U S E R ' S G U I D E

Document Revision 3
September 12, 1995

AIC-8375

Fast IDE Disk Controller IC

Adaptec Confidential

adaptec®

AIC-8375 Features

ATA Interface Block

- ATA Multiword DMA modes 0-2
- Fast IDE PIO modes 0-4
- IORDY for PIO flow control
- Automatic AT R/W command execution
- Automatic AT Task File updates
- 32-byte host FIFO
- LBA or CHS TASK File Modes
- Programmable IRQ automation to comply with different BIOS implementations
- Provides logic for daisy chaining two embedded disk drive controllers
- Hardware selectable PCMCIA 2.0
- Full BIOS compatibility
- On-Chip12 mA Host Drivers
- PCMCIA Attribute Memory stored in buffer memory

Buffer Controller Block

- 8 bit wide or 16 bit wide buffer data bus with parity
- DRAM support with up to 4 Mbyte addressing capability; up to 36 Mbyte/s (18 Mbyte/s PCMCIA) buffer bandwidth using page mode DRAM access
- SRAM support for up to 256 Kbyte direct addressing; up to 50 Mbyte/s (25 Mbyte/s PCMCIA) buffer bandwidth
- Automated Data Flow Management (ADFM) automates disk/host transfers.
- 2K page direct microprocessor access
- Variable Segmentation
- Write Cache Support
- Servo Split count out of buffer

Other Features

- 128-pin QFP and TQFP packages
- Automatic power-down modes
- High-speed, low-power CMOS

EDAC Block

- Optimized 168 bit ECC with Triple Burst on-the-fly (OTF) correction
- 65 bit single burst OTF correction in <1 sector time or 17 bit Double or Triple Burst OTF correction in <1 sector time
- SW correction up to three 17-bit bursts
- Error detection of one 89-bit error, two 41-bit errors, or three 17-bit errors
- ECC seeding validating servo and head track position
- Fault tolerant sync mark detection with 2 byte sync

Disk Controller Block

- Enhanced Headerless Architecture (EDSA)
- 50 Mbits/sec, 100 Mbits/sec and 120 Mbits/sec disk rate in single, dual, and byte-wide NRZ modes respectively
- 31 x 3 byte flexible high-speed RAM-based sequencer
- Defect skipping and/or embedded servo capabilities with Constant Density Recording (CDR)
- 32-byte disk FIFO for speed matching with the buffer manager
- Two-index timer
- Supports (1,7) and (2,7) RLL interfaces
- MR and PRML channel support

Microcontroller Interface Block

- Direct support for Intel or Motorola multiplexed or non-multiplexed interfaces
- Ready line for interfacing to faster microprocessors and for direct microprocessor access to the Buffer
- Programmable open drain interrupt output for host, disk, and buffer
- Address latch outputs for multiplexed microprocessor interface
- Direct buffer addressing in programmable 2K windows with 1K resolution on base address

Table of Contents

List of Tables

List of Figures

REVISION NOTES
for AIC-8375 User's Guide (PN 700191-011A Rev 3) - September 1995

In this document, **all** changes to technical information incorporated by ***Rev 3*** are indicated by a change bar in the left-hand margin. (Superficial or non-technical edits are not indicated.)

This page contains a list of pages having **technical information** that has changed **from *Rev 2* to *Rev 3***.

Changes Incorporated Into Rev 3

Section Updated	Pages Affected	Change Description
Front Cover	-	Chip revision level removed from title.
2.3.1.3	18	Two bulleted items added regarding the CIS.
3.6	143	Removed sentence - "Software correction is optionally available etc." Removed bullet - "Handles user data block sizea from 360 to 744 bytes. etc."
3.6.1.1	144	Changed second bullet to "A maximum of 744 data bytes can be handled by the ECC (the max under ATA spec is currently 512 bytes)."
3.6.1.1	146	Changed 219 NRZ byte times to 320 - changed Table 3-29.
3.6.1.3	149	Added note to Table 3-30.
3.6.1.4	152	Added reference to ECC Correction Threshold to first bulleted item.
3.6.2	153-154	Changed text. Changed Tables 3-31 & 3-32 to 3-31 - 3-34.
4.2.2	188	Deleted "pointed to by the 'Next Address' instruction etc." from "Note." Add SYNCEN pin statement to decode '010'.

SECTION 1
Overview

1.1 Introduction

The Adaptec AIC-8375 is an automated single-chip disk controller designed for high performance embedded ISA/EISA/ATA and PCMCIA drive applications. The AIC-8375 has evolved from the AIC-8371 and is targeted for low cost, high performance headerless drive designs. It incorporates significant new advances over the AIC-8265 series of controller devices in the areas of AT automation, disk format, buffer management, and increased data rate. These advances provide the designer with methods for bridging the increasing gap between host processor performance (MIPS) and disk drive performance (I/Os per second).

The AIC-8375 is capable of executing full track read/write operations, at a maximum disk rate of up to 50 Mbits/sec in single NRZ mode, 100 Mbits/sec in dual NRZ mode, and 120 Mbits/sec in byte wide NRZ mode, with complete automation including full automation of the various subfunctions such as error detection and correction, buffer data flow management, embedded servo or defect algorithms, and AT interface management. The various functional blocks within the device work together automatically to ensure proper data flow management and data integrity. This automation provides the local microprocessor with more bandwidth to apply to other tasks such as servo control functions. Alternately, the improved bandwidth may be applied towards the use of a lower performance microprocessor to further increase the performance/price ratio.

Designed using high speed CMOS technology, the AIC-8375 provides a hierarchy of power down and automatic wake up modes for power sensitive applications.

1.2 General Description and Features

The AIC-8375 works in conjunction with a local microprocessor to perform the ISA/EISA/ATA and PCMCIA interface control, buffer data flow management, disk format/read/write control, and error correction functions of an embedded disk drive controller. The microprocessor communicates with the AIC-8375 by reading from and writing to its various internal registers.

To the microprocessor, the registers of the AIC-8375 appear as unique memory or I/O locations that are randomly accessed and operated upon. By reading from and writing to the registers, the local microprocessor initiates operations and examines the status of the different functional blocks. Once an operation is started, successful completion or an error condition may cause the AIC-8375 to interrupt the local microprocessor, which then examines the status registers of the AIC-8375 and determines an appropriate course of action. The local microprocessor may also poll the device to ascertain successful completion or error conditions.

Figure 1-1 reveals the various blocks within the AIC-8375 along with their generalized interconnection. The blocks described in this figure will be referred to throughout this document.

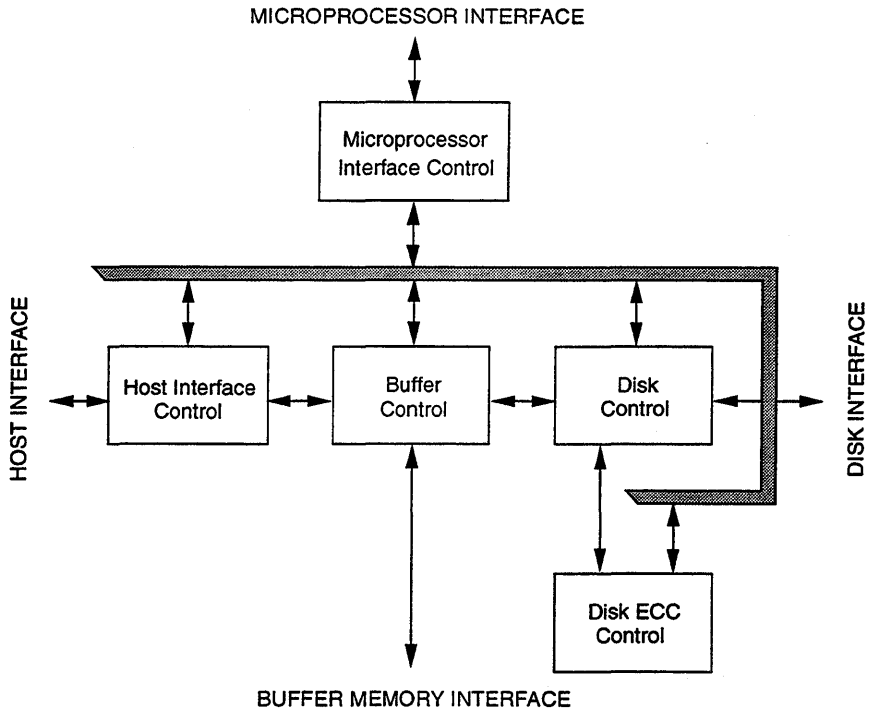

Figure 1-1 AIC-8375 Block Diagram

1.2.1 The Local Microprocessor Interface Block

It is through the Local Microprocessor Interface Block that the local microprocessor communicates to the other control blocks within the device or to the buffer. The AIC-8375 supports Intel and Motorola type 8-bit multiplexed address/data busses. Non-multiplexed Intel and Motorola type interfaces are also supported. This enables the AIC-8375 to interface directly to a variety of microprocessors including, but not limited to, those listed below.

Table 1-1 Compatible Microprocessors

Microprocessor	Mode
Intel 8051/31	multiplexed
Intel 80C196KD20	multiplexed
Intel 80C188	multiplexed
Motorola 68HC11F1	multiplexed/non-multiplexed
Motorola 68HC16	multiplexed/non-multiplexed
Hitachi H8/532	non-multiplexed
National HPC	multiplexed/non-multiplexed
Siemens 88C165, 6, 7	multiplexed/non-multiplexed
TI TMS320C25 (DSP)	non-multiplexed
Adaptec AIC-4420	non-multiplexed

There are other families of microprocessors which can be easily interfaced to the AIC-8375 using minimal external logic. Information provided in the Timing Specifications section of the AIC-8375 Data Sheet can be used to design any required external glue logic which may be required to interface to other types of microprocessors.

To accommodate a variety of microprocessors and embedded applications, the Local Microprocessor Interface Block provides the following features:

- Optional address latch for multiplexed busses.
- Programmable byte order for accesses of multi-byte registers.
- Ready signal for interfacing to various high end microprocessors.
- A Host/Buffer/ Disk Interrupt signal for use by the microprocessor with all interrupts maskable.
- Direct read/write access to any 2 Kbyte window in the buffer.

1.2.2 The Host Interface Block

The Host Interface Block provides the necessary functions for interfacing the AIC-8375 to the ATA (IDE) interface or the PCMCIA interface. Emulation of the IBM PC ISA/EISA Task File is accomplished via a set of registers and logic which is accessible by the local microprocessor. The Task File contains eleven I/O registers that enable the host PC to set drive configurations, execute commands, and check their status.

Both the ATA and PCMCIA interfaces provide a means for the host to access the Task File registers. The PCMCIA communication modes available are:

- Attribute Memory: This is read by the host to determine information about the PCMCIA card and to change the card's configuration. It is not available in ATA mode. The Attribute Memory can be read by the host at any time the AIC-8375 is not in the busy state. The AIC-8375 Attribute Memory consists of two parts:
  - Card Information Structure: This data structure contains readable tuple configuration data. This is a user-defined interface that supports up to 256 bytes of information. The CIS is implemented as the first 256 bytes of buffer memory and is not writable by the host.
  - Configuration Registers: These registers return information about the current configuration. The host can write to these registers to change the current configuration (e.g., change from Memory to a Block I/O addressing mode).
- Primary/Secondary I/O Addressing Mode: This mode is compatiable with existing IDE drives and ATA BIOSes. The primary addresses are 1F0h - 1F7h and 3F6h - 3F7h. The secondary addresses are 170h - 177h and 376h - 377h.
- Block I/O Mode: This mode allows the host to locate the Task File Registers on a 16-byte boundary in the host's I/O memory space. This allows multiple cards to co-exist without I/O conflicts.
- Memory Mode: The Task File Registers are mapped into memory along with a 1KByte block of memory for data transfers. The PCMCIA specification refers to this memory as Common Memory.

The Host Interface Block can be programmed to execute various host read/write commands either completely automatically without any microprocessor intervention, semi-automatically with minimal microprocessor intervention, or manually with the aid of the microprocessor.

Of particular interest to most designers, are the significant advances in AT automation which have been incorporated into the AIC-8375. The highlights of AT automation are:

- Automatic data transfer management for multi-sector Read/Write commands without microprocessor intervention.
- Automatic data transfer management for Read/Write Multiple commands without microprocessor intervention.
- Automatic execution of read commands. (Auto-Read command execution) for cached data in the buffer by matching the first sector.
- "Auto-Write" command execution (First sector of a multiple sector write operation is automated or the transfer of one sector of the selected single sector write operation is automated.)
- Automatic Task File register updates during automatic multi-sector transfers.
- Programmable methods of IRQ assertion to allow automation to work with different BIOS implementations and different device drivers.
- Capability to execute multiple consecutive "Auto-Write" commands without loss of data in the buffer.
- 32-byte host FIFO to allow automation to occur smoothly during discontinuities in transfers on the AT interface.
- Ability to pause the AT automatic transfers between the host and buffer on sector and block boundaries.
- Automation of an extensive portion of the ATA command set.
- Auto-Read Command Interlock to facilitate cache support.

The AIC-8375 supports both PIO and DMA type transfers. The supported DMA type transfers includes ISA and EISA Type A/B DMA modes. DMA transfers and PIO transfers utilize the bus in 8- or 16-bit mode depending upon the command being executed. The bus is automatically switched between 16- and 8-bit mode while performing Read Long and Write Long commands at the time of ECC byte transfers.

Additional functionality is provided in the Host Interface Block by the following features:

- Programmable transfer length for automatic ECC byte transfer on the AT bus.
- Automatically inserted wait states are provided to support the IOCHRDY signal pin functions at any AT interface speed.
- Support for Master/Slave configuration of two embedded disk controller drives.
- Automatic detection of Host status reads.
- Support of both LBA and CHS Task File register formats.
- Automatic detection of both the software AT reset and hardware AT reset.
- 12-mA drivers are provided for direct connection to the ISA/EISA bus. Some AT inputs are Schmitt trigger inputs.
- Support for PIO modes 0 through 4.

When the AIC-8375 is configured for PCMCIA mode the chip is restricted to using a byte wide buffer. The upper byte of the buffer bus is reconfigured as the additional host address signals required by the PCMCIA interface. Additionally the upper two bits of the NRZ are reconfigured as HOE and HWE.

1.2.3 The Buffer Control Block

The Buffer Control block manages the flow of data into and out of the buffer. Significant automation is incorporated which allows buffer activity to take place automatically during read/write operations between the host and the disk. This automation works together with automation within the Host Control and Disk Control blocks to provide more bandwidth for the local microprocessor to perform non-data flow functions.

The buffer control circuitry keeps track of buffer full and empty conditions and automatically works with the Disk Control block to stop transfers to or from the disk when necessary. In addition, transfers to or from the host are automatically stopped or started based on buffer full or empty status.

A prioritized seven port architecture is implemented. All ports, except the refresh port, utilize 22-bit buffer address pointers.

The data path to the buffer RAM can be configured as an 8- or 16-bit path in ATA mode and 8-bit only in PCMCIA mode. A parity bit is available for each of the low and high order eight bits.

DRAM configurations from 64 KBytes to 4 MBytes are supported. Eight-byte Page Mode access along with CAS before RAS type refresh is implemented. In 8-bit mode, the sustained DRAM buffer bandwidth, during concurrent host and disk transfers, is 18 MBytes/sec utilizing a 45 MHz BUFCLK. In 16-bit mode, the sustained DRAM buffer bandwidth, during concurrent host and disk transfers is 36 MBytes/sec while using a 45 MHz BUFCLK.

SRAM configurations of up to 256 KBytes are supported. Two independent RAM chip selects allow the option for two separate SRAMs (up to 128K Bytes each) to be addressed and accessed. In 8-bit mode, the sustained SRAM bandwidth, during concurrent host and disk transfers, is 25 MBytes/sec utilizing a 50 MHz BUFCLK. In 16-bit mode, the sustained SRAM bandwidth, during concurrent host and disk transfers, is 50 MBytes/sec utilizing a 50 MHz BUFCLK. Flexible control options allow various speed SRAMs to be matched with available BUFCLK frequencies to support the desired throughput.

The Buffer Control block incorporates very flexible segmentation support. Two operational segments can be set up to support general read/write, auto write, and read caching (auto read) algorithms. The segment size is programmable to any value up to 4 MBytes with 1-byte resolution (2-byte resolution in 16-bit mode). In addition, there is special segment support for storing disk servo split pointers. The various pointers are designed to automatically wrap at segment boundaries to ensure data integrity without microprocessor intervention.

Additional functionality is provided in the Buffer Control block through the following features:

- Increased automation to support zero latency read operations with minimal latency and true buffer alignment.
- Support for sector sizes up to 1500 bytes in length.
- Capability to support the execution of multiple consecutive Auto-Write commands without loss of data due to overwriting of data.
- 8/16-bit buffer data bus with optional parity feature for DRAM.
- Support for delayed block release.
- Write cache pointer.
- A disk sector counter that can monitor the transfers between the disk and buffer.
- Servo Pointer wrap capability.
- Two independent BCTR registers for use with any two selectable buffer segments.

1.2.4 The Disk Control Block

The AIC-8375 Disk Control block manages the flow of data between the disk and the buffer. It is capable of performing completely automated track read and write operations at a maximum data rate of 50 Mbits/sec in single NRZ mode, 100 Mbits/sec in dual NRZ mode, or 120 Mbits/sec in byte-wide NRZ mode. The AIC-8375 Disk Control block has been redesigned to allow for EDSA headerless support. This design is the first headerless only design in that ID fields or headers are no longer supported. Many flexible features and elements of automation have been incorporated to complement the automation contributed by the host and buffer blocks.

The Disk Control block consists of the programmable sequencer, EDAC circuitry, CDR/data split logic, disk FIFO, and other support logic.

The programmable sequencer contains a 31 by 3 byte programmable SRAM and associated control logic, which is programmed by the user, to automatically control all single track format, read, and write operations. From within the sequencer micro program, the Disk Control block can automatically deal with such real time functions as defect skipping, servo burst data splitting, branching on critical buffer status, address verification, and data compare operations. Each sequencer instruction is capable of an execution count of 8 KBytes. Once the sequencer is started, it executes each word in logical order. At the completion of the current instruction word, it either continues to the next instruction, continues to execute some other instruction based upon an internal or external condition having been met, or stops. During instruction execution or while stopped, registers can be accessed by the local microprocessor to obtain status information reflecting the sequencer operations taking place.

In addition to the flexible sequencer, the Disk Control block contains many other features which are available to satisfy diverse requirements. These include:

- Support for optimized zero latency read operations, with minimal microprocessor intervention.
- Programmable "Wrap To" register.
- Ability to specify at which sector to wrap "wrap to" register, independent of the stopping sector.
- Index counter for power management command support.
- Missing RRCLK detector.
- Time-out support when waiting for Sync, Index, Sector, and End of Servo Burst to relieve microprocessor of overhead associated with managing time outs.
- Optional 2-byte fault tolerant Byte Sync.
- Configurable logic to deal with CDR and embedded servos to allow adaptation to diverse disk formats.
- Ability to automatically stop the sequencer on receipt of an Auto-Write command.
- Split data support from microprocessor or buffer. Buffer split data support utilizes indexing into the buffer servo segment.
- A 32-byte FIFO between the buffer and the Disk Control block smoothes out data flow attributed to discontinuities in data or differences in speed.
- Read Gate and Write Gate are directly controlled from the disk sequencer micro program.

Features that have been added to support EDSA headerless formats include:

- End of Servo (EOS) counter.
- EOS MAX register which allows the EOS counter to wrap at a specified value.
- Ability to clear the EOS counter by Index.
- EOS compare instruction (EOSCMP) in the disk sequencer.
- Current Sector counter (Disk Sequencer Physical Position Counter).
- Ability to reset current CDR instruction

Error detection and correction is handled in the Disk Control block. The AIC-8375 provides a fixed, 3-way interleaved, 168-bit Reed-Solomon code. Automatic on-the-fly hardware correction will take place for up to three symbols in error per interleave. Correction is guaranteed to complete before the ECC Field of the sector following the sector where the error occurred. Optional burst limiting can be used to decrease the probability on misdetection and miscorrection. The ECC can be seeded with a four byte value that consists of the requested sector value plus three bytes specified by the user. An added feature of the AIC-8375 ECC block is the ability to log corrected ECC errors.

1.2.5 Power Management

Power management features are incorporated into each block of the AIC-8375. This allows the designer to tailor the amount of power management to the specific design as required. Other power management features include:

- Independent power management control for each block.
- ECC logic automatically powered down when not in use and powered up when needed.
- Disk sequencer and associated disk logic powered up when the sequencer is started.
- Weak pull-up structure on input pins to prevent undesirable power consumption due to floating CMOS inputs.

2.1 Interfacing to the Local Microprocessor

2.1.1 Selecting the Microprocessor Bus Mode

The AIC-8375 is programmable to be compatible with several different microprocessor interfaces. The AIC-8375 is autoconfigured at power-on time to the specific microprocessor to which it is interfaced. Immediately after *POR is negated, the AIC-8375 samples the voltage on the BD14/PSEL0 and BD15/PSEL1 pins. At this time these two pins function as the PSEL[1:0] signals. If the AIC-8375 is configured for PCMCIA Mode (see Section 2.3.1), PSEL[1:0] maps to BA[6:5]. The value of PSEL[1:0] for each of the microprocessor classes are shown in the table below.

Table 2-1 Microprocessor Class PSEL[1:0] Values

Microprocessor Type	Microprocessor Signal Usage					Configuration Control Signal Pins		
	DSTB	DIR	CS	CSLAT	INT	RSEL	PSEL[1:0]	PMUX
Intel 80C196KB20	*RD/*WR	---	---	1	0	1 = READY	1 1	1
Siemens 88C165, 6, 7	*RD/*WR	---	---	1	0	0 = *READY	1 1	1 or 0
Motorola 68HC11	E	R/*W	*CS	1	0, ↓	---	1 0	1 or 0
Motorola 68HC11F1	*CS	R/*W	---	0	0, ↓	---	0 1	1 or 0
Motorola 68HC16	*DS	R/*W	*CS	0	0	0 = *DSACK0	0 1	0
TI TMS320C25 (DSP)	*DS	R/*W	---	1	0	1 = READY	0 0	0
Adaptec AIC-4420	*RD/*WR	---	---	1	0	---	1 1	0

DSTB = Type of data strobe: *RD/*WR for active low read and write strobes, E for active high read/write strobe, *CS for active low read/write strobe gated with Chip Select and driving the CS or BS input (68HC11F1), and *DS for active low read/write strobe.

DIR = Direction Control: --- for none due to separate *RD and *WR strobes, and R/*W for direction.

CS = Type of Chip Select: *CS for active low, CS for active high, and --- for none supplied by processor.

CSLAT = Chip Select Latch Control: When a "1" the chip selects are latched, otherwise they are not latched. The 68HC11F1 when run in ECLK Chip select mode, cannot latch CS. The 68HC16 cannot latch CS because there is not guaranteed positive setup time to DSTB.

INT = Type of Interrupt: 0 for low level, 1 for high level, ↑ for rising edge detect, ↓ for falling edge detect, --- for none supplied by the processor.

RSEL = Type of Wait State Control: 1 for active high READY, 0 for active low *READY (*DSACK for 68HC16), --- for none supported by the processor so RSEL is don't care.

PSEL = Processor Select Control, PSEL[1:0] inputs driven to this value to select the processor shown.

PMUX = Type of Processor Bus: 0 for non-multiplexed, 1 for multiplexed (0 or 1 if can support both).

These pins are internally pulled high so that external resistors tied to Vdd are not required. However, to operate in Motorola or TI modes, a high value resistor must be connected between the PSEL0 and/or PSEL1 pins and Vss. The value chosen must be high enough so as not to interfere with the operation of the buffer parity function during buffer transfers.

The functionality of the E/*RD/*DS pin and the R/*WR pin is dependent upon which mode is selected. In mode 3 (PSEL[1:0] = '11'), the E/*RD/*DS and R/*WR input pins function as *RD and *WR respectively. In mode 2 (PSEL[1:0] = '10'), they function as E and R/*W inputs respectively. In modes 1 and 0, they function as *DS and R/*W inputs respectively with latched CS for mode 0 and no latched CS for mode 1.

2.1.2 Selecting Multiplexed or Non-Multiplexed Address Mode

Immediately after *POR is negated, the AIC-8375 samples the voltage on the BD13/PMUX pin (BA4/PMUX in PCMCIA mode). If it is found to be low, the device is configured for non-multiplexed operation. If high, the device is configured for multiplexed bus operation. This pin is pulled high by an internal high resistance. Therefore, if it is desired to operate the AIC-8375 in the multiplexed bus mode, an external resistor tied to Vdd is not required. However, to operate in non-multiplexed mode, a high resistance resistor must be connected between the PMUX pin and Vss. The value chosen must be high enough so as not to interfere with the operation of the PMUX pin during buffer transfers.

2.1.3 Enabling the Internal Address Latch

The AIC-8375 has an internal 8-bit latch that latches the low-order byte of the address during a multiplexed mode microprocessor memory access cycle. This latched value is then presented on the eight MA[7:0] outputs and can be used by the rest of the system. This eliminates the need for an external "373" latch for address/data multiplexing. In non-multiplexed mode, the MA[7:0] pins become microprocessor address inputs.

The internal address latch is enabled when the multiplexed address mode is selected. On the falling edge of ALE, the value on the AD[7:0] pins is latched in the internal transparent latch and held on the MA[7:0] outputs.

2.1.4 Chip Select Generation and Memory Mapping

The AIC-8375 utilizes two separate chip selects to access the various areas associated with the device. These chip selects are CS (pin 100) and BS (pin 101). The designer is required to provide address decoding for these two chip selects. The registers and Sequencer RAM within the AIC-8375 are accessed by the microprocessor using the CS input along with the appropriate address. The Buffer RAM is accessed by using the BS input. Refer to Section 3 for more functional details on how the microprocessor accesses the device registers, the Sequencer RAM, and the Buffer RAM.

The polarity of the CS and BS signals is determined when *POR is negated at power-up. At this time, the BD12/CSP pin (BA3/CSP in PCMCIA mode) is sampled. If it is found to be low, the polarity of the CS pin will be active low. If high, CS will be active high. Likewise, the BD11/BSP pin (BA2/BSP in PCMCIA mode) is sampled when *POR is negated at power-up. If this pin is found to be low, BS will be active low, and if high, BS will be active high. These pins incorporate an internal pullup resistor to bias them high. An external pulldown resistor is required if a low sample level is desired.

Refer to Section 4 of this document for more detailed information on the sequencer RAM address mapping. Section 5 of the AIC-8375 Data Sheet contains further details on the AIC-8375 register mapping while using CS.

2.1.5 Using the Ready Signal

The Ready function provides the AIC-8375 with the capability of requesting the microprocessor to insert wait states during the current microprocessor read or write access. The AIC-8375 will drive the Ready pin low to notify the microprocessor that it is "not ready" for it to end the current access. When the microprocessor detects that the AIC-8375 is not ready, it will extend the current read or write access until the AIC-8375 drives the Ready pin high, indicating that it is now ready.

The Ready function is obviously useful when interfacing to very fast microprocessors. However, even if a slower microprocessor is used, there are times when the Ready line is required. This is true when the buffer RAM or sequencer RAM are being accessed. Because access to the buffer is shared between the host, disk, ECC correction logic, and refresh, the Ready line is necessary to hold off the microprocessor when a microprocessor access is attempted while the buffer is being accessed by another source. The Sequencer RAM has internal restrictions as to when it can be accessed during instruction cycles. As a result, the Ready line is asserted to handle these situations.

The READY function polarity is chosen based on the value found on the BD10/RSEL pin (BA1/RSEL in PCMCIA mode) after power-on-reset is negated as shown in Table 2-1.

The Enable Push-Pull READY bit (reg. 51h, R/W, bit 3) is used to configure the READY line for either push-pull or open drain configurations. This is useful for connecting to dedicated or shared interrupt lines.

The Ready line timing that is implemented when accessing the buffer RAM or sequencer RAM (using the BS chip select) is slightly different from that used for the CS chip select. Refer to the appropriate timing diagrams in the *Timing Specifications* section of the AIC-8375 Data Sheet for details on these differences.

2.1.6 Interrupt Sources to the Microprocessor

Two interrupt signal pins are available for sending interrupts to the microprocessor; INTD is the source for disk block related interrupts and INTHB is the source for host block or buffer block related interrupts (the logical 'OR' of all interrupts from these two blocks). All of the device interrupts from the host, disk, and buffer blocks can be output on the INTHB signal pin if the COMBINT bit (reg. 51h, R/W, bit 2) is set.

The polarity of both the INTD and INTHB signal pins is determined by the ACTHIINT bit (reg. 51h, R/W, bit 0).

At power up, these pins default to an active low/open drain state. In open drain mode, an external 1-10K Ohm pullup resistor is required on this pin. These pins can be configured as push-pull if the ENPPINT bit (reg. 51h, R/W, bit 1) is set.

2.2 Buffer RAM Device Selection

The choice of which buffer RAM device to use in a particular design will depend on several factors. These factors include speed, real estate, cost, available system clocks, the need for buffer parity support, and required buffer bandwidth. The following two sections provide technical information on how to interface static RAMs and dynamic RAMs to the AIC-8375. This information will provide a base from which the user can establish buffer memory requirements for a particular application.

2.2.1 Static RAM

A maximum of 256 KBytes of SRAM may be addressed using the available eighteen buffer address lines (BA[17:0]). The implementation of this address space will be determined by the users choice of the following variables:

- One, two, or multi-RAM device implementation
- 8-bit- or 16-bit-wide buffer data path
- SRAM cycle time (2T, 3T, 4T, or 5T)
- Type of SRAM (with or without an *OE pin)

The RAMSEL[2:0] bits (reg. 100h, R/W, bits 5:3) together with the BEN16BIT bit (reg. 100h, R/W, bit 2) establish the physical SRAM configuration. The BEN16BIT control bit establishes an 8- or 16-bit buffer data path. The types of configurations available are summarized in Tables 2-2 and 2-3.

Table 2-2 8-bit SRAM Buffer Configuration

RAMSEL[2:0]	SRAM Types	SRAM Control	BEN16BIT	BA0/*WE1	*WE0/*CAS0
0 0 0	Single X 8 SRAM (with *OE pin)	*MOE controls SRAM *OE pin	0 (8-bit)	used as BA0	used as *WE0
0 0 1	Single X 8 SRAM (without *OE pin)	*MCE controls SRAM *CE pin	0 (8-bit)	used as BA0	used as *WE0
0 1 0	(Reserved)	(Reserved)	(Reserved)	---	---
0 1 1	(Reserved)	(Reserved)	(Reserved)	---	---
1 0 0	Dual 32Kx8 SRAMs	*MCE1 and *MCE2 control SRAM *CE pins	0 (8-bit)	used as BA0	used as *WE0
1 0 1	Dual 64Kx8 SRAMs	*MCE1 and *MCE2 control SRAM *CE pins	0 (8-bit)	used as BA0	used as *WE0
1 1 0	Dual 128Kx8 SRAMs	*MCE1 and *MCE2 control SRAM *CE pins	0 (8-bit)	used as BA0	used as *WE0
1 1 1	(DRAM)				

Table 2-3 16-bit SRAM Buffer Configurations

RAMSEL[2:0]	SRAM Types	SRAM Control	BEN16BIT	BA0/*WE1	*WE0/*CAS0
0 0 0	Single X 16 SRAM (with *OE pin) Two X 8 SRAMs (with *OE pins)	*MOE controls SRAM *OE pin	1 (16-bit)	used as *WE1	used as *WE0
0 0 1	Single X 16 SRAM (without *OE pin) Two X 8 SRAMs (without *OE pins)	*MCE controls SRAM *CE pin	1 (16-bit)	used as *WE1	used as *WE0
0 1 0	(Reserved)	(Reserved)	(Reserved)	---	---
0 1 1	(Reserved)	(Reserved)	(Reserved)	---	---
1 0 0	Dual 32Kx16 SRAMs	*MCE1 and *MCE2 control SRAM *CE pins	1 (16-bit)	used as *WE1	used as *WE0
1 0 1	Dual 64Kx16 SRAMs	*MCE1 and *MCE2 control SRAM *CE pins	1 (16-bit)	used as *WE1	used as *WE0
1 1 0	Dual 128Kx16 SRAMs	*MCE1 and *MCE2 control SRAM *CE pins	1 (16-bit)	used as *WE1	used as *WE0
1 1 1	(DRAM)				

The AIC-8375 can be interfaced to SRAMs which have an *OE (output enable) and *CS (chip select) control input as well as SRAMs which do not have an *OE control input. The *MOE/*RAS/*MCE1 pin is typically connected to the *OE pin of those SRAMS which have this control input. If the SRAM does not have an *OE control input, the *MOE/*RAS/*MCE1 pin is connected to the *CS or *CE pins of the SRAM.

Note that the functions of the *MOE/*RAS/*MCE1 signal pin and the *WE0/*CAS0 signal pin are different in SRAM and DRAM modes.

The SRAM cycle time is a function of the BUFCLK which is provided by the user. The AIC-8375 provides the user with a range of four different SRAM cycle times. These cycle times are typically referred to as "2T, 3T, 4T, and 5T" SRAM modes. The desired cycle time is selected via the BUFCYC[1:0] bits (reg. 100h, R/W, bits 7:6). These cycle times give the user the ability to juggle three critical parameters: SRAM speed, available BUFCLK, and desired buffer bandwidth. Figure 2-1 shows the relationship of these three parameters.

