National Semiconductor

PRELIMINARY
March 1996

PC87415
PCI-IDE DMA Master Mode Interface Controller

1.0 General Description

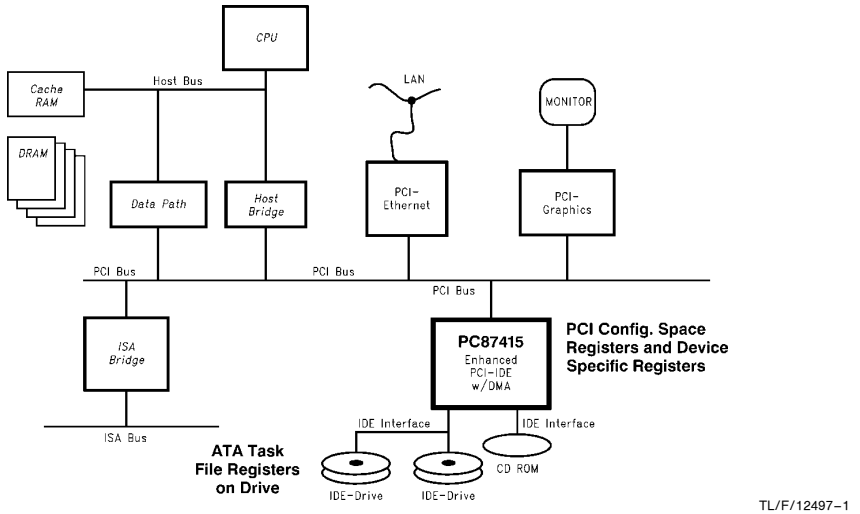
The Enhanced PCI-IDE Interface is a single-chip controller packaged in a 100-pin PQFP. It provides 2 IDE channels for interfacing up to 4 IDE drives, or 2 IDE drives and CD-ROM directly on the PCI Local bus. An enhanced DMA controller on-chip increases system performance by providing full scatter/gather data transfers between IDE devices and system memory without CPU intervention. Four levels of both write posting and read prefetching per channel allow the host CPU to run concurrently with IDE cycles. Programmable timing functions provide maximum flexibility of timing parameters per drive for optimizing the data transfer rate per drive. Both PC compatible addressing and PCI compliant addressing are supported by re-mapping the base addresses. A power control feature allows turning off power to the IDE cables.

The Enhanced PCI-IDE Interface connection to the PCI bus is virtually ''glue-less'', with only one additional TTL data buffer (optional). This high-integration solution reduces component count, eases board design, reduces cost and increases reliability.

The Enhanced PCI-IDE supports faster ATA devices using PIO modes 1, 2, 3, and 4 as well as DMA modes 0, 1, and 2. It comes with a full suite of software drivers for DOS 5.0–6.x, Windows® 3.x, Windows® 95, Windows NT™, OS/2 2.x, Novell® NetWare™ 3.1x–4.x, and SCO UNIX® 3.x.

2.0 Features

- PCI bus interface for up to 4 IDE devices
- 33 MHz, 32-bit PCI bus data path with full parity error reporting
- 16.7 MByte/sec maximum IDE transfer rate
- Support for 2 IDE channels (@ 2 IDE devices per channel)
- Primary or secondary IDE addressing (1F0x/170x) in PC compatible mode
- Re-mappable base registers for full PCI compliance
- Concurrent channel operation (PIO & DMA modes)
- 4 Double Word write FIFO per channel
- 4 Double Word read prefetch FIFO per channel
- Enhanced DMA mode with scatter/gather capability
- ANSI ATA Modes 0 through 4 PIO support (internal DMA not selected)
- IORDY handshaking for PIO
- ANSI ATA Modes 0 through 2 Multiword DMA support (internal DMA selected)
- Individually programmable command and recovery timing for reads and writes per channel/drive for command, control and data
- Individually programmable data sector size for read prefetches per channel
- PC compatible interrupt routing of IRQ14 and IRQ15
- Hardware and software chip enable/disable
- Optional Power Control for IDE Drives
- Fully static logic design
- 100-pin PQFP package



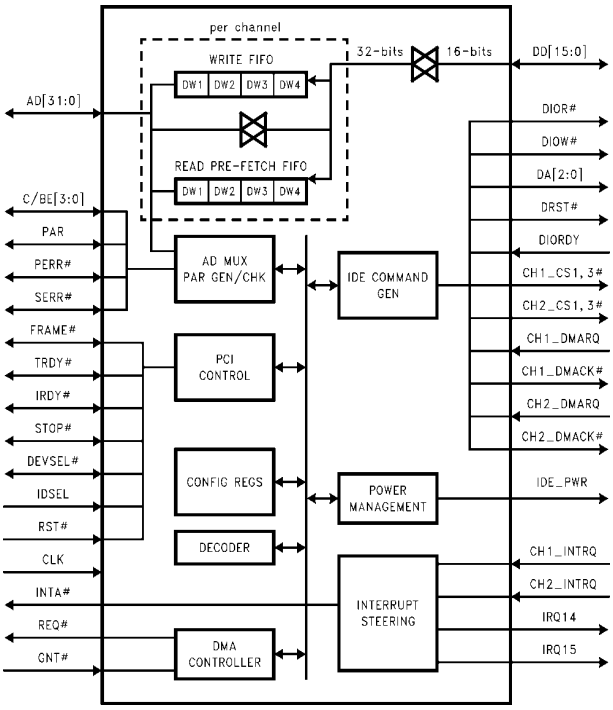
TL/F/12497–1

FIGURE 1. The PC87415 in a PCI Based System

Table of Contents

3.0 System Diagram

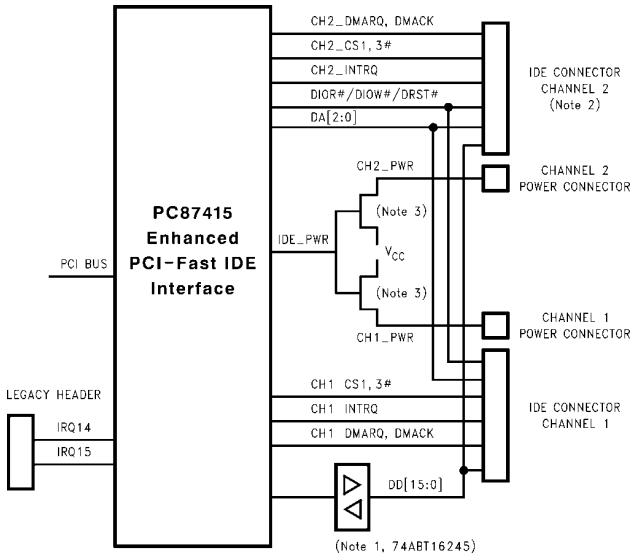
The following diagram shows the functional blocks and associated pins within PC87415.



TL/F/12497–2

FIGURE 2. PC87415 Block Diagram

The following diagram shows how the PCI-IDE interface is used in a system.



TL/F/12497–38

Note 1: Transceivers are optional.

Note 2: Second IDE connector is optional.

Note 3: Transistors are optional.

FIGURE 3. PC87415 System Connections

4.0 Pin Description

4.1 PCI INTERFACE

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>AD[31:0]</td><td>I/O</td><td>MULTIPLEXED ADDRESS AND DATA. The direction of these pins are defined below:
<table>
<tr><th>Phase</th><th>Target</th><th>Bus Master</th></tr>
<tr><td>Address Phase</td><td>Input</td><td>Output</td></tr>
<tr><td>Data Phase:</td><td></td><td></td></tr>
<tr><td>-Read</td><td>Output</td><td>Input</td></tr>
<tr><td>-Write</td><td>Input</td><td>Output</td></tr>
</table></td></tr>
<tr><td>C/BE[3:0]</td><td>I</td><td>COMMAND/BYTE ENABLE are multiplexed Bus commands and Byte enables.</td></tr>
<tr><td>PAR</td><td>I/O</td><td>PARITY is even parity across AD[31:0] and C/BE[3:0]. PAR is an input when AD[31:0] is an input; it is an output when AD[31:0] is an output.</td></tr>
<tr><td>FRAME#</td><td>I/0</td><td>CYCLE FRAME is driven by the initiator to indicate the beginning and duration of an access.</td></tr>
<tr><td>TRDY#</td><td>I/O</td><td>TARGET READY indicates that the current data phase of the transaction is ready to be completed.</td></tr>
<tr><td>IRDY#</td><td>I/O</td><td>INITIATOR READY indicates that the inltiator is ready to complete the current data phase of the transaction.</td></tr>
<tr><td>STOP#</td><td>O</td><td>STOP indicates that the current target is requesting the initiator to stop the current transaction.</td></tr>
<tr><td>DEVSEL#</td><td>I/O</td><td>DEVICE SELECT, when actively driven, indicates the driving device has decoded its address as the target of the current access.</td></tr>
<tr><td>IDSEL</td><td>I</td><td>INITIALIZATION DEVICE SELECT is used as a chip select during configuration read and write transactions. This input can be connected to one of the upper address lines AD[31:11].</td></tr>
<tr><td>PERR#</td><td>I/O</td><td>PARITY ERROR is used for reporting data parity errors during all PCI transactions, except a Special Cycle. PERR# is an output when AD[31:0] and PAR are inputs, it is an input when AD[31:0] and PAR are outputs.</td></tr>
<tr><td>SERR#</td><td>O</td><td>SYSTEM ERROR is used for reporting address parity errors, data parity errors on the Special Cycle command, or any other system error where the result will be catastrophic. When reporting address parity errors, SERR# is an output.</td></tr>
<tr><td>INTA#</td><td>O</td><td>INTERRUPT. Interrupt request A.</td></tr>
<tr><td>REQ#</td><td>O</td><td>REQUEST indicates to the bus arbiter that this device wants to use the bus and become a bus master.</td></tr>
<tr><td>GNT#</td><td>I</td><td>GRANT indicates to this device that access to the bus has been granted.</td></tr>
<tr><td>CLK</td><td>I</td><td>CLOCK. 33 MHz PCI Clock.</td></tr>
<tr><td>RST#</td><td>I</td><td>RESET. PCI Reset.</td></tr>
</table>

4.0 Pin Description (Continued)

4.2 IDE INTERFACE

Name	Type	Description
DD[15:0]	I/O	**DRIVE DATA BUS.** The 16-bit bi-directional data bus to the drive. The lower 8 bits are used for non data 8-bit transfers (e.g., registers, ECC bytes).
DA[0]/ TEST#	I/O	**DRIVE ADDRESS LINE 0 OR TEST.** The DA0 address line is asserted by the host to access a register or data port in the drive. It may also be used for testing: during PCI Reset, the rising edge of RST# samples this pin. If ''low'', all device output pins are forced to a TRI-STATE® level.
DA[1]/ ENABLE	I/O	**DRIVE ADDRESS LINE 1 OR ENABLE.** During normal operation, the DA1 address line is asserted by the host to access a register or data port in the drive. During PCI Reset, the rising edge of RST# samples this pin and places its value in the Command Register bit-0.
DA[2]/ LEGACY#	I/O	**DRIVE ADDRESS LINE 2 OR LEGACY.** Dining normal operation, the DA2 address line is asserted by the host to access a register or data port in the drive. During PCI Reset, the rising edge of RST# samples this pin and places its value in the Programming Interface Register bits 0 and 2.
DIORDY	I	**DRIVE I/O CHANNEL READY.** When the drive is not ready to respond to a data transfer request, this signal is negated (low) to extend the disk transfer cycle of any register access (read or write). When DIORDY is not negated, it remains in a high impedance state.
DIOR#	O	**DRIVE I/O READ.** The read strobe signal for both channels. The falling edge of DIOR# enables data from a register or the data port of the drive onto the PCI-IDE chip.
DIOW#	O	**DRIVE I/O WRITE.** The write strobe signal for both channels. The rising edge of DIOW# clocks data from the PCI-IDE chip into the register or the data port of the drive.
DRST#	O	**DRIVE RESET.** This signal from the PCI-IDE chip is asserted after power up or under software control (see Control Register bits Table A). It is active for as long as the PCI Reset signal, or if the reset bit in the Control Register is set.
CH1__CS1#, CH1__CS3#	O	**CHANNEL 1 CHIP SELECT 1 AND 3.** CH1__CS1# is the chip select signal to select the Command Block Registers. CH1__CS3# is the chip select signal to select the Control Block Registers.
CH2__CS1#, CH2__CS3#	O	**CHANNEL 2 CHIP SELECT 1 AND 3.** CH2__CS1# is the chip select signal to select the Command Block Registers. CH2__CS3# is the chip select signal to select the Control Block Registers.
CH1__INTRQ CH2__INTRQ	I	**DRIVE INTERRUPTS.** These signals are used to interrupt the host system. CH1__INTRQ is asserted only when the drive(s) on channel 1 has a pending interrupt, and the host has cleared nIEN in the drive's Device Control Register. CH2__INTRQ is asserted only when the drive(s) on channel 2 has a pending interrupt, and the host has cleared nIEN in the Device Control Register.
CH1__ DMARQ	I	**CHANNEL 1 DMA REQUEST.** This signal is used when using the internal DMA controller. When this signal is asserted, the selected drive on channel 1 is ready to transfer data.
CH2__ DMARQ	I	**CHANNEL 2 DMA REQUEST.** This signal is used when using the internal DMA controller. When this signal is asserted, the selected drive on channel 2 is ready to transfer data.
CH1__ DMACK#	O	**CHANNEL 1 DMA ACKNOWLEDGE.** This signal is used when using the internal DMA controller. When asserted, it signals the selected drive on channel 1 that data has been accepted, or that data is available.
CH2__ DMACK#	O	**CHANNEL 2 DMA ACKNOWLEDGE.** This signal is used when using the internal DMA controller. When asserted, it signals the selected drive on channel 2 that data has been accepted, or that data is available.
IRQ14	O	**IRQ14** mirrors CH1__INT if Legacy mode is enabled.
IRQ15	O	**IRQ15** mirrors CH2__INT if Legacy mode is enabled.

