since the processor I/O buffers do not meet the external device voltage or drive needs. The ATA\_DDIR signal indicates the direction of the transfer to the buffer/level-shifter. The processor does not supply a dedicated signal to enable/disable the level shifter. The level shifter is enabled or disabled by direct tie-off external to the processor.

### 3.2.2 Hardware Interface to an ATA/ATAPI Device Through a Level-Shifter

The traditional 40-pin or upgraded 80-pin conductor cable is commonly used to interface with hard-disk drive type storage. The correct ribbon cable size to use mostly depends on the speed the application is supposed to support. If ultra-DMA modes 2 and greater are to be used, the 80-conductor ribbon cable must be used; otherwise, there will be problems with data integrity due to data corruption resulting from noise (cross-talk). [Table 8](#) shows a mapping of the processor ATA/ATAPI controller signals with those on a generic hard-disk drive.

### 3.2.3 Software Configuration for Interfacing to a Standard ATA/ATAPI Device Through a Level-Shifter

The following is a recommendation of how the firmware performs a host and device initialization. The initialization steps are:

1. Identify IDE controller clock frequency.
2. Initialize programmable timing registers for PIO mode 0 register access based on the clock frequency.
3. Initialize programmable timing registers for PIO mode 0 data access based on the clock frequency.
4. Enable programmable timing logic.
5. Enable IDE register access.
6. Perform IDENTIFY FEATURE command.
7. Configure device with modes supported by both host and device.
8. Configure host with modes supported by both host and device.

Both the ATA controller and device are now ready to perform data transaction. For a detailed description of the software configuration of both the ATA controller and the external device, see [Section 2.8](#).

**Table 8. ATA/ATAPI Device Interface Connections for a Standard ATA/ATAPI Device Through a Level-Shifter**

Processor ATA/ATAPI Controller Signal	Attached ATA/ATAPI Device Signal	Description
ATA_CS[1:0]	CS[1:0]	Chip select 0 and 1, used by host to select command block and control block registers. During DMA transfers, both $\overline{CS0}$ and $\overline{CS1}$ are negated.
ATA_HA0 ATA_HA1 ATA_HA2	DA[2:0]	DA[2:0] specifies register or data port address.
ATA_D[15:0]	D[15:0]	D[15:0] is a 16-bit bi-directional Host Data Bus.
ATA_DMARQ	DMARQ	ATA_DMARQ is driven by the device to request DMA transfer.
$\overline{ATA\_DMACK}$	$\overline{DMACK}$	$\overline{ATA\_DMACK}$ is driven by the host to acknowledge DMA transfer.
ATA_IORDY	IORDY/ $\overline{DDMARDY}$ / STROBE	IORDY is negated by device to extend PIO transfer. $\overline{DDMARDY}$ flow control for ultra-DMA data-out (write) transfer. DSTROBE is data strobe signal for Ultra DMA data-in transfer.
$\overline{ATA\_HIOR}$	$\overline{DIOR}/\overline{HDMARDY}/$ HSTROBE <sup>(1)</sup>	$\overline{DIOR}$ is PIO transaction Read Strobe. $\overline{HDMARDY}$ flow control for ultra-DMA data-in (read) transfer. HSTROBE is data strobe signal for Ultra DMA data-out transfer.
$\overline{ATA\_HIOW}$	$\overline{DIOW}/\overline{STOP}$ <sup>(1)</sup>	$\overline{DIOW}$ is PIO transaction Write Strobe. STOP is used to signal ultra-DMA burst transfer termination.
ATA_INTRQ	INTRQ <sup>(1)</sup>	Interrupt Request used by device to extend host PIO data.
GPIO signal used as a RESET	$\overline{RESET}$	Reset used by host to reset the device.
ATA_DDIR	N/A	This signal is meaningful if the device to be connected to is higher (greater than 1.8 V) voltage tolerant.

(1) The ATA/ATAPI specification applies different signal names to this signal depending on the mode of operation.

### 3.3 Interfacing to Compact Flash

#### 3.3.1 Hardware Interfacing to Compact Flash

The compact Flash (CF) is a storage medium that is capable of being configured in multiple modes (Flash mode, I/O mode, and true-IDE mode). The CF is required to be configured in true-IDE mode in order for it to interface to a host via an ATA/ATAPI standard interface. Since it is capable of operating in multiple modes, configuring the CF to operate in true-IDE mode requires additional steps if hot insertion capability needs to be utilized. This additional step is due to compact Flash device general configuration and has no ties with the functionality of the IDE controller.

In order for a compact Flash device to be configured in true-IDE mode, the CF-specific signal, ATA\_SEL, should be driven low prior to power cycling. The processor IDE controller does not have a dedicated signal to drive this ATA\_SEL signal. This signal should be tied low external to the processor. Hot insertion is not supported.

The IDE host controller is used to interface to a multitude of storage devices. Devices like hard-disk drives, compact disks, or DVDs use the standard 40-pin or 80-pin ribbon cable to connect with the IDE controller. However, compact Flash (CF) devices use a 50-pin connector since it complies with the PCMCIA standard.

The CF should also be configured to operate in true-IDE mode in order to connect with the IDE controller. For true-IDE mode, the IDE controller terminals are connected to the corresponding CF terminals as defined in the compact Flash specification.

Table 9 shows a mapping of the processor ATA/ATAPI controller signals with those on a compact Flash device.

**Table 9. ATA/ATAPI Device Interface Connections for a Compact Flash Device**

DM646x ATA/ATAPI Controller Signal	Attached ATA/ATAPI Device Signal	Description
$\overline{\text{ATA\_CS}}[1:0]$	CS[2:1]	Chip select 0 and 1, used by host to select command block and control block registers, respectively, within the compact Flash device in true-IDE mode.
ATA_HA0 ATA_HA1 ATA_HA2	DA[2:0]	Lower 3 bits of address are used in compact Flash. Upper address bits A[10:3] should be grounded by host when in true-IDE mode.
ATA_D[15:0]	D[15:0]	Host Data Bus
ATA_DMARQ	Unconnected	DMA unsupported on compact Flash spec 2.0
$\overline{\text{ATA\_DMACK}}$	Unconnected	DMA unsupported on compact Flash spec 2.0
ATA_IORDY	WAITN	I/O Ready signal used by device to extend host PIO data transfer cycles.
$\overline{\text{ATA\_HIOR}}$	IORDY	Host I/O read strobe
$\overline{\text{ATA\_HIOW}}$	IOWR	Host I/O write strobe
ATA_INTRQ	INTRQ	Interrupt Request used by device to extend host PIO data.
GPIO signal used as a RESET	$\overline{\text{RESET}}$	Reset used by host to reset the device. Note that RESET is typically active high for compact Flash devices, but is active low for true-IDE devices.
N/A	$\overline{\text{ATASEL}}$	Grounded Prior to power-up. If hot insertion is supported, it is necessary to reset the CF card for proper configuration.