Since the microprocessor and ECC correction logic can potentially use the buffer, the bandwidth of the buffer, which will be available for host and disk transfers, will vary. Although this is typically only a small portion of the overall bandwidth, it should be taken into account. Figure 2-1 shows the relationship between SRAM buffer bandwidth, buffer cycle time, BUFCLK, and SRAM access time assuming typical microprocessor and correction bandwidth as indicated.

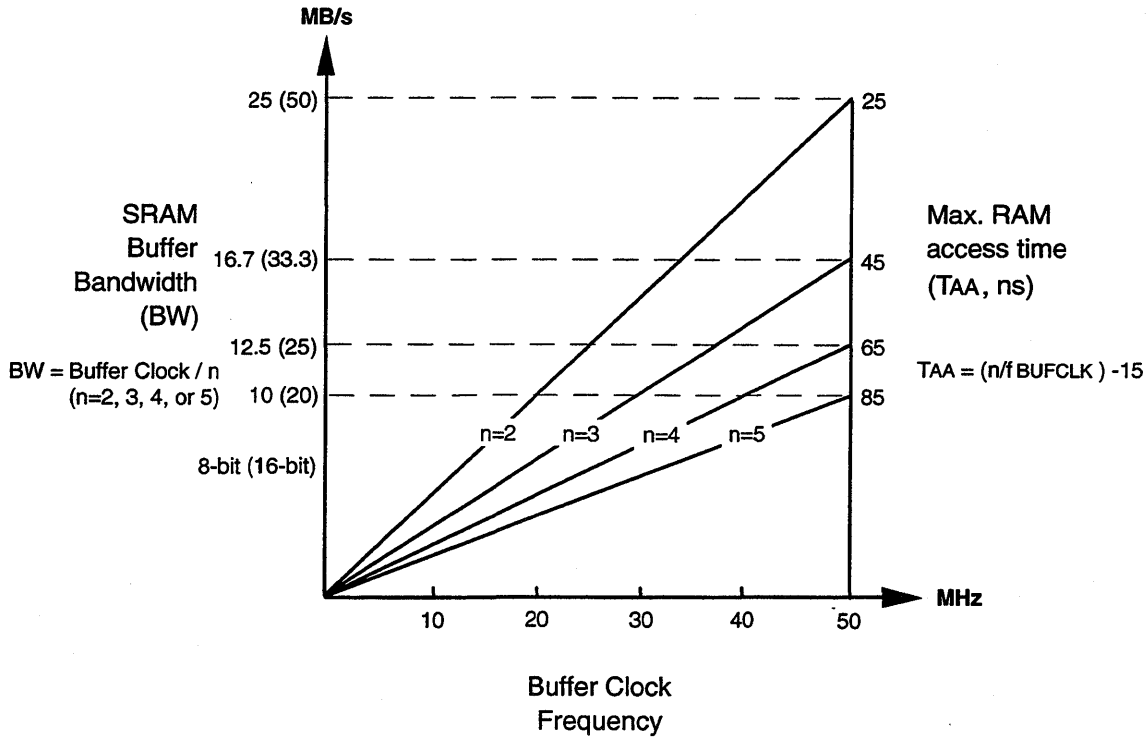

Figure 2-1 SRAM Performance Relationships

The generalized equation for SRAM Bandwidth is as follows:

- Buffer BW = [f_{BUFCLK}/Cycles per access] - [NDRA]

  where NDRA = Average frequency (in MBytes/s) of all *N*on-*D*ata *R*elated *A*ccess due to ECC corrections, microprocessor access, and Servo Pointer access.

The buffer RAM device access time, T_{AA}, is calculated as follows:

- $T_{AA} = T \bullet n - 15 = (n/f_{BUFCLK}) - 15$ ns.

  where T_{CYC} = 2T, 3T, 4T, or 5T ($T = 1/f_{BUFCLK}$)

2.2.2 Dynamic RAM

The AIC-8375 is designed to operate with page mode DRAMs. Disk accesses to the buffer will occur using 8-byte or 8-word page mode accesses. Host accesses consist of page mode transfers of up to 256 bytes. Page mode accesses of less than eight bytes/words can occur at page boundaries, microprocessor accesses, ECC auto-correction accesses, and odd byte/word number accesses. Likewise, host side accesses will vary in length depending upon the dynamic operating conditions.

Up to 4 MBytes of DRAM address space may be configured. Address lines BA[10:0] are used to provide CAS and RAS addresses to the DRAMs. BA[15:11] are not used for DRAM addressing. The DRAM buffer path can be 8- or 16-bits wide. When configured in 8-bit mode via the BEN16BIT control bit (Reg. 100h, R/W, bit 2), the RAS function is established on the *MOE/*MCE1/*RAS pin and the CAS function is provided on the *WE0/*CAS0 pin. When configured for 16-bit mode, the *MOE/*RAS/*MCE1 pin and the *WE0/*CAS0 and BA17/*CAS1 pins are configured as *RAS and *CAS respectively. The BA13/*WRITE pin is used as *WE. The 8-bit and 16-bit modes are summarized in Tables 2-4 and 2-5.

Table 2-4 DRAM Address Bit Specification (8-bit Mode)

BUFFER ADDR. PINS	BA0	BA1	BA2	BA3	BA4	BA5	BA6	BA7	BA8	BA9	BA10
CAS (Addr. Bits)	0	1	2	3	4	5	6	7	17	19	21
RAS (Addr. Bits)	8	9	10	11	12	13	14	15	16	18	20
<---------- 64KB DRAM ---------->											
<---------- 28KB/256KB DRAM ---------->											
<---------- 512KB/1 MB DRAM ---------->											
<---------- 2 MB/4 MB DRAM ---------->											

Table 2-5 DRAM Address Bit Specification (16-bit Mode)

BUFFER ADDR. PINS	BA0	BA1	BA2	BA3	BA4	BA5	BA6	BA7	BA8	BA9	BA10
CAS (Addr. Bits)	16	1	2	3	4	5	6	7	18	20	--
RAS (Addr. Bits)	8	9	10	11	12	13	14	15	17	19	21
<---------- 64KWord DRAM ---------->											
<---------- 28KWord/256KWord DRAM ---------->											
<---------- 512KWord/1 MWord DRAM ---------->											
<---------- 2 MWord DRAM ---------->											

Since the AIC-8375 performs DRAM write operations utilizing "Early Write" mode, the *OE pin of the DRAMs can be tied to ground. Optionally, the *WE signal output can be inverted and connected to the *OE pin of the DRAM. When not writing to the device, the *OE control pin of the DRAM will be asserted but the DRAM will typically be in the output mode only when qualified internally by the *RAS and *CAS signals. If interfacing to DRAMs which do not have an *OE pin, the *WE signal is connected to the *WE input of the DRAM. In this case, the DRAM will be in output mode at the appropriate occurrence of *RAS and *CAS.

Since DRAM refresh will use buffer bandwidth and the microprocessor and ECC correction logic, servo port accesses can all potentially use the buffer. The bandwidth of the buffer, which will be available for host and disk transfers, will vary. Although this is typically only a small portion of the overall bandwidth, it should be taken into account.

The generalized equation for available DRAM buffer bandwidth is as follows:

- DRAM buffer BW = $f_B \bullet [(w \bullet bl)/((a \bullet bl) + oc)]$ - [NDRA]

  where f_B = BUFCLK frequency

  w = Buffer bus width in bytes (1 or 2)

  bl = DRAM page mode access burst length (8)

  a = access cycles (2: no wait states, 3: one wait state)

  oc = overhead bytes (4 or 5)

  NDRA = *N*on *D*ata *R*elated *A*ccesses due to ECC correction, servo, µP, and refresh

For 8-bit mode (no wait states), DRAM buffer BW = $2\ f_B/5$ - [NDRA]

For 8-bit mode (one wait state), DRAM buffer BW = $8\ f_B/29$ -[NDRA]

For 16-bit mode (no wait states), DRAM buffer BW = $4\ f_B/5$ - [NDRA]

For 16-bit mode (one wait state), DRAM buffer BW = $16\ f_B/29$ -[NDRA]

Figure 2-2 shows the relationship between DRAM buffer bandwidth, BUFCLK, and DRAM access time assuming no factor for any microprocessor, servo, correction, and refresh bandwidth. To factor in the effects of these second order terms, the appropriate (NDRA) value must be subtracted from the data points in the graph to arrive at the actual performance curves.

Wait states are enabled by using the Buffer Cycle Time Select bits (Reg. 100h, R/W, bits 6 and 7) while DRAM mode is selected.

The number of BUFCLK periods required to complete a DRAM access of "n" number of bytes is given by the following equations:

DRAM access without wait states:

- BUFCLK Cycles = [4 + 2n]
- $T_{RAC} = 3T + T_{BL} - 10$
- $T_{CAC} = T + T_{BL} - 10$
- $T_{CPA} = 2T - 8$

DRAM access with wait states:

- BUFCLK Cycles = [5 + 3n]
- $T_{RAC} = 4T + T_{BL} - 10$
- $T_{CAC} = 2T + T_{BL} - 10$
- $T_{CPA} = 3T - 8$

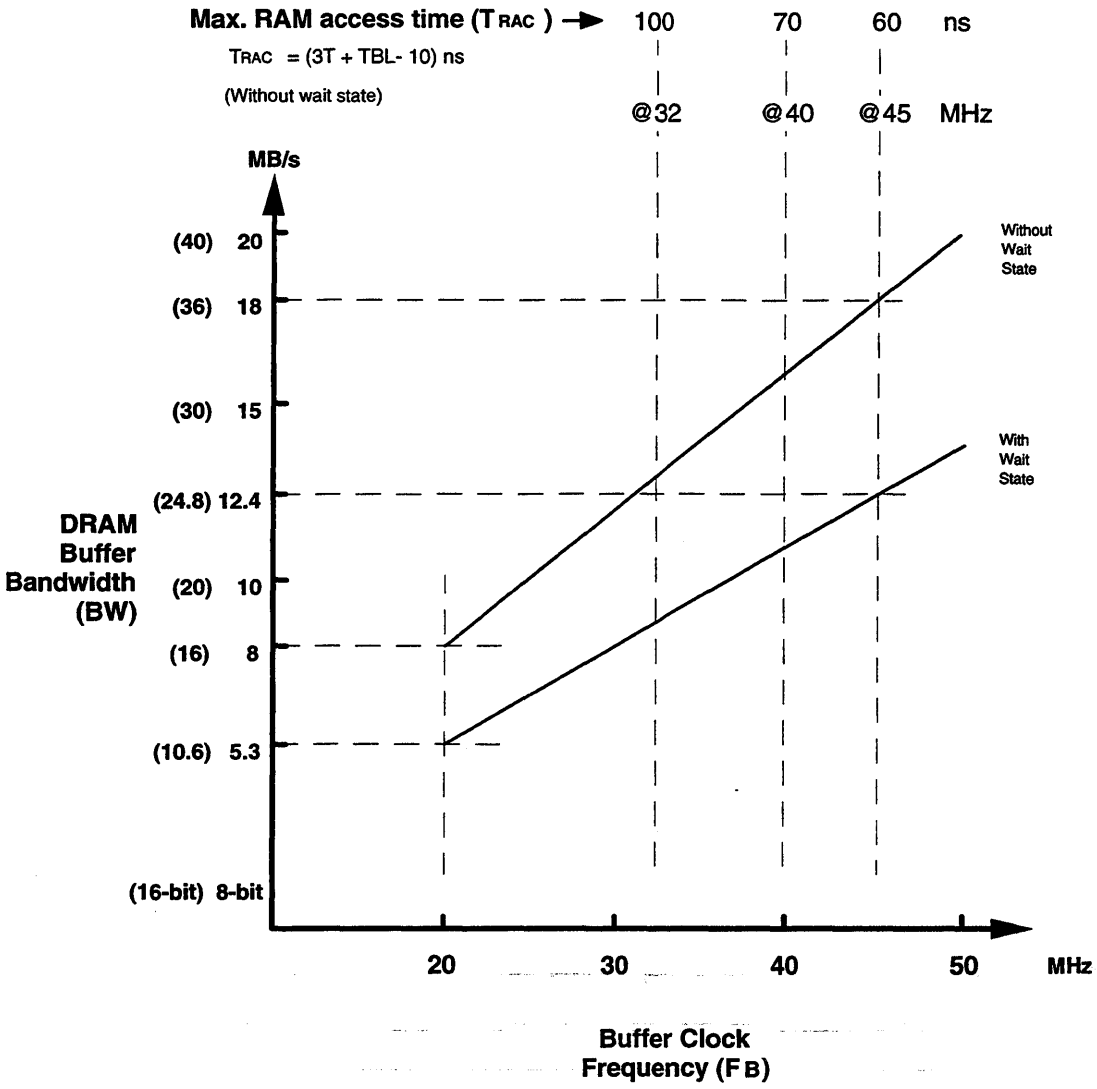

Figure 2-2 DRAM Performance Relationships

2.3 Interfacing to the Host

2.3.1 Selecting ATA or PCMCIA Mode

The AIC-8375 provides two host interfaces: ATA and PCMCIA. This section describes the selection and initialization requirements for each interface.

2.3.1.1 Automatic Detection

The AIC-8375 is designed to operate in either a PCMCIA or ATA configuration. There is currently a well defined PCMCIA 68-pin electrical interface. There is also a proposed 68-pin interface that uses the ATA electrical signals mapped to similar signal pins on a 68-pin PCMCIA compatible connector. This would allow a PCMCIA/ATA card to work in either type of socket for transferring data between systems (i.e., notebook and desktop systems).

The AIC-8375 has a built-in method for detecting whether it has been selected for the PCMCIA or ATA if at power-on time (*POR negated) the BA7 pin is checked. If it is a '1' ATA Mode is selected and if it is a '0' PCMCIA Mode is selected. A 6.8K pulldown resistor is used to bias the pin low.

2.3.1.2 Programmed Selection

After power-on reset, the host can override the automatic interface selection by writing to the HOSTSEL[1:0] bits (reg. E6h, R/W, bits 1:0).

2.3.1.3 Initializing the AIC-8375 for PCMCIA Operation

The AIC-8375 must be initialized for PCMCIA operation after Power-On Reset as follows:

- Set *HCE2/*HCS1 to A9 mode (clear reg. C0h, R/W, bit 3).
- Negate *PDIAG and *DASP signals (set reg. BFh, R/W, bits 6, 5) (default at *POR).
- Disable Master/Slave mode (clear reg. C0h, R/W, bit 1).
- Select 16-bit transfer mode (set reg. C5h, R/W, bit 6). The host selects 8- or 16-bit transfer mode by way of the *HCE1/*HCE2 signals.
- Initialize the CIS in the buffers starting at address 0 (see Section 2.3.7.1).
- Enable HCISPEN to allow the host to read the CIS by setting HPEN (reg. 109h, R/W, bit 5).

2.3.1.4 Initializing the AIC-8375 for ATA Operation

The AIC-8375 must be initialized for ATA operation after Power-On Reset as follows:

- Set *HCE2/*HCS1 to *HCS1 mode (set reg. C0h, R/W, bit 3).
- Negate *PDIAG and *DASP signals (set reg. BFh, R/W, bits 6, 5) (default at *POR).

2.3.2 ATA/IDE Interface Support

The AIC-8375 is designed to directly interface to the ATA bus without any extra glue logic. In other applications which do not adhere strictly to the ATA bus, or in applications which interface directly to the AT bus, it will be necessary to generate the *HCS0 register select signal. The bus drivers incorporated in the AIC-8375 are 12-mA drivers capable of driving the AT bus or ATA interface.

2.3.3 AT Register Mapping

The AIC-8375 is designed to interface directly to the ATA/IDE bus with no additional decoding or logic. This is termed *HCS1 mode. The device can also be used in applications in which the AIC-8375 resides on the same printed circuit board along with a floppy drive controller or other peripheral control devices. In this latter environment, the AIC-8375 may be operated in a mode which eliminates the requirement of having to generate two separate host chip select signals (*HCS0 and *HCS1). In this case, the HA9/*HCS1 pin can be directly connected to address line 9 on the AT bus. This is referred to as HA9 mode. *HCS1 mode is enabled when the HCS1SEL bit (reg. C0h, R/W, bit 3) is set. HA9 mode is selected if the HCS1SEL bit is cleared. These two modes of operation are summarized in Table 2-6.

Table 2-6 AT Register Mapping

1. When HCS1 mode is disabled (HCS1SEL, reg. C0h, R/W, bit 3 cleared), the HA9/*HCS1-pin functions as HA9, and the host registers are mapped as follows:

DRQ	BSY	*HCS0	HA9	HA[2:0]	Read	Write
1	0	0	0	0h	H_DATA	H_DATA
x	0	0	0	1h	H_ERROR	H_FEATRS
x	0	0	0	2h	H_SECCNT	H_SECCNT
x	0	0	0	3h	H_SECNUM	H_SECNUM
x	0	0	0	4h	H_CYLHI	H_CYLHI
x	0	0	0	5h	H_DRVHD	H_DRVHD
x	0	0	0	6h	H_STAT	H_CMD
x	0	0	0	7h	H_STAT	not allowed
0	1	0	0	x	H_STAT	not allowed
x	x	0	1	6h	H_ALSTAT	H_DVCTL
x	x	0	1	7h	H_DRVADD	reserved

2. When HCS1 mode is enabled (HCS1SEL, reg. C0h, R/W, bit 3 set), the HA9/*HCS1 - pin functions as *HCS1, and the host registers are mapped as follows:

DRQ	BSY	*HCS0	*HCS1	HA[2:0]	Read	Write
1	0	0	1	0h	H_DATA	H_DATA
x	0	0	1	1h	H_ERROR	H_FEATRS
x	0	0	1	2h	H_SECCNT	H_SECCNT
x	0	0	1	3h	H_SECNUM	H_SECNUM
x	0	0	1	4h	H_CYLLO	H_CYLLO
x	0	0	1	5h	H_DRVHD	H_DRVHD
x	0	0	1	6h	H_STAT	H_CMD
x	0	0	1	7h	H_STAT	not allowed
0	1	0	1	x	H_STAT	not allowed
x	x	1	0	6h	H_ALSTAT	H_DVCTL
x	x	1	0	7h	H_DRVADD	reserved

2.3.4 ATA Interface Signal Requirements

There are serveral points related to ATA signal management which are addressed below.

2.3.4.1 Bus Drivers and Receivers

The AIC-8375 incorporates 12- and 24-mA. drivers which are designed to connect directly to the AT or ATA bus.

All host interface input signals implement the equivalent of internal weak pullup resistors. This prevents the host interface pins from floating when not being driven either externally or internally. This greatly reduces power consumption caused by floating CMOS signal pins.

The host data bus drivers can be programmed to continuously drive the bus during a host read cycle while *HCS0, HA9/*HCS1, and HA[7:0] are properly addressing the AIC-8375's data port. This feature provides additional support for PIO modes 3 and 4. This function is enabled when the HDBDRVPIO bit (reg. C6h, R/W, bit 0) is set.

The receivers incorporate hysterisis to reduce signal integrity problems.

Refer to the *Timing Specifications* section of the AIC-8375 Data Sheet for more technical parameters on the host interface signals.

2.3.4.2 Specific ATA Signal Requirements

Various signals require specific attention. Most of these are specified in the ATA specification. Any additional information is presented below.

The internal pullup resistors on the HA[2:0] signal pins can be disabled by setting the Disable Host Address Pull-Ups bit (reg. 51h, R/W, bit 7).

The internal pullup resistors on the *PDIAG and *DASP signal pins can be disabled by setting the Disable Host Control Pull-Ups bit (reg. 51h, R/W, bit 6).

All internal pullup resistors can be disabled by setting the Disable All Pull-Ups bit (reg. 50h, R/W, bit 7).

2.3.5 Master/Slave Support

Functionality has been incorporated into the AIC-8375 to support the master/slave (daisy chained) configuration as specified in the ATA specification. In order to operate in the master/slave environment, the drive must be designed with one of three methods for designating whether it is drive '0' (the master) or drive '1' (the slave). These methods are:

- a jumper on the drive
- a switch on the drive
- utilize the Cable Select (CSL) function on the ATA interface

Once this capability has been established, the features of the AIC-8375 which support master/slave operation can be used. The *PDIAG pin (Passed Diagnostics, pin 22) and the *DASP pin (Drive Active/Drive 1 Present, pin 21) support the master/slave protocol.

In the ATA master/slave environment, the *DASP cable line will be used/shared by both drive '0' and drive '1'. It will be used at different times for different purposes. It is driven by the slave to inform the master that it is present. It can also be used by either drive to drive an activity LED. The *DASP pin is driven/read by the *DASP bit (reg. BFh, R/W, bit 5). When set, this bit forces the *DASP into a high impedance state. When cleared, *DASP will be driven low. When read, this bit reflects the state of the *DASP pin.

The *PDIAG cable line is used by the slave drive to inform the master drive that it has passed power-on diagnostics. It is used only for drive to drive communications. When the *PDIAG bit (reg. BFh, W, bit 6) is set, the *PDIAG pin will be in the high impedance state. When *PDIAG is cleared, the *PDIAG pin will be driven low. When read, this bit reflects the state of the *PDIAG pin.

2.3.6 PCMCIA Interface Support

There are three basic ways for the host to communicate with a PCMCIA/ATA card containing an AIC-8375:

- I/O Addressing Modes (Primary/Secondary and Block)
- Memory Addressing Mode
- Attribute Memory (AMEM)

One of the three I/O and Memory Addressing Modes must be selected at all times. The Attribute Memory is accessible to the host at any time the AIC-8375 is using the PCMCIA Interface and is not in a Busy state (on power-up, on Host Reset or when bit 6 of the Host Status 2 register (reg. CEh) is set to 1).

The Addressing Mode is selected by the host and used to access the ATA Task File Registers. At power-on, or after a PCMCIA reset, the AIC-8375 is in the Memory Addressing Mode by default. The host accesses the Task File Registers using memory strobes (*OE, *WE) with *REG not asserted. The card will not respond to any I/O strobes (*IORD, *IOWR) in this mode.

The I/O Addressing Mode is selected by the host writing the proper Configuration Index to bits 3:0 of the Configuration Option register (reg. 200h). The host uses I/O strobes to access the Task File Registers. The card will not respond to any Memory strobes unless *REG is asserted (for Attribute Memory).

Attribute Memory returns the card configuration Information (Card Information Structure or CIS area) and allows the host to change the configuration through the Card Configuration registers (CCR). This mode is accessed using memory strobes with the *REG signal asserted.

Table 2-7 summarizes the various host to card communication methods.

Table 2-7 AIC-8375 Addressing Modes for Host Access

Addressing Mode	Address Signals											Strobe Signals				
	10	9	8	7	6	5	4	3	2	1	0	*REG	*OE	*WE	*IOR	*IOW
AMEM: CCR Read	x	1	x	x	x	x	x	V	V	V	x	0	0	1	1	1
AMEM: CCR Write	x	1	x	x	x	x	x	V	V	V	x	0	1	0	1	1
AMEM: CIS Read	x	0	V	V	V	V	V	V	V	V	x	0	0	1	1	1
I/O: Pri/Sec I/O Read	x	V	V	V	V	V	V	V	V	V	V	0	1	1	0	1
I/O: Pri/Sec I/O Write	x	V	V	V	V	V	V	V	V	V	V	0	1	1	1	0
I/O: Blk I/O Read	x	x	x	x	x	x	x	V	V	V	V	0	1	1	0	1
I/O: Blk I/O Write	x	x	x	x	x	x	x	V	V	V	V	0	1	1	1	0
MEM: Mem Read	V	x	x	x	x	x	x	V	V	V	V	1	0	1	x	x
MEM: Mem Write	V	x	x	x	x	x	x	V	V	V	V	1	1	0	x	x

x = don't care, V = Valid signal, 0 = logic zero, 1 = logic one

2.3.6.1 Primary/Secondary I/O Mode Addressing

The Primary/Secondary Mode is the standard ATA Mode. It should be used when compatibility with existing BIOSes is needed. In this mode, the AIC-8375 internally decodes the 1K I/O address space and only responds when the address is within the selected range. The host selects which address range the chip will decode through the Configuration Index (bits 3-0 of the Configuration Option register (reg. 200h, R/W or the image of this register, reg. E0h, R)). If the Primary I/O mode is selected, the card responds to 1F0h-1F7h and 3F6h-3F7h. The Secondary I/O range is 170h-177h and 376h-377h.

Table 2-8 provides the Primary/Secondary I/O Mode addresses.

Table 2-8 Primary/Secondary I/O Mode Addresses

*REG	A10	A9-A4	A3	A2	A1	A0	Offset	*IOR	*IOW
0	X	1F(17)h	0	0	0	0	0	Even Data	Even Data
0	X	1F(17)h	0	0	0	1	1	Error Register	Features
0	X	1F(17)h	0	0	1	0	2	Sector Count	Sector Count
0	X	1F(17)h	0	0	1	1	3	Sector Number	Sector Number
0	X	1F(17)h	0	1	0	0	4	Cylinder Low	Cylinder Low
0	X	1F(17)h	0	1	0	1	5	Cylinder High	Cylinder High
0	X	1F(17)h	0	1	1	0	6	Drive & Head	Drive & Head
0	X	1F(17)h	0	1	1	1	7	Status	Command
0	X	3F(37)h	0	1	1	0	6	Alt Status	Device Control
0	X	3F(37)h	0	1	1	1	7	Drive Address	Reserved
1	X	X	X	X	X	X		No access	No access

Register 0 is accessed with *HCE1 low and *HCE2 low as a word register on SD15-SD0. This register may also be accessed by a pair of byte accesses to offset 0 with *HCE1 low and *HCE2 high. Note that the address space of this word register overlaps the address space of the Error and Feature byte Registers, which are at offset 1. When Register 0 is accessed twice as a byte register with *HCE1 low, the first byte to be accessed is the even byte of the word and the second byte accessed is the odd byte.

A byte access to Register 0 with *HCE1 high and *HCE2 low accesses the Error register on a read and the Feature register on a write.

If the host selects the Primary I/O Mode configuration, *INPACK is asserted when the host address is in the 1F0h to 1F7h or 3F6h to 3F7h range. If the Secondary I/O mode is selected, *INPACK is asserted for 170h to 177h or 376h to 377h addresses.

2.3.6.2 Block I/O (Independent) Mode Addressing

The Block I/O Address Mode is selected via the Configuration Index bits 3:0 in the Configuration Option register (reg. 200h, R/W or the image of this register, reg. E0h, R). This mode allows address independence by not internally decoding the host's address. It uses the *HCE1/*HCE2 signals (as generated by an external decoder) for address decoding. This allows the card to reside in a 16-byte address space. The host can assign multiple cards to independent I/O blocks without I/O port conflicts.

In the Block I/O Addressing Mode, the *IOIS16 signal determines the possible I/O transfer data width for the PCMCIA Interface. If this pin is asserted, this indicates that the addressed register is capable of 16-bit transfers. The host then asserts *HCE1 and *HCE2 for 16-bit transfers. Otherwise, 8-bit transfers are performed. For the ATA Interface, the *IOIS16 signal is renamed to *IOCS16. The host must perform a 16-bit transfer when this signal is asserted.

Table 2-9 provides the Block I/O Mode addresses.

Table 2-9 Block I/O Mode Addresses

*REG	A10-A4	A3	A2	A1	A0	Offset	*IOR	*IOW
0	X	0	0	0	0	0	Even Data	Even Data
0	X	0	0	0	1	1	Error	Features
0	X	0	0	1	0	2	Sector Count	Sector Count
0	X	0	0	1	1	3	Sector Number	Sector Number
0	X	0	1	0	0	4	Cylinder Low	Cylinder Low
0	X	0	1	0	1	5	Cylinder High	Cylinder High
0	X	0	1	1	0	6	Drive & Head	Drive & Head
0	X	0	1	1	1	7	Status	Command
0	X	1	0	0	0	8	Even Data (dup)	Even Data (dup)
0	X	1	0	0	1	9	Odd Data (dup)	Odd Data (dup)
0	X	1	1	0	1	0Dh	Error (dup)	Features (dup)
0	X	1	1	1	0	0Eh	Alt Status	Device Control
0	X	1	1	1	1	0Fh	Drive Address	Reserved
1	X	X	X	X	X		No access	No access

NOTE: In Table 2-9, 'dup' indicates a duplicate register.

Table 2-10 shows all possible combinations for accessing Task File Data registers. This table is only valid for Block I/O and Memory addressing modes.

Table 2-10 Task File ATA Register Accesses in Block I/O and Memory Addressing Modes

Offset	A0	*HCE1	*HCE2	Data Byte	Data Bus	Comments
0	0	0	1	Even Byte	D7-D0	
0	0	0	1	Odd Byte	D7-D0	FIFO Advance
8	0	0	1	Even Byte	D7-D0	
9	1	0	1	Odd Byte	D7-D0	FIFO Advance
9	1	0	1	Odd Byte	D7-D0	
8	0	0	1	Even Byte	D7-D0	FIFO Advance
9	1	0	1	Odd Byte	D7-D0	
9	1	0	1	Odd Byte	D7-D0	No FIFO Advance
0, 8, or 9	x	0	0	Even and Odd	D15-D0	Word access
0 or 8	0	0	1	Even Byte	D7-D0	
8 or 9	x	1	0	Odd byte	D15-D8	FIFO Advance
8 or 9	x	1	0	Odd Byte	D15-D8	
0 or 8	0	0	1	Even Byte	D7-D0	FIFO Advance
8 or 9	x	1	0	Odd Byte	D15-D8	
8 or 9	x	1	0	Odd Byte	D15-D8	No FIFO Advance
1	1	0	1	Error/Features	D7-D0	No data access
0 or 1	x	1	0	Error/Features	D15-D8	No data access

NOTE: The ECC transfer can be either 8 bits or 16 bits.

Table 2-11 shows the Task File Registers for host systems that can only perform 16-bit accesses. This table is only valid for Block I/O and Memory addressing modes.

Table 2-11 16-Bit Task File Register Accesses in Block I/O and Memory Addressing Modes

Offset	Data Bus 7-0	Data Bus 15-8
0h	Even Data Byte	Odd Data Byte
2h	Sector Count	Sector Number
4h	Cylinder Low	Cylinder High
6h	Drive & Head	Status/Command
8h	Even Data (dup)	Odd Data (dup)
0Ah	not used	not used
0Ch	not used	Error/Features (dup)
0Eh	Alt Status/Drv Ctrl	Drive Address

NOTE: In Table 2-11, 'dup' indicates a duplicate register.

Register 0 is accessed with *HCE1 low and *HCE2 low as a word register on SD15-SD0. This register may also be accessed by a pair of byte accesses to offset 0 with *HCE1 low and *HCE2 high. Note that the address space of this word register overlaps the address space of the Error and Feature byte registers, which are at offset 1. When Register 0 is accessed twice as a byte register with *HCE1 low, the first byte to be accessed is the even byte of the word and the second byte accessed is the odd byte.

A byte access to Register 0 with *HCE1 high and *HCE2 low accesses the Error register on a read and the Feature register on a write.

Registers at offset 8, 9 and 0Dh are non-overlapping duplicates of the data registers at offsets 0 and 1. Register 8 is equivalent to Register 0, while Register 9 accesses the odd byte. If the registers are byte accessed in the order 9 then 8 the data will be transferred odd byte then even byte. Repeated byte accesses to Register 8 or 0 will access consecutive data bytes (even then odd) from the data buffer. Repeated word accesses to Register 8, 9 or 0 will access consecutive words from the data buffer. Repeated byte accesses to Register 9 are not supported. However, repeated alternating byte accesses to Registers 8 then 9 will access consecutive (even then odd) bytes from the data buffer. Repeated alternating byte accesses to Registers 9 then 8 will access consecutive (odd then even) bytes from the data buffer. Byte accesses to Register 9 access only the odd byte of the data. (Refer to Table 2-10 for data access conditions)

In the Block I/O mode, the *INPACK signal is asserted whenever *HCE1 or *HCE2, *REG, and *IOR/ *IOW are all asserted.

2.3.6.3 Memory Mode Addressing

This mode is selected by default after Power-On Reset, a PCMCIA reset condition, or by the host selecting this mode via bit 0 of the Configuration Option register (reg. 200h). This mode is used to support memory-only hosts or PCMCIA 1.0 Sockets (without I/O support).

This memory mode is called Common Memory in the PCMCIA specification and is accessed similar to Attribute Memory except the *REG signal is not asserted. (See Table 2-7.)

This mode does not contain any real memory but is used to map the ATA Task File Registers into a memory block. The memory map is divided into two blocks. The first is a 16-register block that is common to the Block I/O Task File Registers. This block, as shown in Table 2-12, simulates the standard eight ATA Task File Registers (with some of the registers duplicated at other offsets).

Table 2-12 Memory Address Block 1

Memory Address	Task File Register
00h	Even/Odd Data
01h	Error/Feature
02h	Sector Count
03h	Sector Number
04h	Cylinder Low
05h	Cylinder High
06h	Drive & Head
07h	Status/Command
08h	Even/Odd Data (dup)
09h	Odd Data (dup)
0Dh	Error/Feature (dup)
0Eh	Alt Status/Device Control
0Fh	Drive Address

NOTE: In Table 2-12, 'dup' indicates a duplicate register.