4.0 Pin Description (Continued)

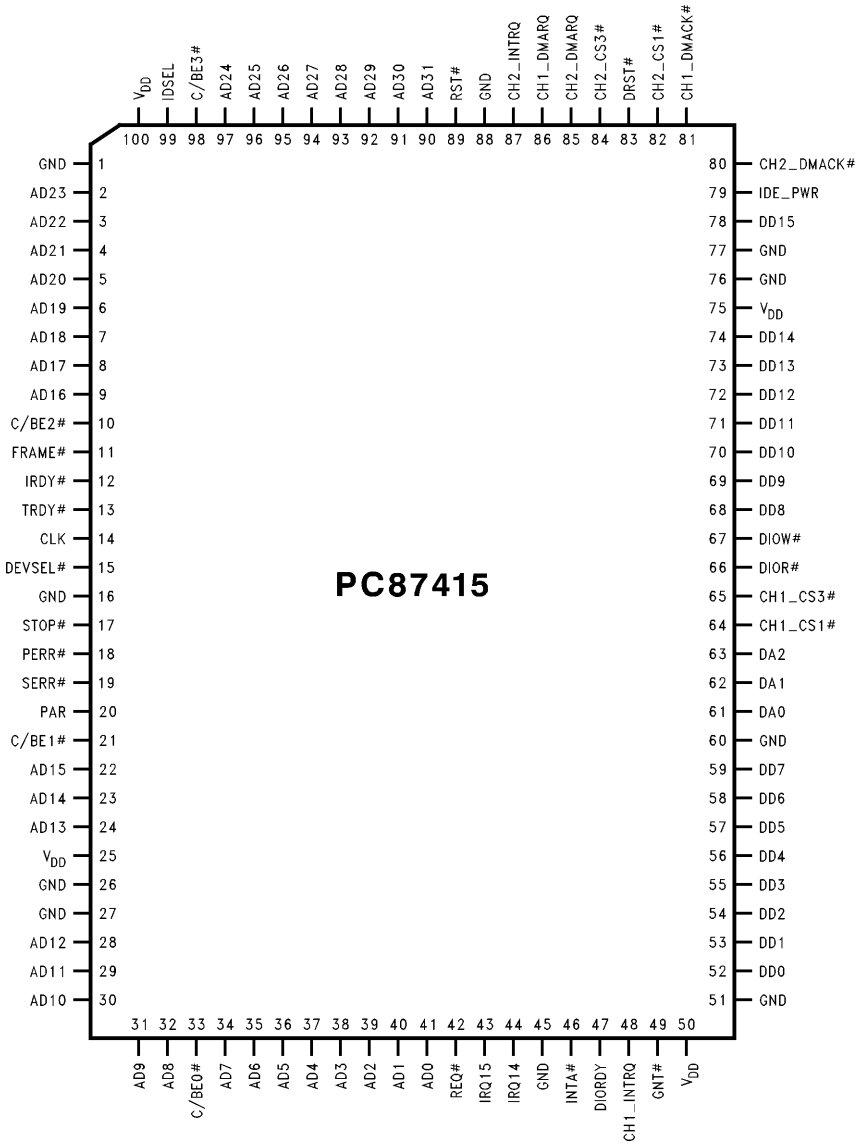
4.3 POWER CONTROL

Name	Type	Description
IDE__PWR	O	**IDE Power.** During power up, this signal is used to disable power to the IDE cables. It may also be programmed on or off (see Control Register bits 3 and 18). It is active for as long as the PCI reset signal, or if the reset bit in the Control Register is set.

4.4 POWER AND GROUND

Name	Type	Description
GND	I	**V_{SS} or GROUND**
V_{DD}	I	**+5V**

4.5 PINOUT



TL/F/12497–28

5.0 Configuration Registers

Reg. # (HEX)	R/W	Description
00–01	R/W	**VENDOR ID**—100Bh
02–03	R/W	**DEVICE ID**—0002h
04–05	R/W	**COMMAND REGISTER (CMD).** The command register provides coarse control over the device's ability to generate and respond to PCI cycles. Bit 0: IO Space. This bit is used to enable the device to respond to PCI I/O cycles. The default value is the sampled value of the ENABLE pin (sampled on the rising edge of RST#). 0: Device is disabled 1: Device is enabled Bit 1: Not used. 0 during reads. Bit 2: Bus Master. Controls the device's ability to act as a master on the PCI bus. 0: Device cannot be bus master. This will disable all DMA operations even if internal DMA is enabled. (default value) 1: Device can become a bus master. If this bit is set and internal DMA is enabled then the device can become a bus master. Bit 3–5: Not used. 000 during reads Bit 6: Parity Error (PERR#) response 0: Ignore parity error (default) 1: Respond to parity error Bit 7: Not used. 0 during reads Bit 8: System Error (SERR#) response 0: Disable system error checking (default) 1: Enable system error checking Bit 9: Not used. 0 during reads. Fast back-to-back transactions to same agent. Bit 10–15: Not used. 000000 during reads.
06–07	R/W	**STATUS REGISTER (SR).** The status register is used to record status information for PCI bus related events. Reads to this register behave normally. Writes to this register cause bit(s) to be reset. A bit is reset whenever the register is written with a ''1'' in the corresponding bit location. Bit 0–6: Reserved Bit 7: Always 0. Bit 8: Data Parity Detected. 0: No parity detected 1: Parity detected Bit 9–10: DEVSEL# timing 00: Reserved 01: Medium decode (default) Bit 11: Signaled Target Abort 0: The device did not terminate a transaction with target abort. 1: The device has terminated a transaction with target abort Bit 12: Receive Target Abort 0: The device has not received a target abort 1: The device has received a target abort when it was a master. Bit 13: Received Master Abort 0: Transaction was not terminated with a master abort 1: Transaction has been terminated with a master abort Bit 14: Signaled System Error (SERR#) 0: System error was not signaled 1: System error was signaled Bit 15: Detected Parity Error (PERR#) 0: No parity error detected 1: Parity error detected

5.0 Configuration Registers (Continued)

Reg. # (HEX)	R/W	Description
08	R	**REV ID**—01h
09	R/W	**PROGRAMMING INTERFACE (PIF)** Bit 0: This bit determines what addresses channel 1 responds to. The default value is the sampled value of the LEGACY# pin (sampled on the rising edge of RST#). 0: Channel 1 responds to addresses 1F0–1F7 and 3F6. Base registers 0 and 1 (10h–17h) are not used—legacy mode 1: Channel 1 responds to addresses programmed in base register 0 and 1—native mode Bit 1: Always 1. Channel 1 has programmable selection of modes. Bit 0 determines which mode (default). This bit is read only. Bit 2: This bit determines what addresses channel 2 responds to. The default value is the sampled value of the LEGACY# pin (sampled on the rising edge of RST#). 0: Channel 2 responds to addresses 170–177 and 376. Base registers 2 and 3 (18h–1Fh) are not used—legacy mode 1: Channel 2 responds to addresses programmed in base register 2 and 3—native mode Bit 3: Always 1. Channel 2 has programmable selection of modes. Bit 2 determines which mode (default). This bit is read only. Bit 4–6: Reserved Bit 7 Always 1. Indicating that the device supports Master IDE.
0A	R	**SUB-CLASS CODE**—01h (IDE controller)
0B	R	**CLASS CODE**—01h (Mass Storage controller)
0C		Not used
0D	R/W	**LATENCY TIMER.** This register specifies, in units of PCI bus clocks, the value of the 8 most significant bits of an 11-bit Latency Timer for this device when the internal DMA controller is used and the device is a bus master.
0E	R	**HEADER TYPE**—00h
0F	R	**BIST**—00h
10–13	R/W	**BASE ADDRESS REGISTER 0 (BAR0).** Used for channel 1 data/command block accesses if channel 1 is programmed to be in native mode. The device decodes 8 bytes of address space for data/command block accesses. This register is always read/write, regardless of the setting of the Native/Legacy bit in the PIF. Bit 0: always 1 Bit 1: always 0 Bit 2: always 0
14–17	R/W	**BASE ADDRESS REGISTER 1 (BAR1).** Used for channel 1 control block access if channel 1 is programmed to be in native mode. The device decodes 4 bytes of address space for control block accesses. This register is always read/write, regardless of the setting of the Native/Legacy bit in the PIF. Bit 0: always 1 Bit 1: always 0
18–1B	R/W	**BASE ADDRESS REGISTER 2 (BAR2).** Used for channel 2 data/command block accesses if channel 2 is programmed to be in native mode. The device decodes 8 bytes of address space for data/command block accesses. This register is always read/write, regardless of the setting of the Native/Legacy bit in the PIF. Bit 0: always 1 Bit 1: always 0 Bit 2: always 0

5.0 Configuration Registers (Continued)

Reg. # (HEX)	R/W	Description
1C–1F	R/W	**BASE ADDRESS REGISTER 3 (BAR3).** Used for channel 2 control block access if channel 2 is programmed to be in native mode. The device decodes 4 bytes of address space for control block accesses. This register is always read/write, regardless of the setting of the Native/Legacy bit in the PIF. Bit 0: always 1 Bit 1: always 0
20–23	R/W	**BASE ADDRESS REGISTER 4 (BAR4).** Used to address the Bus Master IDE control and status registers. Bit 0: always 1 Bit 1: always 0 Bit 2: always 0 Bit 3: always 0
24–3B		**NOT USED**
3C	R/W	**INTERRUPT LINE** (default is 0Eh)
3D	R	**INTERRUPT PIN** Bit 0: Always 1. Designates INTA. Bit 1–7: 0000000
3E–3F		**NOT USED**

5.0 Configuration Registers (Continued)

<table>
<tr><th>Reg. # (HEX)</th><th>R/W</th><th>Description</th></tr>
<tr><td>40–42</td><td>R/W</td><td>CONTROL REGISTER (CTRL). Default value is 00000000h (after Reset)
Bit 0: Reserved
Bit 1: 0 (Always): Reserved
Bit 2: 0 = Software reset to CH1/CH2 off
1 = Software reset to CH1/CH2 on
Bit 3: 0 = IDE__PWR set on
1 = IDE__PWR set off
Note: Bit 18 must be set to “1” to allow Bit 3 to toggle IDE__PWR.
<table><tr><th>B18</th><th>B3</th><th>IDE__PWR</th></tr><tr><td>0</td><td>X</td><td>1 (Default at power up reset)</td></tr><tr><td>1</td><td>0</td><td>1 (IDE__PWR set to on)</td></tr><tr><td>1</td><td>1</td><td>0 (IDE__PWR set to off)</td></tr></table>(*) Asserted for t__25 s.</td></tr>
<tr><td></td><td></td><td>Bit 4: 0 = Map CH1__INT according to legacy/native mapping scheme
1 = Map CH1__INT to INTA regardless if in legacy mode
Bit 5: 0 = Map CH2__INT according to legacy/native mapping scheme
1 = Map CH2__INT to INTA regardless if in legacy mode
Bit 6: 0 = INTA# unmasked
1 = INTA# masked
Bit 7: 0 = Write to vendor ID and device ID registers disabled
1 = Write to vendor ID and device ID registers enabled
Bit 8: 0 = Channel 1 interrupt unmasked
1 = Channel 1 interrupt masked
Bit 9: 0 = Channel 2 interrupt unmasked
1 = Channel 2 interrupt masked
Bit 10: 0 = PCI Base Address Register 2 and 3 enabled
1 = PCI Base Address Register 2 and 3 disabled
Bit 11: 0 = PCI data phase WATCHDOG™ timer disabled
1 = PCI data phase WATCHDOG timer enabled</td></tr>
</table>

5.0 Configuration Registers (Continued)

Reg. # (HEX)	R/W	Description
40–42 (Continued)	R/W	**CONTROL REGISTER**—(Continued) Bit 12: 0 = All accesses mapped to Base Address Register 10 and 14 bypass the data buffers 1 = IDE device accesses (Control Register bit 14 = 0) to offset 0 of the base address window register 10 and 14 are buffered Bit 13: 0 = All accesses mapped to Base Address Register 18 and 1C bypass the data buffers 1 = IDE device accesses (Control Register bit 15 = 0) to offset 0 of the base address window register 18 and 1C are buffered Bit 14: 0 = IDE device mapped to base address window registers 10 and 14 1 = Non-IDE device mapped to base address window registers 10 and 14 Bit 15: 0 = IDE device mapped to base address window registers 18 and 1C 1 = Non-IDE device mapped to base address window registers 18 and 1C Bit 16: 0 = Channel 1 prefetch buffer disabled 1 = Channel 1 prefetch buffer enabled Bit 17: 0 = Channel 2 prefetch buffer disabled 1 = Channel 2 prefetch buffer enabled Bit 18: 0 = Reset/idle State. 1 = IDE__PWR has power on/off function Bit 19: Reserved Bit 29: 0 = Channel 1 drive 1 IORDY handshaking for flow control 1 = Channel 1 drive 1 DMARQ/DMACK handshaking for flow control Bit 21: 0 = Channel 1 drive 2 IORDY handshaking for flow control 1 = Channel 1 drive 2 DMARQ/DMACK handshaking for flow control Bit 22: 0 = Channel 2 drive 1 IORDY handshaking for flow control 1 = Channel 2 drive 1 DMARQ/DMACK handshaking for flow control Bit 23: 0 = Channel 2 drive 2 IORDY handshaking for flow control 1 = Channel 2 drive 2 DMARQ/DMACK handshaking for flow control
43	R	**WRITE BUFFER STATUS.** This status information is used for power management. Before the software attempts to turn off power to the IDE drives it must ensure that the write buffer is empty. Bit 0 = 0: Channel 1 write buffer empty 1: Channel 1 write buffer not empty Bit 1 = 0: Channel 2 write buffer empty 1: Channel 2 write buffer not empty

5.0 Configuration Registers (Continued)

Reg. # (HEX)	R/W	Description
44	R/W	**CHANNEL 1 DEVICE 1 DATA READ TIMING REGISTER** Bit 3–0: Command active time (in PCI CLKs) Default = 0101 (Mode 0) 1111 = 2 CLKs 1110 = 3 CLKs 1101 = 4 CLKs : : 0001 = 16 CLKs 0000 = 17 CLKs Bit 7–4: Command recovery time (in PCI CLKs) Default = 1000 (Mode 0) 1111 = 1 CLKs 1110 = 2 CLKs 1101 = 3 CLKs : : 0001 = 15 CLKs 0000 = 16 CLKs
45	R/W	**CHANNEL 1 DEVICE 1 DATA WRITE TIMING REGISTER** Bit 3–0: Command active time (in PCI CLKs) Default = 0101 (Mode 0) 1111 = 2 CLKs 1110 = 3 CLKs 1101 = 4 CLKs : : 0001 = 16 CLKs 0000 = 17 CLKs Bit 7–4: Command recovery time (in PCI CLKs) Default = 1000 (Mode 0) 1111 = 1 CLKs 1110 = 2 CLKs 1101 = 3 CLKs : : 0001 = 15 CLKs 0000 = 16 CLKs
48	R/W	**CHANNEL 1 DEVICE 2 DATA READ TIMING REGISTER** Bit 3–0: Command active time (in PCI CLKs) Default = 0101 (Mode 0) 1111 = 2 CLKs 1110 = 3 CLKs 1101 = 4 CLKs : : 0001 = 16 CLKs 0000 = 17 CLKs Bit 7–4: Command recovery time (in PCI CLKs) Default = 1000 (Mode 0) 1111 = 1 CLKs 1110 = 2 CLKs 1101 = 3 CLKs : : 0001 = 15 CLKs 0000 = 16 CLKs