### 3.3.2 Software Configuration for Interfacing to Compact Flash

The following is a recommendation of how the firmware performs a host and device initialization. The initialization steps are:

1. Identify IDE controller clock frequency.
2. Initialize programmable timing registers for PIO mode 0 data access.
3. Enable programmable timing logic.
4. Enable IDE register access.
5. Perform IDENTIFY FEATURE command.
6. Configure device with modes supported by both host and device.
7. Configure dost with modes supported by both host and device.

Both the ATA controller and device are now ready to perform data transaction. For a detailed description of the software configuration of both the ATA controller and the external device, see [Section 2.8](#).

## 4 Registers

[Table 10](#) lists the memory-mapped registers for the ATA controller. See the device-specific data manual for the memory address of these registers.

[Table 11](#) lists the registers that are in the attached device, but not in the DM646x processor.

**Table 10. ATA Host Controller Registers**

Offset	Acronym	Register Description	Section
<b>ATA Bus Master Interface DMA Engine Registers</b>			
0h	BMICP	Primary IDE Channel DMA Control Register	<a href="#">Section 4.1</a>
2h	BMISP	Primary IDE Channel DMA Status Register	<a href="#">Section 4.2</a>
4h	BMIDTP	Primary IDE Channel DMA Descriptor Table Pointer Register	<a href="#">Section 4.3</a>
<b>ATA Configuration Registers</b>			
40h	IDETIMP	Primary IDE Channel Timing Register	<a href="#">Section 4.4</a>
47h	IDESTAT	IDE Controller Status Register	<a href="#">Section 4.5</a>
48h	UDMACTL	Ultra-DMA Control Register	<a href="#">Section 4.6</a>
50h	MISCCTL	Miscellaneous Control Register	<a href="#">Section 4.7</a>
54h	REGSTB	Task File Register Strobe Timing Register	<a href="#">Section 4.8</a>
58h	REGRCVR	Task File Register Recovery Timing Register	<a href="#">Section 4.9</a>
5Ch	DATSTB	Data Register Access PIO Strobe Timing Register	<a href="#">Section 4.10</a>
60h	DATRCVR	Data Register Access PIO Recovery Timing Register	<a href="#">Section 4.11</a>
64h	DMASTB	Multiword DMA Strobe Timing Register	<a href="#">Section 4.12</a>
68h	DMARCVR	Multiword DMA Recovery Timing Register	<a href="#">Section 4.13</a>
6Ch	UDMASTB	Ultra-DMA Strobe Timing Register	<a href="#">Section 4.14</a>
70h	UDMATRP	Ultra-DMA Ready-to-Pause Timing Register	<a href="#">Section 4.15</a>
74h	UDMATENV	Ultra-DMA Timing Envelope Register	<a href="#">Section 4.16</a>
78h	IORDYTMP	Primary IO Ready Timer Configuration Register	<a href="#">Section 4.17</a>

**Table 11. ATA Controller Registers in the Attached Device**

Offset	Register Name	Register Description	Read/Write
<b>Primary Channel Command Block Registers</b>			
1F0h	Data	PIO Read/Write Data Register Access	Read/Write
1F1h	Error	Error Register	Read
1F1h	Features	Features Register	Write
1F2h	Sector Count	Sector Count Register	Read/Write
1F3h	LBA Low	LBA Low (known as Sector Number on previous ATA Specs)	Read/Write
1F4h	LBA Mid	LBA Mid (known as Cylinder Low on previous ATA Specs)	Read/Write
1F5h	LBA High	LBA High (known as Cylinder High on previous ATA Specs)	Read/Write
1F6h	Device	Device Register (known as Drive/Head on previous ATA Specs)	Read/Write
1F7h	Status	Status Register	Read
1F7h	Command	Command Register	Write
<b>Primary Channel Control Block Registers</b>			
3F6h	Alt-Status	Alternate Status Register	Read
3F6h	Device Control	Device Control Register	Write

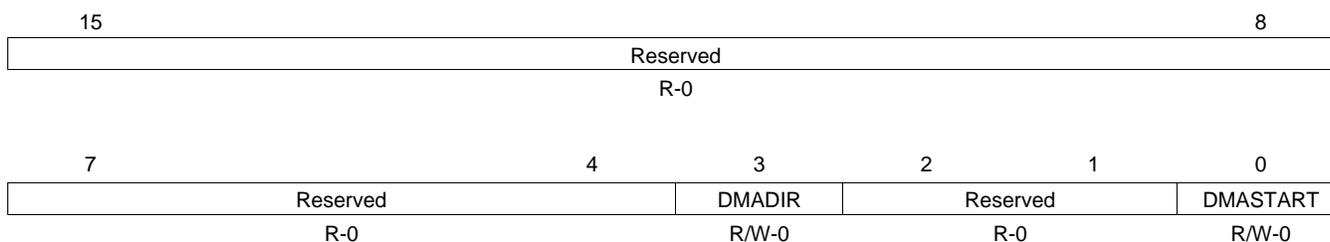
#### 4.1 Primary IDE Channel DMA Control Register (BMICP)

The primary IDE channel DMA control register (BMICP) is a 16-bit wide register used to indicate the direction of the DMA. It is also used to initiate DMA transfers to/from the IDE devices. BMICP is shown in Figure 3 and described in Table 12.

Writing a 1 to the DMASTART bit starts the DMA engine; DMA operation begins when this bit changes from a 0 to a 1. The ATA controller transfers data between the IDE device and memory only when the DMASTART bit is set; a DMA transfer terminates, if the DMASTART bit is written with a 0, and cannot be resumed. The DMASTART bit remains set when a DMA transfer is completed.

If the DMASTART bit is cleared to 0 while a DMA operation is still active (the IDEACT bit in BMISP is 1) and the device has not yet finished its data transfer (the INTRSTAT bit in BMISP is 0), the DMA transaction is aborted and data transferred from the device may not be written to memory. The DMASTART bit should be cleared to 0 after a data transfer is completed (as indicated by the IDEACT or INTRSTAT bits in BMISP). Note that the descriptors must be set up in memory and the descriptor starting address written to the primary IDE channel DMA descriptor table pointer register (BMIDTP) before setting the DMASTART bit.