The second memory block is used only for data transfers (see Table 2-13). It maps the Even and Odd Data Registers into the address space from 400h to 7FFh. It is intended to be used by hosts that can perform memory-to-memory block move operations (i.e., Intel x86 CPUs). The 1 KByte memory space (400h to 7FFh) allows both the source and destination addresses to increment during the transfer. The host starts each sector data transfer (512 bytes) at the beginning of the memory block (400h). The block is 1 KBytes long to support Read/Write Long commands, which require 512 bytes of data plus four or more bytes of ECC.

Table 2-13 Memory Address Block 2

Memory Address	Task File Register
400h	Even Data (dup)
401h	Odd Data (dup)
402h	Even Data (dup)
403h	Odd Data (dup)
404h-7FDh	...repeated...
7FEh	Even Data (dup)
7FFh	Odd Data (dup)

NOTE: The standard ATA protocol handshake (Busy/DRQ) must still be observed for each data block transferred. Also note that 'dup' indicates a duplicate register.

The following table shows how the Task File Registers are accessed in the Memory Addressing Mode. The host can place the memory block on any 2 KByte boundary in Common Memory space by using the *HCE1/ *HCE2 signals, as properly decoded by an external decoder. Otherwise, the host must decode A11 to A26.

Table 2-14 Task File Register Access in Memory Addressing Mode

*REG	A10	A9-A4	A3	A2	A1	A0	Offset	*HOE	*WE
1	0	X	0	0	0	0	0	Even Data	Even Data
1	0	X	0	0	0	1	1	Error	Features
1	0	X	0	0	1	0	2	Sector Count	Sector Count
1	0	X	0	0	1	1	3	Sector Number	Sector Number
1	0	X	0	1	0	0	4	Cylinder Low	Cylinder Low
1	0	X	0	1	0	1	5	Cylinder High	Cylinder High
1	0	X	0	1	1	0	6	Drive & Head	Drive & Head
1	0	X	0	1	1	1	7	Status	Command
1	0	X	1	0	0	0	8	Even Data (dup)	Even Data (dup)
1	0	X	1	0	0	1	9	Odd Data (dup)	Odd Data (dup)
1	0	X	1	1	0	1	0Dh	Dup. Error	Dup. Features
1	0	X	1	1	1	0	0Eh	Alt Status	Device Control
1	0	X	1	1	1	1	0Fh	Drive Address	Reserved
1	1	X	X	X	X	0	8	Even Data (dup)	Even Data (dup)
1	1	X	X	X	X	1	9	Odd Data (dup)	Odd Data (dup)

NOTE: In Table 2-14, 'dup' indicates a duplicate register.

Tables 2-10 and 2-11 in the previous section and their related discussion also apply to Memory Addressing Mode.

Accesses to even addresses between 400h and 7FFh access Register 8. Accesses to odd addresses between 400h and 7FFh access Register 9. Note that this entire data block accesses the Data Register FIFO, which reads/writes the disk data buffer sequentially. The host cannot randomly address the data block.

2.3.7 Attribute Memory

The Attribute Memory is used by the host to read information about the PCMCIA card and to change the card's configuration. The information is accessed using a special memory operation that asserts *REG and *HCE1 along with the *HOE or *HWE signals (see Table 2-7).

To be compatible with 8-bit hosts, the Card Information Structure (CIS) information is stored on even-byte boundaries (0, 2, 4, ...). The content of odd-byte locations is undefined.

Attribute Memory consists of two parts:

1) Card Information Structure (CIS)

2) Card Configuration Registers (CCR)

2.3.7.1 Card Information Structure (CIS)

The CIS consists of 256 bytes of RAM that is read by the host to determine the configuration of the PCMCIA card. It is a user-definable structure which contains configuration information in a metaformat called tuples. (Tuple definitions can be found in the PCMCIA specification.)

The following is some of the information found in the CIS tuples:

- Device type (I/O or Memory)
- Data access speed (cycle time)
- Power requirements (5V, 3.3V, etc.)
- Memory and/or I/O port addresses
- Configuration modes
- Information about the Card Configuration Registers (base address in Attribute Memory)

At power-on time, the microprocessor initializes the CIS with the proper tuple information. The CIS information is stored in the Buffer RAM starting at location 000h through location 0FFh. The host is allowed to read (but not write) the CIS once the AIC-8375 is not busy.

2.3.7.2 Card Configuration Registers (CCR)

The Card Configuration registers are used by the host to configure the PCMCIA card. The host first reads the CIS to determine the configurations supported by the card. The CIS also contains the number of CCRs supported by the card and the base address where the CCRs are located in Attribute Memory.

The host can then write to the Configuration Option register (reg. 200h) to change the card's configuration or to reset the card. The CCR can also be used to control the state of a general control output (HCORB4), which might be used for a purpose such as power control.

2.4 Interfacing to the Drive Electronics

The AIC-8375 can interface with a variety of Data Channel interfaces. The information that follows summarizes the hardware related topics.

2.4.1 NRZ Data Configuration

The data path can be configured as a 1-, 2-, or 8-bit interface. The NRZSEL[1:0] field (reg. 60h, R/W, bits 3:2) is used to establish the width of the disk data interface.

1-Bit NRZ Mode:

In single-bit NRZ mode, data is written and read via a single signal pin (NRZ[0]). The most significant bit is written or read first. External or internal byte sync detection may be used. If internal byte sync detection is used, a 1-byte or 2-byte fault tolerant byte sync method can be used.

2-Bit NRZ Mode:

In dual-bit NRZ mode, data is written and read via two signal pins (NRZ[1:0]). The most significant bit is written via the NRZ[1] signal pin. External or internal byte sync detection may be used. If internal byte sync detection is used, a 1-byte or 2-byte fault tolerant byte sync method can be utilized

8-bit NRZ Mode:

Eight-bit NRZ requires all of the NRZ[7:0] signal pins to transmit and receive the data. External byte sync detection must be used.

2.4.2 Control/Status Signals

RRCLK and NRZ:

To ensure proper operation during disk read and write operations, the relationship between RRCLK and the NRZ[7:0] data signals must be maintained as specified in the timing diagrams. For proper high speed operation, capacitive loading must be kept to a minimum and proper termination must be maintained.

A write clock (WCLK) has been provided to allow for more relaxed timing in single and dual NRZ modes.

Index and Sector:

The AIC-8375 recognizes Index and Sector. They are recognized by their leading edge qualified and followed by at least two BUFCLK periods at the asserted level. The sequencer samples for the existence of this internal version of Index or Sector on Index/Sector instructions.

The polarity of Sector is programmable via the Enable Active High Sector bit (reg. 63h, R/W, bit 6).

Refer to the *Timing Specifications* section of the AIC-8375 Data Sheet for complete timing specifications on Index and Sector.

Input/Output:

The INPUT/OUTPUT pin can be recognized by two different methods. The sequencer samples for the existence of this internal version of Input every byte time during Wait For Input instructions. Secondly, the sequencer may sample the raw unqualified Input signal.

The Input signal can also be used as a write fault status input (asserted for at least 2 BUFCLK periods to be valid) and will stop the sequencer immediately when the Input signal is asserted. This function is enabled by the Stop On Input Write Fault bit (reg. 61h, R/W,
bit 3).

Refer to the *Timing Specifications* section of the AIC-8375 Data Sheet for more timing specifications on the Input pin.

EOS:

The EOS input signal is used to indicate the end of the servo burst area.

The End Of Servo signal is always recognized by its leading edge qualified and followed by at least two BUFCLK periods at the asserted level. The sequencer samples for the existence of this internal version of End Of Servo every byte time during branch on EOS instructions. In this case branch on EOS will occur when using the Alternate Branch instruction '001'.

Refer to the *Timing Specifications* section of the AIC-8375 Data Sheet for more timing specifications on the EOS signal.

External Sync Byte Found Offset:

When using external byte sync detection, the device can accommodate an early SYNCFND signal in all NRZ modes. An early offset of 0-7 RRCLKs can be programmed depending on which NRZ mode is used. The External Sync Byte Found Offset [2:0] bits (reg. 61h, R/W, bits 2:0) are used for this function.

2.5 Read Reference and Buffer Clock

There are several clock relationships which must be adhered to in order to ensure proper operation of the AIC-8375. These important relationships are as follows:

- $BUFCLK_{MAX}$= 50 MHz
- $RRCLK_{MAX}$ (for single NRZ mode) = 50 MHz
- $RRCLK_{MAX}$ (for dual NRZ mode) = 50 MHz
- $RRCLK_{MAX}$ (for byte-wide NRZ mode) = 15 MHz
- BUFCLK ≥ 2 x NRZ Byte Clock frequency,

  where NRZ Byte Clock frequency = RRCLK for byte-wide NRZ,
  RRCLK/4 for dual NRZ,
  RRCLK/8 for single NRZ.
- If not using the Ready line, and the local microprocessor is executing 'word' read/write instructions, the BUFCLK frequency must be high enough so as not to cause mis-clocking of the microprocessor data bytes.
- BUFCLK ≥ [host *IOR or host *IOW frequency] x 4
- The reading and writing of synchronous registers, which are located in the host or buffer blocks, requires that BUFCLK be running.
- The reading and writing of synchronous registers, which are located in the disk block, requires that RRCLK be running.
- If RRCLK and BUFCLK are reduced to implement power saving algorithms, the time required between consecutive register read and write operations must be adhered to as specified in the data sheet.

2.6 Power-On Reset (*POR) Condition

When the *POR input pin is asserted, the AIC-8375 is initialized and all outputs are de-asserted. *POR must be asserted while the AIC-8375 is powering up and for eight BUFCLK cycles after power is on.

Assertion of *POR puts the AIC-8375 into a non-operational state with the host, disk, and buffer control blocks latched into a reset state. As part of firmware initialization, these blocks must be brought out of the reset state by writing a zero to the Disk Block Reset bit (reg. 50h, R/W, bit 2), the Buffer Block Reset bit (reg. 50h, R/W, bit 1), and the Host Block Reset bit (reg. 50h, R/W, bit 0).

Refer to *Register Reset Summary* section in the AIC-8375 Data Sheet for the default values of the register bits upon Power On Reset.

2.7 Configuration/Option Switch Interface

The designer can incorporate up to ten configuration/option switches in an application using the AIC-8375 without the need for external glue logic. This is accomplished by allowing the switches to reside on the buffer data bus (BD[9:0]). They are accessed by the local microprocessor as buffer RAM accesses with the buffer RAM output enable function turned off. Bits BD[15:10] (BA[6:1] if PCMCIA mode) are reserved for microprocessor configuration control.

The required implementation is shown in Figure 2-3. The resistance values are chosen so as not to interfere with any buffer RAM accesses. This necessitates the use of approximately 4K-6.8K Ohm resistors for the pulldown resistors. This resistance value is also used on BD[15:10] should these bits need pulldown resistors. Refer to Section 3 for detailed information on how the local microprocessor reads the configuration/ option switches.

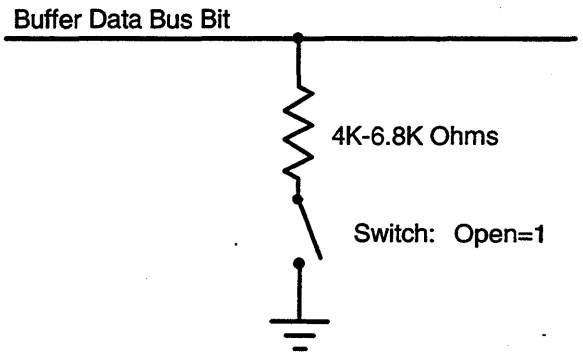

Figure 2-3 Switch Placement on the Buffer Data Bus

2.8 Miscellaneous Design Information

2.8.1 Power/Temperature Requirements

For proper operation, the AIC-8375 requires the following voltage:

- V_{dd} = +5.0 VDC +/- 10%

The following operating temperature should not be exceeded for proper operation:

- 0° C to 70° C

2.8.2 Unconnected Pins

Unused outputs may be left unconnected if desired.

Unused inputs should be connected through a resistor to either GND or VDD. This will prevent unwanted current consumption due to floating CMOS inputs as well as providing more ESD protection.

Unused bi-directional pins need to connected per the guidelines summarized above. The bi-directional pins will default, via application firmware, to either an input or an output.

SECTION 3
Functional Description

3.1 Microprocessor Interface and Control

The microprocessor interface and control logic contains many features which allows the AIC-8375 to function in many diverse applications utilizing different microprocessors and control structures. The following sections provide guidelines for the use of these features.

3.1.1 Microprocessor Interface Block Architecture

The Microprocessor Interface block performs the following functions:

- Configures itself at the end of the POR state for the particular microprocessor selected via the PMUX, PSEL, CSP, BSP, and RSEL signal pins.
- Manages the Ready signal.
- Provides the address latch function if desired in multiplexed mode.
- Manages the interrupts for the other blocks of the device.
- Manages the block reset and block powerdown functions.
- Provides miscellaneous control functions for all the blocks.
- Provides test control functions for all the blocks.

The block diagram of the Microprocessor Interface block is shown in Figure 3-1. This figure illustrates the major functions within the block.

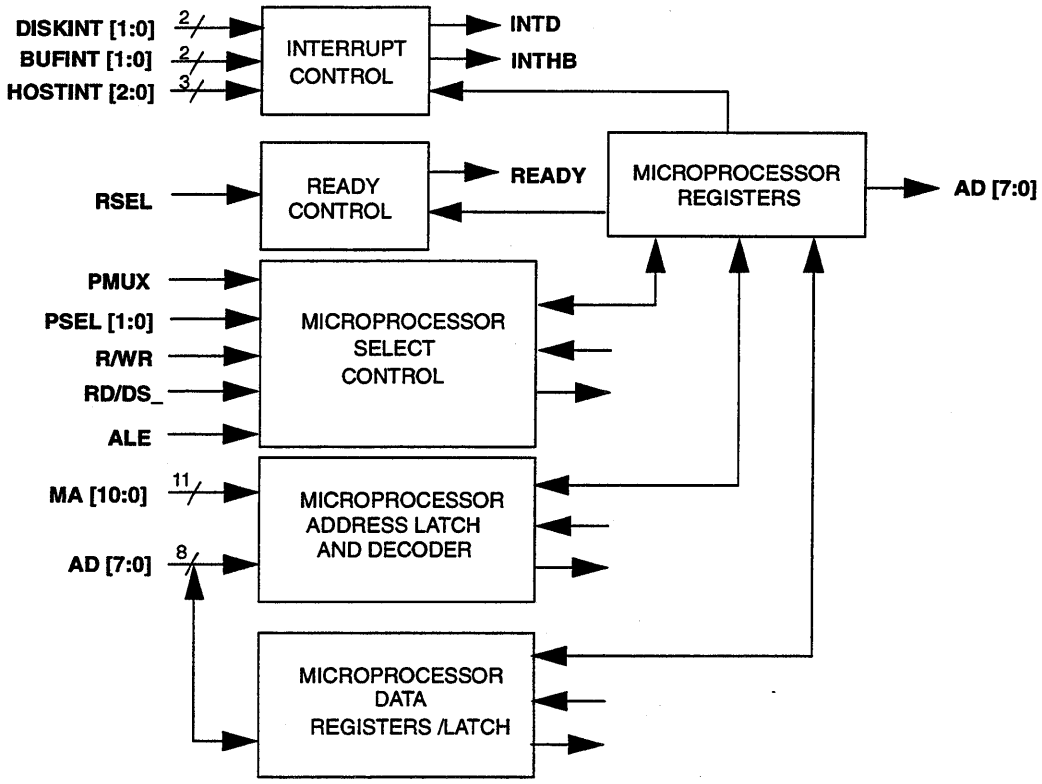

Figure 3-1 Microprocessor Interface Block

3.1.2 Interrupt Generation

The AIC-8375 can be programmed to generate interrupts to the local microprocessor on the INTHB/-INTHBD pin and the INTD pin. The user has control over when and how the interrupts are generated. This enables the user to implement flexible control algorithms. This is now discussed on a block by block basis.

3.1.2.1 Host Block Interrupts

Host interface related interrupts are generated on the INTHB/INTHBD pin. Table 3-1 summarizes the various registers and bits associated with the control and setting up of host block interrupts.

Table 3-1 Registers and Bits Used With Host Block Interrupts

Register or Bit	Description or Use
Host Interrupt 0 Status Register (reg. C8h, R)	The bits in this register indicate which condition caused the Host 0 interrupt.
Host Interrupt 0 Enable Register (reg. C9h, R/W)	The bits in this register enable the corresponding conditions specified by the Host Interrupt 0 Status register to generate an interrupt.
Host Interrupt 0 Status Clear Register (reg. C8h, W)	The bits in this register are used to clear the corresponding interrupt in the Host Interrupt 0 Status register.
Host Interrupt 1 Status Register (reg. CAh, R)	The bits in this register indicate which condition caused the Host 1 interrupt.
Host Interrupt 1 Enable Register (reg. CBh, R/W)	The bits in this register enable the corresponding conditions specified by the Host Interrupt 1 Status register to generate an interrupt.
Host Interrupt 1 Status Clear Register (reg. CAh, W)	The bits in this register are used to clear the corresponding interrupt in the Host Interrupt 1 Status register.
Host Interrupt 2 Status Register (reg. E4h, R)	The bits in this register indicate which condition caused the Host 2 interrupt.
Host Interrupt 2 Enable Register (reg. E5h, R/W)	The bits in this register enable the corresponding conditions specified by the Host Interrupt 2 Status register to generate an interrupt.
Host Interrupt 2 Status Clear Register (reg. E4h, W)	The bits in this register are used to clear the corresponding interrupt in the Host Interrupt 2 Status register.
Host Interrupt Active[2:0] (reg. 52h, R, bits 2:0)	These bits reflect the OR'd status of the two Host Interrupt Status registers. Bits 2:0 corresponds to Host Interrupt 2:0 Status registers respectively.
Enable Host Interrupt[2:0] (reg. 53h, R/W, bits 2:0)	These three bits enable any or all of the Host Interrupt Active bits (reg. 52h, R, bits 2:0) to generate an interrupt on the INTHB/INTHBD pin.
Enable Push-Pull Interrupt Outputs (reg. 51h, R/W, bit 1)	This bit determines if the INTHB/INTHBD pin and the INTD pin is open drain or push-pull.
Enable Active High Interrupt Outputs (reg. 51h, R/W, bit 0)	This bit determines the active polarity of the INTHB/INTHBD pin and the INTD pin.

3.1.2.2 Buffer Block Interrupts

Buffer interface related interrupts are generated on the INTHB/INTHBD pin. Table 3-2 summarizes the various registers and bits associated with the control and setting up of buffer block interrupts.

Table 3-2 Registers and Bits Used With Buffer Block Interrupts

Register or Bit	Description or Use
Buffer Interrupt 0 Status Register (reg. 103h, R)	The bits in this register indicate which condition caused the Buffer 0 interrupt.
Buffer Interrupt 0 Enable Register (reg. 104h, R/W)	The bits in this register enable the corresponding conditions specified by the Buffer Interrupt 0 Status register to generate an interrupt.
Buffer Interrupt 0 Status Clear Register (reg. 103h, W)	The bits in this register are used to clear the corresponding interrupt in the Buffer Interrupt 0 Status register.
Buffer Interrupt 1 Status Register (reg. 106h, R)	The bits in this register indicate which condition caused the Buffer 1 interrupt.
Buffer Interrupt 1 Enable Register (reg. 107h, R/W)	The bits in this register enable the corresponding conditions specified by the Buffer Interrupt 1 Status register to generate an interrupt.
Buffer Interrupt 1 Status Clear Register (reg. 106h, W)	The bits in this register are used to clear the corresponding interrupt in the Buffer Interrupt 1 Status register.
Buffer Interrupt Active[1:0] (reg. 52h, R, bits 4:3)	These bits reflect the OR'd status of the two Buffer Interrupt Status registers. Bit 1 corresponds to Buffer Interrupt 1 Status register and bit 0 corresponds to Buffer Interrupt 0 Status register.
Enable Buffer Interrupt[1:0] (reg. 53h, R/W, bits 4:3)	These two bits enable either or both of the Buffer Interrupt Active bits (reg. 52h, R, bits 4:3) to generate an interrupt on the INTHB/INTHBD pin.
Enable Push-Pull Interrupt Outputs (reg. 51h, R/W, bit 1)	This bit determines if the INTHB/INTHBD pin and the INTD pin is open drain or push-pull.
Enable Active High Interrupt Outputs (reg. 51h, R/W, bit 0)	This bit determines the active polarity of the INTHB/INTHBD pin and the INTD pin.

3.1.2.3 Disk Block Interrupts

Disk interface related interrupts are generated on the INTHBD pin or the INTD pin. Table 3-3 summarizes the various registers and bits associated with the control and setting up of disk block interrupts.

Table 3-3 Registers and Bits Used With Disk Block Interrupts

Register or Bit	Description or Use
Disk Interrupt 0 Status Register (reg. 5Eh, R)	The bits in this register indicate which condition caused the Disk 0 interrupt.
Disk Interrupt 0 Enable Register (reg. 5Fh, R/W)	The bits in this register enable the corresponding conditions specified by the Disk Interrupt 0 Status register to generate an interrupt.
Disk Interrupt 0 Status Clear Register (reg. 5Eh, W)	The bits in this register are used to clear the corresponding interrupt in the Disk Interrupt 0 Status register.
Disk Interrupt 1 Status Register (reg. 66h, R)	The bits in this register indicate which condition caused the Disk 1 interrupt.
Disk Interrupt 1 Enable Register (reg. 67h, R/W)	The bits in this register enable the corresponding conditions specified by the Disk Interrupt 1 Status register to generate an interrupt.
Disk Interrupt 1 Status Clear Register (reg. 66h, W)	The bits in this register are used to clear the corresponding interrupt in the Disk Interrupt 1 Status register.
Disk Interrupt Active[1:0] (reg. 52h, R, bits 6:5)	These bits reflect the OR'd status of the two Disk Interrupt Status registers. Bit 1 corresponds to Disk Interrupt 1 Status register and bit 0 corresponds to Disk Interrupt 0 Status register.
Enable Disk Interrupt[1:0] (reg. 53h, R/W, bits 6:5)	These two bits enable either or both of the Disk Interrupt Active bits (reg. 52h, R, bits 6:5) to generate an interrupt on the INTHB/INTHBD pin or the INTD pin.
Combine All Interrupts (reg. 51h, R/W, bit 2)	This bit allows the Disk Interrupt to be Or'd into the INTHB/INTHBD pin.
Enable Push-Pull Interrupt Outputs (reg. 51h, R/W, bit 1)	This bit determines if the INTHBD pin and the INTD pin is open drain or push-pull.
Enable Active High Interrupt Outputs (reg. 51h, R/W, bit 0)	This bit determines the active polarity of the INTHBD pin and the INTD pin.

3.1.3 Intel and Motorola Mode Memory Mapping

The Swap Register Addresses bit (reg. 51h, R/W, bit 5) selects the order in which the multiple-byte registers appear in memory. When this bit is cleared, all multiple-byte contiguous registers are located in the memory map low byte first. This is useful when interfacing to an Intel style microprocessor. If word type accesses are used, the bytes will get written or read in the correct order. When this bit is set, the bytes are located in the memory map high byte first. In this case, Motorola word accesses can be performed with the bytes being written or read in the correct order.

3.1.4 Microprocessor Access of the Internal Device Registers

There are several important points to be made about accessing the registers within the AIC-8375 device.

- Powering down the various blocks via the power down bits located in register 50h, R/W will not cause the registers to lose their states. They will power up in the same state they were in before power down occurred.
- Writing to any Type 1, 2, or 3 synchronous register or to any register in which the register R/W process will cause some action to occur, should not be attempted if BUFCLK (in the case of a host block or buffer block registers) or RRCLK (in the case of a disk block register) is not running.

Certain registers within the AIC-8375 are called synchronous registers because read or write operations to these registers take place with specific timing restrictions and requirements. Synchronous write registers are registers for which the internal write is initiated after the negation of the write strobe from the microprocessor.

There are three types of synchronous registers. These are defined below and listed in Table 3-4.

Type 1 Synchronous Registers:

Type 1 synchronous registers require the use of the Ready signal pin or the Busy For Microprocessor Access bit (reg. 52h, R, bit 7).

Type 2 Synchronous Registers:

Type 2 synchronous registers require that the write operation to the same register must not be repeated without a delay of at least four BUFCLK or RRCLK periods between write operations. However, a write to another register may occur during that time. If a host block or buffer block register is being accessed, the applicable clock is BUFCLK. If a disk block register is being accessed, the applicable clock is RRCLK.

Type 3 Synchronous Registers:

Type 3 synchronous registers require that RRCLK be present for any action to occur.

Table 3-4 Synchronous Registers

Write Registers			Read Registers
Type 1	Type 2	Type 3	Type 1
A2h-A6h, B0h,	5Eh, 65h, 66h,	60h, 6Eh, 6Fh,	Buffer RAM
ADh-AFh, 103h,	9Bh,	73h	
106h, 108h-10Bh,	C1h, C3h, C6h,		
10Dh, 12Eh, 12Fh,	C8h, CAh, CEh,		
133h, 137h,	E4h, 1F3h-1F7h		
Buffer RAM			

3.1.5 Microprocessor Access of the Buffer RAM

There are two methods of reading and writing the buffer RAM. They are called the direct method and the pseudo-direct method. The direct method utilizes the Ready line while the pseudo-direct method is based on polling a done bit to control the accesses. There are other control bits to deal with 8-bit or 16-bit wide buffers as well as high/low byte ordering.

The following registers and bits are potentially utilized during microprocessor accesses of the buffer RAM:

Table 3-5 Registers and Bits Used During Microprocessor Accesses Of Buffer RAM

Register/Bit Name	Register/Bit Number
Microprocessor Page 0 and Page 1 registers	Reg. 110h, 111h
Enable Pseudo Read From Buffer bit (RDPSEUDO)	Reg. 100h, R/W, bit 0
Enable Buffer Parity Checking bit (ENBPCHK)	Reg. 105h, R/W, bit 0
Busy For Microprocessor Access bit (BUSYMA)	Reg. 52h, R, bit 7
Microprocessor Port Check Error bit (MPARERR)	Reg. 106h, R, bit 3
Clear Micro. Check Error Interrupt bit (CLR_MCHKERR)	Reg. 106h, W, bit 3
Enable Micro. Check Error Interrupt bit (EN_MCHKERR)	Reg. 107h, R/W, bit 3
Swap Register Addresses bit (SWAPREG)	Reg. 51h, R/W, bit 5
Enable 16-Bit Wide Buffer (BEN16BIT)	Reg. 100h, R/W, bit 2
Disable MOE (DISMOE)	Reg. 102h, R/W, bit 1
Microprocessor Port Enable bit (MPEN)	Reg. 109h, R/W, bit 3
Enable 16-Bit Micro. Buffer Access bit (MP16BIT)	Reg. 102h, R/W, bit 2

3.1.5.1 Buffer RAM Read

Direct Method:

The preferred method is to utilize the buffer RAM as true scratchpad RAM. A direct read access of the RAM is performed with the use of the Ready line. This is required since the buffer may be in use by the disk, host, refresh, or correction logic at the time of the attempted microprocessor access. The buffer read operation incorporates the following steps:

- The device is properly configured for the appropriate SRAM or DRAM configuration using the Buffer Cycle Time Select bits and RAM Select bits in register 100h, R/W.
- The following bits must be initialized:

  Enable Pseudo Read From Buffer bit (reg. 100h, R/W, bit 0) = 0
  Microprocessor Port Enable bit (reg. 109h, R/W, bit 3) = 1
  Enable 16-Bit Micro. Buffer Access bit (reg. 102h, R/W, bit 2) = 0 or 1 as desired
  Enable 16-Bit Wide Buffer bit (reg. 100h, R/W, bit 2) = 0 or 1 as desired
  Disable MOE (reg. 102h, R/W, bit 1) = 0
- The Microprocessor Page registers (reg. 110h, 111h) are loaded with the base address of a 2 KByte page in which to do the read operation. Note that there is an overlap of one bit between the area addressed by the Microprocessor Page Register (MPAGE[21:10], regs. 110h/111h, R/W) and the memory address lines (MA[10:0]) providing the offset in the page. This allows the 2 KByte pages to be created on 1 KByte boundries.

- If it is desired to use buffer parity checking during microprocessor buffer reads, the following bits must be configured:

  Enable Buffer Parity Checking bit (reg. 105h, R/W, bit 0) = 1

- Further, if it is desired that a parity error generate an interrupt, the following bits must be initialized:

  Clear Micro. Check Error Interrupt bit (reg. 106h, W, bit 3) = 1
  Enable Micro. Check Error Interrupt bit (reg. 107h, R/W, bit 3) = 1

- The microprocessor can now access the desired location by accessing the AIC-8375 using the BS chip select pin. The 11-bit address specified in the current instruction will be used as the offset into the currently selected a 2 KByte page.

- Ready will be negated upon the assertion of the BS signal. On the assertion of the microprocessor RD signal, the internal execution of the buffer RAM read is initiated. When the internally initiated read is complete and the data is valid on the AD[7:0] bus, Ready is asserted and the microprocessor can complete the read access.

- The Swap Register Addresses bit (reg. 51h, R/W, bit 5) can be set if needed to swap the internal register addresses of the MSB and LSB bytes of the multi-byte registers. This would be done to allow proper byte ordering for microprocessors using word (rather than byte) accesses to the AIC-8375 device.

- The Enable 16-Bit Micro. Buffer Access bit (reg. 102h, R/W, bit 2) can be used to optimize the timing of the RAM read operation if a "word" wide read operation is being performed while the buffer is configured in 16-bit mode via the Enable 16-Bit Wide Buffer bit (reg. 100h, R/W, bit 2). When the microprocessor reads from a low byte address, both the low and high bytes are read from the buffer RAM via one read access. The high byte will be internally latched and the low byte will be returned to the microprocessor. When the microprocessor then reads the high byte address, the internally latched high byte is returned to the microprocessor and no buffer access is performed.

Pseudo-Direct Method:

The buffer RAM can also be read by microprocessors which can't use the Ready line. The procedure is as follows:

- The device is properly configured for the appropriate SRAM or DRAM configuration using the Buffer Cycle Time Select bits and RAM Select bits in reg. 100h, R/W.
- The following bits must be initialized:

  Enable Pseudo Read From Buffer bit (reg. 100h, R/W, bit 0) = 1
  Microprocessor Port Enable bit (reg. 109h, R/W, bit 3) = 1
  Enable 16-Bit Micro. Buffer Access bit (reg. 102h, R/W, bit 2) = 0 or 1 as appropriate
  Enable 16-Bit Wide Buffer bit (reg. 100h, R/W, bit 2) = 0 or 1
  Disable MOE (reg. 102h, R/W, bit 1) = 0
- The Microprocessor Page registers (reg. 110h, 111h) are loaded with the base address of a 2K-byte page in which to do the read operation. Note that there is an overlap of one bit between the area addressed by the Microprocessor Page Register (MPAGE[21:10], regs. 110h/111h, R/W) and the memory address lines (MA[10:0]) providing the offset in the page. This allows the 2 KByte pages to be created on 1 KByte boundries.
- If it is desired to use buffer parity checking during microprocessor buffer reads, the following bits must be configured:

  Enable Buffer Parity Checking bit (reg. 105h, R/W, bit 0) = 1.
- Further, if it is desired that a parity error generate an interrupt, the following bits must be initialized:

  Clear Micro. Check Error Interrupt bit (reg. 106h, W, bit 3) = 1
  Enable Micro. Check Error Interrupt bit (reg. 107h, R/W, bit 3) = 1.
- The microprocessor can now access the desired location by accessing the AIC-8375 using the BS chip select pin. The 11-bit address specified in the current instruction will be used as the offset into the currently selected a 2 KByte page.
- The microprocessor executes a read of the desired location. The data returned is not the correct contents of that location since the actual read has not really completed by this time. A holding register is actually read which contains data from the previous Buffer RAM read operation. This initiates an internal read operation of the Buffer RAM.
- The microprocessor polls the Busy For Microprocessor Access bit (reg. 52h, R, bit 7) until it becomes cleared. This event indicates that the internal read of the desired location has been completed and the data is latched.
- The microprocessor must now execute another read of a buffer RAM location (most likely the next sequential) in order to retrieve the previously requested data which is sitting in the latched holding register, and start another read at the new buffer address.
- The Swap Register Addresses bit (reg. 51h, R/W, bit 5) can be set if needed to swap the internal register addresses of the MSB and LSB bytes of the multi-byte registers. This would be done to allow proper byte ordering for microprocessors using word (rather than byte) accesses to the AIC-8375 device.