5.0 Configuration Registers (Continued)

Reg. # (HEX)	R/W	Description
49	R/W	**CHANNEL 1 DEVICE 2 DATA WRITE TIMING REGISTER** Bit 3–0: Command active time (in PCI CLKs) Default = 0101 (Mode 0) 1111 = 2 CLKs 1110 = 3 CLKs 1101 = 4 CLKs : : 0001 = 16 CLKs 0000 = 17 CLKs Bit 7–4: Command recovery time (in PCI CLKs) Default = 1000 (Mode 0) 1111 = 1 CLKs 1110 = 2 CLKs 1101 = 3 CLKs : : 0001 = 15 CLKs 0000 = 16 CLKs
4C	R/W	**CHANNEL 2 DEVICE 1 DATA READ TIMING REGISTER** Bit 3–0: Command active time (in PCI CLKs) Default = 0101 (Mode 0) 1111 = 2 CLKs 1110 = 3 CLKs 1101 = 4 CLKs : : 0001 = 16 CLKs 0000 = 17 CLKs Bit 7–4: Command recovery time (in PCI CLKs) Default = 1000 (Mode 0) 1111 = 1 CLKs 1110 = 2 CLKs 1101 = 3 CLKs : : 0001 = 15 CLKs 0000 = 16 CLKs
4D	R/W	**CHANNEL 2 DEVICE 1 DATA WRITE TIMING REGISTER** Bit 3–0: Command active time (in PCI CLKs) Default = 0101 (Mode 0) 1111 = 2 CLKs 1110 = 3 CLKs 1101 = 4 CLKs : : 0001 = 16 CLKs 0000 = 17 CLKs Bit 7–4: Command recovery time (in PCI CLKs) Default = 1000 (Mode 0) 1111 = 1 CLKs 1110 = 2 CLKs 1101 = 3 CLKs : : 0001 = 15 CLKs 0000 = 16 CLKs

5.0 Configuration Registers (Continued)

Reg. # (HEX)	R/W	Description
50	R/W	**CHANNEL 2 DEVICE 2 DATA READ TIMING REGISTER** Bit 3–0: Command active time (in PCI CLKs) Default = 0101 (Mode 0) 1111 = 2 CLKs 1110 = 3 CLKs 1101 = 4 CLKs : : 0001 = 16 CLKs 0000 = 17 CLKs Bit 7–4: Command recovery time (in PCI CLKs) Default = 1000 (Mode 0) 1111 = 1 CLKs 1110 = 2 CLKs 1101 = 3 CLKs : : 0001 = 15 CLKs 0000 = 16 CLKs
51	R/W	**CHANNEL 2 DEVICE 2 DATA WRITE TIMING REGISTER** Bit 3–0: Command active time (in PCI CLKs) Default = 0101 (Mode 0) 1111 = 2 CLKs 1110 = 3 CLKs 1101 = 4 CLKs : : 0001 = 16 CLKs 0000 = 17 CLKs Bit 7–4: Command recovery time (in PCI CLKs) Default = 1000 (Mode 0) 1111 = 1 CLKs 1110 = 2 CLKs 1101 = 3 CLKs : : 0001 = 15 CLKs 0000 = 16 CLKs

5.0 Configuration Registers (Continued)

Reg. # (HEX)	R/W	Description
54	R/W	**COMMAND AND CONTROL BLOCK TIMING REGISTER** Bit 0–3: Command active time (in PCI CLKs) Default = 0111 (Mode 0) 1111 = Reserved 1110 = 3 CLKs 1101 = 4 CLKs : : 0001 = 16 CLKs 0000 = 17 CLKs Bit 4–7: Command recovery time (in PCI CLKs) Default = 1011 (Mode 0) 1111 = 3 CLKs 1110 = 4 CLKs 1101 = 5 CLKs : : 0001 = 17 CLKs 0000 = 18 CLKs
55	R/W	**SECTOR SIZE** Bit 3–0: Channel 1 sector size 1111: Not used 1110: 512 bytes 1100: 1024 bytes 1000: 2048 bytes 0000: 4096 bytes Bit 7–4: Channel 2 sector size 1111: Not used 1110: 512 bytes 1100: 1024 bytes 1000: 2048 bytes 0000: 4096 bytes

6.0 Bus Master IDE Control and Status Registers

Offset from BAR4	R/W	Description
00	R/W	**CHANNEL 1 BUS MASTER IDE COMMAND REGISTER.** Default value is 00h. Bit 0: Start/Stop Bus Master 0: Stop Bus Master transfers. Master operation can be halted by writing a 0 to this bit. All state information is lost when a 0 is written. 1: Start Bus Master transfers. This enables bus master operation of the controller. Bus master operation begins when this bit is detected changing from zero to one. The controller will transfer data between the IDE device and memory only when this bit is set. Bit 1–2: Reserved Bit 3: Read or Write Control 0: PCI bus master read 1: PCI bus master write Bit 4–7: Reserved
01		**RESERVED**
02	R/W	**CHANNEL 1 BUS MASTER IDE STATUS REGISTER.** Writes to this register cause bit(s) to be reset. Bits 1 and 2 are reset wherever the register is written with a ''1'' in the corresponding bit location. Default value is 00h. Bit 0: Bus Master IDE Active. This bit is set when the Start bit is written to the Bus Master IDE Command Register. This bit is cleared when the last transfer for a region is performed, where the EOT for that region is set in the region descriptor. It is also cleared when the Start bit is cleared in the Bus Master IDE Command Register. Bit 1: Error. This bit is set when the controller encounters an error in transferring data to/from memory. This bit is cleared when a 1 is written to it by software. Bit 2: Interrupt. This bit is set by the rising edge of CH1__INT. This bit is cleared when a 1 is written to it by software. Bit 3–4: Reserved Bit 5: Drive 1 DMA Capable. Bus Master IDE transfers to drive 1 on channel 1 are qualified by this bit. 0: Drive 1 is not DMA capable 1: Drive 1 is DMA capable Bit 6: Drive 2 DMA Capable. Bus Master IDE transfers to drive 2 on channel 1 are qualified by this bit. 0: Drive 2 is not DMA capable 1: Drive 2 is DMA capable Bit 7: Simplex only. This read-only bit is always 0 indicating that both channels can operate independently and can be used at the same time.
03		Reserved
04–07	R/W	**CHANNEL 1 BUS MASTER IDE PRD TABLE ADDRESS.** Default value is 00000000h. Bit 0–1: Reserved Bit 2–31: Base address of channel 1 Descriptor table. Corresponds to AD[31:2].
08	R/W	**CHANNEL 2 BUS MASTER IDE COMMAND REGISTER.** Default value is 00h. Bit 0: Start/Stop Bus Master 0: Stop Bus Master transfers. Master operation can be halted by writing a 0 to this bit. All state information is lost when a 0 is written. 1: Start Bus Master transfers. This enables bus master operation of the controller. Bus master operation begins when this bit is detected changing from zero to one. The controller will transfer data between the IDE device and memory only when this bit is set. Bit 1–2: Reserved Bit 3: Read or Write Control 0: PCI bus master read 1: PCI bus master write Bit 4–7: Reserved

6.0 Bus Master IDE Control and Status Registers (Continued)

Offset from BAR4	R/W	Description
09		**RESERVED**
0A	R/W	**CHANNEL 2 BUS MASTER IDE STATUS REGISTER.** Writes to this register cause bit(s) to be reset. Bits 1 and 2 are reset whenever the register is written with a ''1'' in the corresponding bit location. Default value is 00h. Bit 0: Bus Master IDE Active. This bit is set when the Start bit is written to the Bus Master IDE Command Register. This bit is cleared when the last transfer for a region is performed, where the EOT for that region is set in the region descriptor. It is also cleared when the Start bit is cleared in the Bus Master IDE Command Register. Bit 1: Error. This bit is set when the controller encounters an error in transferring data to/from memory. This bit is cleared when a 1 is written to it by software. Bit 2: Interrupt. This bit is set by the rising edge of CH2__INT. This bit is cleared when a 1 is written to it by software. Bit 3–4: Reserved Bit 5: Drive 1 DMA Capable. Bus Master IDE transfers to drive 1 on channel 2 are qualified by this bit. 0: Drive 1 is not DMA capable 1: Drive 1 is DMA capable Bit 6: Drive 2 DMA Capable. Bus Master IDE transfers to drive 2 on channel 2 are qualified by this bit. 0: Drive 2 is not DMA capable 1: Drive 2 is DMA capable Bit 7: Simplex only. This read-only bit is always 0 indicating that both channels can operate independently and can be used at the same time.
0B		Reserved
0C-0F	R/W	**CHANNEL 2 BUS MASTER IDE PRD TABLE ADDRESS.** Default value is 00000000h. Bit 0–1: Reserved Bit 2–31: Base address of channel 2 Descriptor table. Corresponds to AD[31:2].

7.0 Functional Description

The Enhanced PCI-IDE controller interfaces up to four IDE devices directly onto the PCI bus. The controller has 4 major blocks:

— PCI Interface
— Data Buffers
— IDE Interface
— DMA Controller

7.1 PCI INTERFACE

7.1.1 Commands

The device acts as either a target or a master on the PCI bus. The following PCI commands are recognized by this device when acting as a target:

— I/O Read
— I/O Write
— Configuration Read
— Configuration Write

When this device is a master it generates the following PCI commands:

— I/O Read
— I/O Write
— Memory Read
— Memory Write

All the I/O cycles are non-bursting (target or as a master). Memory Reads and Writes are all burst capable.

7.1.2 Termination

7.1.2.1 Target Abort in Target Mode

When the device is operating as a target (the IDE channel operating in PIO mode) it can generate a target abort under 3 conditions. These conditions are as follows:

Condition 1: During the address phase, a reserved or unsupported combination on the command inputs has been detected.

Condition 2: During an I/O write or read cycle an illegal combination has been detected on the byte enable inputs and the 2 least significant address lines.

Condition 3: During a configuration write or read cycle, the 2 least significant address lines have been non-zero. This means the configuration cycle address is not double word aligned.

Conditions 1 through 3 cause the target-abort bit to be set in the Status Register (bit 11).

7.1.2.2 Retry in Target Mode

When the device is a PCI target, a WATCHDOG™ timer is provided to prevent the device from holding the cycle for more than 32 PCI clocks. This timer is enabled through the Control Register bit 10. If the timer expires, a retry is generated on the PCI bus.

7.1.2.3 Target Abort in Master Mode

When the device is operating as a master (the IDE channel operating in DMA mode) and the target device issues a target abort, the master cycle will be terminated and the PCI Status register updated indicating that the master was terminated with a target abort.

7.1.2.4 Retry in Master Mode

When the device is doing a master cycle and the target responds with a retry, the cycle is terminated by de-asserting FRAME# and no data is transferred. The device then tries to continue to do the terminated cycle until successful.

7.1.2.5 Disconnect in Master Mode

When the device is doing a master burst cycle and the target issues a disconnect command, the master cycle terminates the burst cycle by de-asserting FRAME#. The device then continues to transfer the remainder of the burst cycle by asserting FRAME# again and tries to do another burst cycle.

7.1.2.6 Master Abort in Master Mode

When the device is operating as a master and the target doesn't respond with DEVSEL# by the end of the 6th PCI clock, the master cycle is terminated and a master abort is signaled by writing into the PCI Status register indicating that the master cycle was terminated.

7.2 DATA BUFFERS

7.2.1 Write Posting

The device provides 4 levels of write posting (16 bytes) per channel. If the buffer is not full, the TRDY# signal is returned to the PCI bus 1 clock after DEVSEL# is asserted during a write operation. This provides additional performance during transfers to the IDE devices. Writes to the data port are only posted. Writes to the command or control block by-pass the posting buffer. If the buffer is not empty and the CPU tries to either access the command or control port of the current channel, in PIO mode, the access is held off until the write buffers of the current channel are flushed out to the IDE bus and then the command or control port access is processed using the by-pass mode. If the channel is in DMA mode then the access to the non data port will go through at a higher priority than the buffered write access.

7.2.2 Prefetch

The device provides 4 levels of prefetch buffers (16 bytes) used for both channels. Reads to the command or control block will by-pass the prefetch buffer. If the buffer is not empty and the CPU tries to either access the command or control port of the current channel, in PIO mode, the access is held off until the prefetch buffers of the current channel are filled and then the command or control port access is processed using the by-pass mode. If the channel is in DMA mode, the access to the non data port will go through even though there is data in the prefetch buffer.

7.2.3 Two Channel Buffer Protocol

The device could respond to accesses to another channel while the buffer of the current channel is being filled or is full. Accesses to the 2 channels are interleaved, giving each channel equal access. Interleaving is done on a cycle by cycle basis. If only one channel is being accessed, then that IDE channel is given the full IDE bus bandwidth.

7.3 IDE INTERFACE

The device interfaces IDE drives onto the PCI bus. It intercepts I/O commands intended for the IDE drives and then emulates the IDE bus cycles to the drive interface at higher speeds (e.g., mode 4). The I/O port addresses monitored are either the legacy addresses or a PCI mapped I/O addresses. The IDE port definition is shown in Table I as defined in the ATA interface specification.