**Figure 3. Primary IDE Channel DMA Control Register (BMICP)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 12. Primary IDE Channel DMA Control Register (BMICP) Field Descriptions**

Bit	Field	Value	Description
15-4	Reserved	0	Reserved
3	DMADIR	0	DMA read (write to IDE device).
		1	DMA write (read from IDE device).
2-1	Reserved	0	Reserved
0	DMASTART	0	Stop DMA operation.
		1	Start DMA operation.

## 4.2 Primary IDE Channel DMA Status Register (BMISP)

The primary IDE channel DMA status register (BMISP) is a 16-bit wide register used to indicate interrupt presence, DMA error condition, as well as DMA activity (state). BMISP is shown in [Figure 4](#) and described in [Table 13](#).

**Figure 4. Primary IDE Channel DMA Status Register (BMISP)**

15	Reserved					8
R-0						
7	4	3	2	1	0	
Reserved		IORDYINT	INTRSTAT	DMAERROR	IDEACT	
R-0		R/W1C-0	R/W1C-0	R/W1C-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing 0 has no effect); -n = value after reset

**Table 13. Primary IDE Channel DMA Status Register (BMISP) Field Descriptions**

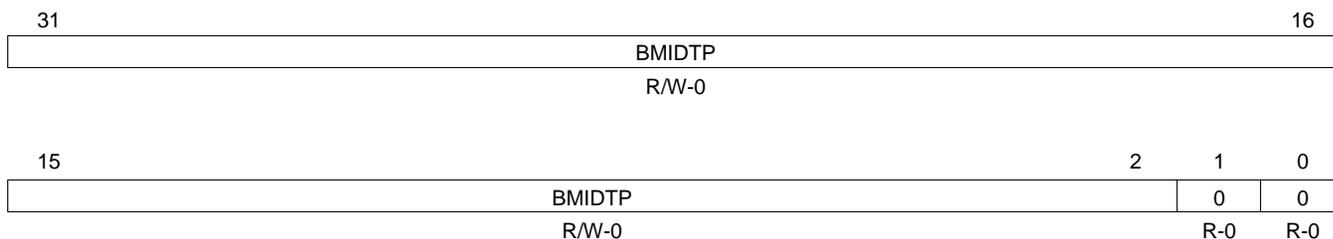
Bit	Field	Value	Description
15-4	Reserved	0	Reserved
3	IORDYINT	0 1	IORDY timeout. Write a 1 to clear this bit. IORDY timer functionality is disabled. IORDY timer functionality is enabled and the WAIT (IORDY) timer times-out before the target device being accessed deasserts ATA_IORDY during a transfer.
2	INTRSTAT	0 1	IDE interrupt status. INTRSTAT is set by the rising edge of the IDE device interrupt pin; write a 1 to clear this bit. After a DMA transfer is initiated, INTRSTAT is set when all data has been transferred to system memory (read commands) or to the device (write commands). If a DMA transfer has not been initiated, software can use INTRSTAT to determine if an IDE device has asserted its interrupt line.  Because INTRSTAT is set at the rising edge of the IDE interrupt, INTRSTAT will not necessarily represent the current state of the IDE interrupt signal. IDE interrupt is inactive. IDE interrupt is active.
1	DMAERROR	0 1	DMA error. DMAERROR is set when the controller encounters an error in transferring data to or from memory; write a 1 to clear this bit.  DMAERROR can only be set due to an internal buffer overflow or underflow. Under normal conditions, this should not occur. No DMA error. DMA buffer underflow or overflow.
0	IDEACT	0 1	IDE active. IDEACT is set when the DMASTART bit in BMICP is written with a 1. IDEACT is cleared when the final DMA transfer for a region is performed (that is, where the EOT bit is set in the region descriptor) and all data has been transferred to system memory (device read) or to the device (device write), or when a DMA transfer is aborted by writing the DMASTART bit with a 0. See <a href="#">Table 6</a> . IDE is inactive IDE is active.

### 4.3 Primary IDE Channel DMA Descriptor Table Pointer Register (BMIDTP)

The primary channel DMA descriptor table pointer register (BMIDTP) is a 32-bit wide register used to describe the starting address of the DMA descriptor table. BMIDTP must be programmed appropriately before setting the DMASTART bit in the primary IDE channel DMA control register (BMICP). Bits 0 and 1 are permanently tied low, as descriptors must be aligned on a 4-byte boundary. BMIDTP is shown in Figure 5 and described in Table 14.

The content of the BMIDTP is updated automatically with the next PRD address after completing a DMA transfer to/from the buffer location pointed by the current PRD if the EOT (End-of-Table) bit is cleared (see section Section 2.6.1.1 for more information). For this reason, proper initialization of the BMIDTP register is required prior to any DMA transaction.

**Figure 5. Primary IDE Channel DMA Descriptor Table Pointer Register (BMIDTP)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14. Primary IDE Channel DMA Descriptor Table Pointer Register (BMIDTP) Field Descriptions**

Bit	Field	Value	Description
31-2	BMIDTP	0-3FFF FFFFh	Descriptor base address pointer.
1-0	Reserved	0	Reserved

#### 4.4 Primary IDE Channel Timing Register (IDETIMP)

The primary IDE channel timing register (IDETIMP) is a 16-bit wide register used to control the enable/disable capability for the IDE interface as well as control for PIO data pre-fetch/post-write capability. IDETIMP is shown in [Figure 6](#) and described in [Table 15](#).

If the pre-fetch/post-write capability is enabled, the CPU can perform burst read/write data transfer since the DMA FIFO will be used to store outgoing and incoming data temporarily.

**Figure 6. Primary IDE Channel Timing Register (IDETIMP)**

15	14					8
IDEEN	Reserved					
R/W-0	R-0					
7	6	5	3	2	1	0
Reserved	PREPOST1	Reserved		PREPOST0	Reserved	
R-0	R/W-0	R-0		R/W-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15. Primary IDE Channel Timing Register (IDETIMP) Field Descriptions**

Bit	Field	Value	Description
15	IDEEN	0 1	IDE decode enable. PIO transactions targeting the IDE device registers (command and control blocks) are not decoded; accesses to these address spaces do not initiate activity on the IDE interface. PIO transactions targeting the IDE device registers (command and control blocks) are decoded and drive the IDE interface.
14-7	Reserved	0	Reserved
6	PREPOST1	0 1	Device 1 PIO prefetch and postwrite enable. PIO data prefetch and postwrite for device 1 is disabled. PIO data prefetch and postwrite for device 1 is enabled.
5-3	Reserved	0	Reserved
2	PREPOST0	0 1	Device 0 PIO prefetch and postwrite enable. PIO data prefetch and postwrite for device 0 is disabled. PIO data prefetch and postwrite for device 0 is enabled.
1-0	Reserved	0	Reserved

#### 4.5 IDE Controller Status Register (IDESTAT)

The IDE controller status register (IDESTAT) is an 8-bit wide register used to return the logic value of various IDE interface signals. IDESTAT is shown in Figure 7 and described in Table 16.