3.1.5.2 Buffer RAM Write

Direct Method:

The preferred method is to utilize the buffer RAM as true scratchpad RAM. A direct write access of the RAM is performed with the use of the Ready line. This is required since the buffer may be in use by the disk, host, refresh, or correction logic at the time of the attempted microprocessor access. The buffer write operation incorporates the following steps:

- The device is properly configured for the appropriate SRAM or DRAM configuration using the Buffer Cycle Time Select bits and RAM Select bits in reg. 100h, R/W.
- The following bits must be initialized:

  Enable Pseudo Read From Buffer bit (reg. 100h, R/W, bit 0) = 0
  Microprocessor Port Enable bit (reg. 109h, R/W, bit 3) = 1
  Enable 16-Bit Micro. Buffer Access bit (reg. 102h, R/W, bit 2) = 0 or 1 as desired
  Enable 16-Bit Wide Buffer bit (reg. 100h, R/W, bit 2) = 0 or 1 as desired
  Disable MOE (reg. 102h, R/W, bit 1) = 0
- The Microprocessor Page registers (reg. 110h, 111h) are loaded with the base address of a 2K-byte page in which to do the write operation. Note that there is an overlap of one bit between the area addressed by the Microprocessor Page Register (MPAGE[21:10], regs. 110h/111h, R/W) and the memory address lines (MA[10:0]) providing the offset in the page. This allows the 2 KByte pages to be created on 1 KByte boundries.
- If it is desired to use buffer parity, no control bit required since the parity bit available for use on each buffer write operation.
- If it is desired to create a false parity error for diagnostic purposes, an inverted parity bit can be generated by setting the Force Buffer Parity Error bit (reg. 105h, R/W, bit 1) = 1
- The microprocessor can now access the desired location by accessing the AIC-8375 using the BS chip select pin. The 11-bit address specified in the current instruction will be used as the offset into the currently selected a 2 KByte page.
- Ready will be negated upon the assertion of the BS signal. After the assertion and then the negation of the microprocessor WR signal, the internal execution of the buffer RAM write is initiated. The negation of WR is synchronized to BUFCLK and a write request is made to the buffer access prioritizer. After the microprocessor port wins access to the buffer, a write to the RAM is performed and the operation is complete. If another microprocessor read or write operation is attempted before the actual write to the RAM takes place, the READY line is negated as soon as the BS input is set. Wait states will be inserted as needed to keep READY negated while the write to the RAM is pending.
- The Swap Register Addresses bit (reg. 51h, R/W, bit 5) can be set if needed to swap the internal register addresses of the MSB and LSB bytes of the multi-byte registers. This would be done to allow proper byte ordering for microprocessors using word (rather than byte) accesses to the AIC-8375 device.
- The Enable 16-Bit Micro. Buffer Access bit (reg. 102h, R/W, bit 2) can be used to optimize the timing of the RAM write operation if a "word" wide write operation is being performed while the buffer is configured in 16-bit mode via the Enable 16-Bit Wide Buffer bit (reg. 100h, R/W, bit 2). In this case, the write to the RAM will take place with the *WE0 and *WE1 signals being asserted simultaneously to write both bytes to the RAM in parallel. The low byte (LS bit of address = 0) will be internally latched and not yet written to the buffer RAM. When the microprocessor writes to the high byte address (LS bit of address = 1), the high data byte is written to the buffer RAM along with the latched low byte via one write access.

Pseudo-Direct Method:

The buffer RAM can also be written by microprocessors that can't use the Ready line. The procedure is as follows:

- The device is properly configured for the appropriate SRAM or DRAM configuration using the Buffer Cycle Time Select bits and RAM Select bits in reg. 100h, R/W.
- The following bits must be initialized:

  Enable Pseudo Read From Buffer bit (reg. 100h, R/W, bit 0) = 1
  Microprocessor Port Enable bit (reg. 109h, R/W, bit 3) = 1
  Enable 16-Bit Micro. Buffer Access bit (reg. 102h, R/W, bit 2) = 0 or 1 as desired
  Enable 16-Bit Wide Buffer bit (reg. 100h, R/W, bit 2) = 0 or 1 as desired
  Disable MOE (reg. 102h, R/W, bit 1) = 0
- The Microprocessor Page registers (reg. 110h, 111h) are loaded with the base address of a 2K-byte page in which to do the write operation. Note that there is an overlap of one bit between the area addressed by the Microprocessor Page Register (MPAGE[21:10], regs. 110h/111h, R/W) and the memory address lines (MA[10:0]) providing the offset in the page. This allows the 2 KByte pages to be created on 1 KByte boundries.
- If it is desired to use buffer parity, no control bit required since the parity bit available for use on each buffer write operation.
- If it is desired to create a false parity error for diagnostic purposes, an inverted parity bit can be generated by setting the following bit;

  Force Buffer Parity Error bit (reg. 105h, R/W, bit 1) = 1
- The microprocessor can now write to the desired location by accessing the AIC-8375 using the BS chip select pin. The 11-bit address specified in the current instruction will be used as the offset into the currently selected a 2 KByte page.
- The microprocessor executes a byte write to the desired location. The actual write to the RAM is prioritized and may be delayed for a short time until a current other RAM access is finishing.
- The microprocessor write data is actually written into a holding register. At this point in time, an internal write operation to the buffer RAM is initiated.
- The microprocessor can now poll the Busy For Microprocessor Access bit (reg. 52h, R, bit 7) until it becomes cleared. This event indicates that the internal write of the desired location has been completed and a new buffer access can be initiated.
- The Swap Register Addresses bit (reg. 51h, R/W, bit 5) can be set if needed to swap the internal register addresses of the MSB and LSB bytes of the multi-byte registers. This would be done to allow proper byte ordering for microprocessors using word (rather than byte) accesses to the AIC-8375 device.
- The Enable 16-Bit Micro. Buffer Access bit (reg. 102h, R/W, bit 2) can be used to optimize the timing of the RAM write operation if a "word" wide write operation is being performed while the buffer is configured in 16-bit mode via the Enable 16-Bit Wide Buffer bit (reg. 100h, R/W, bit 2). In this case, the write to the RAM will take place with the *WE0 and *WE1 signals being asserted simultaneously to write both bytes to the RAM in parallel. The low byte (LS bit of address = 0) will be internally latched and not yet written to the buffer RAM. When the microprocessor writes to the high byte address (LS bit of address = 1), the high data byte is written to the buffer RAM along with the latched low byte via one write access.

3.1.6 Microprocessor Access of the Disk Sequencer RAM

The Disk Sequencer RAM address space is addressed using the CS chip select pin. Thus accessing the Disk Sequencer RAM is accomplished in the same manner as the internal register set.

The Disk Sequencer RAM should not be written to while the Disk Sequencer is running since there is a possibility of corrupting the sequencer words if it is attempted.

3.1.7 Microprocessor Access of the Configuration/Option Switches

Configuration switches may be physically located on the buffer data bus, BD[9:0]. BD[15:10] (BA[6:1] for PCMCIA mode) are reserved for microprocessor interface configuration at Power On Reset time. Certain procedures must be adhered to in order to successfully access the switches without affecting operation of the buffer RAM. These points, relative to DRAM and SRAM environments, are slightly different.

Using Switches With SRAM:

Generally, the switches need to be read by the microprocessor only during post Power-On Reset initialization. By adhering to this philosophy, the switches will be read at a time when the buffer holds no valuable data or is not being accessed by another resource. This will prevent the corruption of switch data or possibly buffer RAM data in the event of an incorrectly written switch access routine.

The following steps illustrate a method for accessing the switches:

1. The chip is enabled for SRAM mode by setting the RAM Select bits (reg. 100h, R/W, bits 5:3) to the value corresponding to current SRAM configuration.

2. Next, the *MOE/*RAS/*MCE1 pin is disabled by setting the Disable MOE bit (reg. 102h, R/W, bit 1). This insures that the output of the SRAM will not interfere with the switch data.

3. Since this is a "dummy" read, the Microprocessor Page registers (regs. 110h, 111h) do not have to be loaded with the base address of a 2K-byte page.

4. The Microprocessor can now access the switches by doing a simulated read of a location in the buffer. The access will take place according to the rules specified in Section 3.1.5 which describes details of the buffer RAM read operation.

5. Once the read of the switches has taken place, the above mentioned registers should be set to the states required for normal operation.

Using Switches With DRAM:

As with SRAM implementations, the switches need to be read by the microprocessor only during post Power-On Reset initialization.

The following steps illustrate a method for accessing the switches with DRAM present:

1. The chip is enabled for DRAM mode by setting the RAM Select bits (reg. 100h, R/W, bits 5:3) to the value corresponding to current DRAM configuration.

2. The Buffer Cycle Time Select bits (reg. 100h, R/W, bits 7:6) should be set to either '00' or '01' to correspond to 6T or 8T DRAM mode.

3. Refresh should be disabled.

4. Next, the *MOE/*RAS/*MCE1 pin is disabled by setting the Disable MOE bit (reg. 102h, R/W, bit 1). This insures that the output of the DRAM will not interfere with the switch data.

5. Since this is a "dummy" read, the Microprocessor Page registers (reg. 110h, 111h) do not have to be loaded with the base address of a 2K-byte page.

6. The Microprocessor can now access the switches by doing a simulated read of a location in the buffer. The access will take place according to the rules specified in Section 3.1.5 which describes details of the buffer RAM read operation.

7. Once the read of the switches has taken place, the above mentioned registers should be set to the states required for normal operation.

3.1.8 Microprocessor Block Initialization

Several registers must be configured when the microprocessor interface is initialized. The following steps illustrate the *general* order of initialization:

- First the CMODE register (reg. 51h, R/W) is set to accommodate the microprocessor interface's hardware implementation. Note that the value of bit 5 (SWAPREG) of this register depends on the type of microprocessor used while the other bits reflect the interface's implementation in hardware.
- When the drive is ready to accept commands the Enable Host/Buffer/Disk Interrupts in the CINTEN register (reg. 53h, R/W, bits 6:0) are set as required. Optionally at this time, the user may wish to read the CREV register (reg. 54h, R) to ensure that the correct revision of the chip has been installed.
- The Enable µP Port bit (reg. 109h, R/W, bit 3) is then set to enable µP/Buffer accesses.

The microprocessor block has now been initialized.

3.2 AT Host Interface and Control

The AIC-8375 incorporates many features for supporting documented features of the ATA and PCMCIA Interface Specifications and for supporting both automated and manual data transfers in various AT, ATA, and PCMCIA environments. The sections that follow provide information on applying these features.

3.2.1 AT Host Interface Block Architecture

The architecture of the AT Host Block is illustrated in Figure 3-2. This figure shows the general partitioning or major functions and data flow paths.

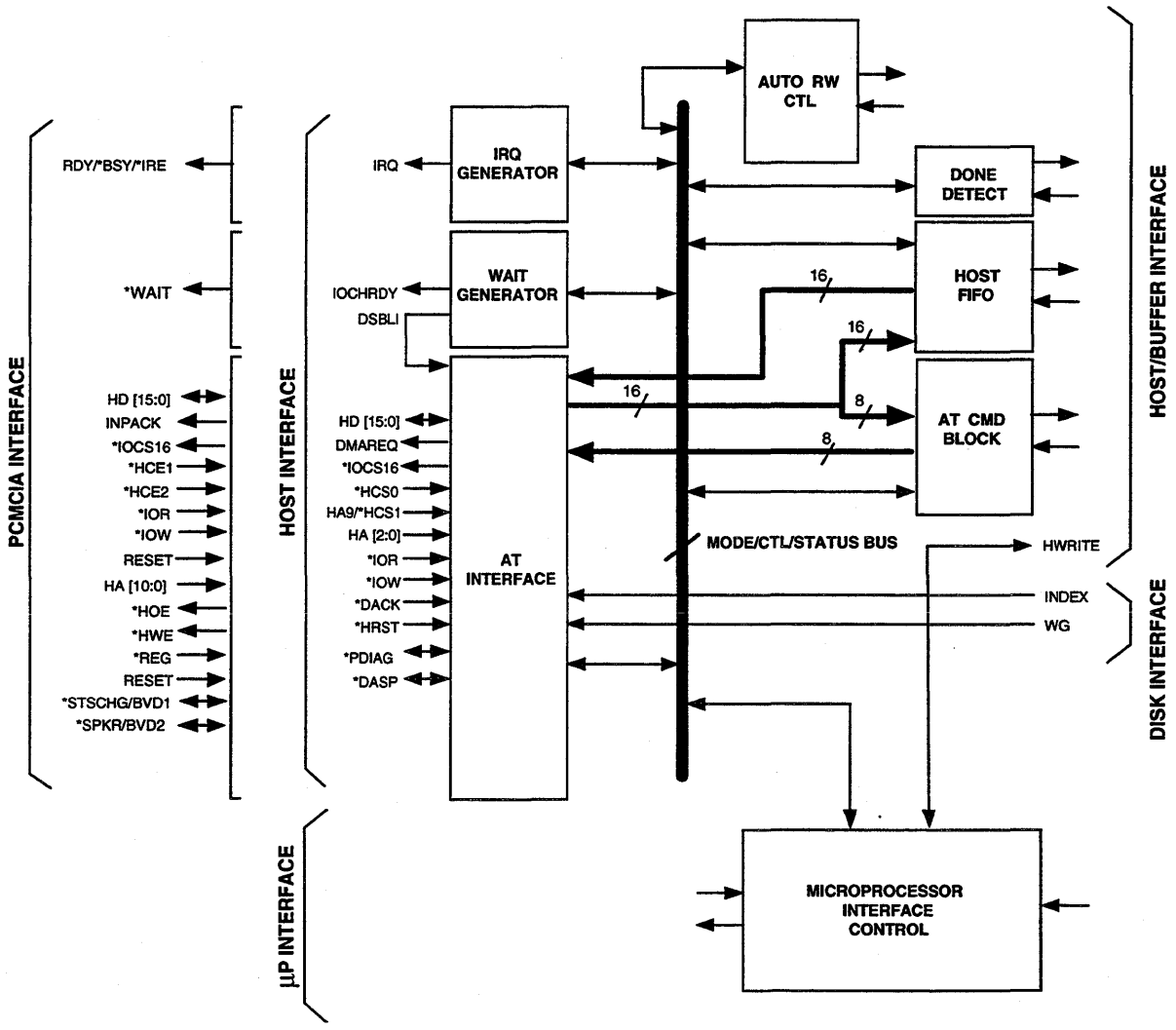

Figure 3-2 AT Host Block Architecture

3.2.2 AT Software Reset and Hardware Reset

The host has two paths for asserting a reset condition on the ATA bus and in the attached drives. The first is via the HRST pin. The second is via the Host Software Reset bit (reg. 3F6h, W, bit 2). If two physical drives are attached to the ATA interface, both will be affected by the two types of host resets.

In general, the host resets will affect only the host interface logic. If any transfer from the disk to host is taking place at the time of the host reset, the buffer to host link will be halted but the transfer from disk to buffer will not be affected.

Table 3-6 summarizes the various device register bits and pins which pertain directly to the two types of host reset.

Table 3-6 Registers and Bits Associated With Host Resets

Register, Bit, or Pin	Description or Use
Host Software Reset bit (reg. 3F6h, W, bit 2)	This bit is set by the host to cause a reset of the connected drives.
Host Software Reset Detected bit (reg. C8h, R, bit 3)	This bit is set when the host sets the Host Software Reset bit. This bit remains set for the duration of the reset. This bit is cleared by assertion of the HRST pin.
Enable Host Software Reset Detected Interrupt bit (reg. C9h, R/W, bit 3)	This bit enables the Host Software Reset Detected bit to generate an interrupt.
Clear Host Software Reset Detected bit (reg. C8h, W, bit 3)	This bit clears the Host Software Reset Detected Interrupt.
Host Reset (HRST) Pin	This is the hardware reset asserted by the host.
Host Reset Detected bit (reg. C8h, R, bit 4)	This bit is an interrupt status bit which is set if the Host Reset pin is asserted. This bit cannot be cleared while the HRST pin is asserted.
Enable Host Reset Detected Interrupt bit (reg. C9h, R/W, bit 4)	This bit enables the Host Reset Detected bit to generate an interrupt.
Clear Host Reset Detected bit (reg. C8h, W, bit 4)	This bit clears the Host Reset Detected Interrupt.

The affects of the two host resets on the device register bits can be found in the Section 4 (Register Reset Summary) of the AIC-8375 Data Sheet.

3.2.3 Master/Slave Drive Option

The AT environment and the ATA specification allow for two hard disk drives to be present. These two drives are normally referred to as the master (drive 0) and the slave (drive 1). This structure is supported via several bits in the AT Task File registers, several bits and registers within the AIC-8375, as well as several pins on the AIC-8375. These are summarized in Table 3-7.

Table 3-7 Registers, Bits, and Pins Used For the AT Master/Slave Environment

Register/Bit /Pin Name	Register/Bit /Pin Number
Master/Slave Enable bit (MSEN)	Reg. C0h, R/W, bit 1
Slave Mode bit (SLAVEMODE)	Reg. C0h, R/W, bit 2
Passed Diagnostics bit (*PDIAG)	Reg. BFh, W, bit 6
*PDIAG Device Pin	Pin 22
Drive Active/Slave Present bit (DASP)	Reg. BFh, W, bit 5
*DASP Device Pin	Pin 21
DRV bit	Reg. 1F6h, R/W, bit 4
Drive 0, Drive 1 Select bits	Reg. 3F7h, R, bits 1:0
Drive 0 Status Register	Reg. B4h, R/W
Drive 1 Status Register	Reg. B5h, R/W
Maximum Sector Number 0 Register	Reg. B1h, R/W
Maximum Sector Number 1 Register	Reg. B2h, R/W
Maximum Head Number Register	Reg. B3h, R/W
Microprocessor Drive/Head Register	Reg. A6h, R/W
Host Status Register	Reg. 1F7h, R

The master/slave option may be physically implemented with two separate controller/drives, or it may be logically implemented on a single controller/drive. In addition, various registers must be used in a certain way. Details on their use are described in the following sections.

3.2.3.1 Physical Implementation

During initialization:

Support for the physical implementation of the master/slave option starts during the execution of post Power-On Reset initialization firmware. If master/slave mode is implemented, the Master/Slave Enable bit (reg. C0h, R/W, bit 1) of each drive must be set. If master/slave mode is enabled, the firmware must be able to ascertain if the drive is the master (drive 0) or slave (drive 1) drive by reading a jumper or switch or utilizing the Cable Select function as specified in the ATA specification. If it is determined that the drive is a slave, the Slave Mode bit (reg. C0h, R/W, bit 2) must be set in the particular drive.

If the drive is configured as drive 1, it must assert the *DASP pin before a time, specified in the ATA specification, has elapsed. The *DASP pin is directly controlled in firmware by the DASP bit (reg. BFh, W, bit 5). Since *DASP is connected to both drives, it is driven by drive 1 to indicate to drive 0 that it is present. If drive 1 does not exist, *DASP can be asserted to drive an activity LED. After *DASP has been negated, either drive 0 or drive 1 may assert *DASP as an indication of drive activity.

The *PDIAG pin is also controlled by drive 1. It is asserted via drive 1 firmware to indicate to drive 0 that it has passed diagnostics. The firmware must drive this pin via the PDIAG bit (reg. BFh, W, bit 6) according to the protocol set forth in the ATA specification.

During normal operation:

In the master/slave environment, the Task File Registers of both drives will be loaded in parallel by the host. The host specifies which drive it intends to have execute the command by specifying that drive via the DRV bit (AT Host Drive/Head register, reg. 1F6h, R/W, bit 4) which is loaded as part of the Task File Register load. When set, only drive 1 will respond to the command. The command is written into the non-selected drives AT Host Command register (reg. 1F7h). However, the command is not executed and the various host interrupts are not generated. Since register 1F6h is loaded first, the DRV bit, along with the Slave Mode and Master/Slave Enable bits (reg. C0h, R/W, bits 2:1), are used to block or allow the execution of the command in the Command register. The drive select state is available for the host via the Drive 0 and Drive 1 Select bits (reg. 3F7h, R, bits 1:0). Various status and control bits are not altered in the non-selected drive even though the Task File registers of that drive are written. At the time the AT Command register (reg. 1F7h) is written, the DRV bit is used to either block or allow the various bits to be modified.

The host may read the readable Task File registers at various times. Whether it reads the Task File registers of drive 0 or drive 1 will depend on the setting of the Master/Slave Enable bit (reg. C0h, R/W, bit 1) and the DRV bit (reg. 1F6h, R/W, bit 4) in the two drives. This is summarized in the register description of the AT Host Status register (reg. 1F7h, R) in Section 5 (Register Descriptions) of the AIC-8375 Data Sheet.

In the master/slave physical implementation, the Drive 0 Status register (reg. B4h, R/W) is used for posting the status of both drive 0 and drive 1. In this mode, the Drive 1 Status register is not used.

Execution of the Host Diagnostic Command:

If the host sends an Execute Diagnostics Command to the drives, unlike all other commands, it will be executed by both drive 0 and drive 1. Upon successful completion of the diagnostics, drive 1 will use *PDIAG to inform drive 0 of this good status. This should occur according to the timing specifications set forth in the ATA specification.

3.2.3.2 Logical Implementation

During initialization:

In applications that logically implement the master/slave mode in a single controller/drive environment, much of the previous initialization process is not needed since two physical drives will not be communicating to each other.

Support for the logical implementation of the master/slave option starts during the execution of post Power-On Reset initialization firmware. If a logical implementation of master/slave mode is required, the Master/Slave Enable bit (reg. C0h, R/W, bit 1) must be cleared. With this bit cleared, all commands received are executed and all host interrupts in the Host Interrupt Status registers (reg. C8h, R and reg. CAh, R) are allowed to occur.

During normal operation:

All drive 0 and drive 1 commands will be executed by the single drive.

The host may read the readable Task File registers at various times. The single set of physical Task File Registers will be shared by drive 0 and drive 1 commands.

The Drive 0 Status register (reg. B4h, R/W) and Drive 1 Status register (reg. B5h, R/W) are used for their respective status's.

Execution of the Host Diagnostic Command:

The Execute Diagnostics Command will be executed and final status will be posted by the drive as in the physical master/slave arrangement.

3.2.4 Transfer Modes

The AIC-8375 incorporates hardware to automatically or manually perform three types of data transfers across the ATA interface. The various bits that control which transfer mode is used and how it used are shown in Table 3-8. The sections which follow summarize how to set up and use these transfer modes.

Table 3-8 Registers and Bits Associated With Transfer Modes

Register, Bit, or Pin	Description or Use
Host Data Bus Drive Enable For PIO Transfer (reg. C6h, R/W, bit 0)	This bit allows the ATA data lines to be driven earlier during execution of a PIO read command.
Host Data Bus Drive Enable For DMA Transfer (reg. C6h, R/W, bit 4)	This bit allows the ATA data lines to be driven earlier during execution of a DMA read command.
Enable Multi-Word DMA (reg. C6h, R/W, bit 1)	This bit enables DMA operation to occur in multi-word mode.
Enable IOCS16 Open Drain (reg. C5h, R/W, bit 7)	This bit Configures the IOCS16 pin for either open drain or push pull operation.
Enable 16-Bit Host Data Transfer (reg. C5h, R/W, bit 6)	This bit allows 16-bit transfers to occur.
PIO Transfer Mode Enable bit (reg. C5h, R/W, bit 3)	This bit enables PIO transfer mode.
Automatic Wait State Enable bit (reg. C5h, R/W, bit 2)	This bit enables using automatic wait states in PIO transfers.
Early IOCHRDY bit (reg. C5h, R/W, bit 0)	This bit allows IOCHRDY to be driven earlier during PIO mode transfers.

3.2.4.1 PIO Transfer Mode

PIO transfers to or from the host are accomplished with host software instructions which generate an *IOW or *IOR pulse for every word (or byte) transferred. Data is transferred either one sector at a time or in programmable blocks of sectors at a time. An established protocol using certain status bits, control bits, and interface signals must be adhered to before, in between, and after each sector (or block) is transferred. Signals involved with PIO transfers on the ATA interface are; HA[2:0], *IOW, *IOR, HD[15:0], IOCHRDY, IRQ, *IOCS16, *HCS0, and *HCS1.

The AIC-8375 is designed to transfer data across the ATA interface in PIO transfer mode at up to 22 MBytes/sec. PIO modes 0 through 4, as specified in the ATA specification, are supported.

The PIO Transfer Mode Enable bit (reg. C5h, R/W, bit 3) must be set to allow PIO type commands to operate.

8- and 16-bit transfers are allowed in PIO transfer mode. Per the ATA specification, sector data is transferred using 16 bits, while ECC bytes (for the Read Long and Write Long commands), command, control, and Status bytes are transferred using 8 bits. The Enable 16-Bit Host Data Transfer bit (reg. C5h, R/W, bit 6) determines the width of the sector data transfer. All other bytes being transferred, which support the command, ignore the state of this bit and the transfer takes place using 8 bits. The *IOCS16 signal is asserted during 16-bit transfers and negated during 8-bit transfers.

The Automatic Wait State Enable bit (reg. C5h, R/W, bit 2) is used to prevent system speed problems in situations where the host is attempting to read or write data when the host FIFO is not ready. While this bit is set, the IOCHRDY signal will be asserted when the current *IOW or *IOR operation to data port 1F0h (host FIFO) has a word (or byte) to be read or has space for a word (or byte). If either condition is not true, IOCHRDY will not be asserted. The host, in turn, will keep the *IOW or *IOR signal asserted until IOCHRDY is asserted. At this time the data will be read or written to the host FIFO.

The Early IOCHRDY bit (reg. C5h, R/W, bit 0) determines when IOCHRDY will be asserted/negated. When set, IOCHRDY will be driven when the address on the *HCS0, *HCS1, and HA[2:0] signal lines address the Host FIFO data port (1F0h). When cleared, IOCHRDY will driven based on the above signals plus *IOW or *IOR having been negated. When interfacing with faster hosts, this bit should be set.

The Host Data Bus Drive Enable For PIO Transfer bit (reg. C6h, R/W, bit 0) determines when the HD[15:0] pins will be driven during a PIO read operation. When set, the HD[15:0] pins will be driven based on the *HCS0, *HCS1, and HA[2:0] signal lines addressing the Host FIFO data port (1F0h). When this bit is cleared, they will driven based on the above signals plus *IOR having been asserted. This is another feature that can be used to help interfacing with faster hosts.

The typical interface protocol for PIO mode transfers as specified in the ATA specification is shown in Figure 3-3.

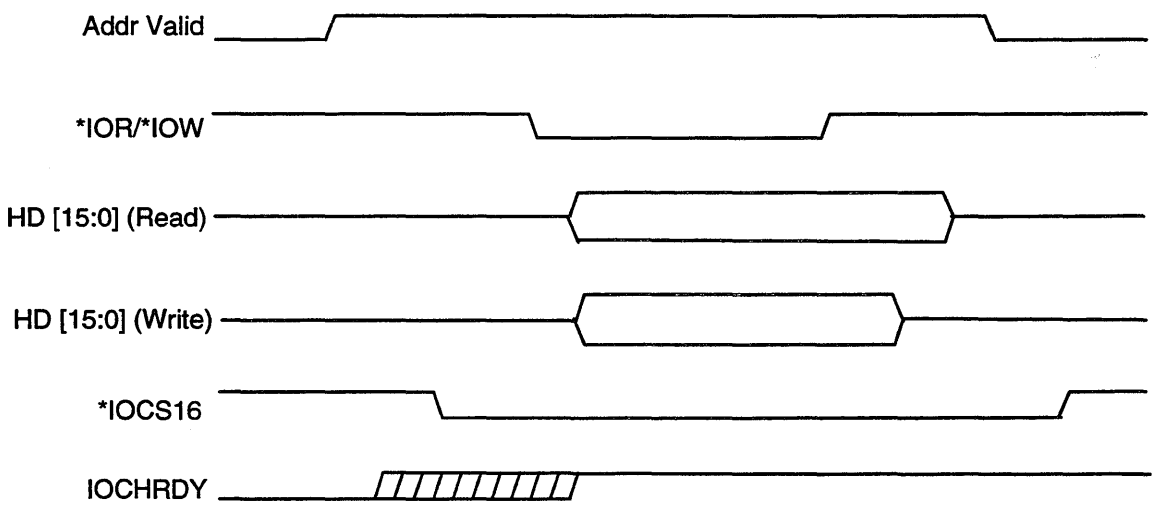

Figure 3-3 Typical PIO Mode Transfer

3.2.4.2 Single-Word DMA Transfer Mode

In this mode, data is transferred across the ATA interface one word at a time. *IOR and *IOW are still used to create a control edge for each word but typically are generated via a hardware DMA controller device in the host rather than in software. In addition, DMARQ and *DMACK are used to handshake the transfer. DMARQ is asserted by the AIC-8375 to inform the host that there is a word in the host FIFO to read or there is space in the host FIFO in which to write a word. *DMACK is a handshake signal sent from the host informing the AIC-8375 that the operation is done. Data can be transferred without regard to sector boundaries. An established protocol using certain status bits, control bits, and interface signals must be adhered to before and after the entire data is transferred. Signals involved with single-word DMA transfers on the ATA interface are; *IOW, *IOR, HD[15:0], DMARQ, and *DMACK.

The AIC-8375 is designed to transfer data across the ATA interface in Single-Word DMA mode at over 20 MBytes/sec. Single-Word DMA modes 0 through 2, as specified in the ATA specification, are supported

If the PIO Transfer Mode Enable bit (reg. C5h, R/W, bit 3) is set, and a DMA command is received, this bit will be ignored and a DMA transfer will take place. If this bit is reset, a DMA transfer will always occur. In addition, the Enable Multiword DMA bit (reg. C6h, R/W, bit 1), must be cleared to allow the AIC-8375 to perform the DMA command in Single-Word mode rather than Multi-Word mode.

Only 16-bit transfers are allowed in DMA transfer mode. Per the ATA specification, sector data is transferred using 16 bits, while the actual command is sent to the AIC-8375 using 8 bits.

Automatic wait state generation is inherently built into the Single-Word DMA transfer mode protocol since it uses the DMARQ/*DMACK handshake.

The Host Data Bus Drive Enable For DMA Transfer bit (reg. C6h, R/W, bit 4) determines when the HD[15:0] pins will be driven during a DMA read operation. When set, the HD[15:0] pins will be driven when the *DMACK signal line is driven addressing the Host FIFO data port (1F0h). When this bit is cleared, they will driven based on the above signals plus *IOR having been asserted. This is another feature that can be used to help interfacing with faster hosts.

The typical interface protocol for Single-Word DMA mode transfers as specified in the ATA specification is shown in Figure 3-4.

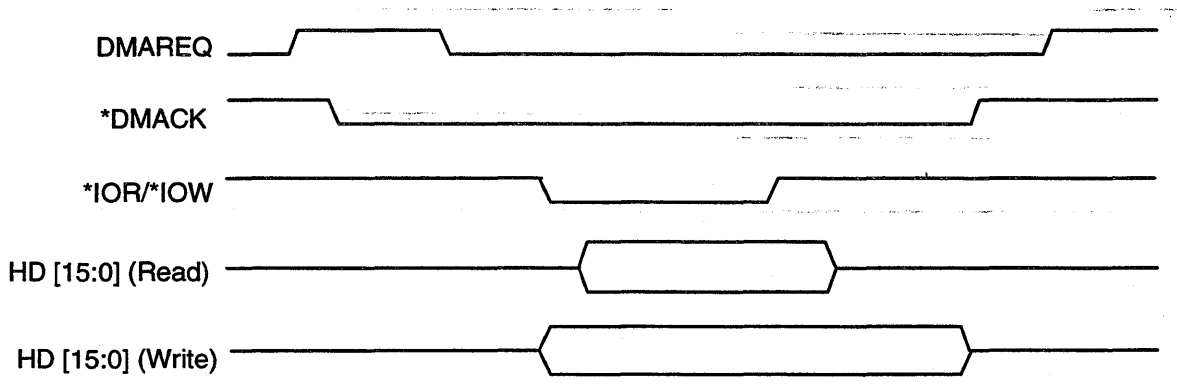

Figure 3-4 Typical Single-Word DMA Mode Transfer

3.2.4.3 Multi-Word DMA Transfer Mode

This mode differs from Single-Word DMA mode transfers in that the DMARQ and *DMACK are not used to handshake every word being transferred, but are used to throttle the transfer of any number of words. A single N-sector transfer, without any interruptions, would use only one pair of DMARQ/ *DMACK handshakes. As before, data is transferred across the ATA interface one word at a time.

The AIC-8375 is designed to transfer data across the ATA interface in Multi-Word DMA mode at over 20 MBytes/sec. Multi-Word DMA modes 0 through 2, as specified in the ATA specification, are supported

The Enable Multiword DMA bit (reg. C6h, R/W, bit 1), must be set to allow the AIC-8375 to perform the DMA command in this mode. The PIO Transfer Mode Enable bit (reg. C5h, R/W, bit 3) will be ignored whenever a DMA command is received.

Automatic wait state generation is inherently built into the DMA Multi-Word transfer mode protocol since it uses the DMARQ/*DMACK handshake.

The Host Data Bus Drive Enable For DMA Transfer bit (reg. C6h, R/W, bit 4) determines when the HD[15:0] pins will be driven during a DMA read operation. When set, the HD[15:0] pins will be driven when the *DMACK signal line is driven addressing the Host FIFO data port (1F0h). When this bit is cleared, they will driven based on the above signals plus *IOR having been asserted. This is another feature that can be used to help interfacing with faster hosts.

The typical interface protocol for DMA Multi-Word mode transfers as specified in the ATA specification is shown in Figure 3-5.