7.0 Functional Description (Continued)

TABLE I. ATA Port Definition

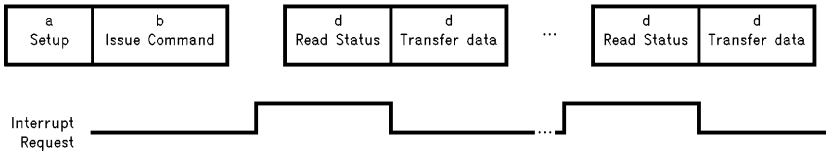
Address	Read Function	Write Function
COMMAND BLOCK REGISTERS		
1F0	Data (16 bits)	Data (16 bits)
1F1	Error Register	Features
1F2	Sector Count	Sector Count
1F3	Sector Number	Sector Number
1F4	Cylinder Low	Cylinder Low
1F5	Cylinder High	Cylinder High
1F6	Drive/Head	Drive/Head
1F7	Status	Command
CONTROL BLOCK REGISTERS		
3F6	Alternate Status	Device Control
3F7	Drive Address (and floppy DSKCHG)	Not Used

7.3.1 IDE Protocol

This section is provided as a reference only. The device passes these commands to the drive.

Commands that transfer data can be grouped into different classes according to the protocol followed for command execution. The different classes are:

— PIO Single sector read
— PIO Block Mode read
— PIO Single sector write
— PIO Block Mode write
— DMA Mode



TL/F/12497–3

The above sequence is repeated if accesses are directed to another channel.

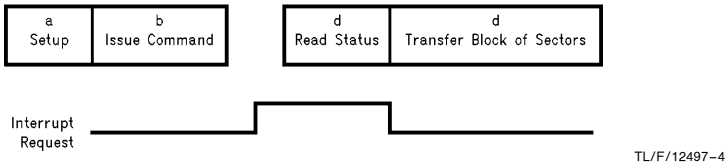
7.3.1.2 PIO Block Mode Reads

This class includes the following command:

— Read multiple

Execution includes the transfer of a block of sectors of data from the drive to the CPU.

a. The CPU writes any required parameters to the Features, Sector Count, Sector Number, Cylinder and Drive/Head registers.
b. The CPU writes the command code to the Command Register
c. When the block of sectors of data is available the drive issues an interrupt to the CPU.
d. After detecting the interrupt, the CPU reads the Status Register, then reads the sectors of data via the Data Register. In response to the Status Register being read, the drive clears the interrupt request.



TL/F/12497–4

7.3.1.1 PIO Single Sector Reads

This class includes the following commands:

— Identify drive
— Read buffer
— Read sector(s)

Execution includes the transfer of one or more sectors of data from the drive to the CPU.

a. The CPU writes any requlred parameters to the Features, Sector Count, Sector Number, Cylinder and Drive/Head registers.
b. The CPU writes the command code to the Command Register.
c. When a sector of data is available the drive issues an interrupt to the CPU.
d. After detecting the interrupt, the CPU reads the Status Register, then reads one sector of data via the Data Register. In response to the Status Register being read, the drive clears the interrupt request.
e. If transfer of another sector is required, the above sequence is repeated from c.

7.0 Functional Description (Continued)

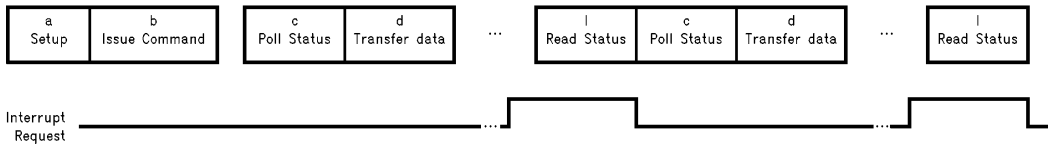
7.3.1.3 PIO Single Sector Writes

This class includes the following commands:

— Format
— Write buffer
— Write Sector(s)

Execution includes the transfer of one or more sectors of data from the CPU to the drive.

a. The CPU writes any required parameters to the Features, Sector Count, Sector Number, Cylinder and Drive/Head registers.
b. The CPU writes the command code to the Command Register.
c. The CPU polls the drive to see if the drive is ready (DRQ is set in the drive Status Register).
d. The CPU writes one sector of data via the Data Register.
e. When the drive has completed processing the sector it generates an interrupt request.
f. After detecting the interrupt, the CPU reads the Status Register. In response to the Status Register being read, the drive clears the interrupt request.
g. If transfer of another sector is required, the above sequence is repeated from c.



TL/F/12497–5

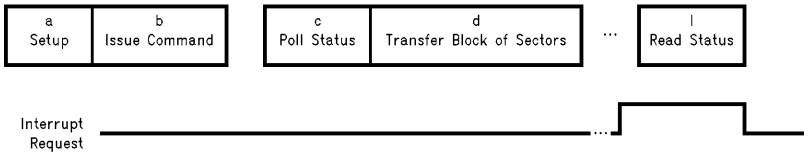
7.3.1.4 PIO Block Mode Writes

This class includes the following command:

— Write multiple

Execution includes the transfer of a block of sectors of data from the CPU to the drive.

a. The CPU writes any required parameters to the Features, Sector Count, Sector Number, Cylinder and Drive/Head registers.
b. The CPU writes the command code to the Command Register.
c. The CPU polls the drive to see if the drive is ready (DRQ is set in the drive Status Register).
d. The CPU writes one block of sectors of data via the Data Register.
e. When the drive has completed processing the block of sectors it generates an interrupt request.
f. After detecting the interrupt, the CPU reads the Status Register. In response to the Status Register being read, the drive clears the interrupt request.
g. If transfer of another block of sectors is required, the above sequence is repeated from c.



TL/F/12497–6

7.0 Functional Description (Continued)

7.3.1.5 DMA Mode

This class includes the following commands:

— Read DMA

— Write DMA

Data transfers using the DMA commands differ in two ways from PIO transfers:

— Data transfers are performed using the built-in DMA engine

— No intermediate sector interrupts are issued on multi-sector commands

The protocol is separated into 3 phases as follows:

a. Command phase
   1. The CPU initializes the DMA engine in the device (see Bus Master Operation section)
   2. The CPU updates the command Block Registers
   3. The CPU writes the command code to the Command Register

b. Data phase
   1. See DMA Engine Protocol section

c. Status phase
   1. The drive generates the interrupt to the CPU
   2. The CPU issues a Stop command to the DMA engine
   3. The CPU reads the Status Register and Error Register

TL/F/12497–7

7.3.2 Independent Timing per Drive

The device provides independent timings for reads, writes and command/control cycles per channel and per drive on each channel. All command and control block accesses are 8 bits and use one timing register to control the timing parameters. All accesses to the data register (offset 0) are always 16 bits and use the timing registers 44h–4Bh.

Each drive on a channel has two separate timing controls (one for read cycles and one for write cycles). A snooping mechanism is used to switch between drives in a channel. After reset, all accesses to channel 1 use ''channel 1-device 1'' timing, and all accesses to channel 2 use ''channel 2-device 1'' timing. When a command write to channel 1 with offset address of 6 is issued with bit 4 set (DRV) the controller will begin to use channel 1 device 2 timing for all successive accesses to that channel. Writing to channel 1 with offset of 6 with bit 4 cleared (DRV) will cause the controller to use channel 1 device 1 timing. This same mechanism is used for channel 2 separately.

7.3.3 Flow Control

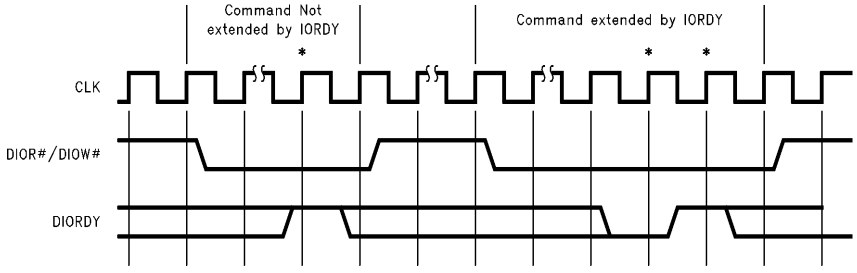
7.3.3.1 PIO

IORDY handshaking is provided for flow control between the controller and the IDE drive when the device is in PIO mode. The IORDY handshaking is the current specified flow control for PIO cycles in the ATA-2 specification.

7.3.3.1.1 IORDY

IORDY handshaking protocol is as follows:

1. The device receives a read/write data/command accessing the drive.
2. The device begins to generate IDE cycles using the timing associated with the channel number, drive number and cycle type (read, write, data or command).
3. Prior to completing the IDE cycle, the device samples DIORDY on the second to last clock of the cycle active time (one clock prior to the clock cycle that would de-assert the command) at which time the device either extends the cycle or completes depending on DIORDY. The cycle is extended as long as DIORDY is low (not ready).
4. When DIORDY is sampled high the cycle is completed.



* Sample DIORDY

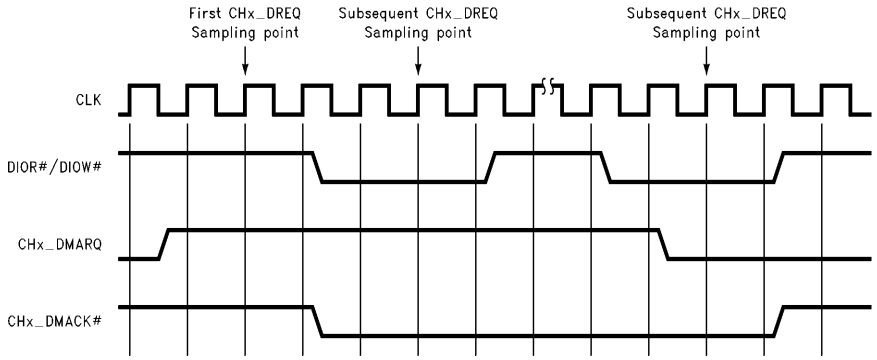
TL/F/12497–8

7.0 Functional Description (Continued)

7.3.3.2 DMA

The DMA handshaking protocol is as follows:

1. The device receives a command to start a DMA transfer.
2. The IDE controller waits for DMARQ from the drive to be asserted before issuing the data cycle.
3. When DMARQ is asserted, the controller asserts DMACK and begins to do data transfers until either there is no more data to be transferred or DMARQ is de-asserted. DMARQ is sampled on a cycle by cycle bases.
4. When the last command is complete and DMARQ is de-asserted, the device de-asserts DMACK#.



TL/F/12497–10

7.3.4 Interrupt Routing

Channel 1 and 2 interrupts can be routed to either IRQ14, IRQ15 respectively or to INTA on the PCI bus in various combinations. Table II shows all the combinations of interrupt routing.

TABLE II. Interrupt Routing

CMD(0) Enable	CTRL(6) Mask INTA	CTRL(8) Mask CH1__INTRQ	CTRL(9) Mask CH2__INTRQ	CTRL(4) Route Opt 0	CTRL(5) Route Opt 1	PIF(0) Native CH1	PIF(2) Native CH2	INTA#	IRQ14	IRQ15	Notes
0	x	x	x	x	x	x	x	x	x	x	1
1	0	0	0	0	0	0	0	x	CH1__INTRQ	CH2__INTRQ	2
1	0	0	1	0	0	0	0	x	CH1__INTRQ	x	2
1	0	1	0	0	0	0	0	x	x	CH2__INTRQ	2
1	0	1	1	0	0	0	1	x	x	x	2
1	0	0	0	1	0	0	0	!(CH1__INTRQ)	x	CH2__INTRQ	2
1	0	0	0	0	1	0	0	!(CH2__INTRQ)	CH1__INTRQ	x	2
1	0	0	0	1	1	0	0	!(CH1__INTRQ + CH2__INTRQ)	x	x	2
1	0	0	0	x	x	1	1	!(CH1__INTRQ + CH2__INTRQ)	x	x	3
1	0	0	1	x	x	1	1	!(CH1__INTRQ)	x	x	3
1	0	1	0	x	x	1	1	!(CH2__INTRQ)	x	x	3
1	0	1	1	x	x	1	1	x	x	x	3
1	1	x	x	x	x	1	1	x	x	x	3

Note 1: Device is disabled.

Note 2: Device responds to: Channel 1 to 1Fx and 3F6, Channel 2 to 17x and 376.

Note 3: Device responds to: Channel 1 to BAR0 and BAR1, Channel 2 to BAR2 and BAR3.

7.3.5 Prefetching

The device has the ability to prefetch IDE data during reads to the IDE data port (1F0, 170, BAR0 offset 0, BAR2 offset 0). Since the CPU will always request data in sector blocks (sector size is programmable) the device can initiate the next read operation to the drive while the CPU is writing the currently fetched data out to system memory. The device provides a sector count in order not to prefetch beyond the sector boundary, which could cause a fatal error on some disks.

If prefetch is enabled, upon the detection of a read to the IDE data port the device will begin prefetching. Prefetching will continue until either the sector count expires or the IDE command register (offset 07) is loaded with the following commands:

E4h Read buffer

C4h Read multiple

20h Read sector(s) (w/retry)

21h Read sector(s) (w/o retry)

In either of these two cases, prefetch will stop and the sector count re-loaded. Prefetching will resume upon a detection of a read to the IDE data port and if the prefetch buffer is not full. If the sector is disrupted for any reason, the sector counter will reset on the next read command, in order that reads will not pass the end of sector somewhere down the line. This is not normally a problem in PC-compatible systems, since the system will always transfer the entire amount of data requested. But for additional control, these read commands are monitored for synchronization.

7.3.6 Legacy/Native Mapping Scheme

The device indicates to the BIOS that it is an IDE device by returning 01 in the Class code byte (indicating that it is a mass storage controller) and a 01 in the Sub-Class code byte (indicating that it is an IDE controller) of the Class Code register. The Programming interface byte of the Class Code register indicates whether the device supports legacy and/or native mode. There is a pair of bits per IDE channel in the Programming Interface register that indicate to the BIOS which mode(s) the device supports. Bits 0 and 1 of the Programming Interface register correspond to channel 1, bits 2 and 3 correspond to channel 2 (see Programming Interface Register description for details of each bit).

When the device is programmed in legacy mode, the device will respond to the following addresses:

Primary: 1F0h–1F7h, 3F6h

Secondary: 170h–177h, 376h

When channel 1 is in legacy mode, the Base Address Registers 0–1 are not used, and when channel 2 is legacy mode Base Address Registers 2–3 are not used. When in legacy mode, each channel can be disabled independently by software. To turn off a legacy channel, software writes a one to the mode bit of the Programming Interface register (bits 0 or 2). When a channel is in native mode, it will not respond to legacy addresses, but will claim addresses defined by the Base Address Register 0–1 for channel 1 and Base Address Register 2–3 for channel 2.