**Figure 7. IDE Controller Status Register (IDESTAT)**

7	6	5	4	3	0
Reserved	DMARQ	INTRQ	DMACKN	Reserved	
R-0	R-0	R-0	R-1	R-0	

LEGEND: R = Read only; -n = value after reset

**Table 16. IDE Controller Status Register (IDESTAT) Field Descriptions**

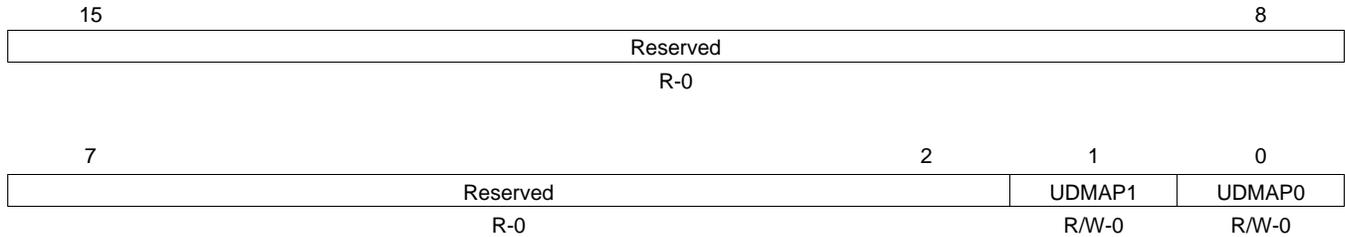
Bit	Field	Value	Description
7	Reserved	0	Reserved
6	DMARQ	0 1	Exact value of the ATA_DMARQ signal line being sent from the device to the host controller. ATA_DMARQ signal line is logic low. ATA_DMARQ signal line is logic high.
5	INTRQ	0 1	Exact value of the INTRQ signal line being sent from the device to the host controller. INTRQ signal line is logic low. INTRQ signal line is logic high.
4	DMACKN	0 1	Exact value of the $\overline{\text{ATA\_DMACK}}$ signal line being sent from the host controller to the device. The default value is inactive high. $\overline{\text{ATA\_DMACK}}$ signal line is logic low. $\overline{\text{ATA\_DMACK}}$ signal line is logic high.
3-0	Reserved	0	Reserved

#### 4.6 Ultra-DMA Control Register (UDMACTL)

The ultra-DMA control register (UDMACTL) is a 16-bit wide register used to enable each individual drive for ultra-DMA transfers. For multiword DMA (not ultra-DMA) operation, UDMACTL should be programmed with a 0. UDMACTL is shown in [Figure 8](#) and described in [Table 17](#).

From the device/drive perspective, the type of DMA transfer (multiword or ultra-DMA) to be used on the next transfer depends on the latest DMA configuration invoked by the host firmware using the SET FEATURES command. Consult the ATA/ATAPI specification for more information.

**Figure 8. Ultra-DMA Control Register (UDMACTL)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 17. Ultra-DMA Control Register (UDMACTL) Field Descriptions**

Bit	Field	Value	Description
15-2	Reserved	0	Reserved
1	UDMAP1	0	Primary device 1 ultra-DMA enable. Ultra-DMA is disabled for device 1. Multiword DMA operation is enabled for device 1.
		1	Ultra-DMA is enabled for device 1.
0	UDMAPO	0	Primary device 0 ultra-DMA enable. Ultra-DMA is disabled for device 0. Multiword DMA operation is enabled for device 0.
		1	Ultra-DMA is enabled for device 0.

#### 4.7 Miscellaneous Control Register (MISCCTL)

The miscellaneous control register (MISCCTL) is a 32-bit wide register used to provide miscellaneous configuration for the IDE controller PIO 8-bit, PIO 16-bit, and multiword DMA write hold time. All three transactions share the same field. The value programmed should be large enough to satisfy all three transfers. In addition, MISCCTL has fields that control the enable/disable capability of the programmable timing registers. MISCCTL is shown in Figure 9 and described in Table 18.

**Figure 9. Miscellaneous Control Register (MISCCTL)**

31	23	22	20	19	18	16	
Reserved			HWNHLD1P	Rsvd	HWNHLD0P		
R-0			R/W-0	R-0	R/W-0		
						1	0
15						TIMORIDE	
						R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

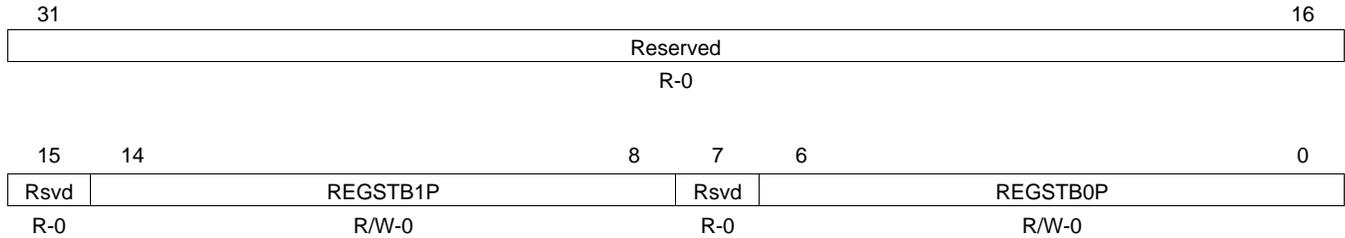
**Table 18. Miscellaneous Control Register (MISCCTL) Field Descriptions**

Bit	Field	Value	Description
31-23	Reserved	0	Reserved
22-20	HWNHLD1P	0-7h 0 1h-7h	ATA_HIOW write data hold time for slave controller. HWNHLD1P specifies the number of clock cycles minus 1 that the write data is held past the deassertion of the ATA_HIOW strobe for the slave controller during PIO and multiword DMA transfers.  This field is only valid when the TIMORIDE bit is set to 1.  0 1 clock cycle. 1h-7h 2 clock cycles to 8 clock cycles.
19	Reserved	0	Reserved
18-16	HWNHLD0P	0-7h 0 1h-7h	ATA_HIOW write data hold time for master controller. HWNHLD0P specifies the number of clock cycles minus 1 that the write data is held past the deassertion of the ATA_HIOW strobe for the master controller during PIO and multiword DMA transfers.  This field is only valid when the TIMORIDE bit is set to 1.  0 1 clock cycle. 1h-7h 2 clock cycles to 8 clock cycles.
15-1	Reserved	0	Reserved
0	TIMORIDE	0 1	IDE interface timing control.  0 IDE interface timing is controlled by the internal timing parameters of the IDE controller. 1 IDE interface timing is controlled by the timing override registers (REGSTB, REGRCVR, DMASTB, DATRCVR, DMASTB, DMARCVR, UDMASTB, UDMATRP and UDMATENV).