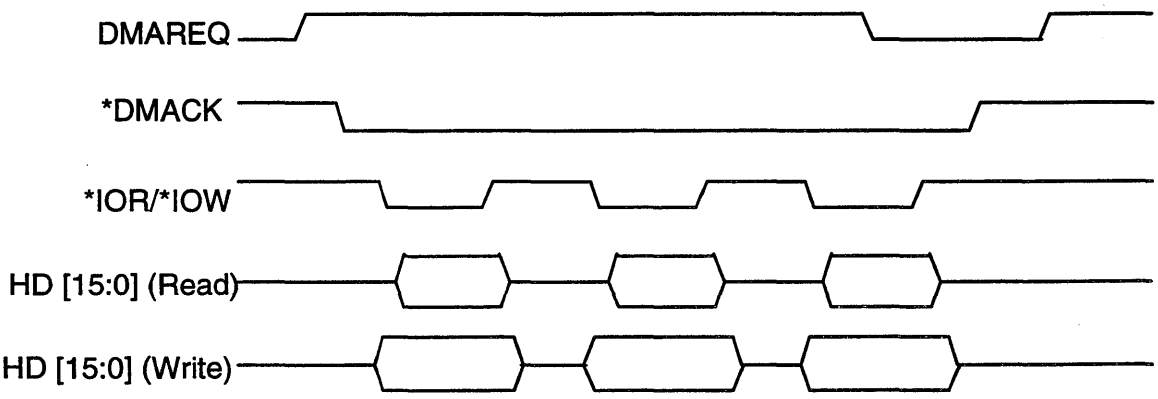

Figure 3-5 Typical Multi-Word DMA Mode Transfer

3.2.5 AT Host Automation

The AIC-8375 incorporates significant host automation which greatly reduces the burden on the microprocessor of performing ATA interface and data handling tasks. This automation can be applied to the first sector (or block) of a multi-sector transfer and/or to all subsequent sectors (or blocks) of the entire transfer. Automation can be turned off if so desired.

The sections which follow describe the terminology used, the extent of automation, details of its use, and some examples illustrating its use.

3.2.5.1 PIO Transfer Mode Automation

PIO read operation flow differs slightly from PIO write operation flow. These flows are shown in Figures 3-6 and 3-7. Following them are details describing what happens at each strategic point and what the user has control of.

PIO Write Transfer Mode Automation:

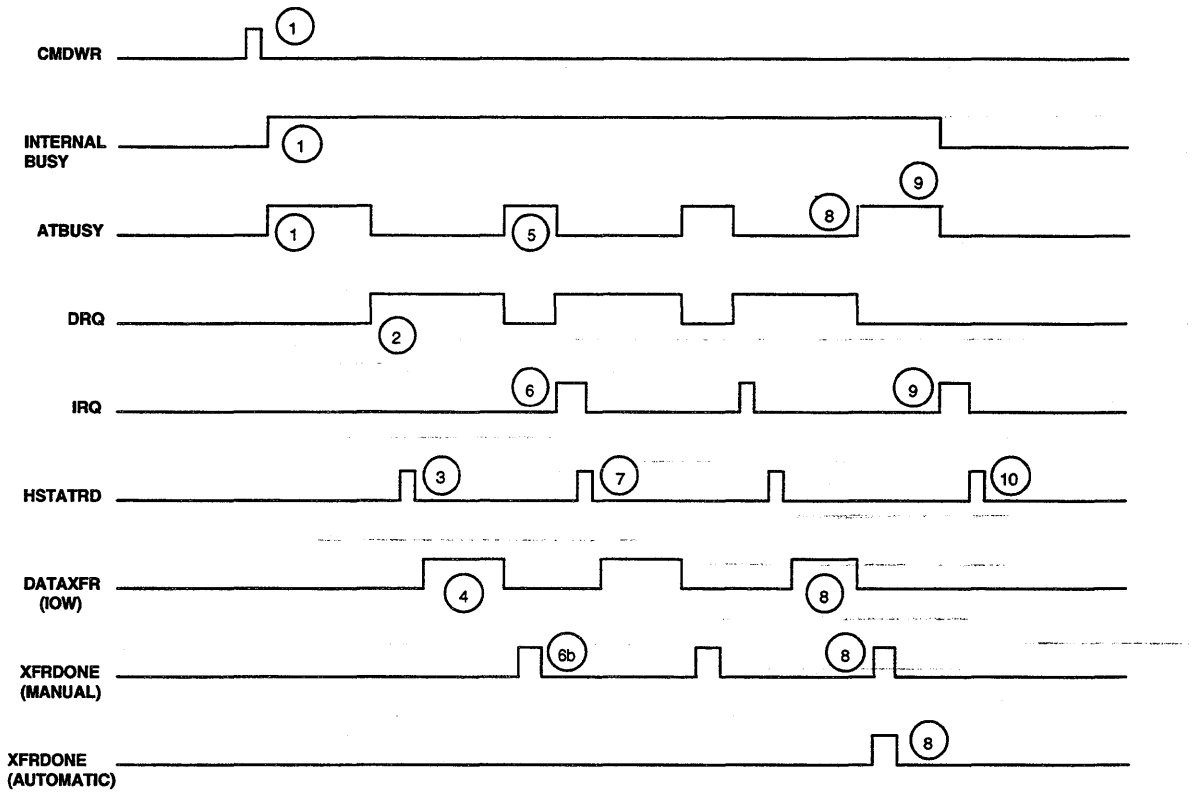

Figure 3-6 PIO Write Transfer

1. The host loads the Task File Registers, loading the AT Host Command register (reg. 1F7h, W) last with a PIO Write command. This sets the AT Busy bit (reg. 1F7h, R, bit 7) / (reg. CEh, R, bit 6) and the Internal Busy bit (reg. C1h, R, bit 7). The Selection Phase Detected bit (reg. C8h, R, bit 2) is also set along with the appropriate bits in the Host Status 0 register (reg. CCh, R) that indicate which command was received. Many other bits are cleared at this time if the DRV bit indicates that the command is for the indicated drive.

2. If the command has been enabled for Auto-Write execution via the Enable Auto-Write bit (reg. C2h, R/W, bit 6) or the Enable Auto Write Multiple bit (reg. C2h, R/W, bit 2), AT Busy is negated and DRQ (reg. 1F7h, R, bit 3) is asserted automatically after a very short time (about 200 ns). If Auto-Write execution is not enabled, the microprocessor must start the transfer using the Start AT Transfer bit (reg. C3h, R/W, bit 2).

3. The host might read the AT Status register (reg. 1F7h, R) to ascertain if DRQ is set, or it may begin the transfer without checking. If the register is read, the Host Status Read Detected bit (reg. CAh, R, bit 2) is set.

4. The host will now transfer either one sector or one block of sectors (if the command was a Write Multiple command). During the transfer of the data bytes, the *IOCS16 signal will be asserted to indicate a 16-bit transfer is taking place.

5. After one sector or one block of data is transferred, AT Busy will be asserted and DRQ will be negated automatically.

6. (a) If the Enable Automatic Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) is set and the Stop After One Block bit is cleared, the Task File Registers will be automatically updated at this point in time, and the Transfer Done bit (reg. C8h, R, bit 0) will not get set. It will take automation about 25 BUFLCK periods (approximately 600 ns.) minimum to stay in this state. The Busy Timer can be used to extend this time. The status of the host FIFO is checked and if it and the buffer can accept more data, IRQ and DRQ will be asserted and AT Busy will be negated automatically under the timing controls established by the IRQ Mode bits (reg. C4h, R/W, bits 1:0).

   (b) If the Enable Automatic Multi-Sector Transfer bit is not set or if the Stop After One Block bit is set, the automation will pause and the microprocessor will have to participate in the inter-gap protocol. The Transfer Done bit (reg. C8h, R, bit 0) will be set at this time. If the Disable Automatic Command Block Update bit (DISAUTOUPD, C0, R/W, bit 6) is not set, the Task File Registers will be automatically updated. If this bit is set, the microprocessor will have to manually update the Task File Registers. To restart the transfer, the microprocessor sets the Restart AT Transfer bit (reg. C3h, R/W, bit 1). To start the next transfer, the hardware will assert IRQ and DRQ and negate AT Busy when the buffer is ready.

7. The host will either be waiting for IRQ or polling for DRQ to begin the next transfer. Transfer will continue until after the next sector or block where steps 5-7 will be repeated as appropriate.

8. After the last sector (or block) has been transferred from the host, the Transfer Done bit (reg. C8h, R, bit 0) will be set, AT Busy is asserted and DRQ is negated automatically. The microprocessor may write the data to the drive and then post status or do write caching and immediately post ending status and then write the data to the drive. Posting status involves updating the Task File Registers for the last time so they point to the last sector and contain appropriate status.

9. AT Busy and Internal Busy must now be negated be setting the Reset Busy bit (reg. C1h, R/W, bit 6). IRQ is manually set by the microprocessor by setting the Host Interrupt Request bit (reg. C1h, R/W, bit 4).

10. The host typically will read the AT Status register (reg. 1F7h, R) which will in turn reset IRQ.

PIO Read Transfer Mode Automation:

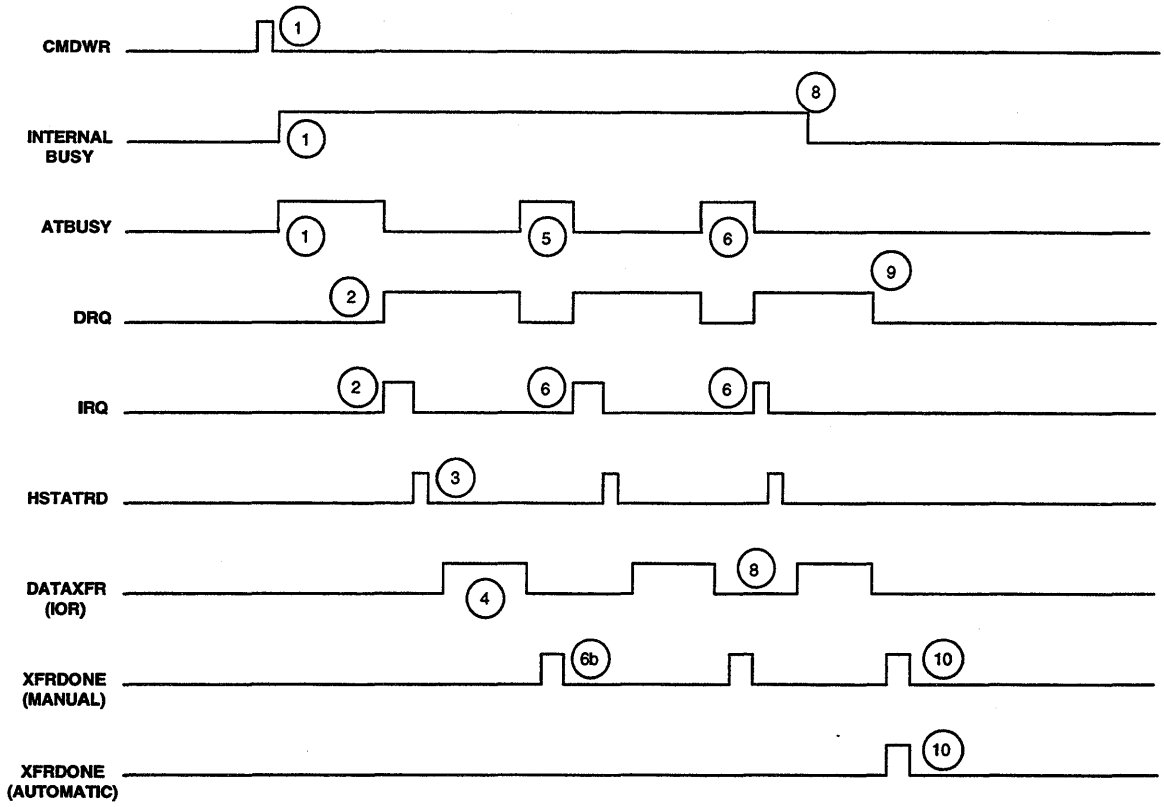

Figure 3-7 PIO Read Transfer

1. The host loads the Task File registers, loading the AT Host Command register (reg. 1F7h, W) last with a PIO Read command. This sets the AT Busy bit (reg. 1F7h, R, bit 7) / (reg. CEh, R, bit 6) and the Internal Busy bit (reg. C1h, R, bit 7). The Selection Phase Detected bit (reg. C8h, R, bit 2) is also set along with the appropriate bits in the Host Status 0 register (reg. CCh, R) that indicate which command was received. Many other bits are cleared at this time only if the DRV bit matches the selected drive.

2. When data is available in the buffer and host FIFO, AT Busy is negated while DRQ (reg. 1F7h, R, bit 3) and IRQ are asserted. This is accomplished via the microprocessor by setting the Start AT Transfer bit (reg. C3h, R/W, bit 2) or is automatically done if Auto-Reads are enabled and the requested data is in the buffer. If the read interlock feature is turned on and the command transfer length is for one sector or one block, the Host automation logic will assert XFRDONE (reg. C8h, R, bit 0) and stop at this point and not allow the transfer to continue.

3. The host either waits for IRQ or polls for DRQ in order to begin the transfer. If IRQ is used, the host typically reads the AT Status register (reg. 1F7h, R) to ascertain if DRQ is set. If the register is read, the Host Status Read Detected bit (reg. CAh, R, bit 2) is set and IRQ is negated.

4. The host will now read either one sector or one block of sectors (if the command was a Read Multiple command).

5. After data is transferred, AT Busy will be asserted and DRQ will be negated automatically.

6. (a) If the Enable Automatic Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) is set, the Task File registers will be automatically updated at this point in time, and the Transfer Done bit (reg. C8h, R, bit 0) will not get set. It will take automation about 120 BUFLCK periods (approximately 3 us.) minimum to stay in this state. The Busy Timer can be used to extend this time. The status of the host FIFO is checked and if it has more data, IRQ and DRQ will be asserted and AT Busy will be negated automatically. The IRQ timing is established by which IRQ mode (reg. C4h, R/W, bits 1:0) is active at the time.

   (b) If the Enable Automatic Multi-Sector Transfer bit is not set, the automation will stop and the microprocessor will have to participate in the inter-gap protocol. The Transfer Done bit (reg. C8h, R, bit 0) will be set at this time. The Task File registers will be automatically updated before the pause occurs. After checking to make sure that the host FIFO and buffer have data to send, the microprocessor sets the Restart AT Transfer bit (reg. C3h, R/W, bit 1). This will assert IRQ and DRQ and negate AT Busy.

7. The host will either be waiting for IRQ or polling for DRQ to begin the next transfer. Transfer will continue until after the next sector or block where steps 5-7 will be repeated as appropriate.

8. If the read interlock feature is turned on, the Host automation may stop command execution before the last sector or block is transferred to the Host. If it has stopped the transfer, the transfer is restarted as described in 6b above. (Refer to Section 3.2.7 for information on the read interlock feature.) After the host starts taking the last sector (or block), the Internal Busy bit (reg. C1h, R, bit 7) will be automatically negated by the hardware.

9. After the last sector (or block) has been transferred to the host, DRQ is automatically negated.

10. The Transfer Done bit (reg. C8h, R, bit 0) will be set after all the data has been transferred to the host.

Various Control and Status Bits Available for Use in PIO Transfer Mode:

The basic PIO transfer mode flow illustrated in Figures 3-6 and 3-7 can be altered with the use of various control and status bits. These are listed in Table 3-9.

Table 3-9 Control and Status Bits Available For Use In PIO Transfer Mode

Register and/or Bit(s)	Location
Host Timer	B9h, R
Host Busy Timer	BAh, W
Host Interrupt Time	BBh, R/W
Start AT Transfer bit	C3h, R/W, bit 2
Restart AT Transfer bit	C3h, R/W, bit 1
Stop After One Block bit	C1h, R/W, bit 2
Reset Host Int. Request bit	C1h, R/W, bit 3
Host Interrupt Request bit	C1h, R/W, bit 4
Enable Auto-Verify/Same bit	C2h, R/W, bit 0
Enable Auto-Read Multiple bit	C2h, R/W, bit 1
Enable Auto-Write Multiple bit	C2h, R/W, bit 2

Register and/or Bit(s)	Location
Enable Auto-Read bit	C2h, R/W, bit 5
Enable Auto-Write bit	C2h, R/W, bit 6
En. Auto. Multi-Sector XFR bit	C2h, R/W, bit 7
En. Auto Pause On Error bit	C3h, R/W, bit 4
IRQ Mode bits	C4h, R/W, bits 1:0
Read Interlock bit	C6h, R/W, bit 7
Enable Auto Read Interlock bit	C6h, R/W, bit 6
Auto-Read Interlock Status bit	C6h, R/W, bit 2
Auto-Read Cmd Received bit	C8h, R, bit 5
Pause Host Transfer bit	C3h, R/W, bit 0

3.2.5.2 Task File Update Control

As part of the specified ATA protocol, the ATA Task File registers are updated at specific times during command execution.

There are three methods of updating these registers. The manual method requires that the microprocessor write to each register at the appropriate time. The semi-automatic method allows the Task File registers to updated automatically at a specific time determined by the microprocessor. With this method, the automatic update will occur when the microprocessor sets the Update Host Count bit (reg. C6h, R/W, bit 5). In addition, the Disable Automatic Command Block Update bit (reg. C0h, R/W, bit 6) must be set. The third method will take place while the Disable Automatic Command Block Update bit is cleared and a command is being executed which has been enabled for automatic execution or manual execution. In this mode, no microprocessor intervention is required and the register updates will automatically occur at the appropriate times during the transfer. The hardware will appropriately update the Task File registers at the proper time dependent on the state of the Enable Multi-Sector Transfer bit (reg. C2h, R/W, bit 7).

Table 3-10 summarizes when the Task File registers are updated under various command and user programmable environments.

Table 3-10 Task File Register Update Conditions

Condition	When Task File Update Occurs
Automated Multi-Sector Read Sector(s) Command	Automatically before each sector is transferred to the host and before IRQ is set
Automated Multi-Sector Read Multiple Command	Automatically before each block of sectors is transferred to the host and before IRQ is set
Manual Read Sector(s) Command	Manually before each sector is transferred to the host and before IRQ is set
Manual Read Multiple Command	Manually before each block is transferred to the host and before IRQ is set
Read Long Command	Manually before the sector is transferred to the host
Automated Multi-Sector Read DMA Command	Automatically before each block of sectors is transferred to the host and before DMAREQ is set
Manual Read DMA Command	Manually before each sector is transferred to the host and before DMAREQ is set
Automated Multi-Sector Write Sector(s) Command	Automatically after each sector is transferred from the host and manually after last one transferred
Automated Multi-Sector Write Multiple Command	Automatically after each block of sectors is transferred from the host and manually after last one is transferred
Manual Write Sector(s) Command	Manually after each sector is transferred from the host but before setting IRQ
Manual Write Multiple Command	Manually after each block is transferred from the host but before setting IRQ
Write Long Command	Manually after the sector is transferred from the host but before setting IRQ
Automated Multi-Sector Write DMA Command	Automatically after each sector is transferred from the host and manually after last one is transferred
Manual Write DMA Command	Manually after each sector is transferred from the host and before DMAREQ is set
Inter-Gap time during Write (PIO or DMA) Cmds when XFER paused by Host Pause XFER bit	Automatically after the sector is transferred and before the pause at the sector or block boundary
Inter-Gap time during Read (PIO or DMA) Cmds when XFER paused by Host Pause XFER bit	Automatically before the sector is transferred to the host and before IRQ or DMAREQ is set

If manual updating of the ATA Task File registers is required, the Disable Automatic Command Block Update bit (reg. C0h, R/W, bit 6) must be set.

The ATA Task File registers will be updated correctly when operated in either LBA or CHS command modes.

The Start Sector Number register (reg. ADh, R/W), Start Cylinder Low register (reg. AEh, R/W), Start Cylinder High register (reg. AFh, R/W), and Start Drive/Head register (reg. B0h, R/W) will be updated whenever the ATA Task File registers are updated while the Disable Automatic Command Block Update bit (reg. C0h, R/W, bit 6) is cleared. If the Disable Start Register Update bit (reg. C0h, R/W, bit 4) is set, the Start registers are not updated during execution of a Read command.

The Task File registers can also be set up to increment during a Read Verify command. This is accomplished by having the Enable Read Verify Command Automatic Task File Update bit (reg. C0h, R/W, bit 5) set during the execution of the Read Verify command.

3.2.5.3 IRQ Mode / Interblock Gap Automation

The AIC-8375 incorporates the ability to program the characteristics of the Interblock protocol in PIO transfer mode. The programmability of the hardware allows the AT automation to work in many diverse ATA environments. This is essential since the ATA specification can be interpreted in different ways and as a result all designs do not behave the same way.

The hardware which supports the Interblock protocol is available for use during both read and write PIO operations.

IRQ Modes:

The IRQ Mode bits (reg. C4h, R/W, bits 1:0) are used to establish how IRQ will be generated at sector or block boundaries during both read and write PIO transfers. These are the main overriding bits which control how the flow will be performed in the interblock gap. These bits establish what are called mode 0, 1, and 2.

Mode 0 is the simplest of the modes. IRQ will be automatically generated when the Busy Timer counts down to zero. Actually, it is the Host Timer (reg. B9h, R) that counts down. It is loaded with the value that is in the Host Busy Time register (reg. BAh, W) right after the last byte of the sector has been transferred. When the Host Timer register counts down to zero, IRQ and DRQ will be asserted and AT Busy will be negated automatically provided the host FIFO is in the proper state. For a write operation this means that the host FIFO is not full, and during a read operation this means that the host FIFO is full. It is only required that the host FIFO be full to start the next sector transfer during the read operation. It could in fact be running near empty during any portion of the remaining sector transfer.

IRQ mode 0 provides the lowest performance of all three IRQ modes. This is generally due to the fact that different hosts read the AT Status register at different times after a sector is transferred. Using this mode requires that the Busy Time be long enough to cover the worst case situation. This in turn penalizes those hosts that read the AT Status register very quickly after a sector transfer is completed. Using too short of a value for the Busy Time could cause a slower host to erroneously clear the IRQ signal. This would cause the host to hang waiting for the IRQ that it just cleared and it would cause the drive to hang waiting for the host to respond to the same IRQ that was just set. There is a way of protecting against this using the IRQ Timer. This will be described later.

Mode 1 is probably the most used of all the modes. Using this mode, IRQ will be generated after the Busy Timer has timed out and after the host has read the AT Status register (reg. 1F7h, R) and when the host FIFO is ready. The Busy Timer value establishes the minimum time until IRQ could be generated and requiring that the host must have read the AT Status register provides a method of guaranteeing that IRQ will not get asserted too early for slower hosts. The time event from which the host reading the AT Status register is measured from is when IRQ is asserted for the current transfer just completed.

Mode 2 provides the highest level of performance since the next IRQ is generated only on the host having read the AT Status register and that the host FIFO is ready. The host reading of status must occur while IRQ is asserted for the current transfer just completed. This mode allows for the smallest of interblock gaps. However, some hosts, if not complying with the recommended ways of using the ATA interface, may not be able to use this mode.

IRQ Mode Support Features:

As previously mentioned, there is a problem if the host reads status for the prior sector after IRQ has been generated for the next sector; both the host and drive will hang waiting for each other to respond. The Host Timer can be used to safeguard against this situation. The Host Interrupt Time register (reg. BBh, W) is loaded with a value that will determine when another IRQ will be generated in the event this problem occurs. This value is loaded into the Host Timer register (reg. B9h, R) when the IRQ signal is asserted. At this time the counter will start counting down. If the host has not transferred a word to or from the host FIFO before this counter expires, an IRQ Timeout Interrupt (reg. C8h, R, bit 7) is generated.

3.2.5.4 DMA Transfer Mode Automation

DMA read operation flow differs slightly from DMA write operation flow. These flows are shown in Figures 3-8 and 3-9. Following them are details describing what happens at each strategic point and what the user has control of.

DMA Write Transfer Mode Automation:

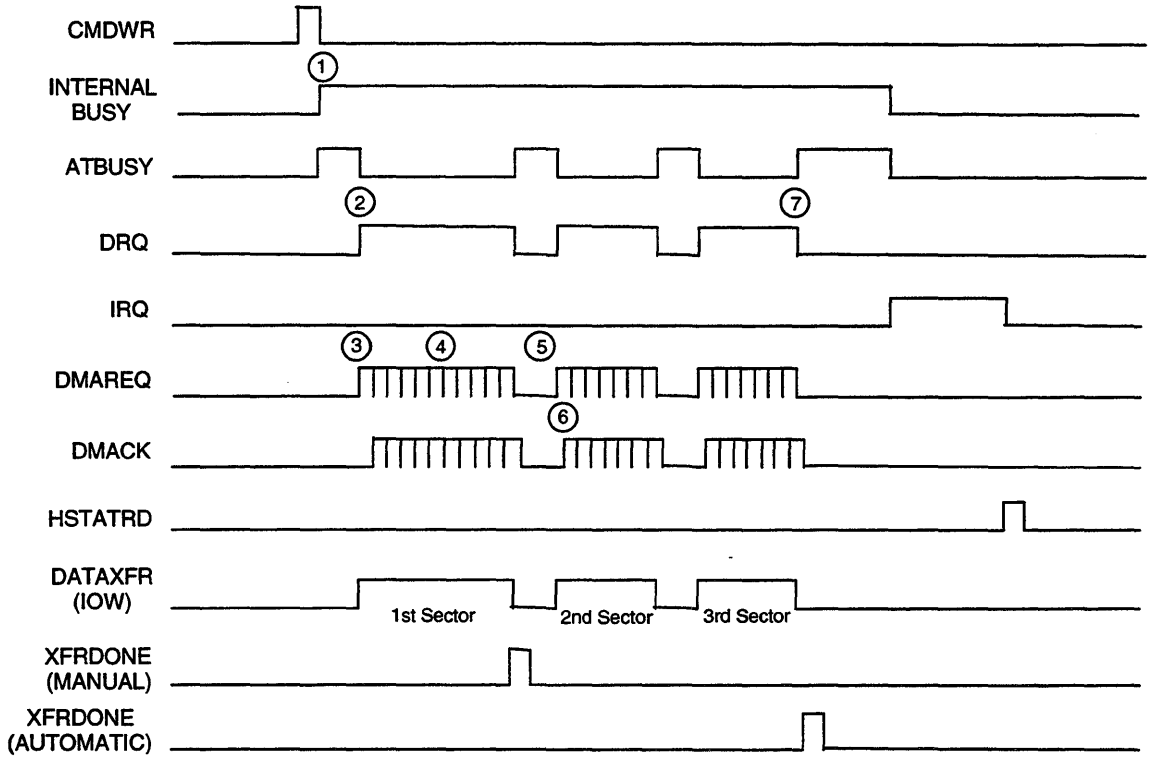

Figure 3-8 DMA Write Transfer

1. The host loads the Task File registers, loading the AT Host Command register (reg. 1F7h, W) last with a DMA write command. This sets the AT Busy bit (reg. 1F7h, R, bit 7) / (reg. CEh, R, bit 6) and the Internal Busy bit (reg. C1h, R, bit 7). The Selection Phase Detected bit (reg. C8h, R, bit 2) is also set along with the appropriate bits in the Host Status 0 register (reg. CCh, R) that indicate which command was received. Many other bits are cleared at this time. However, various status and control bits will only be affected in the drive selected by the DRV bit.

2. If the command has been enabled for Auto-Write execution via the Enable Auto-Write DMA bit (reg. C2h, R/W, bit 4), AT Busy is negated and DRQ (reg. 1F7h, R, bit 3) is asserted automatically after a very short time (about 200 ns). DMAREQ is asserted along with DRQ. If Auto-Write DMA execution is not enabled, the microprocessor must start the transfer using the Start AT Transfer bit (reg. C3h, R/W, bit 2).

3. The host might read the AT Status register (reg. 1F7h, R). If the register is read, the Host Status Read Detected bit (reg. CAh, R, bit 2) is set. In any event, the host will wait for the DMAREQ signal to be asserted which indicates that the host FIFO is ready to accept data.

4. The host will now start to transfer the data and will continue with possible gaps as required by the host FIFO status. One of two DMAREQ / *DMACK handshake protocols will be used.

5. The transfer will either pause or automatically continue to transfer data after one sector of data has been transferred from the host.

   (a) If the Enable Automatic Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) is set and the Stop After One Block bit (reg. C1h, R/W, bit 2) is cleared, the hardware will automatically prepare to accept the next sector. The Transfer Done bit (reg. C8h, R, bit 0) will not get set. It will take automation about 25 BUFLCK periods (approximately 600 ns.) minimum to stay in this state. DMAREQ and DRQ will be negated and AT Busy will be asserted while the Task File registers are updated.

   (b) If the Enable Automatic Multi-Sector Transfer bit is not set, or if it is set and the Stop After One Block bit (reg. C1h, R/W, bit 2) is set, the automation will stop and the microprocessor will have to eventually restart the transfer. The Transfer Done bit (reg. C8h, R, bit 0) will be set at this time. The Task File registers will be automatically updated. The microprocessor sets the Restart AT Transfer bit (reg. C3h, R/W, bit 1) to start transfer and the hardware waits until the buffer is ready before asserting DRQ and DMARQ and negating AT Busy.

6. The assertion of DMAREQ will allow the host to start sending more data. From this point on, the transfer will be checked on sector boundaries. The transfer is managed via the DMAREQ / DMACK handshake protocol on any word transfer as dictated by the status of the host FIFO and allowed to continue only if there is at least one sector of data in the buffer.

7. After all the required data is transferred, AT Busy will be set, the Transfer Done bit (reg. C8h, R, bit 0) will be set, and DRQ will be negated automatically. The microprocessor may write the data to the drive and then post status or do write caching and immediately post ending status and then write the data to the drive. Posting status involves updating the Task File registers for the last time so they point to the last sector and contain appropriate status, clearing AT Busy, and asserting IRQ.

DMA Read Transfer Mode Automation:

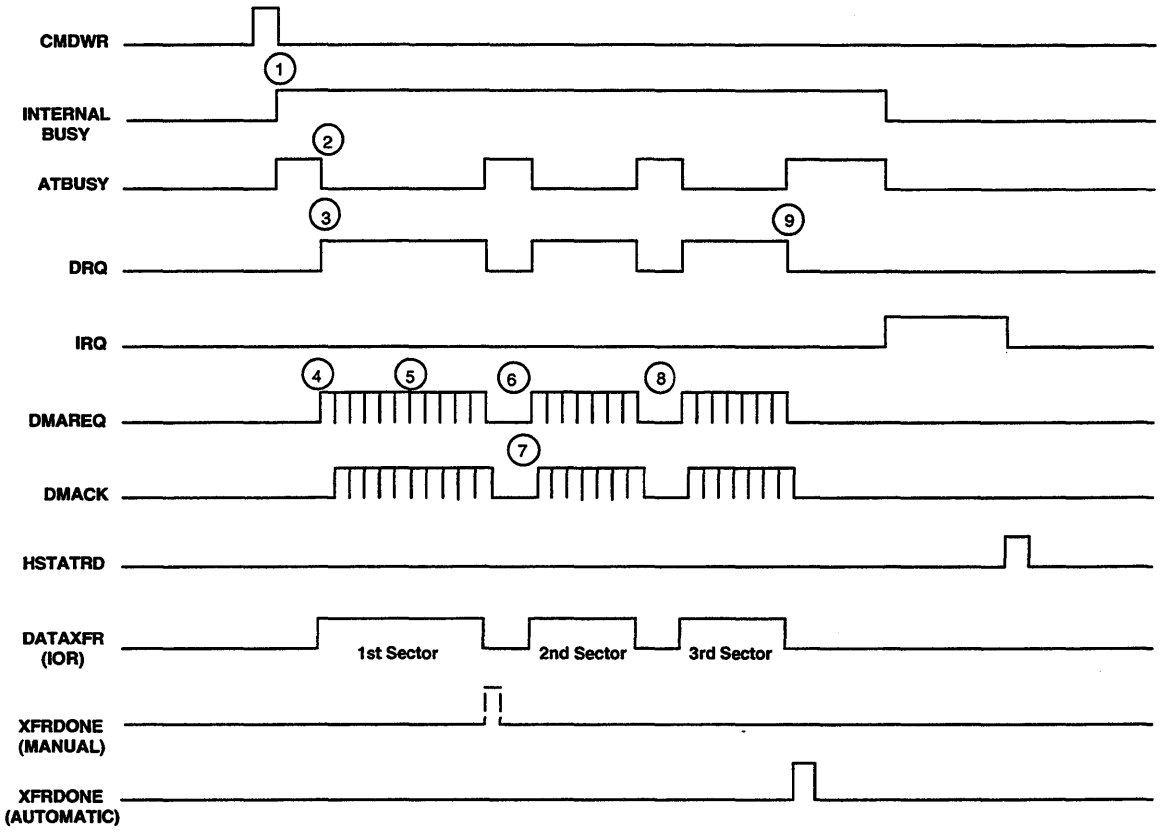

Figure 3-9 DMA Read Transfer

1. The host loads the Task File registers, loading the AT Host Command register (reg. 1F7h, W) last with a DMA read command. This sets the AT Busy bit (reg. 1F7h, R, bit 7) / (reg. CEh, R, bit 6) and the Internal Busy bit (reg. C1h, R, bit 7). The Selection Phase Detected bit (reg. C8h, R, bit 2) is also set along with the appropriate bits in the Host Status 0 register (reg. CCh, R) that indicate which command was received. Many other bits are cleared at this time. However, various status and control bits will only be affected in the drive selected by the DRV bit.

2. The Task File is updated if the DISAUTOUPD bit (reg. C0h, R/W, bit 6) is not set prior to ATBUSY being negated.

3. When data is available in the buffer and host FIFO, AT Busy is negated while DRQ (reg. 1F7h, R, bit 3) is asserted. This is accomplished via the microprocessor by setting the Start AT Transfer bit (reg. C3h, R/W, bit 2) or is automatically done if Auto-Reads are enabled and the requested data is in the buffer. If the read interlock feature is on and the command transfer length is for one sector, the transfer will not be automatically started in the case of the Auto-Read.