7.4 DMA CONTROLLER

The built-in DMA controller allows the IDE controller to transfer data to/from the IDE drive by becoming a PCI bus master. The DMA supports scatter/gather which provides the capability of transferring multiple memory regions between system memory and the IDE drive without CPU intervention. The DMA controller can read the memory address and word count from an array of region descriptors, located in system memory, called the Physical Region Descriptor Table. This allows the DMA controller to sustain DMA transfers until all the memory regions in the PRD Table are transferred. The format of each entry in the Physical Region Descriptor Table is shown in Table III.

TABLE III. Physical Region Descriptor Table Entry

Byte 3	Byte 2	Byte 1	Byte 0
Memory Region Physical Base Address [31:1]			0
EOT	Reserved	Byte Count [15:1]	N/A

Each Physical Region Descriptor entry is 8 bytes in length. The PC87415 allows for 8192 table entries. Table IV below describes the descriptor in more detail. The descriptor table must be aligned on a 4 byte boundary and the table cannot cross a 64k boundary in memory. This limitation exists only if the table in memory is not contiguous. If the table crosses a page boundary and that memory page is located in a different physical memory location, the PC87415 does not have the ability to gather the table from memory. If the table is contiguous in memory, then the table could straddle a 64k boundary.

TABLE IV. Physical Region Descriptor

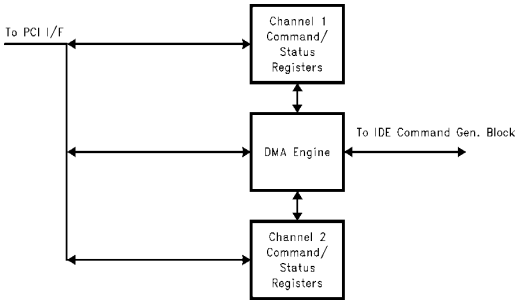
Byte #	Description
0–3	Bit 0: always 0 Bit 1–31: Memory Region Physical Base Address (corresponds to AD[31:1])
4–7	Bit 0: not used Bit 1–15: Byte count Bit 16–30: reserved Bit 31: EOT 0 = There are more entries in the Physical Region Descriptor table 1 = There are no more entries in the Physical Region Descriptor table

7.0 Functional Description (Continued)

The Memory Region specified by the descriptor cannot cross a 64k memory boundary. The Memory Region Physical Base Address is aligned on a 2 byte boundary.

7.4.1 DMA Engine

The built-in DMA controller supports 2 channels and each channel can operate independently. The descriptors are memory resident and are read every time a transfer takes place. The DMA engine is used to initiate and carry out data transfers between the IDE device and system memory. Separate command and status registers are used per channel allowing both channels to time share the DMA engine if both channels need to transfer data. When bus master IDE is enabled on a channel, that channel's write and prefetch buffers are used to either post or prefetch data to/from the disk. When two channels are requesting DMA services from the DMA engine, then after the first request is serviced by completing the entire transfer (EOT = 1 and byte count = 0) the second channel begins the transfer. *Figure 4* below shows the DMA block.



TL/F/12497–11

FIGURE 4. DMA Block

7.4.1.1 Alignment

The device will transfer data to/from memory in double word (4 bytes) transfers. If the Memory Physical Region Base Address is double word aligned and the Byte count is double word aligned then the transfer will be double words on the PCI bus using the PCI burst cycle. If either the Memory Physical Region Base Address or the Bytes count are not double word aligned, the PC87415 will still transfer double words on the PCI bus. This would be a failure condition. The software program needs to ensure that the Memory Physical Region Base Address and Byte count both be double word aligned.

7.4.2 DMA Engine Protocol

The DMA engine works together with the prefetch buffers and the write buffers to optimize IDE transfers and minimize PCI bus utilization. The following sections show how the DMA engine behaves under three different scenarios.

7.4.2.1 Scenario 1

The device request the bus and no one else is on the bus (the bus is idle). No other masters are requesting the bus throughout the entire transfer of a sector.

7.4.2.1.1 Bus Master Writes

The device arbitrates for the PCI bus and at the same time begins reading the data from the drive and filling the prefetch buffers. After the bus is granted, the device reads the descriptor from memory, loads the address pointer and byte count, and then gets off the PCI bus (if the prefetch buffer is not half full). When the prefetch buffer is half full, the device arbitrates onto the PCI bus (if not on the bus yet), and when granted starts writing out the prefetched data (2 DWords) out onto the PCI bus while prefetching data from the IDE drive. When the prefetch buffer is empty the device relinquishes the bus. If it wasn't the last descriptor (EOT is 0) the device continues prefetching from the drive while reading the new descriptor from memory. *Figure 5* shows the DMA flow. The total PCI bus utilization for this type of transfer can be calculated by the following equation:

PCI Bus Utilization = [ARB + [Read Descriptor]] + [256 * [ARB + Tw + Twb]

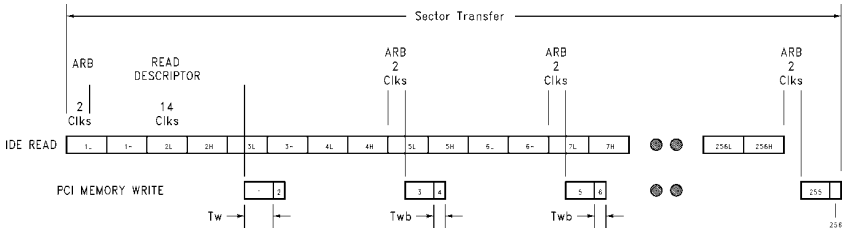
Where:

ARB = number of clocks for arbitration

Read Descriptor = number of clocks to read the descriptor

Tw = number of clocks for first non burst write

Twb = number of clocks for burst write



TL/F/12497–30

FIGURE 5. Bus Master Writes—Scenario 1

7.4.2.1.2 Bus Master Reads

The device arbitrates onto the bus and waits for a grant. When the bus is granted, the device reads the descriptor from system memory and loads the address pointer and byte count. It then begins to read data from system memory and buffering it in the posted write buffers until the buffer is full. When the buffer is full, the device will release the PCI bus. When the device sees data in the buffer, it begins to write to the drive. The device will request the PCI bus whenever the posted write buffer is either empty or half full. If the posted write buffer is empty (initial condition or due to another master on the PCI bus), then the device reads in 4 Dwords from memory, filling the buffers. If the posted write buffer is half full, then the device reads in 2 Dwords from memory, filling the buffers. When the byte count is 0 and it was the last descriptor (EOT is 1), the device relinquishes the bus (after the write FIFO is filled) while the write buffers are being emptied out onto the IDE bus. If it wasn't the last descriptor (EOT is 0), the device reads the new descriptor from memory while the device is writing out to the drive the contents of the write buffers. *Figure 6* shows the DMA flow. The total PCI bus utilization for this type of transfer can be calculated by the following equation:

Bus Utilization = [ARB + (Read Descriptor) + Tr + 3*Trb] + [255 * [ARB + Tr + Trb]

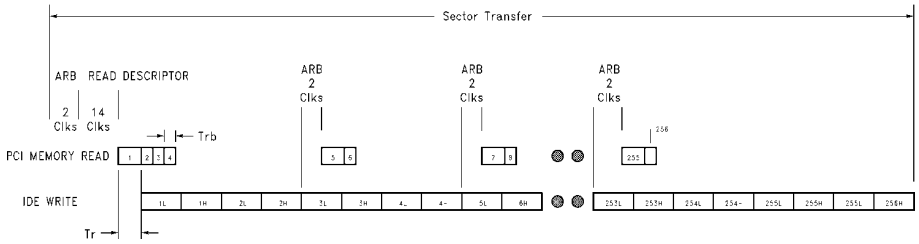
Where:

ARB = number of clocks for arbitration

Read Descriptor = number of clocks to read descriptors from system memory

Tr = number of clocks for first non burst read

Trb = number of clocks for read burst



TL/F/12497–31

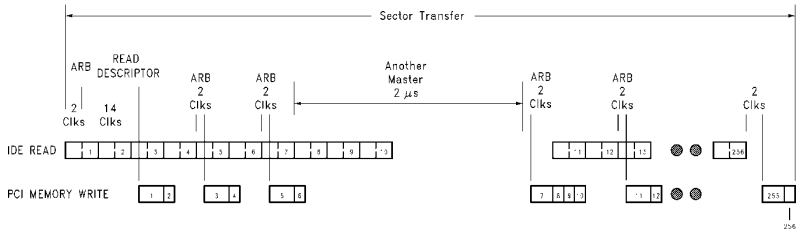
FIGURE 6. Bus Master Reads—Scenario 1

7.4.2.2 Scenario 2

The device requests the bus and no one else is on the bus (the bus is idle). While the transfer takes place, another master requests the bus. The latency timer is set to 2 μs.

7.4.2.2.1 Bus Master Writes

The device arbitrates for the PCI bus and at the same time begins reading the data from the drive and filling the prefetch buffers. After the bus is granted, the device reads the descriptor from memory, loads the address pointer and byte count and then releases the bus. When the prefetch buffer is half full, the device arbitrates onto the PCI bus and starts writing out the prefetch data (2 Dwords) out onto the PCI bus. When the prefetch buffer is emptied onto the PCI bus, the device releases the PCI bus. If another device becomes a bus master while the device is writing out to memory, the device will continue prefetching data from the drive until the prefetch buffers are full at which time it stops prefetching. The device continues to request the bus if the prefetch buffers are full. When it receives a grant, the device writes out the data (4 Dwords) from the prefetch buffers until the buffer is empty. When the prefetch buffer is empty and the byte count is 0 and it was the last descriptor (EOT is 1), the device relinquishes the bus. If it wasn't the last descriptor (EOT is 0) the device continues prefetching from the drive while reading the new descriptor from memory. *Figure 7* shows the DMA flow.

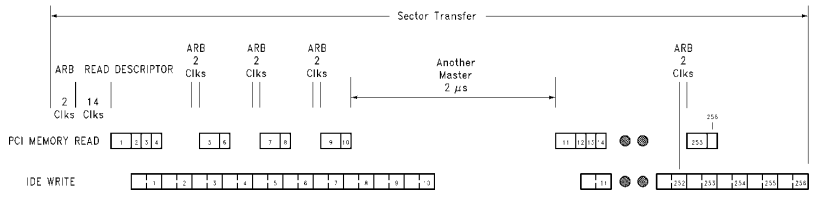


TL/F/12497–32

FIGURE 7. Bus Master Writes—Scenario 2

7.4.2.2.2 Bus Master Reads

The device arbitrates onto the bus and waits for a grant. When the bus is granted, the device reads the descriptor from system memory and loads the address pointer and byte count. It then begins to read data from system memory and buffering it in the posted write buffers until the buffer is full. When the FIFO is full, the device releases the PCI bus. When the device sees data in the buffer, it begins to write to the drive. The device will request the PCI bus whenever the posted write buffer is either empty or half full. If the posted write buffer is empty (initial condition or due to another master on the PCI bus interrupting the transfer), then the device reads in 4 Dwords from memory, filling the buffers. If the posted write buffer is half full, then the device reads in 2 Dwords from memory, filling the buffers. If another device becomes a bus master while reading from memory, the device will continue to write data to the drive emptying the write buffers. The device continues to request the PCI bus if the byte count is 0 and EOT = 0 and the buffer is empty. When it receives a grant, the device bursts in data (4 Dwords) into the write buffer until it is full. When the byte count is 0 and it was the last descriptor (EOT is 1), the device relinquishes the bus while the write buffers are being emptied out onto the IDE bus. If it wasn't the last descriptor (EOT is 0), the device reads the new descriptor from memory while the device is writing out to the drive the contents of the write buffers. *Figure 8* shows the DMA flow.



TL/F/12497–33

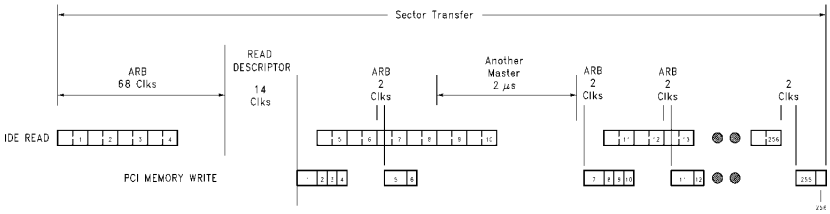
FIGURE 8. Bus Master Reads—Scenario 2

7.4.2.3 Scenario 3

The device requests the bus while another master started a bus transaction. While the transfer takes place, another master requests the bus. The latency timer is set to 2 µs.

7.4.2.3.1 Bus Master Writes

The device arbitrates for the PCI bus and at the same time begins reading the data from the drive and filling the prefetch buffers. The device waits until the other master completes and the bus is granted. After the bus is granted, the device reads the descriptor from memory, loads the address pointer and byte count, and then starts writing out the prefetched data out onto the PCI bus. (4 Dwords). The device will request the PCI bus everytime the prefetch buffer is either half full or full. When the prefetch buffer is half full, then the device will write out 2 Dwords to memory. When the prefetch buffer is full (due to another master owning the PCI bus), then the device will write out 4 Dwords to memory. If another device becomes a master while the device is writing out to memory, the device will continue to prefetch data from the drive until the prefetch buffers are full at which time it stops prefetching. The device continues to request the bus if the prefetch buffer is full. When it receives a grant, the device writes out the data (4 Dwords) from the prefetch buffers until the byte count is 0. When the byte count is 0 and it was the last descriptor (EOT is 1), the device relinquishes the bus. If it wasn't the last descriptor (EOT is 0) the device continues prefetching from the drive while reading the new descriptor from memory. *Figure 9* shows the DMA flow.