#### 4.8 Task File Register Strobe Timing Register (REGSTB)

The task file register strobe timing register (REGSTB) is a 32-bit wide register used in conjunction with the TIMORIDE bit in the miscellaneous control register (MISCCTL). REGSTB is used to define the PIO 8-bit transaction strobe assertion time width in clock cycles for both the master and slave devices. REGSTB is shown in Figure 10 and described in Table 19.

**Figure 10. Task File Register Strobe Timing Register (REGSTB)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

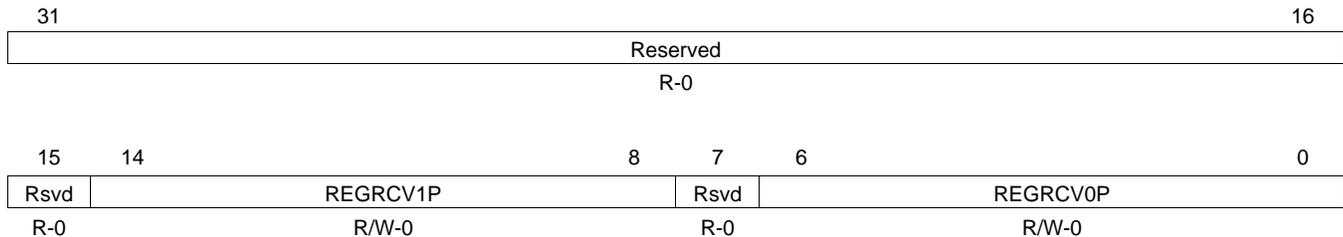
**Table 19. Task File Register Strobe Timing Register (REGSTB) Field Descriptions**

Bit	Field	Value	Description
31-15	Reserved	0	Reserved
14-8	REGSTB1P	0-7Fh  0 1h-7Fh	Register access <u>slave</u> strobe width for the controller. REGSTB1P specifies the number of clock cycles minus 1 that the ATA_HIOR and ATA_HIOW is asserted during accesses to the IDE slave's 8-bit task file registers.  This field is only valid when the TIMORIDE bit in MISCCTL is set to 1.  1 clock cycle. 2 clock cycles to 128 clock cycles.
7	Reserved	0	Reserved
6-0	REGSTB0P	0-7Fh  0 1h-7Fh	Register access <u>master</u> strobe width for the controller. REGSTB0P specifies the number of clock cycles minus 1 that the ATA_HIOR and ATA_HIOW is asserted during accesses to the IDE master's 8-bit task file registers.  This field is only valid when the TIMORIDE bit in MISCCTL is set to 1.  1 clock cycle. 2 clock cycles to 128 clock cycles.

#### 4.9 Task File Register Recovery Timing Register (REGRCVR)

The task file register recovery timing register (REGRCVR) is a 32-bit wide register used in conjunction with the TIMORIDE bit in the miscellaneous control register (MISCCTL). REGRCVR is used to define the PIO 8-bit transaction strobe deassertion time width in clock cycles for both the master and slave devices. REGRCVR is shown in Figure 11 and described in Table 20.

**Figure 11. Task File Register Recovery Timing Register (REGRCVR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

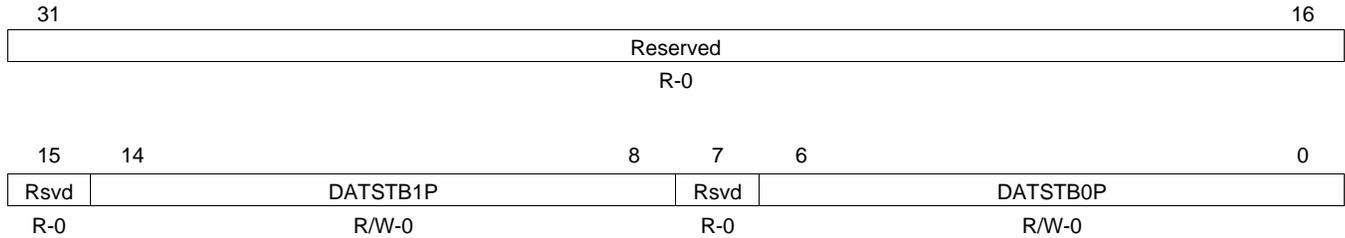
**Table 20. Task File Register Recovery Timing Register (REGRCVR) Field Descriptions**

Bit	Field	Value	Description
31-15	Reserved	0	Reserved
14-8	REGRCV1P	0-7Fh	Register access <u>slave</u> recovery time for the controller. REGRCV1P specifies the number of clock cycles minus 1 between ATA_HIOR or ATA_HIOW negation and assertion during accesses to the IDE slave's 8-bit task file registers.  This field is only valid when the TIMORIDE bit in MISCCTL is set to 1.
		0	1 clock cycle.
		1h-7Fh	2 clock cycles to 128 clock cycles.
7	Reserved	0	Reserved
6-0	REGRCV0P	0-7Fh	Register access <u>master</u> recovery time for the controller. REGRCV0P specifies the number of clock cycles minus 1 between ATA_HIOR or ATA_HIOW negation and assertion during accesses to the IDE master's 8-bit task file registers.  This field is only valid when the TIMORIDE bit in MISCCTL is set to 1.
		0	1 clock cycle.
		1h-7Fh	2 clock cycles to 128 clock cycles.

#### 4.10 Data Register Access PIO Strobe Timing Register (DATSTB)

The data register PIO strobe timing register (DATSTB) is a 32-bit wide register used in conjunction with the TIMORIDE bit in the miscellaneous control register (MISCCTL). DATSTB is used to define the PIO 16-bit data transaction strobe assertion time width in clock cycles for both the master and slave devices. DATSTB is shown in [Figure 12](#) and described in [Table 21](#).