4. The host waits for the DMAREQ signal to begin the transfer using DMACK.

5. During the read, the DMARQ / *DMACK handshake will be performed per the appropriate DMA mode. There may be gaps in the data as required by the status of the host FIFO.

6. The transfer will either pause or automatically continue to transfer data after one sector of data has been transferred to the host.

   (a) If the Enable Automatic Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) is set, the hardware will automatically prepare to send the next sector. The Transfer Done bit (reg. C8h, R, bit 0) will not get set. It will take automation about 3 µsec minimum to stay in this state. DMAREQ and DRQ will be negated and AT Busy will be asserted while the Task File registers are updated.

   (b) If the Enable Automatic Multi-Sector Transfer bit is not set, the automation will stop, the Transfer Done bit (reg. C8h, R, bit 0) will be set, and the microprocessor will have to eventually restart the transfer. The Task File registers will automatically be updated before the next sector is started. The microprocessor sets the Restart AT Transfer bit (reg. C3h, R/W, bit 1) and will assert DRQ and DMAREQ, and negate AT Busy when the buffer hardware is ready.

7. The assertion of DMAREQ will allow the host to start reading more data. From this point on, the transfer will be checked on sector boundaries. The transfer is managed via the DMAREQ / DMACK handshake protocol on any word transfer as dictated by the status of the host FIFO and allowed to continue only if there is at least one sector of data in the buffer.

8. If the read interlock feature is turned on, the Host automation will pause command execution before the last sector is transferred. The transfer can be restarted as discussed in item 6b above.

9. After data is transferred, AT Busy will be set and DRQ will be negated automatically. The Transfer Done bit (reg. C8h, R, bit 0) will be set at this time. The Internal Busy bit (reg. C1h, R, bit 7) and the AT Busy bit must be cleared by the microprocessor setting the Reset Busy bit (reg. C1h, R/W, bit 6). The microprocessor must conclude the command by asserting IRQ.

Various Control and Status Bits Available for Use in DMA Transfer Mode:

The same bits that affect PIO transfer flow affect DMA transfer flow. These are listed in Table 3-9. The basic DMA transfer mode flow illustrated in Figures 3-8 and 3-9 can be altered with the use of various control and status bits. These are listed in Table 3-11.

Table 3-11 Additional Control and Status Bits For Use In DMA Transfer Mode

Register and/or Bit(s)	Location
Start AT Transfer bit	C3h, R/W, bit 2
Restart AT Transfer bit	C3h, R/W, bit 1
Stop After One Block bit	C1h, R/W, bit 2
Reset Host Int. Request bit	C1h, R/W, bit 3
Auto-Read Cmd Received bit	C8h, R, bit 5
Host Interrupt Request bit	C1h, R/W, bit 4
Enable Auto-Read DMA bit	C2h, R/W, bit 3

Register and/or Bit(s)	Location
Enable Auto-Write DMA bit	C2h, R/W, bit 4
Pause Host Transfer bit	C3h, R/W, bit 0
En. Auto. Multi-Sector XFR bit	C2h, R/W, bit 7
Enable Auto Pause On Error bit	C3h, R/W, bit 4
Read Interlock bit	C6h, R/W, bit 7
Auto-Read Interlock Status bit	C6h, R/W, bit 2
Enable Auto Read Interlock bit	C6h, R/W, bit 6

3.2.5.5 Auto-Write Command Execution

"Auto-Write" are the words used to describe a particular type of automation which can be performed with the various ATA write commands. If a write command is enabled for Auto-Write execution, then upon receipt of that command, the hardware will automatically assert and negate the ATA interface signals before and during the transfer of the first, and only the first, sector (or block) of data. Auto-Write execution can be applied to both PIO or DMA type commands.

It is important to realize that Auto-Write command execution automates only the first sector (or block) transfer of a multi-sector write command or the only sector (or block) transfer of a single sector (or block) write command. If Automatic Multi-Sector Write Transfers are enabled by the conclusion of the first sector (or block) transfer, the transfer of all sectors following the first sector (or block) will be automated. If not enabled, the microprocessor must intervene after the first sector (or block) and either facilitate a manual transfer of the remaining sectors or enable automatic multi-sector transfers. Refer to the figures in Sections 3.2.5.1 through 3.2.5.4 for details on signal protocol.

The data received from the host during the first automatic transfer will go into the Auto-Write Segment that has been configured via the Host Segment Control register (reg. 133h, R/W) and the associated Beginning and End Of Segment Pointers (A or B). The Host Pointer (reg. 130h - 132h, R/W) will be used as the address into the buffer. Upon receipt of a command that has been enabled for Auto-Write execution, the Host Pointer will be loaded with either the Write Cache Pointer (reg. 148h-14Ah), the associated Beginning Of Segment Pointer, or not modified at all.

The various ATA Write commands must be enabled for Auto-Write execution before they will be executed in that manner. This is done by setting the appropriate bits in Host Control 2 register (reg. C2h, R/W). Individual bits have been provided to allow some commands to be executed in Auto-Write mode while others will not. This allows the user to tailor execution of commands to the particular application.

If the Enable Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) is set, the command which has been enabled for Auto-Write execution will not stop after the first sector (or block) but will continue to automatically transfer the remaining sectors as allowed by buffer conditions. However, under these same conditions, if the Stop After One Block bit (reg. C1h, R/W, bit 2) is set, the transfer will stop after the first sector (or block) is set. Thus, the user has even more flexibility in controlling the flow of the transfer.

3.2.5.6 Auto-Read Command Execution

"Auto-Read" are the words used to describe a particular type of automation which can be performed with the various ATA read commands. If a PIO or DMA read command is enabled for Auto-Read execution, then upon receipt of that command, the hardware will automatically compare the disk address in the Task File registers against the address in the Start registers (reg. ADh - B0h, R/W). If they match, the hardware will automatically begin the transfer of data out of the buffer to the host and will assert and negate the ATA interface signals before and during the transfer of the first, and only the first, sector (or block) of data. The Auto-Read operation implies a read cache operation.

When enabled via the appropriate Auto-Read enable bit in the HCTL2 register (reg. C2h, R/W), the start of execution will begin automatically if the Cylinder High register (reg. 1F5h, R/W), the Cylinder Low register (reg. 1F4h, R/W), the Sector Number register (reg. 1F3h, R/W), and the Drive/Head register (reg. 1F6h, R/W) in the received command match the Start registers (reg. ADh, AEh, AFh, and B0h). If the Enable LBA Mode bit (reg. C3h, R/W, bit 3) is set, the Start registers must be in LBA format and the entire Drive/Head register is compared to the Start Drive/Head register (reg. B0h, R/W). If the Enable LBA Mode bit (reg. C3h, R/W, bit 3) is cleared, the Start registers must be in CHS format and only bits 4:0 of the Drive/Head register is compared to bits 4:0 of the Start Drive/Head register (reg. B0h, R/W).

It is important to realize that Auto-Read command execution automates only the first sector (or block) trans-

fer of a multi-sector read command or the only sector (or block) transfer of a single sector (or block) read command. If Automatic Multi-Sector Read Transfers are enabled by the conclusion of the first sector (or block) transfer, the transfer of all sectors following the first sector (or block) will be automated. If not enabled, the microprocessor must intervene after the first sector (or block) and either facilitate a manual transfer of the remaining sectors or enable automatic multi-sector transfers. Refer to the figures in Section 3.2.5.1 through 3.2.5.4 for details on signal protocol.

Another prerequisite for the automatic sending of the data to the host is that the buffer must be partially set up ahead of time. That is, the cached data must be in the host segment and the selected BCTR register must be primed with the number of valid sectors that are in that segment.

The cached data must reside in the Host segment. This segment is determined by the Host Segment Control register (reg. 133h, R/W) and the associated Beginning and End Of Segment Pointers. The Host Pointer (reg. 130h -132h, R/W) will be used as the address into the buffer.

The various ATA Read commands must be enabled for Auto-Read execution before they will be executed in that manner. This is done by setting the appropriate bits in Host Control 2 register (reg. C2h, R/W). Individual bits have been provided to allow some commands to be executed in Auto-Read mode while others will not. This allows the user to tailor execution of commands to the particular application.

If the Enable Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) is set, the command which has been enabled for Auto-Read execution will not stop after the first sector (or block) but will continue to automatically transfer the remaining sectors as allowed by buffer conditions. The Stop After One Block bit (reg. C1h, R/W, bit 2) can't be used for Auto-Read command execution to stop the transfer after one sector (or block).

The AIC-8375 incorporates a read interlock feature that can be used in several ways to pause the transfer before the last sector (or block) is transferred. This feature is used to prevent an entire Auto-Read from taking place without the local microprocessor knowing about it.

The Read Interlock bit (reg. C6h, R/W, bit 7) can be used during any PIO or DMA Read command (Auto-Read, automated read, or manual read) transfer to pause the transfer immediately before the last sector (or block) is to be transferred. This is useful to prevent sequential Read commands from executing without the processor ever having any knowledge that more than one command was serviced. If an Auto-Read command of one sector is received while this bit is set, the sector will not be transferred and the Transfer Done bit (reg. C8h, R/W, bit 0) is set. If an Auto-Read command is received after a prior Auto-Read command has been paused by this bit, but not allowed to finish the remaining sector transfer, the new Auto-Read command will not be allowed to start and Transfer Done will be set. If a Single Sector/Block Read command is received while the Read Interlock bit is set, the command will not be started and Transfer Done is set.

The Enable Auto-Read Interlock bit (reg. C6h, R/W, bit 6) is used to enable the read interlock feature for only Auto-Read commands. This bit is ignored if the Read Interlock bit (reg. C6h, R/W, bit 7) is set. If the local microprocessor writes a "1" to the Auto-Read Interlock Clear/Status bit (reg. C6h, R/W, bit 2) before the last sector (or block) is to transferred, the pause will not take place. The Auto-Read Interlock Clear/Status bit is set when an Auto-Read command is received. If a Single Sector/Block Read command is received while the Enable Auto-Read Interlock bit is set, the command will not be started and Transfer Done is set.

3.2.5.7 Automatic Multi-Sector Write Transfers

The entire transfer of data from the host to the buffer for a write command can be fully automated. This is accomplished by automating the first sector (or block) transfer via Auto-Write execution and by automating the remainder of the sectors by operating with the Enable Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) set. The data will be transferred until completion according to the flow illustrated in the figures in Section 3.2.5.1 through 3.2.5.4. Automatic Multi-Sector Transfer mode applies to both PIO and DMA type commands. During automatic multi-sector transfers, the Task File registers will be automatically updated between sector transfers while BUSY is asserted.

During execution, the data transfer will only be stopped for the following reasons: a hardware detected error has occurred, a microprocessor initiated stop has occurred, the buffer is not ready, or the command has been completed. The Pause Host Transfer bit (reg. C3h, R/W, bit 0) can be used to pause the transfer at a sector or block boundary if some action is required on the part of the microprocessor.

3.2.5.8 Automatic Multi-Sector Read Transfers

The entire transfer of data from the buffer to the host for a read command can be fully automated. This is accomplished by automating the first sector (or block) transfer via Auto-Read execution and by automating the remainder of the sectors by operating with the Enable Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) set. The data will be transferred until completion according to the flow illustrated in the figures in Section 3.2.5.1 through 3.2.5.4. Automatic Multi-Sector Transfer mode applies to both PIO and DMA type commands. During automatic multi-sector transfers, the Task File registers will be automatically updated between sector transfers while BUSY is asserted.

During execution, the data transfer will only be stopped for the following reasons: a hardware detected error has occurred, a microprocessor initiated stop has occurred, the buffer is not ready, the command has been completed or the last Sector/Block has been encountered with the Read Interlock or Auto-Read Interlock feature enabled. The Pause Host Transfer bit (reg. C3h, R/W, bit 0) can be used to pause the transfer at a sector or block boundary if some action is required on the part of the microprocessor.

3.2.6 AT Host Manual Operation

The AT automation can be turned off if manual execution of commands is desired. This would typically be done for support of special vendor unique commands, and for the Set Features and Identify Drive commands for example.

For manual operation, the microprocessor must interrogate the AT Command register (reg. A7h, R/W) to find out what command has been sent. If there is a data transfer involved, the microprocessor must adhere to the flow described in Section 3.2.5. The Enable Automatic Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) must be cleared. In addition, the Task File registers must be manually updated at the proper times.

3.2.7 Starting and Stopping Transfers Between the Host and Buffer

During manual execution of a command the microprocessor must restart the transfer for each sector or block and possibly stop the transfer if an error condition arises. During automatic transfers, the microprocessor might have to stop the transfer in order to facilitate some action and then restart it. The AIC-8375 incorporates features to perform the above functions.

Starting a New Transfer:

A command transfer being started manually is started by setting the Start AT Transfer bit (reg. C3h, R/W, bit 2) after all other appropriate bits and conditions have been set up. At the time this bit is used, the SECCNT register (reg. 1F2h, R) must have a non-zero value in it, otherwise the transfer will not be started. The exception to this is the case where the command is a one sector (or block) read command. Even though the microprocessor will change the SECCNT register to zero while manually updating the Task File registers before setting the Start AT Transfer bit, the transfer will automatically start.

If the command is a write command, the IRQ signal pin and bit (BFh, R, bit 3) are not asserted. DRQ will be automatically asserted. In the event IRQ mode '0' or '1' is enabled, the Busy Timer will not be used during this first part of the transfer.

If the command is a read command, the IRQ signal pin and bit and the DRQ bit will be automatically asserted according the rules governing the enabled IRQ mode.

If it is desired to start a Write Command transfer with IRQ getting set, the Restart AT Transfer bit (reg. C3h, R/W, bit 1) should be used.

Restarting a Transfer:

A command transfer which has been stopped can be started manually by setting the Restart AT Transfer bit (reg. C3h, R/W, bit 1) after all other appropriate bits and conditions have been set up. At the time this bit is used, the SECCNT register (reg. 1F2h, R) must have a non-zero value in it, otherwise the transfer will not be restarted.

If the command is a write command, the IRQ signal pin and bit (BFh, R, bit 3) and DRQ are automatically asserted per the IRQ mode '0' or '1' that is enabled.

If the command is a read command, the IRQ and DRQ will be asserted according the rules governing the enabled IRQ mode.

If it is desired to restart a Write Command transfer without IRQ getting set, the Start AT Transfer bit (reg. C3h, R/W, bit 2) should be used.

The Host Transfer Started bit (reg. CAh, R, bit 0) will be set when the host starts either a DMA or PIO transfer.

Stopping the Transfer:

A transfer in progress can be stopped by setting the Pause Host Transfer bit (reg. C3h, R/W, bit 0).

For a PIO write transfer, the transfer will stop at the next available sector (or block) boundary. The actual pause will occur after the Task File registers have been updated. AT Busy will be set and DRQ will be in the negated state after the pause.

For a PIO read transfer, the transfer will stop at the next available sector (or block) boundary. The Task File registers will be updated after the transfer is restarted. AT Busy will be set and DRQ will be in the negated state after the pause.

The Read Interlock bit (reg. C6h, R/W, bit 7) can be used during PIO read command transfers to pause the transfer immediately before the last sector (or block) is to be transferred. This is useful to prevent sequential Auto-Read PIO commands from executing without the processor ever having any knowledge that more than one command was serviced.

Another feature of the AIC-8375 is the Auto-Read Interlock bit which is used to prevent the completion of an Auto-Read command. When the Enable Auto-Read Interlock bit (reg. C6h, R/W, bit 6) is set, automatic multi-sector transfer will stop at the last Inter-Sector Gap only if the Auto-Read Command Received Interrupt (reg. C8h, R, bit 5) is generated. It does not stop if data transfers were initiated by firmware for normal read commands. But if the microprocessor clears the AUTORDINTLKSTAT bit (reg. C6h, R/W, bit 2) before the last Inter-Sector Gap, data transfers of the last sector or block continues without pause. When the Read Interlock bit (reg. C6h, R/W, bit 7) is set, this bit is ignored.

For a DMA write transfer, the transfer will stop at the next available sector boundary. The actual pause will occur after the Task File registers have been updated. AT Busy will be set and DRQ will be in the negated state after the pause. In addition DMAREQ will be negated.

For a DMA read transfer, the transfer will stop at the next available sector boundary. The Task File registers will be updated after the transfer is restarted. AT Busy will be set and DRQ and DMAREQ will be in the negated state after the pause.

The Read Interlock bit (reg. C6h, R/W, bit 7) can also be used during DMA read transfers to stop the transfer before the last sector of data is sent to the host to prevent back to back Auto-Read DMA commands from confusing the microprocessor.

If the Enable Auto Pause On Error bit (reg. C3h, R/W, bit 4) is set, the host data transfer will be stopped at the next sector (or block) boundary if any of the following four error bits are set; the Error bit (reg. C1h, R/W, bit 0), the Host FIFO Error bit (reg. C8h, R/W, bit 1), the Auto-Write Overun bit (reg. CAh, R, bit 3) and the Host Port Check Error bit (reg. 106h, R, bit 0).

Refer to the transfer flow illustrated in the figures in Section 3.2.5.1 through 3.2.5.4.

3.2.8 PCMCIA Interface Functionality

If the PCMCIA interface is selected, the host still communicates with the AIC-8375 via the Task File registers. These registers can be accessed using any of the three addressing modes supported by the AIC-8375: Primary/Secondary I/O, Block I/O, and Memory. The use of all three of these addressing modes is covered in detail in Section 2 of this document.

3.2.8.1 Addressing Modes

Any of the three supported addressing modes can be used with the PCMCIA interface. A summary of the operation of each of these modes is provided here. A detailed description of each is provided in Section 2.

3.2.8.2 Primary/Secondary I/O Mode Addressing

The Primary/Secondary Mode is the standard ATA mode. It should be used when compatibility with existing BIOSes is needed. This mode decodes the lowest 1 KByte of I/O address space and only responds when the address is within the selected range. The host selects which address range the chip will decode through the Configuration Index (bits 3:0 in the Configuration Option register at host address 200h). If Primary I/O mode is selected, the card responds to 1F0-1F7h and 3F6-3F7h. The Secondary I/O range is 170-177h and 376-377h. When a valid I/O read address is decoded by the AIC-8375, the *INPACK signal is asserted.

3.2.8.3 Block I/O (Independent) Mode Addressing

The Block I/O Addressing Mode permits the Task File registers to be located at any block of 16 I/O addresses. This mode requires the host socket to perform the address decoding. It uses the *HCE1 and *HCE2 signals to select the AIC-8375, to determine whether an 8- or 16-bit access is occurring, and to enable one (for 8-bit accesses) or both (for 16-bit accesses) of the byte lanes. This allows the host to assign multiple cards to independent I/O blocks while avoiding I/O port conflicts.

If a PCMCIA interface is used in the Block I/O mode, the *IOIS16 output tells external circuitry whether a particular access to the AIC-8375 can be 16 bits wide. If this pin is asserted, the addressed register is capable of 16-bit transfers and the host, at its option, can initiate either an 8- or 16-bit I/O transfer. For the ATA interface *IOIS16 is renamed to *IOCS16; in this case the host must transfer 8 or 16 bits as specified by *IOCS16.

3.2.8.4 Memory Mode Addressing

This mode is active by default after a Power-On Reset, a PCMCIA reset condition, or by host selection in the Configuration Option register (reg. 200h, R/W). This mode is used to support memory-only hosts or PCMCIA 1.0 Sockets (that do not offer I/O support). It is used to map the ATA Task File registers into a 16 byte memory block and a 1 KByte memory block used to transfer data.

3.2.9 Host / Buffer ECC Transfers

During Read Long and Write Long command execution, the data bytes and ECC bytes, for the indicated disk address, are transferred between the Host and the buffer. The number of ECC bytes transferred will normally be dictated by the particular Host BIOS implementation. This value may differ from the actual number of ECC bytes used in the disk sector format.

The Host ECC Size bits (reg. D2h, R/W, bit 5:0) are used to specify the number of ECC bytes to be transferred between the Host and the buffer during the Read Long and Write Long commands.

The protocol of these commands is slightly different from that of the normal read and write commands and warrant some discussion.

3.2.9.1 Write Long Transfers

Write Long command execution will take place as shown in Figure 3-10.

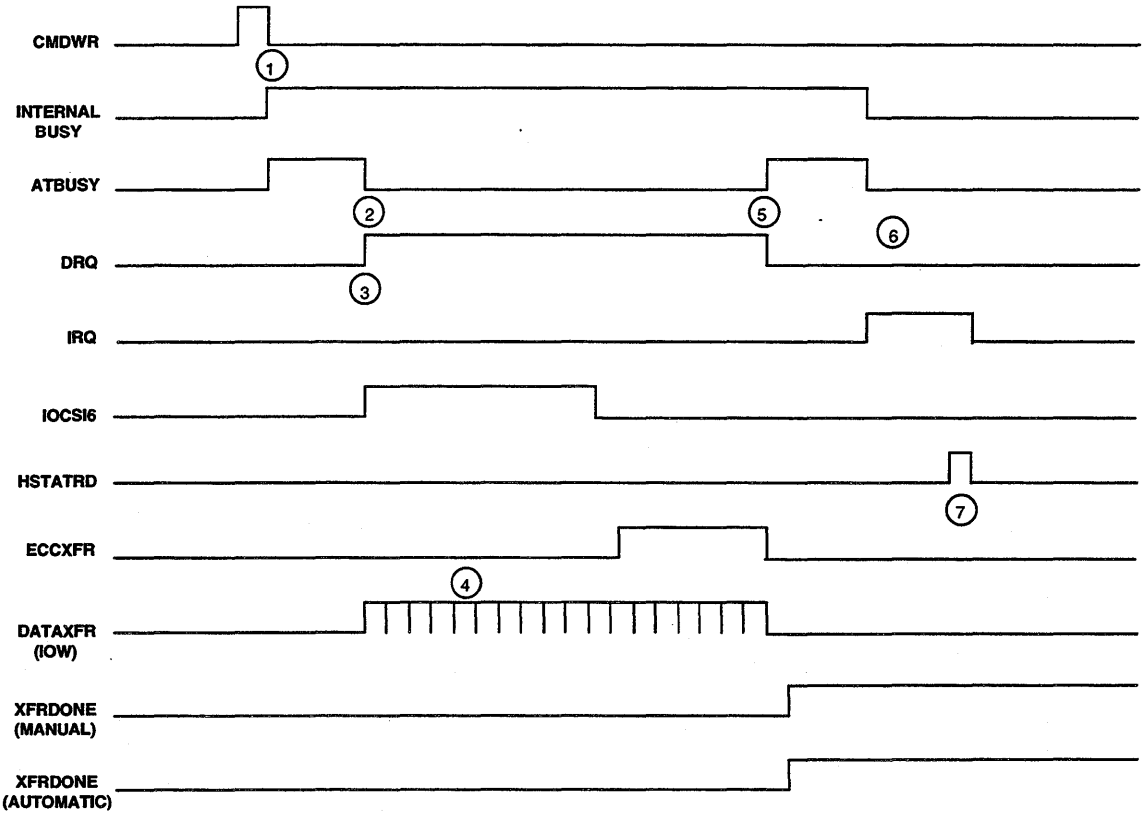

Figure 3-10 Write Long Transfers

1. The host loads the Task File registers, loading the AT Host Command register (reg. 1F7h, W) last with the Write Long command. This sets the AT Busy bit (reg. 1F7h, R, bit 7) / (reg. CEh, R, bit 6) and the Internal Busy bit (reg. C1h, R, bit 7). The Selection Phase Detected bit (reg. C8h, R, bit 2) is also set along with the appropriate bits in the Host Status 0 register (reg. CCh, R) that indicate which command was received. Many other bits are cleared at this time. However, various status and control bits will only be affected in the drive selected by the DRV bit.

2. If the command has been enabled for Auto-Write execution via the Enable Auto-Write bit (reg. C2h, R/W, bit 6), AT Busy is negated and DRQ (reg. 1F7h, R, bit 3) is asserted automatically after a very short time (about 200 ns). If Auto-Write execution is not enabled, the microprocessor must start the transfer using the Start AT Transfer bit (reg. C3h, R/W, bit 2).

3. The host might read the AT Status register (reg. 1F7h, R) to ascertain if DRQ is set, or it may begin the transfer without checking. If the register is read, the Host Status Read Detected bit (reg. CAh, R, bit 2) is set.

4. The host will now begin to transfer the data field and ECC bytes for that sector. The number of ECC bytes that the host will send depends on which BIOS or device driver is being used. The ECC bytes will be transferred across the interface in 8-bit mode. During the ECC byte transfer time, the *IOCS16 signal will be negated to indicate an 8-bit transfer.

5. After the data and ECC bytes have been transferred, AT Busy will be asserted and DRQ will be negated automatically. The Transfer Done bit (reg. C8h, R, bit 0) will be set at this time. The microprocessor may write the data to the drive and then post status or do write caching and immediately post ending status and then write the data to the drive.

6. AT Busy and Internal Busy are cleared by setting the Reset Busy bit (reg. C1h, R/W, bit 6) and IRQ is manually set by the microprocessor by setting the Host Interrupt Request bit (reg. C1h, R/W, bit 4).

7. The host typically will read the AT Status register (reg. 1F7h, R) which will in turn reset IRQ.

3.2.9.2 Read Long Transfers

Read Long command execution will take place as shown in Figure 3-11.

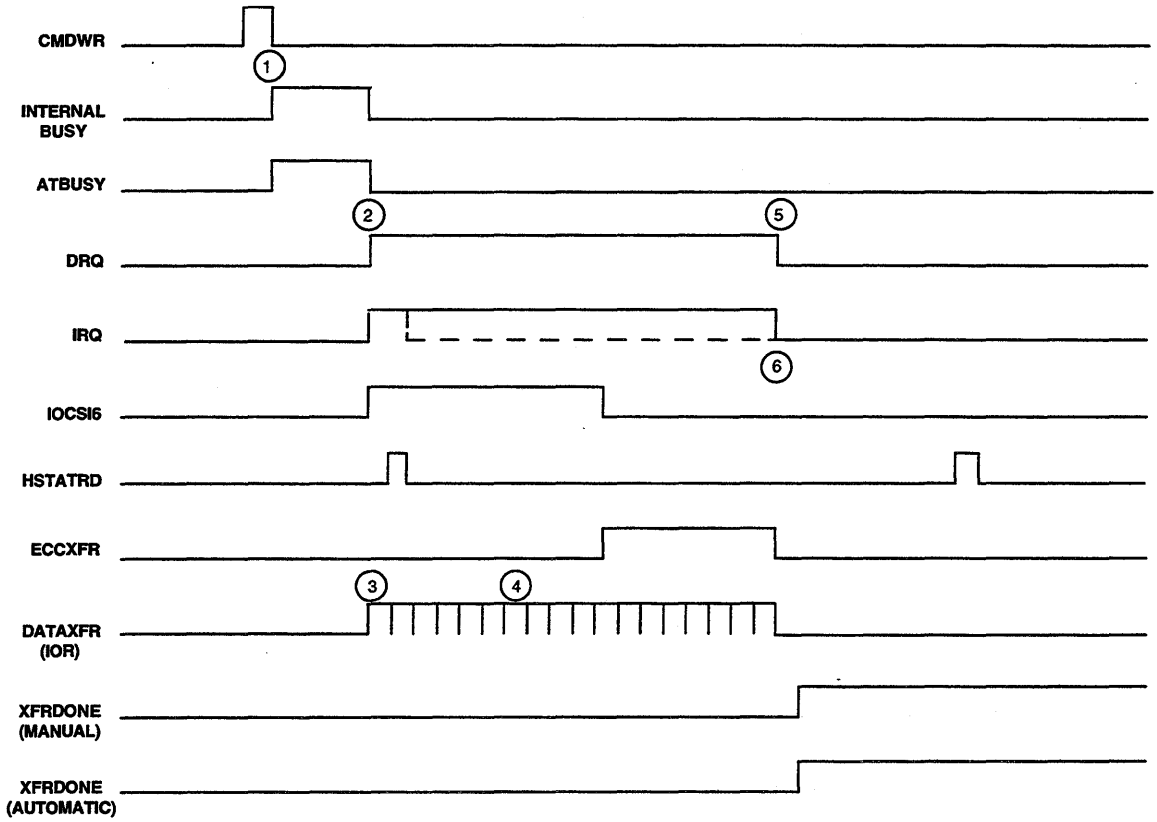

Figure 3-11 Read Long Transfers

1. The host loads the Task File registers, loading the AT Host Command register (reg. 1F7h, W) last with a Read Long command. This sets the AT Busy bit (reg. 1F7h, R, bit 7) / (reg. CEh, R, bit 6) and the Internal Busy bit (reg. C1h, R, bit 7). The Selection Phase Detected bit (reg. C8h, R, bit 2) is also set along with the appropriate bits in the Host Status 0 register (reg. CCh, R) that indicate which command was received. Many other bits are cleared at this time. However, various status and control bits will only be affected in the drive selected by the DRV bit. The microprocessor must update the Task File registers before the transfer begins.

2. When data is available in the buffer and host FIFO, AT Busy is negated while DRQ (reg. 1F7h, R, bit 3) and IRQ are asserted. This is accomplished via the microprocessor by setting the Start AT Transfer bit (reg. C3h, R/W, bit 2).

3. The host either waits for IRQ or polls for DRQ in order to begin the transfer. If DRQ is used, the host typically reads the AT Status register (reg. 1F7h, R) to ascertain if DRQ is set. If the register is read, the Host Status Read Detected bit (reg. CAh, R, bit 2) is set and IRQ is negated.

4. The host will begin to read the data and ECC bytes for the indicated sector. During the transfer of the data bytes, *IOCS16 will be asserted to indicate a 16-bit transfer. During the ECC byte transfer, *IOCS16 will be negated to indicate an 8-bit transfer.

5. After data is transferred, DRQ will be negated automatically.

6. IRQ will either be negated by the host having read status or the last byte of read transfer having occurred, or manually via the microprocessor. The hardware and the Reset Host Interrupt Request bit (reg. C1h, W, bit 3) are controlling bits for this action.

3.2.10 Host FIFO Operation

The Host FIFO is a 32-byte FIFO which interfaces the ATA host bus to the buffer RAM. The state of the host FIFO (eg. Full or Empty status) plays a part in certain sequences of host/buffer automation. For example, the Buffer Transfer bit (reg. C1h, R/W, bit 6), is set during transfers between the buffer and host. Host FIFO full and empty status will play a part in when this bit is set or cleared.

The host and the buffer data paths can either be an 8-bit or a 16-bit path. The host FIFO is designed to operate with any combination of host or buffer data path. Any path conversion is taken care of in the Host FIFO circuit. To accomplish this, the Host FIFO is organized as a 16-bit x 16-word FIFO with byte addressing capability.

The status of the Host FIFO is used as a gating factor during the Interblock gap time between sector transfers. This status, together with other ATA protocol signals and states, is used to control the data transfer to and from the host.

During write operations, the host FIFO will be considered ready as long as there is at least one word (or in some cases one byte) of space available. The Transfer Done status (reg. C8h, R, bit 0) will not get set until the Host Transfer Counter (reg. D0h, D1h) decrements to zero and the remaining bytes for the current sector have been transferred to the buffer.

During read operations, the Host FIFO must be full before it is considered as ready to leave the Interblock gap state. The Transfer Done status (reg. C8h, R, bit 0) will not get set until the Host Transfer Counter (reg. D0h, D1h) decrements to zero and the remaining bytes for the current sector have been transferred to the host.

After the Interblock Gap time is done and the data transfer is taking place, the state of the Host FIFO will be used to regulate each individual word (or byte) transfer. During PIO transfers, if the Host FIFO is not ready, IOCHRDY will be negated so the host can keep *IOW or *IOR asserted until it is ready. During a DMA transfer, if the Host FIFO is not ready, DMAREQ will not be asserted for the next word transfer until the host FIFO is ready.

At the beginning of a transfer, the Host Transfer Counter will be loaded with the contents of the Sector Size register (reg. 120h, 121h).

The Host FIFO interfaces to the buffer RAM via the buffer control block. The operation of the Host FIFO can be altered either automatically or manually. The Enable High Priority for Host bit (reg. 100h, R/W, bit 1) is used for this purpose.

Normally, the order of buffer access priority is Servo, Disk, DRAM Refresh, Correction, Microprocessor, Host, and Host CIS. The Enable High Priority for Host bit (reg. 100h, R/W, bit 1) can be used to alter this priority. When this bit is set, the host will be given higher priority than the Correction and microprocessor ports whenever the Host FIFO is available to transfer 16 bytes or more to/from the buffer. Under these conditions, the buffer access priority will be Servo, Disk, DRAM Refresh, Host High, Correction, Microprocessor, and Host CIS. When this bit is cleared, or when it is set and the Host FIFO is available to transfer less

than 16 bytes, the host port will have the lowest priority. The Enable High Priority for host bit may be operated in the set state if the buffer bandwidth exceeds the sum of the disk and host transfer rates. If the buffer bandwidth is less than the sum of the disk and host transfer rates, this bit must be reset to prevent the correction port from getting locked out of the buffer. In this case, the host must use the ATA IOCHRDY signal during PIO transfers, or DMA transfer mode to slow down the host transfer rate.

When operating in DRAM mode, the Host FIFO interfaces to the buffer RAM using page mode transfers of one to 256 bytes. This feature helps ensure quick response to fast hosts and also helps in facilitating the internal buffer contention algorithm. When operating in DRAM mode, every Host FIFO/Buffer access is initiated as a 256-byte page mode access. This access will be terminated if a page boundary is encountered, the end of the data transfer is reached, or a higher priority buffer port needs servicing. Host FIFO buffer access priority is checked on every byte transfer time.