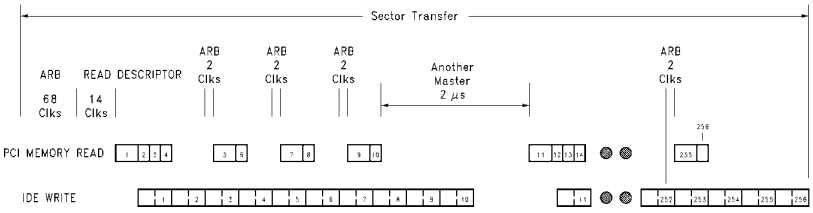


TL/F/12497–34

FIGURE 9. Bus Master Writes—Scenario 3

7.4.2.3.2 Bus Master Reads

The device requests the PCI bus and waits until the other master completes its transaction. When the bus is granted, the device reads the descriptor from system memory and loads the address pointer and byte count. It then begins to read data from system memory and buffers it in the posted write buffers until the buffer is full. When the buffer is full, the device releases the PCI bus. When the device sees data in the buffer, it begins to write to the drive. The device will request the PCI bus whenever the posted write buffer is either empty or half full. If the posted write buffer is empty (initial condition or due to another master on the PCI bus interrupting the transfer), then the device reads in 4 Dwords from memory, filling the buffers. If the posted write buffer is half full, then the device reads in 2 Dwords from memory, filling the buffers. If another master gets ownership of the bus after it released the bus, the device continues to write data to the drive emptying the write buffers. The device continues to request the PCI bus if the byte count is not 0 and EOT = 0 and the buffer is empty. When it receives a grant, the device bursts in data (4Dwords) into the write buffer until it is full. When the byte count is 0 and it was the last descriptor (EOT is 1) the device relinquishes the bus while the write buffers are being emptied out onto the IDE bus. If it wasn't the last descriptor (EOT is 0), the device reads the new descriptor from memory while the device is writing out to the drive the contents of the write buffers. *Figure 10* shows the DMA flow.



TL/F/12497–35

FIGURE 10. Bus Master Reads—Scenario 3

7.4.2.4 PCI Bus Request

7.4.2.4.1 PCI Master Reads

When the device wants to transfer data from memory to the drive, it will request the PCI bus under the following conditions:

[(write FIFO empty or half full) and ((EOT is 0) or (byte count is not 0))]

The device will request the PCI bus when the write FIFO is empty or half full and the DMA is active (not EOT or byte count is not 0). When the device is granted the bus, it will assert FRAME# and read either 2 Dwords (if the FIFO was half full) or 4 Dwords (if the FIFO was empty) from memory and place it into the write FIFO. As soon as 1 Dword is placed into the FIFO the FIFO will no longer be empty, so therefore the PCI request will be deasserted until the data from the FIFO are all written out onto the disk. The device will keep asserting FRAME# until all 2 or 4 Dwords are read from memory. If the device encounters a retry (STOP# asserted) while reading the 2 or 4 Dwords, it will release FRAME# and wait until the FIFO is half full or empty again. When the FIFO is half full, the device will request the PCI bus and will try to read 2 more Dwords from memory. When the FIFO is empty, the device will request the PCI bus and will try to read 4 more Dwords from memory.

Each Dword read cycle is first qualified with the DMA being active. If the DMA goes inactive (EOT is 1 and byte count is 0) during any one of the 2 or 4 Dword reads from memory, the device will stop the reads and deassert FRAME#. The condition for deasserting FRAME# is expressed by the following:

[(burst in 2 or 4 Dwords) or ((EOT is 1) and (byte count 0))]

If the byte count is programmed to be larger than the drive transfer length, the device will continue to read data from memory and posting it into the write FIFO as long as the FIFO is empty and the DMA is active. When the FIFO is full or the DMA goes inactive, the device will release the PCI bus. Since the drive has completed the transfer, it will generate an interrupt to the CPU even though some data is still resident in the write FIFO. When the CPU reads the status registers of the device it will see the interrupt bit set and the active bit either cleared or set. The active bit will be cleared if the device was able to write everything to the FIFO with the last PCI request. The active bit will be set if the device was not able to write everything into the FIFO because the FIFO was not emptied and therefore could not request the PCI bus. The contents of the write FIFO will be cleared when the device sees a stop command and another start command to that channel.

7.4.2.4.2 PCI Master Writes

When the device wants to transfer data from the drive into memory, it will request the PCI bus under the following conditions:

[(prefetch buffer is full or half full) or ((CHx__INT) and ((EOT is 0) or (byte count is not 0)))]

The device will request the bus when the prefetch buffer is full or half full or one of the channel generates an interrupt while the DMA is still active (not EOT or byte count is not 0). When the device is granted the PCI bus, it will assert FRAME# and write 2 Dwords (if the prefetch buffer is half full) or 4 Dwords (if the prefetch buffer is full) to memory. As soon as the first Dword is written out to memory, the read prefetch buffer will no longer be full or half full, so therefore the PCI request will be deasserted until the read prefetch buffer is written out to memory. The device will keep asserting FRAME# until all 4 Dwords are written to memory. If the device encounters a retry (STOP# asserted) while writing the 2 or 4 Dwords, it will release FRAME# and wait until the read prefetch buffer is either full or half full again. When the prefetch buffer is full, the device will request the PCI bus and try to write 4 more Dwords to memory. When the prefetch buffer is half full, the device will request the PCI bus and try to write 2 or more Dwords to memory.

Each Dword write cycle is first qualified with the DMA being active or the prefetch buffer not empty. The condition for deasserting FRAME# is expressed by the following:

[(burst in 2 or 4 Dwords) or (prefetch buffer is empty) or ((EOT is 1) and (byte count is 0))]

If the byte count was programmed to be less than the drive transfer length, the device will stop writing data out to memory as soon as byte count is 0 and will release the PCI bus but will continue to prefetch from the drive because DMREQ is still asserted (drive is not finished). The DMA controller will then stop and walt for software to do something. When the software times out, it will read the DMA status register. When the device sees a stop command and another start command it will clear the contents of the prefetch buffers.

7.4.3 Master Aborts

When a DMA channel is transferring data to a PCI target and the target does not respond with DEVSEL#, a master abort condition occurs. During master abort the following actions are taken:

1. Set bit 13 of Status register (07h) indicating that the transaction was terminated by a master abort.
2. Set the error bit in the Bus Master IDE status register for the appropriate channel.
3. Clear the Active bit in the Bus Master IDE status register for the appropriate channel.
4. Relinquish the PCI bus (de-assert REQ#).
5. De-assert DMACK# to the IDE drive and stop IDE cycles.
6. Clear buffers of the appropriate channel.

7.4.4 Target Aborts

When a DMA channel is transferring data to a PCI target and the target issues a target abort the following actions are taken:

1. Set bit 12 of Status reg. (07) indicating that the transaction was terminated by a target abort.
2. Set the error bit in the Bus Master IDE status register for the appropriate channel.
3. Clear the Active bit in the Bus Master IDE status register for the appropriate channel.
4. Relinqulsh the PCI bus (de-assert REQ#).
5. De-assert DMACK# to the IDE drive and stop IDE cycles.
6. Clear buffers of the appropriate channel.

7.4.5 Data Synchronization

The following is the protocol for data synchronization:

i. During DMA, delay the interrupt from the IDE until the FIFO is empty and the EOT = 1 and BC = 0

ii. No retry cycles generated when reading the DMA status registers (DMA status registers are always readable)

When the device is doing PCI master writes, the device buffers data from the drive using the prefetch buffers. If an IDE interrupt is encountered while the prefetch buffers are not emptied the device delays the interrupt (INTA or IRQ14 or IRQ15) until the FIFO is emptied and until EOT equals 1 and Byte Count equals 0. This is done to ensure that when software reads the status register of the device (caused by receiving an IDE interrupt) the device has already flushed the prefetch buffers.

There are four cases that need to be considered when software reads the status register of the device: normal, byte count is less than IDE transfer size, byte count is greater than IDE transfer size, and transfer in progress.

7.4.5.1 Normal

In normal completion CHx__INT is delayed until the channel x FIFO is empty and EOT equals 1 and Byte Count equals 0. The status register will indicate Channel x Interrupt bit set and the Channel x Active bit set indicating a normal completion.

7.4.5.2 Byte Count is Less Than IDE Transfer Size

When the Byte Count is programmed to be less than the IDE transfer size by more than 4 DWords, an IDE interrupt will not be generated and the software will time out. When the software reads the device's status register it will see the Channel x Interrupt bit not set and the Channel x Active bit not set.

When the Byte Count is programmed to be less than the IDE transfer size by less than 4 DWords, an IDE interrupt will be received by the device but will be blocked because the FIFO is not empty. The software will time out because it has not received any interrupt. It will then read the status register and will see the Channel x Interrupt bit set and the Channel x Active bit not set. This will be interpreted as an error condition (even though the status register reads back as normal completion) because no interrupt has been received.

7.4.5.3 Byte Count is Greater Than IDE Transfer Size

When the Byte Count is programmed to be greater than the IDE transfer size, an IDE interrupt will be received by the device but will be blocked because Byte Count is not 0. The software will time out because it has not received any interrupt. It will then read the status register and will see the Channel x Interrupt bit set and the Channel x Active bit set.

7.4.5.4 Transfer in Progress

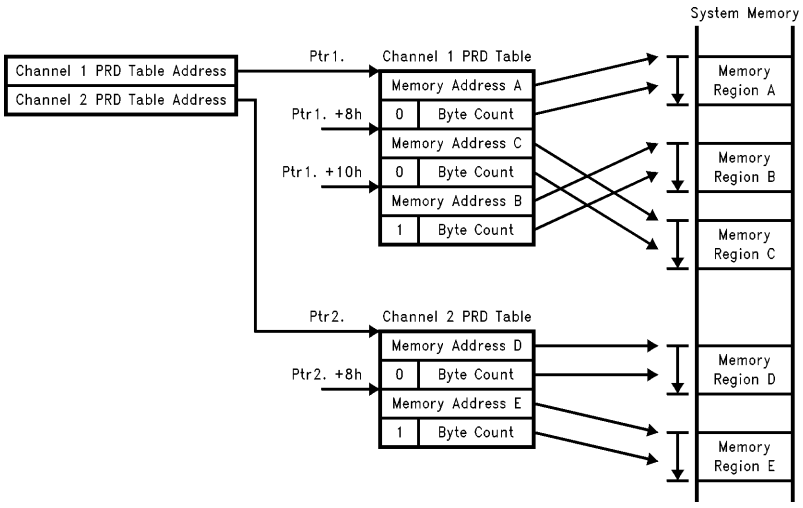
When the software reads the status register of the device while the DMA is in progress, it will read Channel x Interrupt bit not set and the Channel x Active bit set.

7.4.6 Bus Master Programming Sequence

To initiate a bus master transfer between memory and an IDE DMA slave device, the following steps are required:

1. Software prepares a PRD table in system memory. Each PRD is 8 bytes long and consists of an address pointer to the starting address and transfer count of the memory buffer to be transferred. In any given PRD table, two consecutive PRDs are offset by 8 bytes and are aligned on a 4-byte boundary.
2. Software provides the starting address of the PRD table by loading the PRD table pointer register. The direction of the data transfer is specified by setting the Read/Write Control bit in the Bus Master IDE Command register for the appropriate channel. Clear the Interrupt and Error bits in the Bus Master IDE Status register for the appropriate channel.
3. Software issues the appropriate DMA transfer command to the disk device.
4. Engage the bus master function by writing a one to the Start bit in the Bus Master IDE Command register for the appropriate channel.
5. The device arbitrates onto the PCI bus and gets ready to transfer.
6. When the bus is granted, the device reads one entry of the Physical Region Descriptor table pointed by the Physical Region Descriptor Table Address.
7. The device transfers data to/from memory pointed by the Memory Region Physical Base Address responding to DMARQ from the IDE device.
8. Transfers continue until the byte count is 0 from the last Physical Region Descriptor (EOT is set to 1) at which time the devices relinquishes ownership of the PCI bus.
9. At the end of the transfer the IDE drive signals an interrupt.
10. In response to the interrupt, software reads the device Status register and the drive status to determine if the transfer completed successfully and then resets the Start/Stop bit in the command register.

Figure 11 shows an example of a link list DMA transfer. In the example below, channel 1 transfers data between 3 memory regions, and channel 2 transfers between 2 memory regions. Each memory region could hold either a single sector or multiple sectors depending on the byte count of each memory region.



TL/F/12497–18

FIGURE 11. Link List Example

8.0 Electrical Characteristics

8.1 ABSOLUTE MAXIMUM RATINGS (5V 10%)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{DD})	−0.5V to +7.0V
Input Voltage (V_I)	−0.5V to V_{DD} + 0.5V
Output Voltage (V_O)	−0.5V to V_{DD} + 0.5V
Storage Temperature (T_{STG})	−65°C to +165°C
Lead Temperature (T_L) (Soldering, 10 seconds)	+260°C

8.2 RECOMMENDED OPERATING CONDITIONS

	Min	Typ	Max	Units
Supply Voltage	4.5	5.0	5.5	V
Operating Temperature (T_A)	0		+70	°C

8.3 DC ELECTRICAL CHARACTERISTICS

Symbol	Parameter	Conditions	Min	Max	Units
V_{CC}	Supply Voltage		4.75	5.25	V
V_{IL}	Input Low Voltage		−0.5	0.8	V
V_{IH}	Input High Voltage		2.0	V_{CC} + 0.5	V
I_{IH}	Input High Current	V_{IN} = 2.7		70	μA
I_{IL}	Input Low Current	V_{IN} = 0.5		−70	μA
V_{OH}	Output High Voltage	I_{OUT} = −2 mA	2.4		V
V_{OL}	Output Low Voltage	I_{OUT} = 3 mA, 6 mA		0.55	V
I_{CC}	Supply Current			10	mA
C_{IN}	Input Pin Capacitance			10	pF
C_{CLK}	CLK Pin Capacitance		5	12	pF
C_{IDSEL}	IDSEL Pin Capacitance			8	pF
L_{PIN}	Pin Inductance			20	nH

8.0 Electrical Characteristics (Continued)

8.4 AC TIMING SPECIFICATIONS

8.4.1 PCI Timing Specifications

The following AC timing specifications have been fully tested in silicon.

Tested Timing Parameters

Symbol	Parameter	Min	Max	Units	Notes
t1a	PCI signals: C/BE, FRAME#, TRDY# Input Setup Time to CLK	5		ns	1
	CLK High to IRQ14 Valid		15	ns	

The following AC timing specifications are based on simulation data except where noted.