**Figure 12. Data Register Access PIO Strobe Timing Register (DATSTB)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

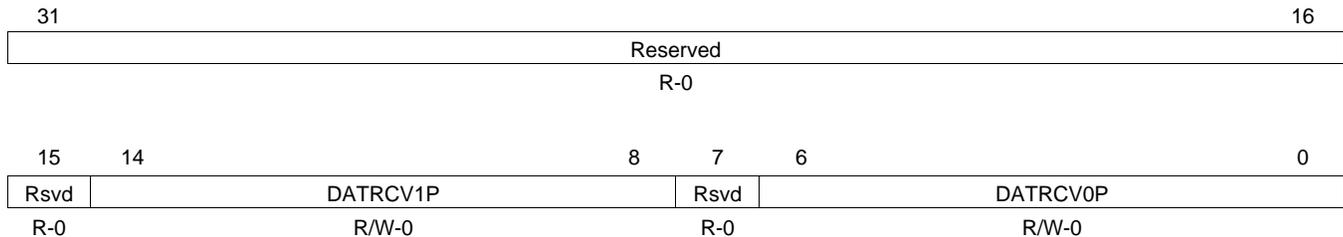
**Table 21. Data Register Access PIO Strobe Timing Register (DATSTB) Field Descriptions**

Bit	Field	Value	Description
31-15	Reserved	0	Reserved
14-8	DATSTB1P	0-7Fh  0 1h-7Fh	Slave data register access strobe width for the controller. DATSTB1P specifies the number of clock cycles minus 1 that the ATA_HIOR and ATA_HIOW is asserted during accesses to the IDE slave's 8-bit data register.  This field is only valid when the TIMORIDE bit in MISCCTL is set to 1.  1 clock cycle. 2 clock cycles to 128 clock cycles.
7	Reserved	0	Reserved
6-0	DATSTB0P	0-7Fh  0 1h-7Fh	Master data register access strobe width for the controller. DATSTB0P specifies the number of clock cycles minus 1 that the ATA_HIOR and ATA_HIOW is asserted during accesses to the IDE master's 8-bit data register.  This field is only valid when the TIMORIDE bit in MISCCTL is set to 1.  1 clock cycle. 2 clock cycles to 128 clock cycles.

#### 4.11 Data Register Access PIO Recovery Timing Register (DATRCVR)

The data register PIO recovery timing register (DATRCVR) is a 32-bit wide register used in conjunction with the TIMORIDE bit in the miscellaneous control register (MISCCTL). DATRCVR is used to define the PIO 16-bit data transaction strobe deassertion time width in clock cycles for both the master and slave devices. DATRCVR is shown in [Figure 13](#) and described in [Table 22](#).

**Figure 13. Data Register Access PIO Recovery Timing Register (DATRCVR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

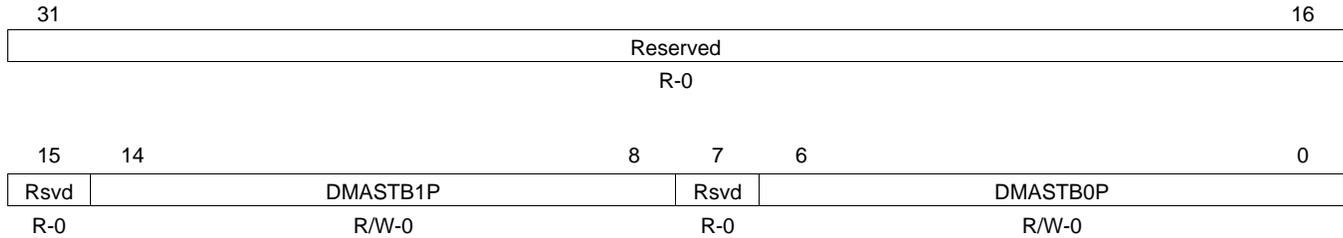
**Table 22. Data Register Access PIO Recovery Timing Register (DATRCVR) Field Descriptions**

Bit	Field	Value	Description
31-15	Reserved	0	Reserved
14-8	DATRCV1P	0-7Fh  0 1h-7Fh	Slave data register access recovery time for the controller. DATRCV1P specifies the number of clock cycles minus 1 between ATA_HIOR or ATA_HIOW negation and assertion during accesses to the IDE slave's 8-bit data register.  This field is only valid when the TIMORIDE bit in MISCCTL is set to 1.  1 clock cycle. 2 clock cycles to 128 clock cycles.
7	Reserved	0	Reserved
6-0	DATRCV0P	0-7Fh  0 1h-7Fh	Master data register access recovery time for the controller. DATRCV0P specifies the number of clock cycles minus 1 between ATA_HIOR or ATA_HIOW negation and assertion during accesses to the IDE master's 8-bit data register.  This field is only valid when the TIMORIDE bit in MISCCTL is set to 1.  1 clock cycle. 2 clock cycles to 128 clock cycles.

#### 4.12 Multiword DMA Strobe Timing Register (DMASTB)

The DMA strobe timing register (DMASTB) is a 32-bit wide register used in conjunction with the TIMORIDE bit in the miscellaneous control register (MISCCTL). DMASTB is used to define the multiword DMA data transaction strobe assertion time width in clock cycles for both the master and slave devices. DMASTB is shown in Figure 14 and described in Table 23.

**Figure 14. Multiword DMA Strobe Timing Register (DMASTB)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

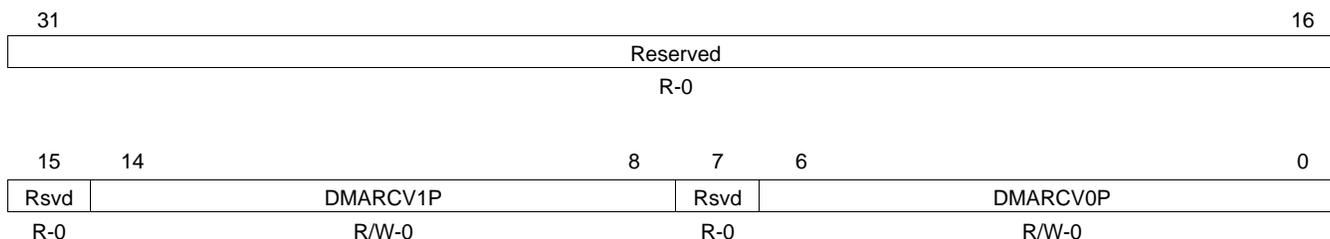
**Table 23. Multiword DMA Strobe Timing Register (DMASTB) Field Descriptions**

Bit	Field	Value	Description
31-15	Reserved	0	Reserved
14-8	DMASTB1P	0-7Fh  0 1h-7Fh	Slave DMA access strobe width for the controller. DMASTB1P specifies the number of clock cycles minus 1 that the $\overline{ATA\_HIO\!R}$ and $\overline{ATA\_HIO\!W}$ is asserted during accesses to the IDE slave device during a DMA data transfer.  This field is only valid when the TIMORIDE bit in MISCCTL is set to 1.  0 1 clock cycle. 1h-7Fh 2 clock cycles to 128 clock cycles.
7	Reserved	0	Reserved
6-0	DMASTB0P	0-7Fh  0 1h-7Fh	Master DMA access strobe width for the controller. DMASTB0P specifies the number of clock cycles minus 1 that the $\overline{ATA\_HIO\!R}$ and $\overline{ATA\_HIO\!W}$ is asserted during accesses to the IDE master device during a DMA data transfer.  This field is only valid when the TIMORIDE bit in MISCCTL is set to 1.  0 1 clock cycle. 1h-7Fh 2 clock cycles to 128 clock cycles.