The Host FIFO will automatically take care of dealing with mixed size transfers during the Read and Write Long commands when the data bytes are transferred in 16-bit mode and the ECC bytes are transferred in 8-bit mode.

Host FIFO Error Conditions:

There are several error conditions related to the host FIFO which are monitored and can be flagged. These error conditions can be programmed in the various interrupt control registers to generate an interrupt to the microprocessor. If the Enable Auto Pause On Error bit (reg. C3h, R/W, bit 4) is set while any of these conditions occur, the host transfer will be stopped at the next sector (or block) boundary and the Transfer Done bit (reg. C8h, R/W, bit 0) will be set. The pause will occur if any of the following four bits are set: the Error bit (reg. C1h, R/W, bit 0), the Host FIFO error bit (reg. C8h, R, bit 1), the AUTOWROVRN bit (reg. CAh, R, bit 3), and the HCHKERR bit (reg. 106h, R/W, bit 0). If the Enable Auto Error Set bit (reg. C3h, R/W, bit 5) is set, then if any of these conditions occur, the AT Error bit (reg. 1F7h, R) will automatically be set.

The Host FIFO Error bit (reg. C8h, R, bit 1), is set whenever a data underrun or overrun condition occurs.

The Host Port Check Error bit (reg. 106h, R, bit 0) is set if, while the Enable Buffer Parity Checking bit (reg. 105h, R/W, bit 0) is set, a parity error is detected when the Host Port (to the buffer) reads a byte from the buffer.

The Auto-Write Overrun Error bit (reg. CAh, R, bit 3) is set whenever the host starts writing to the Data Port (reg. 1F0h, R/W) after an Auto-Write command has been received and is automatically starting execution before the host FIFO and buffer is ready.

3.2.11 Host Logical Block Addressing

The AIC-8375 device can be configured to accept ATA commands in LBA (logical Block Address) format or in CHS (Cylinder/Head/Sector) format.

The Enable LBA Mode bit (reg. C3h, R/W, bit 3) is used to switch between LBA mode and CHS mode of operation. When this bit is set, the device will adapt to the mode specified in the LBA bit (reg. 1F6, R/W, bit 6). When this bit is cleared, the device will operate only in CHS mode. Table 3-12 summarizes details of the two modes of operation.

Table 3-12 CHS and LBA Mode Details

State of AIC-8375 Enable Mode Bit	ENLBAMODE = 1 (Reg. C3h, R/W, bit 3 = 1)		ENLBAMODE = 0 (Reg. C3h, R/W, bit 3 = 0)	
State of LBA Field in command from host	CMD LBA = 1 (Reg. 1F6h, R/W, bits 7:5 = "111")	CMD LBA = 0 (Reg. 1F6h, R/W, bits 7:5 = "101")	CMD LBA = 1 (Reg. 1F6h, R/W, bits 7:5 = "XXX")	CMD LBA = 0 (Reg. 1F6h, R/W, bits 7:5 = "XXX")
Task File Usage	LBA	CHS	CHS	CHS
Proper Format of Start Registers (Reg. ADh-B0h)	Drive/HD (7:5) = "111" SECNUM = LBA (7:0) CYL LO = LBA (15:8) CYC HI = LBA (23:16) Drive/HD (3:0) = LBA (27:24)	Drive/HD (7:5) = "101" SECNUM = SECNUM CYL LO = CYL LO CYC HI = CYC HI Drive/HD (3:0) = Head	Drive/HD (7:5) = "XXX" SECNUM = SECNUM CYL LO = CYL LO CYC HI = CYC HI Drive/HD (3:0) = Head	Drive/HD (7:5) = "XXX" SECNUM = SECNUM CYL LO = CYL LO CYC HI = CYC HI Drive/HD (3:0) = Head
Start Reg to Task File Reg. Comparison	ADh Comp. 1F3h AEh Comp. 1F4h AFh Comp. 1F5h B0h Comp. 1F6h	ADh Comp. 1F3h AEh Comp. 1F4h AFh Comp. 1F5h B0h Comp. 1F6h	ADh Comp. 1F3h AEh Comp. 1F4h AFh Comp. 1F5h B0h (4:0) Comp. 1F6h (4:0)	ADh Comp. 1F3h AEh Comp. 1F4h AFh Comp. 1F5h B0h (3:0) Comp. 1F6h (3:0)
Notes			Illegal condition - See note 1.	

Note 1: Per ATA specifications, host should not send LBA commands when controller is not set up in LBA mode.

The task file registers are incremented differently in LBA or CHS mode.

In CHS mode, the Sector Number register (reg. 1F3h, R/W) will increment to the value in the MAXSEC 0 register (reg. B1h, R/W) or the MAXSEC 1 register (reg. B2h, R/W) and then wrap to one. This increments the Head Select bits (bits 3:0) in the Drive/Head register (reg. 1F6h, R/W). This field will in turn wrap to zero after it increments past the value specified by either the MAXHEAD_0 or MAXHEAD_1 bits in the MAXHEAD register (reg. B3h, R/W). This will be followed by cascaded incrementing and wrapping of the Cylinder Low register (reg. 1F4h, R/W) followed by the Cylinder High register (reg. 1F5h, R/W).

In LBA mode, the task file registers are treated as a linear address register. The Sector Number register (reg. 1F3h, R/W) is incremented first without any comparison to either the MAXSEC 0 or MAXSEC 1 registers. It will wrap to zero after incrementing through FFh. Incrementing and wrapping will continue in the same manner with the Cylinder Low register (reg. 1F4h, R/W) followed by the Cylinder High register (reg. 1F5h, R/W) and finally the low order four bits of the Drive/Head register (reg. 1F6h, R/W). The four bits in the Drive/Head register will wrap to zero after incrementing through 0Fh, without any compare to either the MAXHEAD_0 or MAXHEAD_1 fields in the MAXHEAD register.

3.2.12 Host Block Initialization

This section presents a sample initialization setup for the host block. The host block contains many AT automation features. Some of the major features are Auto Write, Auto Read, host-buffer ADFM, automatic task file register updates for reads/writes including the read verify command, auto IRQ generations, CHS/ LBA mode support, single and dual drive support, and PIO/DMA data transfer mode support.

The sample flow presented here supports the following commonly used features.

- Auto-Write using an Auto-Write segment set to circular buffer.
- ADFM on the host and buffer blocks using BCTRA for Auto-Writes.
- ADFM on the host and buffer blocks using BCTRB for reads.
- 16-bit host data transfers with automatic switching to 8-bit ECC data transfer.
- CHS mode and LBA mode.
- Auto IRQ generation.
- Automatic task file register updates.
- Single drive support.
- Normal PIO and Single word DMA modes.
- Auto Busy clear for host read commands.

In the flow table below, if any bits in a register are not specifically mentioned to be either set or cleared, their original value must be retained while setting or clearing the specified bits.

Table 3-13 Host Block Initialization

Initialization Function	Register Action
Enable automatic task file register updates, active low Host Chip Select 1, and single Master drive mode.	Set reg. C0h = 0Ah.
Disable stop after one block host data transfer.	Write 00h to reg. C1h.
Enable Auto-writes with multi sector transfer.	Set reg. C2h = D5h.
Enable auto error pause & CHS/LBA mode.	Write 38h to reg. C3h.
Set IRQ mode. Set Busy value and IRQ Timer values.	Set reg. C4h = 01h. Set reg. BAh = desired value. Set reg. BBh. = desired value.
Enable 16 bit Host data transfer, auto IRQ, PIO mode, auto wait states, early IOCHRDY.	Set reg. C5h = 5Dh.
Enable Auto-read interlock feature, reset host FIFO, enable single word DMA, and enable the host data bus drive enable for both DMA and PIO commands.	Set reg. C6h = 59h.
Set Read/Write Multiple block size.	Set reg. B6h = currently selected block size.
Set maximum logical sector and head.	Set reg. B1h, B2h, B3h = max logical sector and head.
Set Host ECC size for Read and Write Long commands.	Set reg. D2h, bits 5:0 = desired ECC count value.
Initialize task file registers.	Set reg. A1h = 00h, reg. A2h = 01h, reg. A3h = 01h, reg. A4h = 00h, reg. A5h = 00h, and reg. A6h = 00h.
Enable Host interrupts.	Set reg. C9h = 7Fh. Set reg. 53h bits 1, 0.
Enable Host port for auto commands.	Set reg. 109h bit 0.
Ensure the PDIAG/DASP lines are set properly at the appropriate time according to the ATA specification.	Ensure reg. BFh bits 6, 5 are properly set.

3.2.13 ATA Write Command Execution

The details of ATA write command automation was presented in Section 3.2.5. Using that as background information, the sections which follow will summarize the important aspects of each command and describe any additional information which should be known for executing the Host/Buffer portion of the specific ATA write commands.

3.2.13.1 Auto-Write Sector(s)

- PIO Transfer mode is used.
- The Host Segment Control register (reg. 133h, R/W) determines where the Auto-Write data will be written in the buffer.

- Prior to beginning the transfer, the Host Pointer will either be loaded with the Write Cache Pointer or the selected Beginning Of Segment Pointer, or will not be modified.
- The transfer can be automatically paused after the first sector is transferred using the Stop After One Block bit (reg. C1h, R/W, bit 2).
- The entire transfer will take place automatically if the Enable Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) is set.
- The ATA Task File registers will automatically be updated after each sector is transferred. The ATA inter-sector protocol will automatically be executed by the hardware between each sector transfer.
- The Host Write Operation bit (reg. C3h, R/W, bit 7) is automatically set upon receipt of the command.
- The Host Transfer Count register (reg. D0h-D1h, R) indicates how many bytes of the current sector remain to be transferred between the Host and the Host FIFO.
- The Host Byte Counter (reg. 124h-125h, R/W) indicates the number of bytes remaining in the current sector to be transferred between the Host FIFO and the Buffer RAM.
- After the last sector is transferred, the final ATA Task File register update must occur. The microprocessor must clear BSY and then set IRQ.

3.2.13.2 Manual Write Sector(s)

- PIO Transfer mode is used.
- The Host segment in the buffer is used for the transfer. This is determined via the Host Segment Control register (reg. 133h, R/W).
- The Host Pointer (reg. 130h-132h, R/W) must be set up manually to point to the starting location in the host segment where data is to be transferred.
- The transfer is manually restarted every sector using the Restart AT Transfer bit (reg. C3h, R/W, bit 1).
- The entire transfer will take place manually since the Enable Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) is not set. After the last sector is transferred, the final ATA Task File register update must occur. If the DISAUTOUPD bit (reg. C0h, R/W, bit 6) is not set, the Task File registers will be updated by the hardware. If the DISAUTOUPD bit is set, the microprocessor must do the update. The microprocessor must then set IRQ.
- The ATA Task File register updates can occur automatically at the inter-sector gap time if the Disable Automatic Command Block Update bit (reg. C0h, R/W, bit 6) is not set. If this bit is set, the microprocessor must perform the update completely manually or assert the Update Host Count bit (reg. C6h, R/W, bit 5) to initiate the update.
- The Host Write Operation bit (reg. C3h, R/W, bit 7) is automatically set upon receipt of the command.
- The Host Transfer Count register (reg. D0h-D1h, R) indicates how many bytes of the current sector remain to be transferred between the Host and the Host FIFO.
- The Host Byte Counter (reg. 124h-125h, R/W) indicates the number of bytes remaining in the current sector to be transferred between the Host FIFO and the Buffer RAM.

3.2.13.3 Auto-Write Long

- PIO Transfer mode is used.
- The Host Segment Control register (reg. 133h, R/W) determines where the Auto-Write data will be written in the buffer.
- Prior to beginning the transfer, the Host Pointer will either be loaded with the Write Cache Pointer or the selected Beginning Of Segment Pointer, or will not be modified.
- The transfer is a one sector transfer type command and will be fully automated except for the end of the command. The final setting of IRQ will not occur automatically. The microprocessor must do this manually at the end of the command.
- The Enable Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) will have no effect.
- Only the initial beginning of ATA protocol will automatically be executed by the hardware. *IOCS16 will automatically be negated during the transfer of the 8-bit ECC bytes.
- The Host Write Operation bit (reg. C3h, R/W, bit 7) is automatically set upon receipt of the command.
- The Host Transfer Count register (reg. D0h-D1h, R) indicates how many bytes of the current sector remain to be transferred between the Host and the Host FIFO. During "user data" time, this register will contain the remaining byte count for the 512-byte data field. During the "ECC" time, this register will contain the number ECC bytes remaining to be transferred. In addition, the number of ECC bytes of the current sector remaining to be transferred between the Host FIFO and the Buffer RAM is available in the Host ECC Count[5:0] field (reg. D3h, R, bits 5:0).
- The Host Byte Counter (reg. 124h-125h, R/W) indicates the number of bytes remaining in the current sector to be transferred between the Host FIFO and the Buffer RAM.
- The number of ECC bytes that will be transferred is governed by the host. This number should be known and set into the Host ECC Size [5:0] field (reg. D2h, R/W, bits 5:0) at initialization time.

3.2.13.4 Manual Write Long

- PIO Transfer mode is used.
- The Host segment in the buffer is used for the transfer. This is determined via the Host Segment and Control register (reg. 133h, R/W).
- The Host Pointer (reg. 130h-132h, R/W) is manually set to the starting location in the Host segment.
- The transfer is a one sector transfer type command. The microprocessor must facilitate all the ATA protocol handshaking. The final setting of IRQ will not occur automatically. The microprocessor must do this manually at the end of the command.
- The Enable Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) will have no effect.
- The Host Write Operation bit (reg. C3h, R/W, bit 7) is automatically set upon receipt of the command.
- The Host Transfer Count register (reg. D0h-D1h, R) indicates how many bytes of the current sector remain to be transferred between the Host and the Host FIFO. During "user data" time, this register will contain the remaining byte count for the 512-byte data field. During the "ECC" time, this register will contain the number ECC bytes remaining to be transferred. In addition, the number of

ECC bytes of the current sector remaining to be transferred between the Host FIFO and the Buffer RAM is available in the Host ECC Count [5:0] field (reg. D3h, R, bits 5:0).

- The Host Byte Counter (reg. 124h-125h, R/W) indicates the number of bytes remaining in the current sector to be transferred between the Host FIFO and the Buffer RAM.
- The number of ECC bytes that will be transferred is governed by the host. This number should be known and set into the Host ECC Size [5:0] field (reg. D2h, R/W, bits 5:0) at initialization time.

3.2.13.5 Auto-Write Multiple

- PIO Transfer mode is used.
- The Host Segment Control register (reg. 133h, R/W) determines where the Auto-Write data will be written in the buffer.
- Prior to beginning the transfer, the Host Pointer will either be loaded with the Write Cache Pointer or the selected Beginning Of Segment Pointer, or will not be modified.
- The transfer can be automatically paused after the first block is transferred using the Stop After One Block bit (reg. C1h, R/W, bit 2).
- The entire transfer will take place automatically since the Enable Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) is set.
- The ATA Task File registers will automatically be updated on block boundaries and the ATA inter-block protocol will automatically be executed by the hardware between block transfers.
- The Host Write Operation bit (reg. C3h, R/W, bit 7) is automatically set upon receipt of the command.
- The Host Transfer Count register (reg. D0h-D1h, R) indicates how many bytes of the current sector remain to be transferred between the Host and the Host FIFO.
- The Host Byte Counter (reg. 124h-125h, R/W) indicates the number of bytes remaining in the current sector to be transferred between the Host FIFO and the Buffer RAM.
- The Host Block Size register (reg. B6h, R/W) determines the number of sectors per block.

3.2.13.6 Manual Write Multiple

- PIO Transfer mode is used.
- The Host segment in the buffer is always used for the transfer. This is determined via the Host Segment and Control register (reg. 133h, R/W).
- The Host Pointer (reg. 130h-132h, R/W) is manually set to the beginning of the Host segment.
- The transfer is restarted before every block using the Restart AT Transfer bit (reg. C3h, R/W, bit 1).
- The entire transfer will take place manually since the Enable Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) is not set.
- The ATA Task File register updates can occur automatically at the inter-sector gap time if the Disable Automatic Command Block Update bit (reg. C0h, R/W, bit 6) is not set. If this bit is set, the microprocessor must perform the update completely manually or assert the Update Host Count bit (reg. C6h, R/W, bit 5) to initiate the update.

- The Host Write Operation bit (reg. C3h, R/W, bit 7) is automatically set upon receipt of the command.
- The Host Transfer Count register (reg. D0h-D1h, R) indicates how many bytes of the current sector remain to be transferred between the Host and the Host FIFO.
- The Host Byte Counter (reg. 124h-125h, R/W) indicates the number of bytes remaining in the current sector to be transferred between the Host FIFO and the Buffer RAM.
- The Host Block Size register (reg. B6h, R/W) determines the number of sectors per block.
- The Internal Busy bit (reg. C1h, R, bit 7) and the AT Busy bit must be cleared by the microprocessor setting the Reset Busy bit (reg. C1h, R/W, bit 6). The microprocessor must conclude the command by asserting IRQ.

3.2.13.7 Auto-Write DMA

- DMA Transfer mode is used. The PIO Transfer Mode Enable bit (reg. C5h, R/W, bit 3) will be ignored.
- The Host Segment Control register (reg. 133h, R/W) determines where the Auto-Write data will be written in the buffer.
- Prior to beginning the transfer, the Host Pointer will either be loaded with the Write Cache Pointer or the selected Beginning Of Segment Pointer, or will not be modified.
- The transfer can be automatically paused after the first sector is transferred using the Stop After One Block bit (reg. C1h, R/W, bit 2).
- The entire transfer will take place automatically since the Enable Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) is set.
- The ATA Task File registers will automatically be updated on sector boundaries even though it is a DMA command and not looked at by the host during the transfer.
- The hardware will automatically pause and restart the transfer based on buffer full/empty status using the DMAREQ/*DMACK handshake used for DMA transfers.
- The Host Write Operation bit (reg. C3h, R/W, bit 7) is automatically set upon receipt of the command.
- The Host Transfer Count register (reg. D0h-D1h, R) indicates how many bytes of the current sector remain to be transferred between the Host and the Host FIFO.
- The Host Byte Counter (reg. 124h-125h, R/W) indicates the number of bytes remaining in the current sector to be transferred between the Host FIFO and the Buffer RAM.
- The Internal Busy bit (reg. C1h, R, bit 7) and the AT Busy bit must be cleared by the microprocessor setting the Reset Busy bit (reg. C1h, R/W, bit 6). The microprocessor must conclude the command by asserting IRQ.

3.2.13.8 Manual Write DMA

- DMA Transfer mode is used. The PIO Transfer Mode Enable bit (reg. C5h, R/W, bit 3) will be ignored.
- The Host segment in the buffer is used for the transfer. The Host Segment Control register (reg. 133h, R/W) determines where the Auto-Write data will be written in the buffer.
- The Host Pointer (reg. 130h-132h, R/W) is manually set to the starting location in the Host segment.
- The entire transfer will take place manually since the Enable Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) is not set.
- The transfer is restarted every sector using Start AT Transfer bit (reg. C3h, R/W, bit 2), rather than the Restart AT Transfer bit, since IRQ assertions are not wanted.
- If the DISAUTOUPD bit (reg. C0h, R/W, bit 6) is not set, the Task File registers will be updated by the hardware. If the DISAUTOUPD bit is set, the microprocessor must do the update.
- The hardware will automatically pause and restart the transfer based on buffer full/empty status using the DMAREQ/*DMACK handshake used for DMA transfers.
- The Host Write Operation bit (reg. C3h, R/W, bit 7) is automatically set upon receipt of the command.
- The Host Transfer Count register (reg. D0h-D1h, R) indicates how many bytes of the current sector remain to be transferred between the Host and the Host FIFO.
- The Host Byte Counter (reg. 124h-125h, R/W) indicates the number of bytes remaining in the current sector to be transferred between the Host FIFO and the Buffer RAM.
- The Internal Busy bit (reg. C1h, R, bit 7) and the AT Busy bit must be cleared by the microprocessor setting the Reset Busy bit (reg. C1h, R/W, bit 6). The microprocessor must conclude the command by asserting IRQ.

3.2.14 ATA Read Command Execution

The details of ATA read command automation was presented in Section 3.2.5. Using that as background information, the sections which follow will summarize the important aspects of each command and describe any additional information which should be known for executing the Host/Buffer portion of the specific ATA read commands.

3.2.14.1 Auto-Read Sector(s)

- PIO Transfer mode is used.
- The Host segment in the buffer is used for the transfer. This is determined via the Host Segment Control register (reg. 133h, R/W).
- The Host Pointer (reg. 130h-132h, R/W) must be set up manually to point to the starting location in the host segment where data is to be transferred.
- The read operation will be automatically started if Auto-Read has been enabled via the Enable Auto-Read bit (reg. C2h, R/W, bit 5) prior to receipt of a Read Sectors command the Task File address registers of the read command match the Start registers (reg. ADh-B0h, R/W). This implies that the desired data has been cached in the buffer.
- The data must be cached in the Host segment, the Host Pointer must be pre-set to the starting location of the cached data, and the associated BCTR and BKMAX registers must have some positive number in them reflecting how many sectors are cached in the Host segment.
- The entire transfer will take place automatically since the Enable Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) is set. Data that is present in the buffer will automatically be taken out of the buffer and sent to the host without any microprocessor intervention.
- Both the ATA Task File registers and the Start registers will automatically be updated and the ATA inter-sector protocol will automatically be executed by the hardware.
- The Host Write Operation bit (reg. C3h, R/W, bit 7) is automatically cleared upon receipt of the command.
- The Host Transfer Count register (reg. D0h-D1h, R) indicates how many bytes of the current sector remain to be transferred between the Host and the Host FIFO.
- The Host Byte Counter (reg. 124h-125h, R/W) indicates the number of bytes remaining in the current sector to be transferred between the Host FIFO and the Buffer RAM.
- Unlike the write commands, the Read Sectors command will finish by having the Host take the last sector of data and requiring no further ATA handshaking. The microprocessor does not have to do anything after the last sector is taken. This can present problems in some applications since an entire Auto-Read Sectors command could be executed and the microprocessor might not be aware that it took place. The Enable Auto-Read Interlock bit (reg. C6h, R/W, bit 6) can be used to stop an Auto-Read Sectors command before the last sector is transferred. This provides the microprocessor with an interlock mechanism which ensures it knows that the Auto-Read Sectors command is taking place. The local microprocessor can then write a "1" to the Auto-Read Interlock Status/ Clear bit (reg. C6h, R/W, bit 2) before the last sector is encountered to prevent the pause from occurring. If this bit is not cleared before the last sector begins transferring, the operation will pause and the microprocessor will have to use the Restart at Transfer bit (reg. C3h, R/W, bit 1) to transfer the last sector.

3.2.14.2 Automated Read Sector(s)

- An automated Read Sector(s) command is one which the data is not in the buffer, an access off of the media is required, and the data transfer is handled automatically.
- PIO Transfer mode is used.
- The Host segment in the buffer is used for the transfer. This is determined via the Host Segment Control register (reg. 133h, R/W).
- The Host Pointer (reg. 130h-132h, R/W) is manually set to the beginning of the Host segment.
- The entire transfer will take place automatically since the Enable Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) is set. Data will automatically be taken out of the buffer and sent to the Host without any microprocessor intervention.
- Both the ATA Task File registers and the Start registers will automatically be updated and the ATA inter-sector protocol will automatically be executed by the hardware.
- The Host Write Operation bit (reg. C3h, R/W, bit 7) is automatically cleared upon receipt of the command.
- The Host Transfer Count register (reg. D0h-D1h, R) indicates how many bytes of the current sector remain to be transferred between the Host and the Host FIFO.
- The Host Byte Counter (reg. 124h-125h, R/W) indicates the number of bytes remaining in the current sector to be transferred between the Host FIFO and the Buffer RAM.
- Unlike the write commands, the Read Sectors command will finish by having the Host take the last sector of data and requiring no further ATA handshaking. The microprocessor does not have to do anything after the last sector is taken. The Read Interlock bit (reg. C6h, R/W, bit 7) can be used to stop an automated Read Sectors command before the last sector is transferred. This provides the microprocessor with an interlock mechanism that allow action to be taken if necessary before command is finished. The transfer will be paused before the last sector is transfered if this bit is set. However, if the local microprocessor clears the Read Interlock bit before the last sector transfer is started, the pause will not occur. The Restart AT Transfer bit (reg. C3h, R/W, bit 1) is used to restart the transfer of the last sector.

3.2.14.3 Manual Read Sector(s)

- A manual Read Sector(s) command is one which the microprocessor handles the inter-sector gap ATA protocol and manually updates the ATA Task File registers.
- PIO Transfer mode is used.
- The Host segment in the buffer is used for the transfer. This is determined via the Host Segment Control register (reg. 133h, R/W).
- The Host Pointer (reg. 130h-132h, R/W) is manually set to the beginning of the Host segment.
- The entire transfer will take place manually, one sector at a time, since the Enable Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) is not set.
- The transfer is restarted before every sector using the Restart AT Transfer bit (reg. C3h, R/W, bit 1).
- The ATA Task File register updates occur automatically at the inter-sector gap time if the Disable Automatic Command Block Update bit (reg. C0h, R/W, bit 6) is not set. If this bit is set, the microprocessor must perform the update completely manually or assert the Update Host Count bit (reg. C6h, R/W, bit 5) to initiate the update.

- The Host Write Operation bit (reg. C3h, R/W, bit 7) is automatically cleared upon receipt of the command.
- The Host Transfer Count register (reg. D0h-D1h, R) indicates how many bytes of the current sector remain to be transferred between the Host and the Host FIFO.
- The Host Byte Counter (reg. 124h-125h, R/W) indicates the number of bytes remaining in the current sector to be transferred between the Host FIFO and the Buffer RAM.

3.2.14.4 Manual Read Long

- A manual Read Long command is one which transfers data plus ECC bytes for the indicated disk address and the microprocessor handles the ATA protocol and manually updates the ATA Task File registers.
- PIO Transfer mode is used.
- The Host segment in the buffer is used for the transfer. This is determined via the Host Segment Control register (reg. 133h, R/W).
- The Host Pointer (reg. 130h-132h, R/W) is manually set to the beginning of the Host segment.
- The entire transfer will take place manually, the Enable Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) is ignored.
- The transfer is started by using the Start AT Transfer bit (reg. C3h, R/W, bit 2).
- The Read Long command is always a one sector transfer and therefore the Task File registers are never changed. The microprocessor must update the Status register (reg. 1F7h, R).
- *IOCS16 will automatically be negated during the transfer of the 8-bit ECC bytes.
- The Host Write Operation bit (reg. C3h, R/W, bit 7) is automatically cleared upon receipt of the command.
- The Host Transfer Count register (reg. D0h-D1h, R) indicates how many bytes of the current sector remain to be transferred between the Host and the Host FIFO. During "user data" time, this register will contain the remaining byte count for the 512-byte data field. During the "ECC" time, this register will contain the number ECC bytes remaining to be transferred. In addition, the number of ECC bytes of the current sector remaining to be transferred between the Host FIFO and the Buffer RAM is available in the Host ECC Count [5:0] field (reg. D3h, R, bits 5:0).
- The Host Byte Counter (reg. 124h-125h, R/W) indicates the number of bytes remaining in the current sector to be transferred between the Host FIFO and the Buffer RAM.
- The number of ECC bytes that will be transferred is governed by the host. This number should be known and set into the Host ECC Size [5:0] field (reg. D2h, R/W, bits 5:0) at initialization time.

3.2.14.5 Auto-Read Multiple

- PIO Transfer mode is used.
- The Host segment in the buffer is used for the transfer. This is determined via the Host Segment Control register (reg. 133h, R/W).
- The read operation will be automatically started if Auto-Read has been enabled via the Enable Auto-Read Multiple bit (reg. C2h, R/W, bit 1) prior to receipt of a Read Multiple command the Task File address registers of the Read Multiple command match the Start registers (reg. ADh-B0h, R/W). This implies that the desired data has been cached in the buffer.

- The data must be cached in the Host segment, the Host Pointer must be pre-set to the starting location of the cached data, and the associated BCTR and BKMAX registers must have some positive number in them reflecting how many sectors are cached in the Host segment.
- The entire transfer will take place automatically since the Enable Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) is set. Data will automatically be taken out of the buffer and sent to the host without any microprocessor intervention.
- Both the ATA Task File registers and the Start registers will automatically be updated on block boundaries and the ATA inter-sector protocol will automatically be executed by the hardware.
- The Host Write Operation bit (reg. C3h, R/W, bit 7) is automatically cleared upon receipt of the command.
- The Host Transfer Count register (reg. D0h-D1h, R) indicates how many bytes of the current sector remain to be transferred between the Host and the Host FIFO.
- The Host Byte Counter (reg. 124h-125h, R/W) indicates the number of bytes remaining in the current sector to be transferred between the Host FIFO and the Buffer RAM.
- Unlike the write commands, the Read Multiple command will finish by having the Host take the last sector of data and requiring no further ATA handshaking. The microprocessor does not have to do anything after the last sector is taken. This can present problems in some applications since an entire Auto-Read Multiple command could be executed and the microprocessor might not be aware that it took place. The Enable Auto-Read Interlock bit (reg. C6h, R/W, bit 6) can be used to stop an Auto-Read Multiple command before the last block (or partial block) is transferred. This provides the microprocessor with an interlock mechanism which ensures it knows that the Auto-Read Multiple command is taking place. The local microprocessor can then write a "1" to the Auto-Read Interlock Status/Clear bit (reg. C6h, R/W, bit 2) before the last block (or partial block) is encountered to prevent the pause from occurring. If this bit is not cleared before the last block begins transferring, the operation will pause and the microprocessor will have to use the Restart at Transfer bit (reg. C3h, R/W, bit 1) to transfer the last block.
- The Host Block Size register (reg. B6h, R/W) determines the number of sectors per block.

3.2.14.6 Manual Read Multiple

- A manual Read Multiple command is one in which the data is sent in blocks across the interface and the microprocessor handles the inter-sector gap ATA protocol and manually updates the ATA Task File registers.
- PIO Transfer mode is used.
- The Host segment in the buffer is used for the transfer. This is determined via the Host Segment Control register (reg. 133h, R/W).
- The Host Pointer (reg. 130h-132h, R/W) is manually set to the beginning of the host segment.
- The entire transfer will take place manually, one block at a time, since the Enable Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) is not set.
- The transfer is restarted before every block using the Restart AT Transfer bit (reg. C3h, R/W, bit 1).
- The ATA Task File register updates occur automatically at the inter-sector gap time if the Disable Automatic Command Block Update bit (reg. C0h, R/W, bit 6) is not set. If this bit is set, the microprocessor must perform the update completely manually or assert the Update Host Count bit (reg. C6h, R/W, bit 5) to initiate the update.

- The Host Write Operation bit (reg. C3h, R/W, bit 7) is automatically cleared upon receipt of the command.
- The Host Transfer Count register (reg. D0h-D1h, R) indicates how many bytes of the current sector remain to be transferred between the Host and the Host FIFO.
- The Host Byte Counter (reg. 124h-125h, R/W) indicates the number of bytes remaining in the current sector to be transferred between the Host FIFO and the Buffer RAM.
- The Host Block Size register (reg. B6h, R/W) determines the number of sectors per block.

3.2.14.7 Auto-Read DMA

- DMA Transfer mode is used. The PIO Transfer Mode Enable bit (reg. C5h, R/W, bit 3) will be ignored.
- The Host segment in the buffer is used for the transfer. This is determined via the Host Segment Control register (reg. 133h, R/W).
- The Host Pointer (reg. 130h-132h, R/W) must be set up manually to point to the starting location in the host segment where data is to be transferred.
- The read operation will be automatically started if Auto-Read has been enabled via the Enable Auto-Read DMA bit (reg. C2h, R/W, bit 3) prior to receipt of a Read DMA command and the Task File address registers of the read command match the Start registers (reg. ADh-B0h, R/W). This implies that the desired data has been cached in the buffer.
- The data must be cached in the Host segment, the Host Pointer must be pre-set to the starting location of the cached data, and the associated BCTR and BKMAX registers must have some positive number in them reflecting how many sectors are cached in the Host segment.
- The entire transfer will take place automatically since the Enable Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) is set. Data will automatically be taken out of the buffer and sent to the host without any microprocessor intervention.
- Both the ATA Task File registers and the Start registers will automatically be updated on sector boundaries even though it is a DMA command and not looked at by the host during the transfer.
- The Host Write Operation bit (reg. C3h, R/W, bit 7) is automatically cleared upon receipt of the command.
- The Host Transfer Count register (reg. D0h-D1h, R) indicates how many bytes of the current sector remain to be transferred between the Host and the Host FIFO.
- The Host Byte Counter (reg. 124h-125h, R/W) indicates the number of bytes remaining in the current sector to be transferred between the Host FIFO and the Buffer RAM.
- The Auto-Read DMA command will finish by having the host take the last sector of data and setting IRQ. The microprocessor does not have to do anything after the last sector is taken. This can present problems in some applications since an entire Auto-Read DMA command could be executed and the microprocessor might not be aware that it took place. The Enable Auto-Read Interlock bit (reg. C6h, R/W, bit 6) can be used to stop an Auto-Read DMA command before the last sector is transferred. This provides the microprocessor with an interlock mechanism which ensures it knows that the Auto-Read DMA command is taking place. The local microprocessor can then write a "1" to the Auto-Read Interlock Status/Clear bit (reg. C6h, R/W, bit 2) before the last sector is encountered to prevent the pause from occurring. If this bit is not cleared before the last sector begins transferring, the operation will pause and the microprocessor will have to use the Restart at Transfer bit (reg. C3h, R/W, bit 1) to transfer the last sector.