PCI Timing Parameters

Symbol	Parameter	Min	Max	Units	Notes
t0	CLK Period	30		ns	
t_{HIGH}	CLK High Time	11		ns	
t_{LOW}	CLK Low Time	11		ns	
t1a	PCI Signals Input Setup Time to CLK (bussed signals)	7, 5		ns	1
t1b	PCI Signals Input Setup Time to CLK (Point to Point)	10, 12		ns	GNT# = 10 ns GNT# = 12 ns
t2a	PCI Signals, CLK to Output Valid (Busted Signals)	2	11	ns	
t2b	PCI Signals, CLK to Output Valid (Point to Point)	2	12	ns	
	CLK Slew Rate	1	4	V/ns	
	RST# Slew Rate	50		mV/ns	
t_{ON}	Float to Active Delay	2		ns	
t_{OFF}	Active to Float Delay		28	ns	
t_{H}	Input Hold Time from CLK	2		ns	
t_{RST}	Reset Active Time After Power Stable	1		ms	
$t_{RST-CLK}$	Reset Active Time After CLK Stable	100		µs	
$t_{RST-OFF}$	Reset Active to Output Float Delay		40	ns	

Note 1: Characterization of the PC87415 silicon shows a setup time of 5 ns for the following PCI signals: C/BE, FRAME#, and TRDY#. All other PCI bus signals have a setup time of 7 ns based on simulation data.

8.0 Electrical Characteristics (Continued)

8.4.2 ATA/IDE Timing Specifications

The following AC timing specifications are based on simulation data.

ATA/IDE Timing Parameters

Symbol	Parameter	Min	Max	Units	Notes
t3	CLK to DA[2:0], CHx__CS1,3#, DIOR#, DIOW#, DD[15:0] Valid	5	14	ns	
t4	DIORDY Setup Time to CLK Rising	12		ns	
t5	DIORDY Hold Time from CLK Rising	5		ns	
t6	DA[2:0], CHx__CS1,3# Setup Time to DIOR#, DIOW# Falling (Data Cycles)	1	3	CLK	1
t7	DIOR#, DIOW# Active Time (Data Cycles)	2	16	CLK	
t8	DIOR#, DIOW# Recovery Time (Data Cycles)	1		CLK	
t9	DD[15:0] Read Data Setup Time to CLK Rising	10		ns	
t10	DD[15:0] Read Data Hold Time from CLK Rising	0		ns	
t11	DA[2:0], CHx__CS1,3# Setup Time to DIOR#, DIOW# Falling (Non Data Cycles)	1	3	CLK	1
t12	DIOR#, DIOW# Active Time (Non Data Cycles)	2		CLK	
t13	DIOR#, DIOW# Recovery Time (Non Data Cycles)	1		CLK	
t14	CHx__DMARQ Setup Time for the First Occurrence of CHx__DMARQ	7		ns	2
t15	CHx__DMARQ Setup Time of the 2nd and Subsequent Occurrences of CHx__DMARQ	7			3
t16	CLK to CHx__DMACK#		18	ns	

Note 1: Setup time is derived from the recovery field (upper 3 bits).

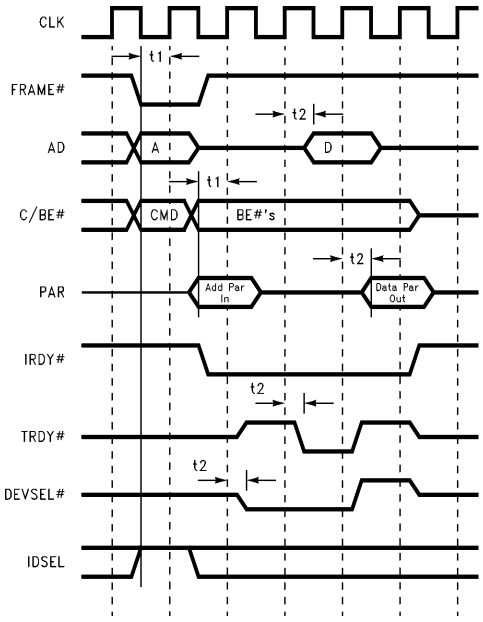
Value of Recovery Field			Setup Time in PCI Clocks
Bit 3	Bit 2	Bit 1	
0	X	X	4 CLKs
1	0	X	3 CLKs
1	1	0	2 CLKs
1	1	1	1 CLKs

Note 2: This is an asynchronous signal. The setup time is used as a reference for timing relationship of CHx__DMARQ and the command generation.

Note 3: This is a synchronous sampling point and setup time must be met for proper operation.

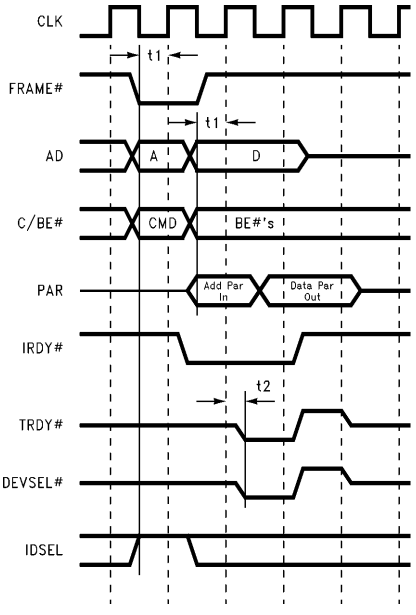
8.0 Electrical Characteristics (Continued)

8.4.3 Configuration Register Read Cycles



TL/F/12497–19

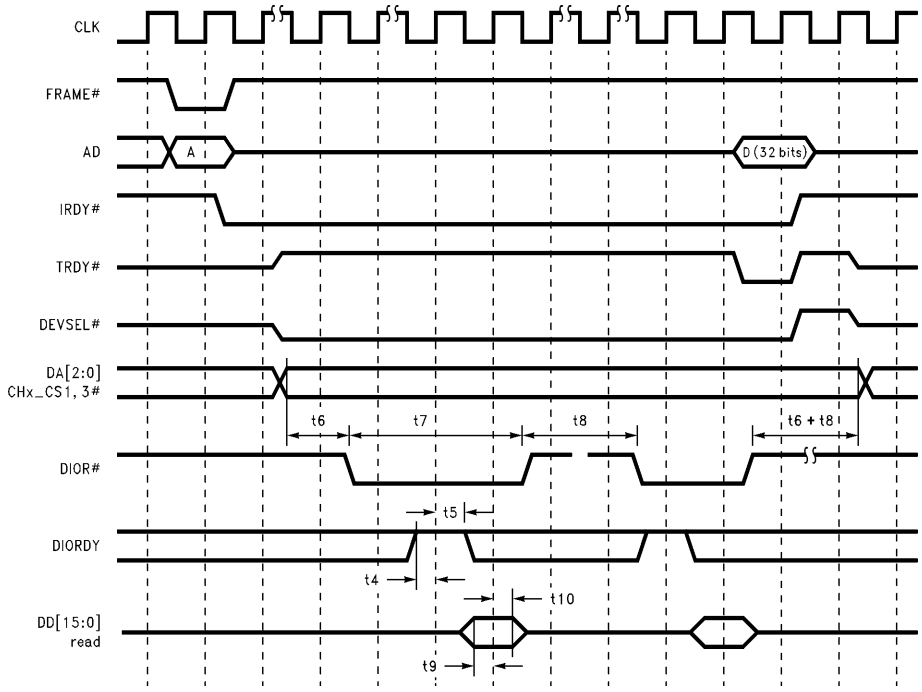
8.4.4 Configuration Register Write Cycles



TL/F/12497–20

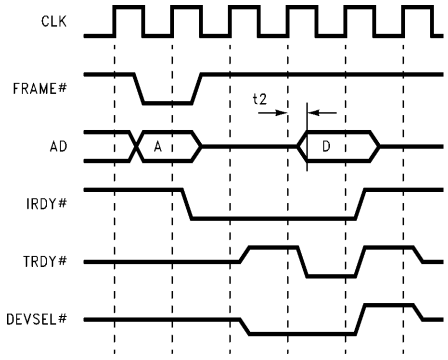
8.0 Electrical Characteristics (Continued)

8.4.5 Prefetch Cycles (Read Buffer Empty)



TL/F/12497–21

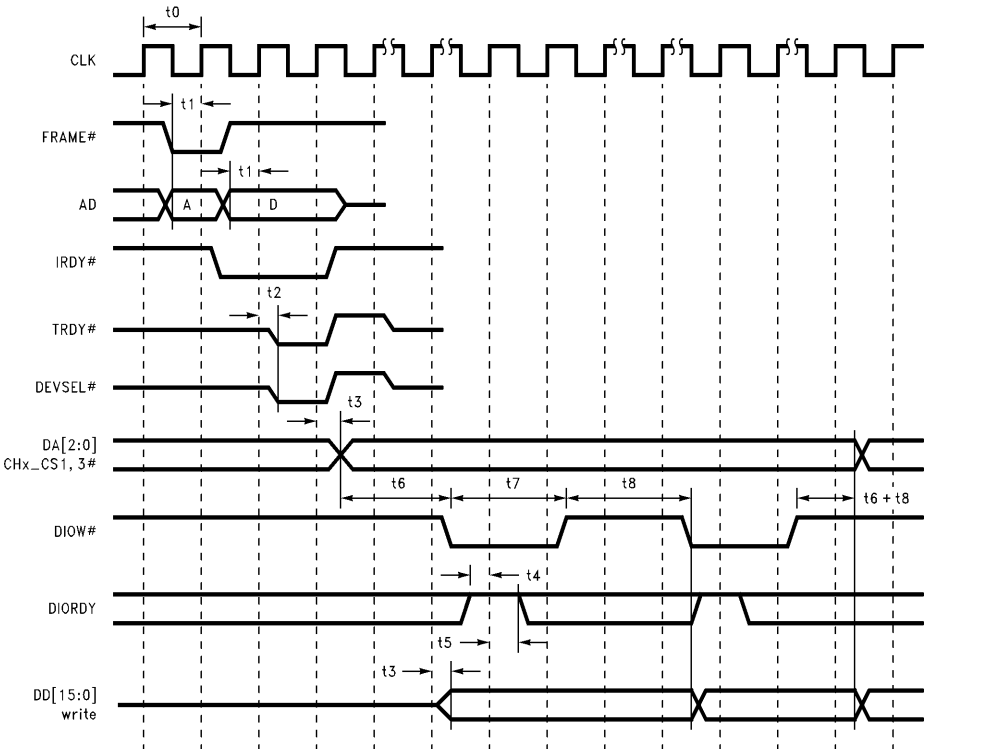
8.4.6 Prefetch Cycles (Read Buffer Not Empty)



TL/F/12497–22

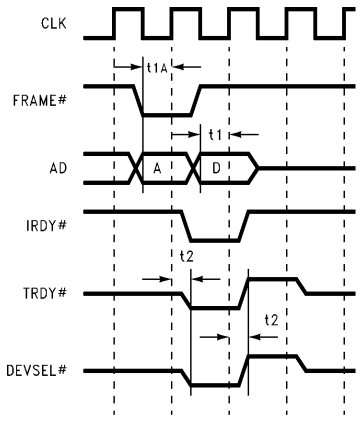
8.0 Electrical Characteristics (Continued)

8.4.7 Posted Write Cycles (Medium Decode)



TL/F/12497–23

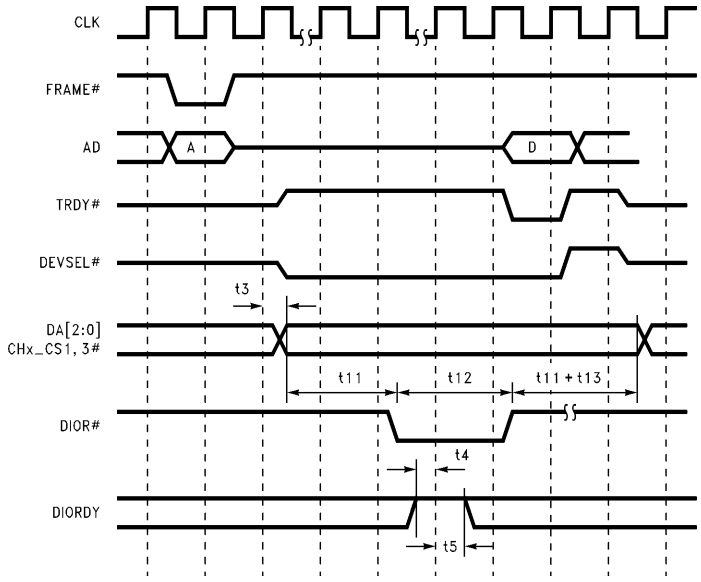
8.4.8 Posted Write Cycles (Fast Decode)



TL/F/12497–24

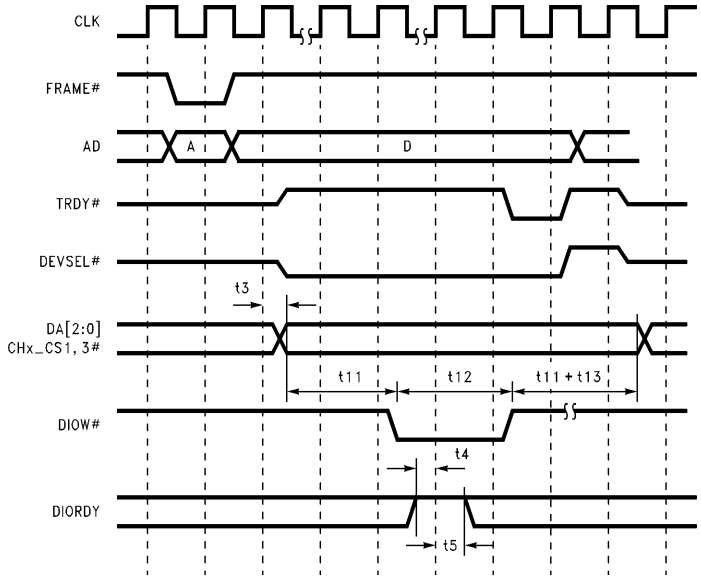
8.0 Electrical Characteristics (Continued)

8.4.9 IDE Non Data Read Cycles



TL/F/12497–25

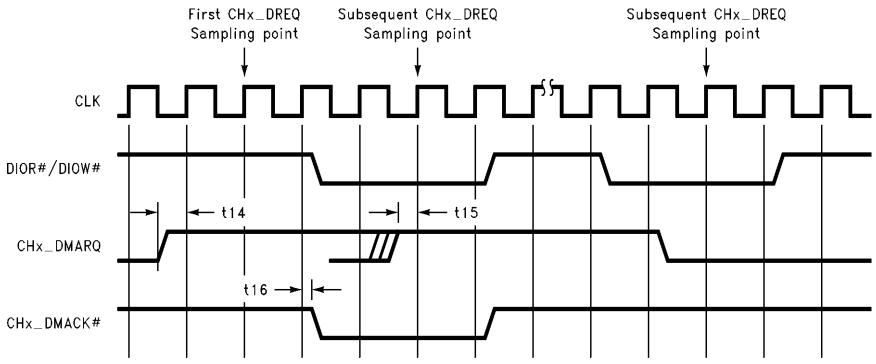
8.4.10 IDE Non Data Write Cycles



TL/F/12497–26

8.0 Electrical Characteristics (Continued)

8.4.11 DMA Cycles



TL/F/12497–27

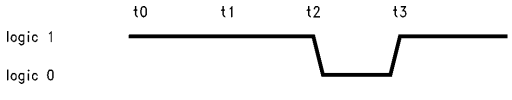
9.0 Application Information

9.1 POWER CONTROL FOR IDE DRIVES

The IDE__PWR pin may be programmed ON or OFF using bits in the CONTROL REGISTER, to allow external transistors to switch power to the IDE power connectors and cables. This feature is useful or ''Green-PC's'' and notebook computers which need to minimize power consumption.