### 4.13 Multiword DMA Recovery Timing Register (DMARCVR)

The DMA recovery timing register (DMARCVR) is a 32-bit wide register used in conjunction with the TIMORIDE bit in the miscellaneous control register (MISCCTL). DMARCVR is used to define the multiword DMA data transaction strobe deassertion time width in clock cycles for both the master and slave devices. DMARCVR is shown in Figure 15 and described in Table 24.

**Figure 15. Multiword DMA Recovery Timing Register (DMARCVR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

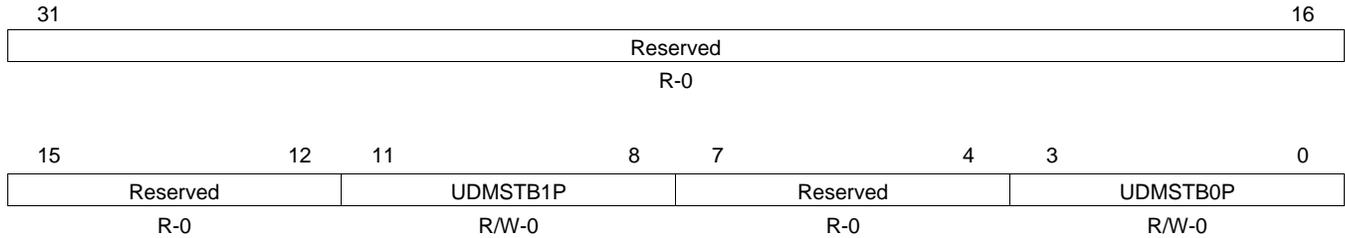
**Table 24. Multiword DMA Recovery Timing Register (DMARCVR) Field Descriptions**

Bit	Field	Value	Description
31-15	Reserved	0	Reserved
14-8	DMARCV1P	0-7Fh  0 1h-7Fh	Slave DMA access recovery time for the controller. DMARCV1P specifies the number of clock cycles minus 1 between ATA_HIOR or ATA_HIOW negation and assertion during accesses to the IDE slave device during a DMA data transfer.  This field is only valid when the TIMORIDE bit in MISCCTL is set to 1.  1 clock cycle.  2 clock cycles to 128 clock cycles.
7	Reserved	0	Reserved
6-0	DMARCV0P	0-7Fh  0 1h-7Fh	Master DMA access recovery time for the controller. DMARCV0P specifies the number of clock cycles minus 1 between ATA_HIOR or ATA_HIOW negation and assertion during accesses to the IDE master device during a DMA data transfer.  This field is only valid when the TIMORIDE bit in MISCCTL is set to 1.  1 clock cycle.  2 clock cycles to 128 clock cycles.

#### 4.14 Ultra-DMA Strobe Timing Register (UDMASTB)

The ultra-DMA strobe register (UDMASTB) is a 32-bit wide register used in conjunction with the TIMORIDE bit in the miscellaneous control register (MISCCTL). UDMASTB is used to define the ultra-DMA data transaction half-strobe width ( $t_{cyc}$ ) in clock cycles for both the master and slave devices. UDMASTB is shown in Figure 16 and described in Table 25.

**Figure 16. Ultra-DMA Strobe Timing Register (UDMASTB)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

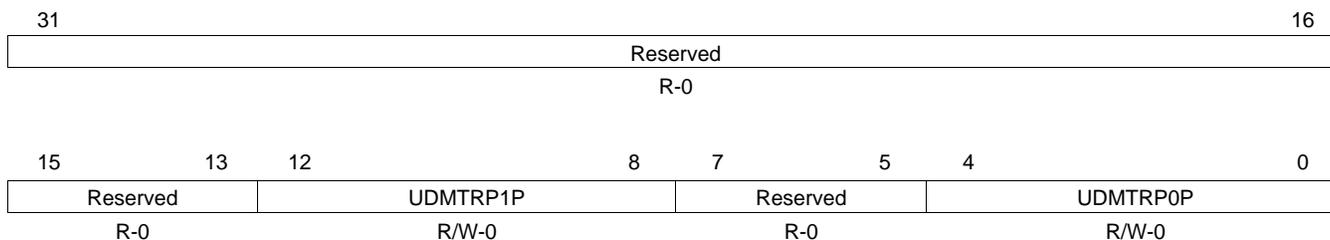
**Table 25. Ultra-DMA Strobe Timing Register (UDMASTB) Field Descriptions**

Bit	Field	Value	Description
31-12	Reserved	0	Reserved
11-8	UDMSTB1P	0-Fh	Slave ultra-DMA access strobe width for the controller. UDMSTB1P specifies the number of clock cycles minus 1 that the ATA_IORDY is asserted during accesses to the IDE slave device during an ultra-DMA data transfer.  This field is only valid when the TIMORIDE bit in MISCCTL is set to 1.
		0	1 clock cycle.
		1h-Fh	2 clock cycles to 16 clock cycles.
7-4	Reserved	0	Reserved
3-0	UDMSTB0P	0-Fh	Master ultra-DMA access strobe width for the controller. UDMSTB0P specifies the number of clock cycles minus 1 that the ATA_IORDY is asserted during accesses to the IDE master device during an ultra-DMA data transfer.  This field is only valid when the TIMORIDE bit in MISCCTL is set to 1.
		0	1 clock cycle.
		1h-Fh	2 clock cycles to 16 clock cycles.

#### 4.15 Ultra-DMA Ready-to-Pause Timing Register (UDMATRP)

The ultra DMA ready-to-pause timing register (UDMATRP) is a 32-bit wide register used in conjunction with the TIMORIDE bit in the miscellaneous control register (MISCCTL). UDMATRP is used to define the minimum time in clock cycles after ATA\_IORDY is negated after which the recipient may assert ATA\_HIOW or negate ATA\_DMARQ for both the master and slave devices. UDMATRP is shown in [Figure 17](#) and described in [Table 26](#).

