List of Figures

List of Tables

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 _____ *Tip:* Enter the literature number in the search box provided at

The current documentation that describes the DM646x DMSoC, related peripherals, and other technical collateral, is available in the C6000 DSP product folder at: _____

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.

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

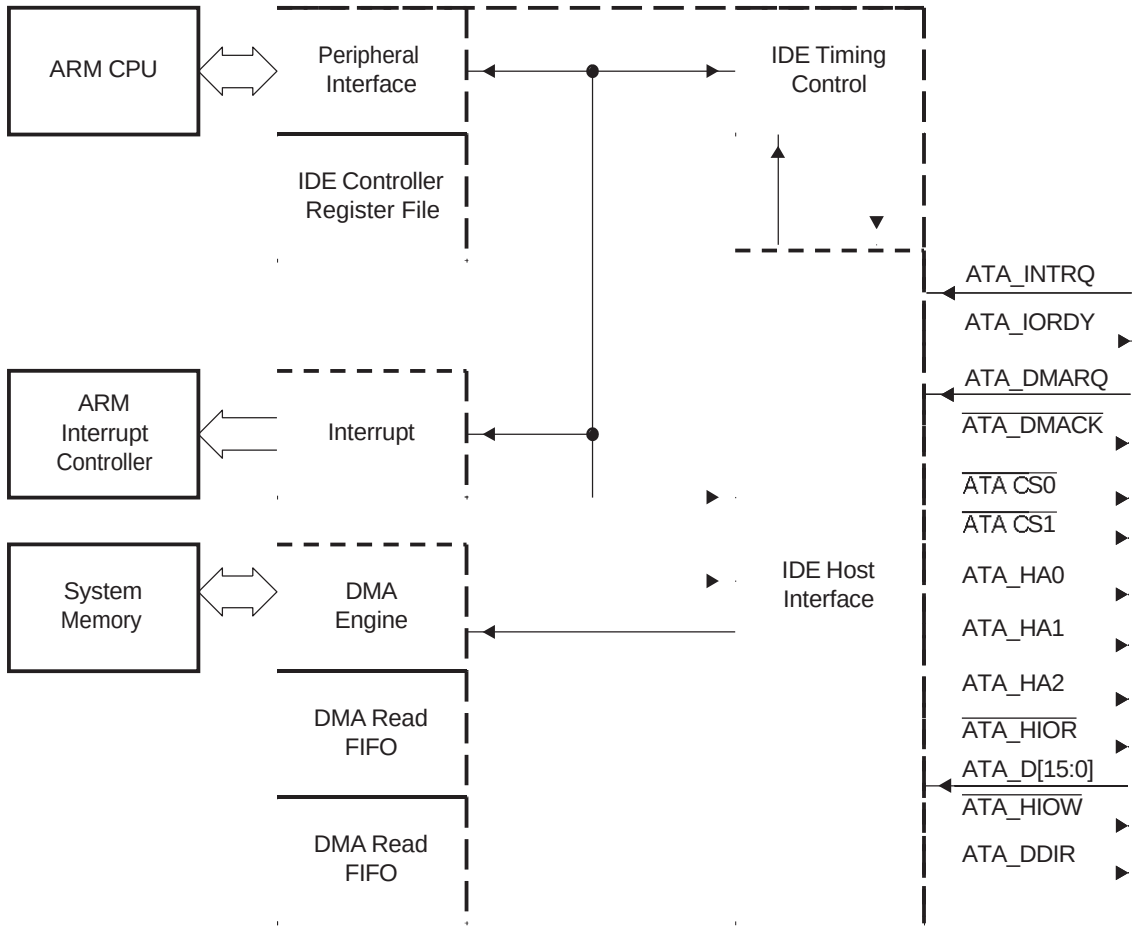
The IDE host controller logic supports PIO, multiword DMA and ultra-DMA (ultra-ATA) modes. The ATA controller has the following features:

- 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
$\overline{\text{ATA\_CS0}}$ ATA_CS1	Output	Chip Select Signals 0 and 1 These are the chip select signals from the host used to select the Command Block or Control Block registers. ATA_CS0 is asserted during Command Block register accesses. ATA_CS1 is asserted during Control Block register accesses. When ATA_DMACK is asserted, ATA_CS0 and ATA_CS1 are both asserted and transfers will be 16 bits wide.
ATA_HA0 ATA_HA1 ATA_HA2	Output	Device Address Bits[2:0] This is a 3-bit binary coded address asserted by the host to access a register or data port in the attached device.
ATA_D[15:0]	Input/Output	Host Read/Write Data Bus 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, ATA_D[7:0] will be used for 8-bit register transfers.
ATA_DMARQ	Input	Device DMA Request 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 ATA_HIOR and ATA_HIOW. This signal is used in a handshake manner with ATA_DMACK, that is, the device will wait until the host asserts ATA_DMACK before negating ATA_DMARQ, and re-asserting ATA_DMARQ if there is more data to transfer. For multiword DMA transfers, the ATA_DMARQ/ATA_DMACK handshake is used to provide flow control during the transfer. For ultra-DMA, the ATA_DMARQ/ATA_DMACK handshake is used to indicate when the function of interface signals changes. This signal will be released when the device is not selected.
$\overline{\text{ATA\_DMACK}}$	Output	Host DMA Acknowledge This signal is used by the host in response to ATA_DMARQ to initiate DMA transfers. For multiword DMA transfers, the ATA_DMARQ/ATA_DMACK handshake is used to provide flow control during the transfer. For ultra-DMA, the ATA_DMARQ/ATA_DMACK handshake is used to indicate when the function of interface signals changes. When $\overline{\text{ATA\_DMACK}}$ is asserted, $\overline{\text{ATA\_CS0}}$ and $\overline{\text{ATA\_CS1}}$ is not asserted and transfers are 16 bits wide.
ATA_IORDY	Input	Device I/O ready during PIO transaction DMA ready during ultra-DMA write DMA strobe during ultra-DMA read ATA_IORDY 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 ATA_IORDY signal. Devices that support PIO modes 3 and above are required by the ATA/ATAPI specification to support ATA_IORDY. 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 ATA_IORDY to pause an Ultra DMA data-out burst. 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 ATA_IORDY latch the data from ATA_D[15:0] into the host. The device may stop generating ATA_IORDY edges to pause an ultra DMA data-in burst. This signal is released when the device is not selected.

Table 1. Supported ATA Controller Signals (continued)

Terminal Name	Direction from IDE Controller	Description
ATA_HIOR	Output	PIO Read Transaction Indicator DMA Ready during Ultra-DMA Read DMA Data Strobe during Ultra-DMA Write
		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.
		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.
		When performing ultra-DMA data write transaction, the host uses the ATA_HIOR to strobe the ultra-DMA data-out burst.
		Both the rising and falling edges of ATA_HIOR 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.
ATA_HIOW	Output	PIO Write Transaction Indicator Stop Ultra-DMA Data Read/Write Bursts
		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.
		The host will negate ATA_HIOW prior to initiation of an ultra DMA burst. The host will negate ATA_HIOW 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.
ATA_INTRQ	Input	Attached Device Interrupt Request
		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.
		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.
		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 ATA_HIOW 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 ATA_HIOW that writes the device register.
		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.
ATA_DDIR	Output	External Level Shifter Direction Indicator
		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.
		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.

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

Name	Description
$\overline{\text{RESET}}$	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 ı10.10 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 ı20 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 ı15 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 ı10.10 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 ı10.10 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 ı30 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 ı10.10 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 ı 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 HWNHLD*n*P 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 $\overline{\text{ATA\_HIOR}}$ and $\overline{\text{ATA\_HIOW}}$ 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 HWNHLD*n*P 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 ɔ15 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 ɔ15 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 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 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 ɔ10.10 ns = 70.70 ns). The minimum requirement on t_{KW} is 25 ns, this translates to a minimum of 3 clock cycles (3 cycles ɔ10.10 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 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 = 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 ɔ10.10 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 ɔ10 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 ɔ10 ns = 20.20 ns to 5 ɔnbsp;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 ɔ15 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 ɔ15 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 ɔ15 ns = 30 ns to 4 cycles ɔnbsp;15 ns = 60 ns).

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