At power on, as the power supply rises to 5V the IDE__PWR pin assumes a default logic 1 state. After reset, the default value of the control register is 00000000h, meaning that you have to set bit 18 = 1 to define the pin to be operating under IDE__PWR ON/OFF control using bit 3. This will result in power to the IDE peripheral being turned on after power up, but even as bit 18 is changed from a 0 to a 1, due to the reversed control polarity used by bit 3, the output level of this pin will remain at a logic 1. In order to ''turn-off'' the V_{CC} supply to the IDE connectors it is necessary to program bit 3 to a 1 which sets the output level of the pin to a logic 0. To turn the V_{CC} supply to the IDE connectors ''back-on'' set bit 3 to a 0 which will in turn set the output pin to a logic 1.

The table below shows the various bit settings and resulting output level for the operation of IDE__PWR pin at different instances in time, t:



TL/F/12497–36

	t0	t1	t2	t3	Comment
Bit 18	0	1	1	1	
Bit 1	0	0	X	X	After bit 18 is set to a 1, bit 1 becomes a don't-care
Bit 2	0	0	0	0	Bit 2 is always set to a 0
Bit 3	0	0	1	0	Bit 3 used to turn ON and OFF the V_{CC} control to the IDE connectors
Mode	Default Value at power up reset	Set pin as correct mode to control output pin IDE__PWR	Set bit 3 = 1 to turn IDE__PWR to logic 0 to set V_{CC} OFF to IDE connectors	Set bit 3 = 0 to turn IDE__PWR to logic 1 to set V_{CC} ON to IDE connectors	

9.2 NATIVE MODE INTERRUPT SUPPORT

The PC87415 can be configured in either legacy mode or native mode. In Legacy mode it routes its interrupts to the ISA Bus (''IRQ14'' for Channel 1 and ''IRQ15'' for Channel 2).

The PC87415 can be set up to power-up and function in Native Mode by following the procedure outlined below:

1. The PC87425 ''DA2'' I/O pin (pin #63) must be pulled high through a 10 kΩ resistor. The state of the ''DA2'' input (LEGACY#) upon ''RESET'' transitioning high will be loaded into the PCI Configuration Space Register #9 (Programming Interface Register, PIF). The pull-up Resistor on pin ''DA2'' will cause a logic ''1'' to be programmed into bits 0 and 2 of the PIF register, thus configuring the PC87415 to be in Native Mode and respond to accesses in Base Address Register 0 and 1 for Channel 1 and BAR 2 and 3 for Channel 2.
2. The circuit board should route the PC87415 interrupt to the PCI Bus INTA#. The BIOS or system must be able to route the PCI INTA# input to an appropriate IRQ vector.
3. The BIOS must be able to program the PC87415 PCI Configuration Space ''Base Address Registers 0–3 and 4''.

Given the above steps the PC87415 will be able to operate in 16-bit PIO Mode 0. In order to access the higher performance possible in the PC87415 the BIOS or Driver must be able to access the PCI Configuration Space above 3FH.

The Control and Timing registers allow the driver or BIOS to program the PC87415 in higher performance PIO modes and use full 32-bit PCI accesses and the internal FIFO's of the device. Lastly, the driver must allow accessing through the PC87415 PCI Configuration Base Address Register #4 to enable the DMA capabilities of the chip (same as Legacy Mode Driver).

The PC87415 and the drivers supplied with it fully support legacy mode. The Windows 95 and Windows NT drivers also support native mode operation.

Level vs. Edge Sensitive Interrupts

To be in conformance with the PCI specification the PC motherboard chipsets should be able to handle ''Level Sensitive Interrupts''. However, some older chipsets do not cor rectly handle Level Sensitive Interrupts, particularly in the instance where the PCI Interrupt pin (INTA#) may be asserted and several interrupts are pending via this pin.

For those older chipsets which do not correctly handle level sensitive interrupts the system designer needs to use an external logic solution, such as a GAL programmable logic device, to guarantee generating an edge for each Interrupt that is pending. The GAL should monitor all the interrupts that are tied to a singular PCI Interrupt pin and the PC87415 INTA# output. The GAL also monitors the PC87415 INTA# output and generates its own version of INTA# to the PCI bus.

9.0 Application Information (Continued)

The GAL monitors the interrupts that are causing INTA# to be asserted. If multiple of these interrupt requests are asserted then every time one of these requests negates, a 10 Clock (PCI CLOCK) negation pulse is produced on the INTA# pin. This gives an edge for each interrupt request and has been shown to work with PC motherboards that do not work properly with PCI level sensitive Interrupts. Note, INTA# will stay negated if no more interrupt requests are pending.

9.3 DMA BUS MASTER CONTROL AND STATUS REGISTER

This is a minor errata that relates to the DMA Bus Master Control and Status Register, offset from the PCI Configuration Space Base Address Register #4.

For Channel #1, the registers offset 00H and 02H from BAR4.

For Channel #2, the registers offset 08H and 0AH from BAR4.

The Registers 02H and 0AH (bits 1 and 2) are the Channel 1 and 2 Error and Interrupt bits respectively. Both these bits should be cleared when writing a ''1'' to the particular bit or bits.

Note: THE RESETTING OF THESE TWO BITS HAVE BEEN ERRONEOUSLY CODED IN THE PC87415 DESIGN!

Instead, to reset bits 1 or 2 in register 02H (or 0AH) the user must write a ''1'' in bits 1 or 2 of register 00H (or 08H).

In other words, to reset these two bits, '0000 0006' should be written, instead of '0006 0000'. The reset bits (bit 1 and bit 2) are reserved bits which are not being used in the PC87415 design and as such this should not affect other functionality of the device.

This software modification is included in all the drivers supplied by National Semiconductor for use with the PC87415.

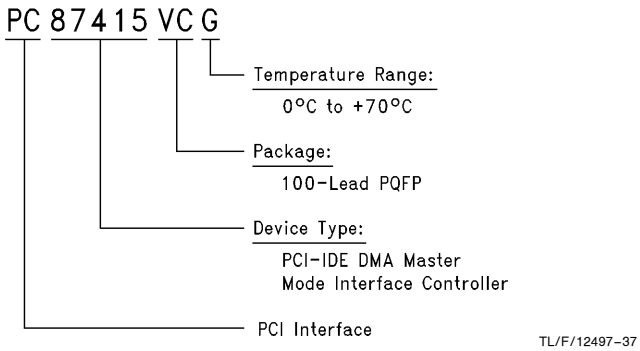
10.0 Errata

10.1 TARGET MODE

a. In response to an illegal CBE combination during access from another Master, the device generates STOP and a Target Abort but does not generate DEVSEL.

Ordering Code Information

(a) Device

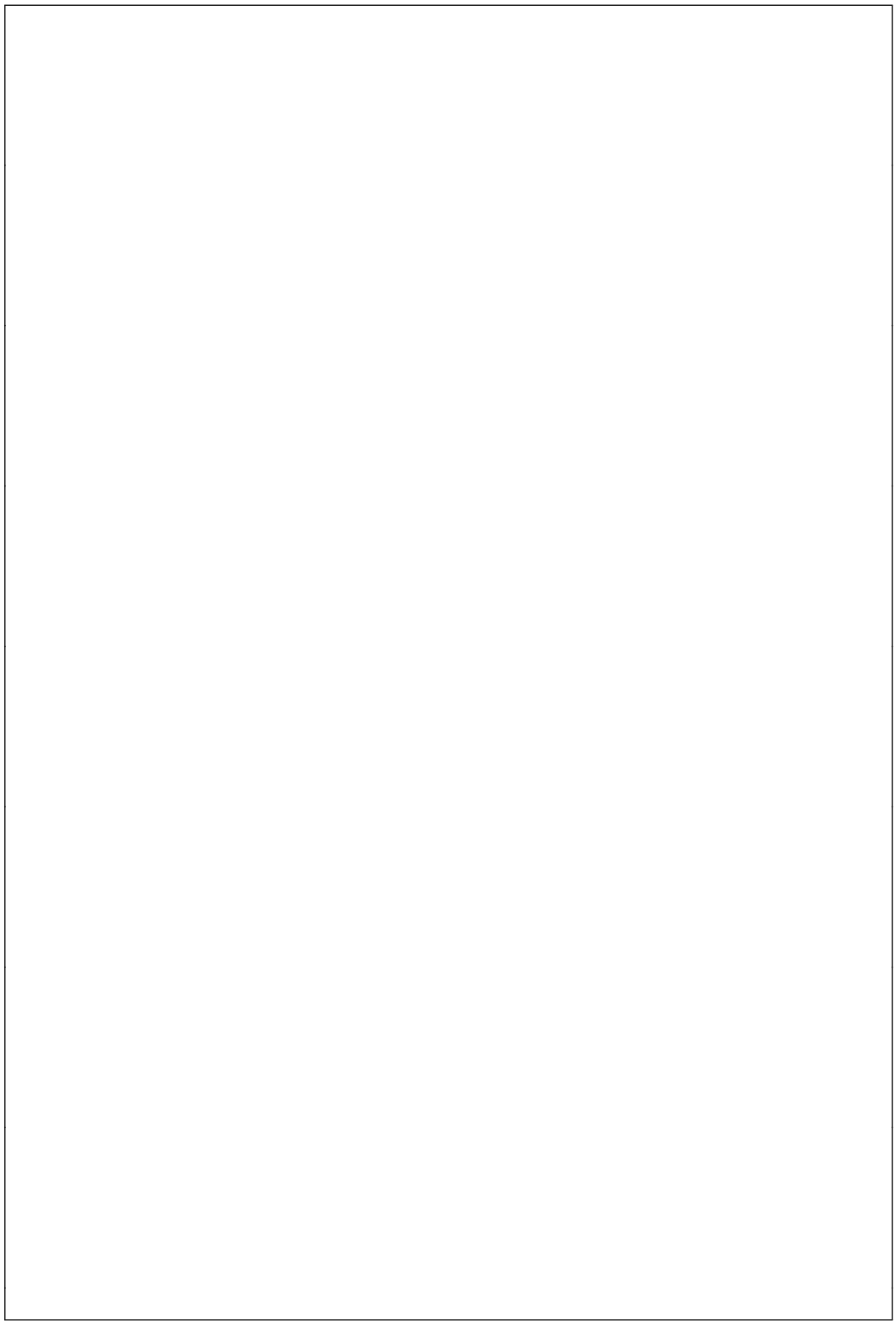


TL/F/12497–37

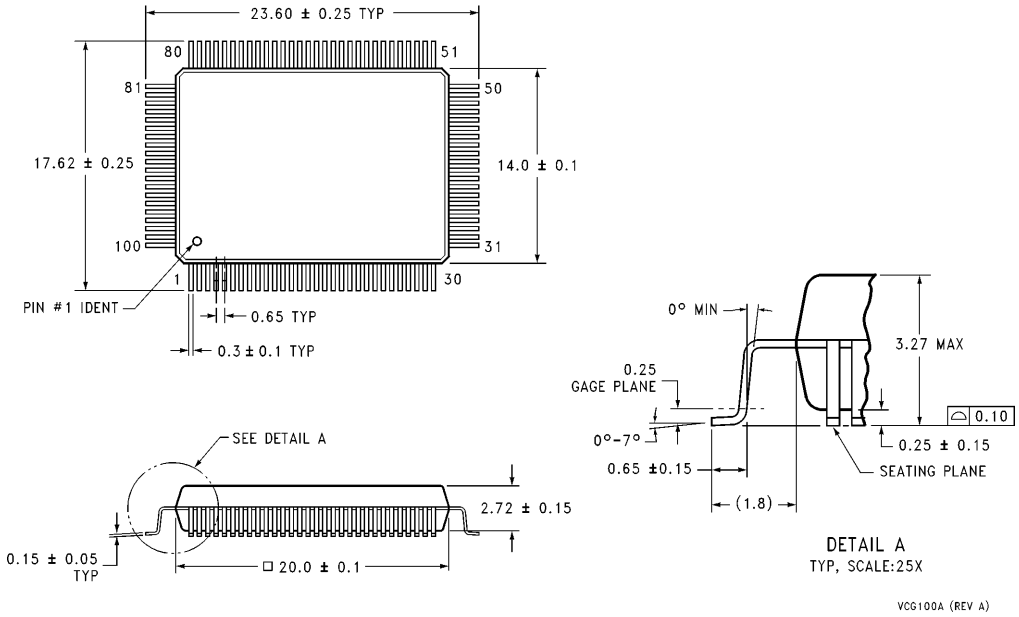
(b) Software Drivers

PC87415-SW.A set of discs with software drives for:

- DOS
- Windows 3.x
- Windows 95
- Windows NT
- OS-2
- Netware
- SCO Unix



Physical Dimensions inches (millimeters)



100-Lead (14mm x 14mm) Molded Plastic Quad Flatpak, JEDEC
Order Number PC87415
NS Package Number VCG100A

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

http://www.national.com

National Semiconductor Europe
Fax: +49 (0) 180-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 180-530 85 85
English Tel: +49 (0) 180-532 78 32
Français Tel: +49 (0) 180-532 93 58
Italiano Tel: +49 (0) 180-534 16 80

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor Japan Ltd.
Tel: 81-043-299-2308
Fax: 81-043-299-2408