**Figure 17. Ultra-DMA Ready-to-Pause Timing Register (UDMATRP)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 26. Ultra-DMA Ready-to-Pause Timing Register (UDMATRP) Field Descriptions**

Bit	Field	Value	Description
31-13	Reserved	0	Reserved
12-8	UDMATRP1P	0-1Fh	Slave ultra-DMA ready-to-pause time for the controller. UDMATRP1P specifies the number of clock cycles minus 1 between the HREADY and STOP signals during ultra-DMA accesses to the IDE slave device.  This field is only valid when the TIMORIDE bit in MISCCTL is set to 1.
		0	1 clock cycle.
		1h-1Fh	2 clock cycles to 32 clock cycles.
7-5	Reserved	0	Reserved
4-0	UDMATRP0P	0-1Fh	Master ultra-DMA ready-to-pause time for the controller. UDMATRP0P specifies the number of clock cycles minus 1 between the HREADY and STOP signals during ultra-DMA accesses to the IDE master device.  This field is only valid when the TIMORIDE bit in MISCCTL is set to 1.
		0	1 clock cycle.
		1h-1Fh	2 clock cycles to 32 clock cycles.

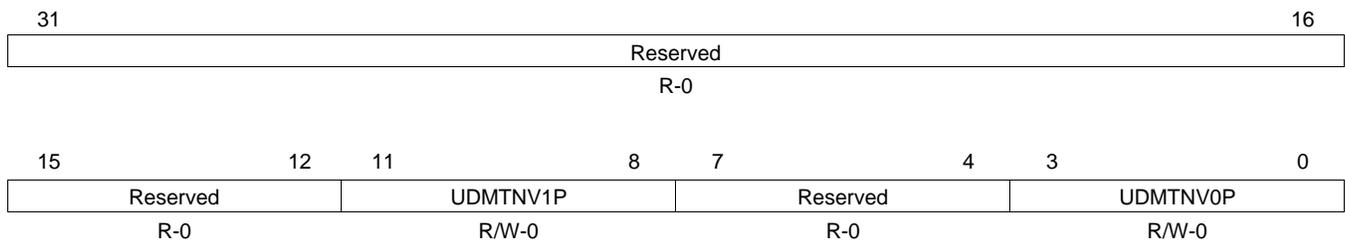
#### 4.16 Ultra-DMA Timing Envelope Register (UDMATENV)

The ultra-DMA timing envelope register (UDMATENV) is a 32-bit wide register used in conjunction with the TIMORIDE bit in the miscellaneous control register (MISCCTL). UDMATENV is used to define the time in clock cycles when:

- the host asserts  $\overline{\text{ATA\_DMACK}}$  until it negates  $\overline{\text{ATA\_HIOW}}$  and asserts  $\overline{\text{ATA\_HIOR}}$  at the beginning of an ultra-DMA data-in burst during a data-in transfer for both the master and slave devices
- the host asserts  $\overline{\text{ATA\_DMACK}}$  until it negates  $\overline{\text{ATA\_HIOW}}$  at the beginning of an ultra-DMA data-out burst for both the master and slave devices

UDMATENV is shown in [Figure 18](#) and described in [Table 27](#).

**Figure 18. Ultra-DMA Timing Envelope Register (UDMATENV)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 27. Ultra-DMA Timing Envelope Register (UDMATENV) Field Descriptions**

Bit	Field	Value	Description
31-12	Reserved	0	Reserved
11-8	UDMTENV1P	0-Fh	Slave ultra-DMA $t_{env}$ timing parameter for the controller. UDMTENV1P specifies the number of clock cycles minus 1 for the $t_{env}$ timing parameter during ultra-DMA transfers to the IDE slave device.  This field is only valid when the TIMORIDE bit in MISCCTL is set to 1.
		0	1 clock cycle.
		1h-Fh	2 clock cycles to 16 clock cycles.
7-4	Reserved	0	Reserved
3-0	UDMTENV0P	0-Fh	Master ultra-DMA $t_{env}$ timing parameter for the controller. UDMTENV0P specifies the number of clock cycles minus 1 for the $t_{env}$ timing parameter during ultra-DMA transfers to the IDE master device.  This field is only valid when the TIMORIDE bit in MISCCTL is set to 1.
		0	1 clock cycle.
		1h-Fh	2 clock cycles to 16 clock cycles.



## Appendix A Revision History

[Table A-1](#) lists the changes made since the previous version of this document.

**Table A-1. Document Revision History**

Reference	Additions/Modifications/Deletions
<a href="#">Section 3.2.1</a>	Changed subsection title. Changed first and second sentence.

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

### Products

Amplifiers	<a href="http://amplifier.ti.com">amplifier.ti.com</a>
Data Converters	<a href="http://dataconverter.ti.com">dataconverter.ti.com</a>
DLP® Products	<a href="http://www.dlp.com">www.dlp.com</a>
DSP	<a href="http://dsp.ti.com">dsp.ti.com</a>
Clocks and Timers	<a href="http://www.ti.com/clocks">www.ti.com/clocks</a>
Interface	<a href="http://interface.ti.com">interface.ti.com</a>
Logic	<a href="http://logic.ti.com">logic.ti.com</a>
Power Mgmt	<a href="http://power.ti.com">power.ti.com</a>
Microcontrollers	<a href="http://microcontroller.ti.com">microcontroller.ti.com</a>
RFID	<a href="http://www.ti-rfid.com">www.ti-rfid.com</a>
RF/IF and ZigBee® Solutions	<a href="http://www.ti.com/lprf">www.ti.com/lprf</a>

### Applications

Audio	<a href="http://www.ti.com/audio">www.ti.com/audio</a>
Automotive	<a href="http://www.ti.com/automotive">www.ti.com/automotive</a>
Broadband	<a href="http://www.ti.com/broadband">www.ti.com/broadband</a>
Digital Control	<a href="http://www.ti.com/digitalcontrol">www.ti.com/digitalcontrol</a>
Medical	<a href="http://www.ti.com/medical">www.ti.com/medical</a>
Military	<a href="http://www.ti.com/military">www.ti.com/military</a>
Optical Networking	<a href="http://www.ti.com/opticalnetwork">www.ti.com/opticalnetwork</a>
Security	<a href="http://www.ti.com/security">www.ti.com/security</a>
Telephony	<a href="http://www.ti.com/telephony">www.ti.com/telephony</a>
Video & Imaging	<a href="http://www.ti.com/video">www.ti.com/video</a>
Wireless	<a href="http://www.ti.com/wireless">www.ti.com/wireless</a>

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265  
Copyright © 2009, Texas Instruments Incorporated