- The hardware will automatically pause and restart the transfer based on buffer full/empty status using the DMAREQ/*DMACK handshake used for DMA transfers.

3.2.14.8 Manual Read DMA

- DMA Transfer mode is used. The PIO Transfer Mode Enable bit (reg. C5h, R/W, bit 3) will be ignored.
- The Host segment in the buffer is used for the transfer. This is determined via the Host Segment Control register (reg. 133h, R/W).
- The Host Pointer (reg. 130h-132h, R/W) must be set up manually to point to the starting location in the host segment where data is to be transferred.
- The entire transfer will take place manually since the Enable Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) is not set.
- The transfer is restarted before every sector using Start AT Transfer bit (reg. C3h, R/W, bit 2), rather than the Restart AT Transfer bit, since IRQ assertions are not wanted.
- The ATA Task File registers do not need to be updated between each sector; only at the end of the command. These registers will be updated if the DISAUTOUPD bit (reg. C0h, R/W, bit 6) is set.
- The Host Write Operation bit (reg. C3h, R/W, bit 7) is automatically cleared upon receipt of the command.
- The hardware will automatically pause and restart the transfer based on buffer full/empty status using the DMAREQ/*DMACK handshake used for DMA transfers.
- The Host Transfer Count register (reg. D0h-D1h, R) indicates how many bytes of the current sector remain to be transferred between the Host and the Host FIFO.
- The Host Byte Counter (reg. 124h-125h, R/W) indicates the number of bytes remaining in the current sector to be transferred between the Host FIFO and the Buffer RAM.
- The Internal Busy bit (reg. C1h, R, bit 7) and the AT Busy bit must be cleared by the microprocessor setting the Reset Busy bit (reg. C1h, R/W, bit 6). The microprocessor must conclude the command by asserting IRQ.

3.2.14.9 Read Verify

- The Read Verify command can be executed in several ways. Firstly, the sector can be read off the disk and just checked for any ECC errors. Secondly, the data can be read off the disk without correction turned on and compared byte by byte to a sector in the buffer. Thirdly, data can be read off the disk with ECC checking turned on and a byte by byte compare will take place with any read incorrect bytes being internally corrected prior to the compare.
- No data will be transferred to the Host. The ATA Task File registers can be optionally set up to automatically update for every sector verified. In this case, upon completion of the command, they point to the location of the last sector that was verified. This feature is invoked via the Enable Read Verify Command Automatic Task File Update bit (reg. C0h, R/W, bit 5).
- Refer to the ATA specification on Read Verify Command protocol.

3.3 Data Buffer Interface and Control

The data buffer control circuitry is important in that it is the focal control for all transfers occurring between the disk and the host. This block of logic works with various control and status signals of the host and disk blocks to manage the flow of data between the host and the disk.

The buffer block may be used in a completely automated mode or in various lesser automated modes and manual mode. This flexibility allows the designer to utilize the device in the mode which best suits the application.

3.3.1 Buffer Block Architecture

The buffer block's primary function is to manage the flow of data into and out of the buffer for all read/write command execution. Other functions of this block, which may not be readily apparent are as follows:

- Providing a means of storing Split Data counts in the buffer and handshaking with the Disk block for the retrieval of these bytes.
- Providing a means of storing Sector Skip information in the buffer and handshaking with the Disk block for retrieval of this information.
- Managing segmentation and the various address pointers for each segment.
- Arbitrating all accesses of the buffer such that there is no contention problem.
- Interfacing with the ECC correction logic so that data can be corrected in the buffer.

The functional block diagram of the Buffer block is shown in Figure 3-12.

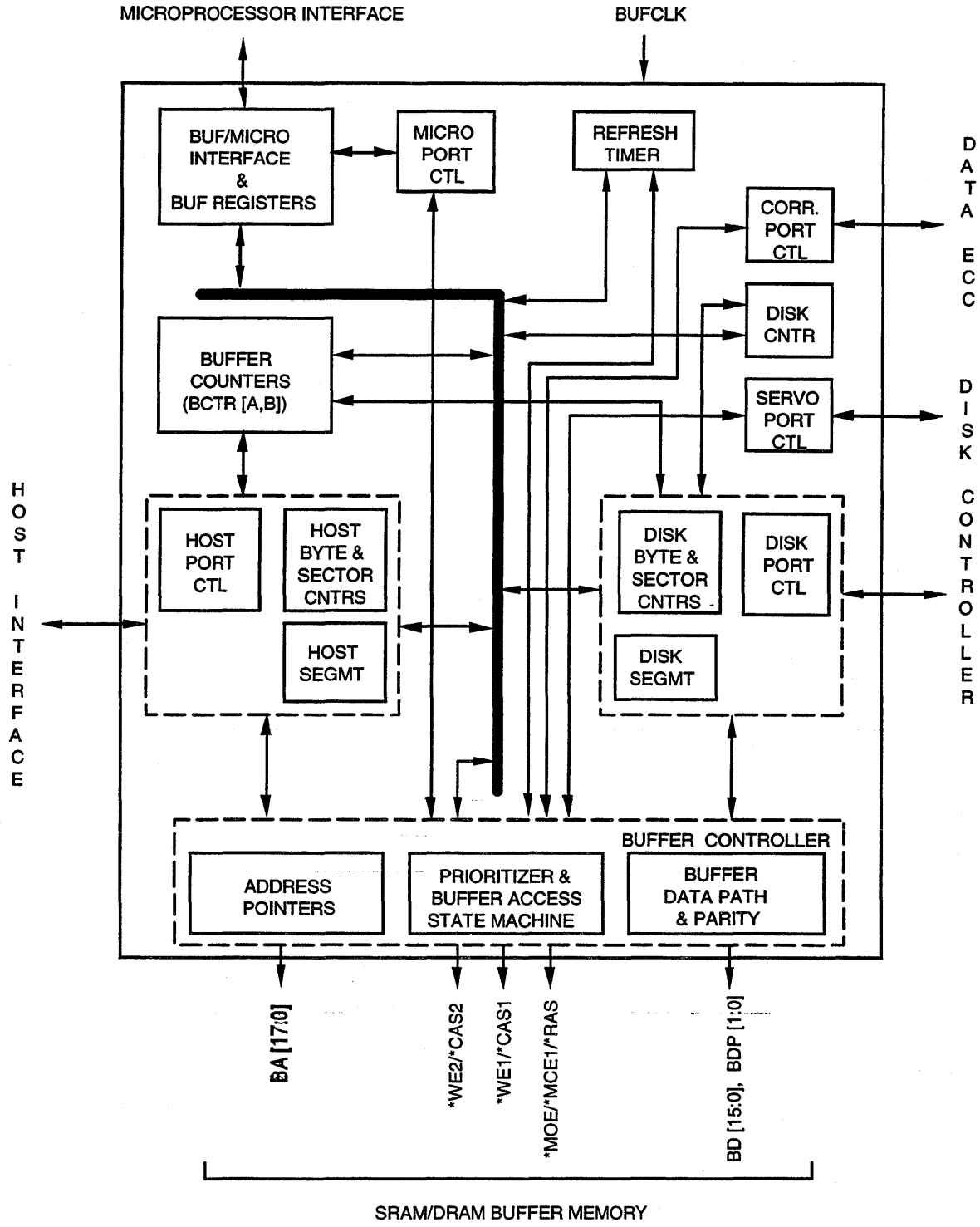

Figure 3-12 Buffer Block Architecture

3.3.2 Buffer Segmentation

The buffer can be logically divided into any number of segments. At any given time, three segments (Segment A, Segment B, Servo Segment) can be controlled automatically via hardware. Segment A and Segment B can be configured as an Auto-Write Host segment, a Host segment (used as an Auto-Read Host segment, a non-Auto-Write Host segment or a non-Auto-Read Host segment), or a Disk segment via a set of control registers. These segments can be adjacent to each other, overlapping, or separated by bytes not included in any segment. The Servo segment is utilized for implementing split servo control out of the buffer and also for headerless format management.

The local microprocessor can access any of these segments through the 2K-byte page window. Any non-segmented area of the buffer may be used as a scratchpad area for the microprocessor.

The Segment Control Structure:

The various Host, Disk, and Servo segments, are established and controlled via a set of registers. Table 3-14 summarizes the various registers that are used to configure the three types of segments.

Table 3-14 Registers Used To Configure Segments

Register or Bit Name	Location	Resident Block
Servo Page registers	112h, 113h, R/W	Buffer
Servo End of Segment registers	11Ch, 11Dh, 11Eh,11Fh,R/W	Buffer
Host Segment Control register	133h, R/W	Buffer
Disk Segment Control register	137h, R/W	Buffer
Begin of Segment A registers	138h, 139h, 13Ah, R/W	Buffer
End of Segment A registers	13Ch, 13Dh, 13Eh, R/W	Buffer
Begin of Segment B registers	140h, 141h, 142h, R/W	Buffer
End of Segment B registers	144h, 145h, 146h, R/W	Buffer

The Host Segment:

The Host Segment is used to transfer data to or from the host and the buffer memory while a non-Auto-Write command is executed. The Host Segment can be operated as a linear non-wrapping segment or as a circular buffer with full pointer wrapping and full/empty status capabilities. The attributes and location of the Host segment must be established prior to using the segment.

The attributes of the Host segment are established via three control bits. The Host Segment Select bit (reg. 133h, R/W, bit 2) is used to establish whether Segment A or Segment B is used for the Host segment. The Host BCTR Select bit (reg. 133h, R/W, bit 1) is used to select either BCTRA or BCTRB as the Buffer Counter for the Host segment. The Enable Host Room Logic bit (reg. 133h, R/W, bit 0) enables the use of the buffer full/empty logic for the Host segment.

For a non-Auto-Write command, the Auto-Write Select bit (reg. 10Bh, R/W, bit 0) should be in the cleared state. The state of this bit determines which set of control bits are used to configure the segment for the Buffer/Host transfer. When set to 0, the HSEGSEL, HBCTRSEL, and ENHROOM bits (reg. 133h, R/W, bits 2:0) will be used to configure the segment.

The location and extent of the Host segment is determined by the Beginning Of Segment and End Of Segment Pointer registers for the associated segment (Segment A or B).

The Host Pointer (reg. 130h-132h, R/W) is used by the Host interface to access data within the Host segment. For any non-Auto-Write command, the existing value of the Host Pointer is used as the starting

address of the transfer; it is not automatically modified by the hardware.

Refer to Table 3-14 for a summary of the registers and register bits used for establishing the Host segment.

The Auto-Write Host Segment:

The Auto-Write Host Segment is used to transfer data from the host to the buffer during execution of an Auto-Write command. The Auto-Write Host Segment can be operated as a linear non-wrapping segment or as a circular buffer with full pointer wrapping and full/empty status capabilities. The attributes and location of the Auto-Write Host segment must be established prior to using the segment.

The attributes of the Auto-Write Host segment are established via five control bits. The first three control bits are equivalent to the three bits used for establishing the Host segment. The Auto-Write Segment Select bit (reg. 133h, R/W, bit 5) is used to establish whether Segment A or Segment B is used for the Auto-Write Host segment. The Auto-Write BCTR Select bit (reg. 133h, R/W, bit 4) is used to select either BCTRA or BCTRB as the Buffer Counter for the Auto-Write Host segment. The Enable Auto-Write Room Logic bit (reg. 133h, R/W, bit 3) enables the use of the buffer full/empty logic for the Auto-Write Host segment.

There are two additional control bits used for the Auto-Write Host segment which are not used for the Host segment. They are the Auto-Write Mode Select bits (reg. 133h, R/W, bits 7:6) and the Auto-Write Select bit (reg. 133h, R/W, bit 0).

The Auto-Write Mode Select bits (reg. 133h, R/W, bits 7:6) are used to establish what value gets loaded into the Host Pointer at the beginning of the Auto-Write command.

The Auto-Write Select bit (reg. 10Bh, R/W, bit 0) is automatically set upon receipt of an Auto-Write command. The state of this bit determines which set of control bits are used to configure the segment for the Buffer/Host transfer. When set to 1, the AWSEGSEL, AWBCTRSEL, and ENAWROOM bits (reg. 133h, R/W, bits 5:3) will be used to configure the segment.

The location and extent of the Auto-Write Host segment is determined by the Beginning Of Segment and End Of Segment Pointer registers for the associated segment (Segment A or B).

The Host Pointer (reg. 130h-132h, R/W) is used by the Host interface to access data within the Auto-Write-Host segment. The Host Pointer can potentially be automatically modified at the start of an Auto-Write command. The Auto-Write Mode Select bits (reg. 133h, R/W, bits 7:6) are used to establish what value gets loaded into the Host Pointer at the beginning of the Auto-Write command. If the value of this AWMODE-field is [00], the Host Pointer will not be modified and the write accesses will begin at the address currently pointed to. If the AWMODE field is [01], the Host Pointer is loaded with the value in the chosen Beginning Of Segment Pointer. If the AWMODE field is [10], the Host Pointer will be loaded with the Write Cache Pointer (reg. 148h-14Ah, R/W).

The Write Cache Pointer is used to provide a dedicated address pointer for write cache implementations. This eliminates the problems associated with using only one host pointer and the problems associated with having to manage it via firmware.

The Write Cache Pointer will only be updated on sector boundaries. This prevents starting any next transfer at a non-sector boundary in the event the previous sector transfer was aborted at other than a sector boundary.

During an Auto-Write command transfer, while the Auto-Write Select bit (reg. 10Bh, R/W, bit 0) is set, the Host Pointer registers (reg. 130h-132h, R/W) and the Host Byte Counter Clear bit (reg. 108h, R/W, bit 6) are prevented from being written to.

Refer to Table 3-14 for a summary of the registers and register bits used for establishing the Auto-Write Host segment.

The Disk Segment:

The Disk Segment is used to transfer data between the buffer and the Disk. The Disk Segment can be operated as a linear non-wrapping segment or as a circular buffer with full pointer wrapping and full/empty status capabilities. The attributes and location of the Disk segment must be established prior to using the segment. The Disk Pointer (reg. 134h-136h, R/W) is used by the Disk FIFO to access the Disk Segment.

The attributes of the Disk segment are established via three control bits. The Disk Segment Select bit (reg. 137h, R/W, bit 2) is used to establish whether Segment A or Segment B is used for the Disk segment. The Disk BCTR Select bit (reg. 137h, R/W, bit 1) is used to select either BCTRA or BCTRB as the Buffer Counter for the Disk segment. The Enable Disk Room Logic bit (reg. 137h, R/W, bit 0) enables the use of the buffer full/empty logic for the Disk segment.

The location and extent of the Disk segment is determined by the Beginning Of Segment and End Of Segment Pointer registers for the associated segment (Segment A or B).

The Disk Pointer (reg. 134h-136h, R/W) is used by the Disk block to access data within the Disk segment. The existing value of the Disk Pointer is used as the starting address of the transfer; it is not automatically modified by the hardware.

Refer to Table 3-14 for a summary of the registers and register bits used for establishing the Host segment.

The Servo Segment:

The Servo Segment is used to store the Data Split values used with disk formats that require servo bursts which split the data field or ECC field. It is also used to store sector skip information for headerless support.

The Servo segment base location is established by the Servo Page registers (reg. 112h-113h, R/W). These registers provide the upper nine bits, VPAGE[21:13], which establish which 8K area of the buffer the Servo segment is located within.

The Servo Beginning Of Segment registers (reg. 11Ch-11Dh, R/W), VBOS[12:0], provide the location within the 8K area where the Servo segment begins.

The Servo End Of Segment registers (reg. 11Eh-11Fh, R/W), VEOS[12:0], establishes the end of the Servo segment within the 8K area.

The Servo segment is accessed via the Servo Pointer register (reg. 12Ch, 12Dh, R/W). The Servo Pointer is actually used as an offset pointer into a segment whose base address is determined by the Servo Page registers (reg. 112h, 113h, R/W). When the Servo Pointer reaches the End Of Servo Segment value, it wraps back to the Beginning Of Servo Segment value.

The Servo Segment is typically accessed by the microprocessor or by the Disk Sequencer logic. The Disk Sequencer initiates fetches to this segment at specific times during the reading or writing of a sector in order to retrieve the Data Split values and skip sector control information associated with that sector. Refer to Section 3.5 for detailed information on this process.

The Host CIS Segment:

The Host CIS Segment is not a true segment in that there is no Pointer or Beginning or End of Segment Pointers associated with it. The Host CIS segment exists only while operating the device in PCMCIA mode and is the lowest 256 bytes of the buffer RAM. It contains the Card Information Structure (CIS) that is read by the PCMCIA Host. This information is loaded into the buffer RAM via the local microprocessor during initialization. While operating in PCMCIA mode, care must be taken to not overlap any other segment over this 256 byte area. This area is fixed at location 0 in the Buffer memory map.

Buffer Counter Registers and Buffer Segment Size:

The Buffer Counter registers are used by the buffer automation to control the transfers of sectors into and out of the buffer segments. There are two Buffer Counter registers; BCTRA (reg. 114h, 115h, R/W) and BCTRB (reg. 118h, 119h, R/W). The selected BCTR is incremented whenever a sector is written to the disk or whenever a good sector has been read from the disk. A good sector is defined as a sector that is error free or one that has been corrected in the buffer. The selected BCTR is decremented whenever a sector is sent to or received from the host.

The Maximum Buffer Count registers are used by the buffer automation to manipulate the size of the working area within a segment. This working area can be thought of as a revolving area within the segment area. There are two Maximum Buffer Count registers; BKMAXA (reg. 116h, 117h, R/W) and BKMAXB (reg. 11Ah, 11Bh, R/W). Each of the BKMAX registers is used with its respective BCTR register.

3.3.3 Host Port Transfers

The Host Port control block handles the data transfers between the buffer RAM and the host FIFO. It will manage both 8-bit wide and 16-bit wide transfers. The direction of the data transfer is determined by the host block and provided to the Buffer Access State Machine.

The Host Port makes requests to the Prioritizer for a buffer read or write access when all of the following conditions have been met:

- The Host Port must be enabled by setting the Host Port Enable bit (reg. 109h, R/W, bit 0).
- The HFAVAIL condition is true. That is, the Host FIFO is available for transferring data to/from the buffer, i.e., a read or writer transfer is enabled in the host block and the Host FIFO is not full (if reading from the buffer) or empty (if writing to the buffer).

The following conditions will disable Host Port requests being sent to the Prioritizer:

- An error occurs which sets the Host Port Check Error bit (reg. 106h, R, bit 0) while the Disable Host Error Stop bit (reg. 105h, R/W, bit 2) is not set.

Setting The Sector Size:

The number of data bytes in a sector is established by the Sector Size registers (reg. 120h, 121h, R/W). This value should only reflect the number of data bytes in the block and must exclude the ECC bytes.

The Host Byte Counter:

The Host Byte Counter is an 11-bit down counter that is used to keep track of the number of bytes within a sector that are transferred between the buffer RAM and the host FIFO. It can be read via the Host Byte Count 0 and 1 registers (reg. 124h, 125h, R).

The Host Byte Counter is initialized to zero when the microprocessor sets the HBCCLR bit (reg. 108h, R/W, bit 6). It is also automatically cleared at the start of an Auto-Write command.

When the first word of a sector is transferred between the host FIFO and the buffer, the Host Byte Counter will wrap to a value that corresponds to the number of data bytes remaining to be transferred for that sector. That value is a function of the BEN16BIT bit (reg. 100h, R/W, bit 2). This relationship is summarized in Table 3-15.

Table 3-15 HBC Wrap Value Selection

BEN16BIT (Reg. 100h, R/W, bit 2)	HBC Wrap Value
0	SECSIZE - 1
1	SECSIZE - 2

After the Host Byte Counter wraps, it decrements by 1 (8-bit mode) or by 2 (16-bit mode) for every word transferred. It will become zero when the entire sector has been transferred between the buffer Ram and the host FIFO. In 16-bit mode, if the sector contains an odd number of bytes, the Host Byte Counter will decrement from 1 to 0 when the last byte of the sector is transferred.

The Host Byte Counter is used to keep track of sector boundaries and is also used for auto decrementing the selected BCTR for the Host Port, if any, for every good sector transferred between the buffer RAM and the Host FIFO.

The Host Transfer Count Register:

The Host Transfer Counter (regs. D0h, R, bits 7:0, and D1h, R, bits 1:0) is a 10-bit counter that indicates the number of bytes remaining to be transferred between the Host FIFO and the Host for the current sector transfer. The Host Transfer Counter is automatically loaded with the contents of the Host Byte Counter (reg. 124h-125h, R) immediately before the Host transfer is started.

During execution of a Read Long or Write Long command, this counter is automatically loaded with the contents of the ECC Size register (reg. D2h, R/W) after the data portion of the transfer has concluded. This value will now indicate the number of ECC bytes remaining to be transferred between the Host FIFO and the Host. When this has occurred, the HECCTIME bit (reg. D1h, R, bit 7) will be set and the Host Transfer Count register will now indicate the ECC transfer count. For every ECC byte transferred, the value in the host Transfer Count register will decrement.

The Host Block Size Register:

The Host Block Size register (reg. B6h, R/W) is loaded by the microprocessor to indicate the number of sectors per block that will be used for any subsequent Read Multiple or Write Multiple commands.

The Host Block Count Register:

The Host Block Count register (reg. B7h, R) is an 8-bit counter that counts the Host sectors within a block during execution of a Write Multiple or Read Multiple command. This counter will be loaded with the contents of the Host Block Size register (reg. B6h, R/W) at the beginning of a Host block transfer during a Read Multiple or Write Multiple transfer. It will decrement for every sector within the block that is transferred.

There are various key registers and associated bits which pertain to transfers between the Host FIFO and the buffer RAM which were not presented in Table 3-14. These registers and bits are the ones most frequently utilized for these transfers. They are summarized in Table 3-16.

Table 3-16 Key Registers and Bits used to Manage Host/Buffer Transfers

Register or Bit Name	Location	Resident Block
Host Port Check Error bit	106h, R, bit 0	Buffer
Disable Host Error Stop bit	105h, R/W, bit 2	Buffer
Host No Room bit	10Bh, R, bit 6	Buffer
Host Port Enable bit	109h, R/W, bit 0	Buffer
Sector Size Size registers	120h, 121h, R/W	Buffer
Host Byte Counter Clear bit	108h, R/W, bit 6	Buffer
Host FIFO Reset bit	C6h, R/W, bit 3	Host
Host Block Size register	B6h, R/W	Host
Host Block Counter	B7h, R	Host
Host Write Operation bit	C3h, R/W, bit 7	Host
Enable Auto Pause On Error bit	C3h, R/W, bit 4	Host
Auto-Write Overrun Error bit	CAh, R, bit 3	Host
Host FIFO Error bit	C8h, W, bit 1	Host
Host FIFO Byte Count [5:0] bits	CFh, R, bits 5:0	Host
Host Transfer Count 0 register	D0h, R	Host
Host Transfer Count 1 register	D1h, R, bits 1:0	Host
Host Pointer registers	130h-132h, R/W	Buffer
Enable 16-Bit Buffer bit	100h, R/W, bit 2	Buffer

There are several automated methods of managing the transfer between the buffer and the Host FIFO. One method uses complete automation via the ADFM logic and the other is the manual method.

Automated Methods:

The transfers between the Host and Host FIFO will be started automatically if the appropriate Auto-Write or Auto-Read Enable bits are set at the time the ATA command is received. The Host Write Operation bit (reg. C3h, R/W, bit 7) will automatically be set to the proper state to establish the direction of the transfer only if the command decode is a recognized Read or Write command. The transfer will begin automatically provided the Host Port Enable bit (reg. 109h, R/W, bit 0) is set along with the other corresponding buffer conditions that indicate the buffer is ready to transfer data. The transfer will be sustained beyond the first sector (or block) provided the Enable Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) is set.

Manual Method:

The buffer/host path can also be managed manually by the local microprocessor. In this mode, automation of the Host interface as well as the host/buffer path is disabled. In this case, all Auto-Write Enable and Auto-Read Enable bits as well as the Enable Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) are cleared. The microprocessor must manage and control various signals between the individual sector (or block) transfers. The Host Write Operation bit (reg. C3h, R/W, bit 7) must be set or cleared to establish the direction of the transfer only if any of the command decodes are not recognized by the AT Command register (Reg, 1F7h, W). The buffer is enabled for transferring when the microprocessor sets Host Port Enable (reg. 109h, R/W, bit 0). The actual transfer is initiated by setting the Start AT Transfer bit (reg. C3h, R/W, bit 2) or the Restart AT Transfer bit (reg. C3h, R/W, bit 1). The microprocessor controls the transfer of each sector (or block) by asserting and negating the various ATA interface signals at their proper times.

3.3.4 Disk Port Transfer

The Disk Port control block handles the data transfers between the buffer RAM and the Disk FIFO. It will manage both 8-bit wide and 16-bit wide transfers. The direction of the data transfer is determined by the Disk block and provided to the Buffer Access State Machine.

The Disk Port makes requests to the Prioritizer for a buffer read or write access when all of the following conditions have been met:

- The Disk Port must be enabled by setting the Disk Port Enable bit (reg. 109h, R/W, bit 1).
- The Disk FIFO is requesting a transfer of bytes (or words) to/from the buffer.

The following conditions will disable Disk Port requests being sent to the Prioritizer:

- An error occurs which sets the Disk Port Check Error bit (reg. 106h, R, bit 1) while the Disable Disk Error Stop bit (reg. 105h, R/W, bit 3) is not set.
- The Disk No Room bit (reg. 10Bh, R, bit 7) becomes set during a write or verify operation.

Setting The Sector Size:

The number of data bytes in a sector is established by the Sector Size registers (reg. 120h, 121h, R/W). This value should only reflect the number of data bytes in the block and must exclude the ECC bytes.

The Disk Byte Counter :

The Disk Byte Count register is an 11-bit down counter that is used to keep track of the number of bytes within a sector that are transferred between the buffer RAM and the Disk FIFO. It can be read via the Disk Byte Count 0 and 1 registers (reg. 126h, 127h, R).

The Disk Byte Count registers are initialized to zero when the microprocessor sets the DBCCLR bit (reg. 108h, R/W, bit 5). The Disk FIFO will also be cleared when the DBCCLR bit is set.

When the first word of a sector is transferred between the disk and the buffer, the Disk Byte Count register will wrap to a value that corresponds to the number of bytes remaining to be transferred for that sector. That value is a function of the BEN16BIT bit (reg. 100h, R/W, bit 2). This relationship is summarized in Table 3-17.

Table 3-17 DBC Wrap Value Selection

BEN16BIT (Reg. 100h, R/W, bit 2)	DBC Wrap Value
0	SECSIZE - 1
1	SECSIZE - 2

After the first word of a sector is transferred, the Disk Byte Count register decrements by 1 (8-bit mode) or by 2 (16-bit mode) for every word transferred. It will become zero when the entire sector has been transferred between the buffer Ram and the Disk FIFO.

The Disk Byte Count register is used to keep track of sector boundaries for the Correction Port which performs ECC correction on data in the buffer.

The Disk Byte Count and Disk Sector Count registers are used together to keep track of data block boundaries and are used for auto incrementing the selected BCTR for the Disk Port, if any, for every good data block transferred between the buffer RAM and the Disk FIFO.

There are various key registers and associated bits which pertain to transfers between the Disk FIFO and the buffer RAM. These registers and bits are the ones most frequently utilized for these transfers. They are summarized in Table 3-18.

Table 3-18 Key Registers and Bits used to Manage Disk/Buffer Transfers

Register or Bit Name	Location	Resident Block
Disk Port Check Error bit	106h, R, bit 1	Buffer
Disable Disk Error Stop bit	105h, R/W, bit 3	Buffer
Disk No Room bit	10Bh, R, bit 7	Buffer
Disk Port Enable bit	109h, R/W, bit 1	Buffer
Buffer Block Size registers	120h, 121h, R/W	Buffer
Disk Byte Counter Clear bit	108h, R/W, bit 5	Buffer
Disk Block Counter Clear bit	108h, R/W, bit 4	Buffer
Sector Ok bit	66h, R/W, bit 1	Disk
Disk Byte Counter	126h, 127h, R	Buffer
Reset Disk FIFO bit	60h, R/W, bit 6	Disk
Disable FIFO Flush bit	61h, R/W, bit 6	Disk
Stop On Disk/Buffer Data Path Error bit	7Eh, R/W, bit 5	Disk
Disk Pointer registers	140h-142h, R/W	Buffer
Disk FIFO Overrun/Underrun Error bit	7Dh, R, bit 7	Disk
Disk FIFO Count[5:0]	7Dh, R, bits 5:0	Disk
Disk Up Counter 0, 1	12Eh, 12Fh, R/W	Buffer
Enable Disk Up Counter bit	102h, R/W, bit 7	Buffer
Enable 16-Bit Buffer bit	100h, R/W, bit 2	Buffer
Disk Sector Counter	10Dh, R	Buffer
Disk Sector Counter Done bit	103h, R, bit 7	Buffer
Enable Disk Sector Counter Done Interrupt	104h, R/W, bit 7	Buffer

The Disk Sector Counter:

The Disk Sector Counter (reg. 10Dh, R) is useful for automatically keeping track of the number of sectors transferred between the buffer and the disk. The Disk Sector Counter starts counting only after it is loaded with a non-zero value.

During a disk write operation, the counter decrements when the last byte of a sector is transferred from the buffer into the Disk FIFO. This will occur when the Disk Byte Count register (regs. 126h, 127h, R/W) decrements to 0.

During a disk read or verify operation, the Disk Sector Counter decrements after an error-free sector is read from the disk or after a sector with an error has been automatically corrected in the buffer.

Once it reaches zero, a Disk Counter Done interrupt (reg. 103h, R, bit 7) can be generated if enabled by the Enable Disk Sector Counter Interrupt bit (reg. 104h, R/W, bit 7).

The Disk Up Counter:

The Disk Up Counter (reg. 12Eh, 12Fh, R/W) is a 10-bit up counter which is also used to keep track of the sectors that have been put into the buffer which require no ECC correction or which have been successfully corrected. This counter is enabled for use via the ENDUCTR bit (reg. 102h, R/W, bit 7). Whenever the ENDUCTR bit is cleared, the Disk Up Counter will be set to 00h.

Disk Block Counter/Delayed Block Release:

The Disk Block Counter (reg. 10Eh, R) can be used as a mechanism for controlling the release of sectors to the Host. During the time while the Enable Delayed Block Release bit (reg. 108h, R/W, bit 7) is set, the Disk Block Counter will be incremented for every "good" sector read from the disk. During this time, the associated BCTR will not be incremented. During this time, a count of unreleased sectors will be accumulating in the Disk Block Counter register. When appropriate, these sectors can be released to the Host by setting the Release Blocks To Host bit (reg. 108h, R/W, bit 3). Setting this bit will cause the count stored in the Disk Block Counter to be added to the value in the associated BCTR.

3.3.5 Correction Port Transfers

The Correction Port is used by the EDAC circuitry to access the buffer during ECC correction time. The Correction Port Enable bit (reg. 109h, R/W, bit 2) must be set to allow the Correction Port accesses to take place. A sector that is read from the disk may have multiple bytes in error that can randomly be located within the sector. The actual error correction is performed though up to nine read/modify/write operations to the buffer RAM (Three errors per interleave). The EDAC block makes a correction request for each byte to be corrected. The information provided to the Correction Port control logic at the time of the request includes the relative location of the byte within the sector, and the error pattern to exclusive OR with that byte in order to correct it.

The Correction Port control will read the location and then write back the modified value to the same location. For each correction request received from the EDAC block, the Correction Port logic will generate the buffer pointer address for the corresponding location in the buffer and submit a read request to the Prioritizer. After the Correction Port wins priority for the access, the byte is read from the buffer and an acknowledge is received from the Buffer Access State Machine along with the data byte. The data is XORed with the error pattern from the EDAC block and a write request is made to the Prioritizer. After winning priority, the modified byte is written back to the same location in the buffer and acknowledged. The correction is now complete and an acknowledge is generated to the EDAC block.

If the buffer RAM is configured as a 16-bit buffer, an entire word will be read from the buffer. Only the byte requiring correction will be modified before the entire word is written back to the buffer.

If a parity error is detected during the read cycle, while the Enable Buffer Parity Checking bit (reg. 105h, R/W, bit 0) is set, the correction process is stopped and the Correction Port Check Error bit (reg. 106h, R, bit 2) is set and will generate an interrupt if desired.

The correction process must be complete before the last byte of the next sector is written into the buffer. If this timing is violated, the Correction Port Overrun Error bit (reg. 103h, R, bit 2) will be set and an interrupt will be generated if enabled. The correction process will be halted if this condition occurs.

3.3.6 Servo Port Transfers

The Servo Port control block manages requests from the Disk block to read the Data Split values from the Servo Segment. For this function to occur, the Servo Port Enable bit (reg. 109h, R/W, bit 4) must be set.

The Disk block will request a number of bytes be fetched from the Servo segment. This is initiated by and sometime after the Load CDR FIFO decode (SEQCTLC = '001') is executed.

The Servo Page register (reg. 112h, 113h, R/W) will be used as the base address for the access. This base address has a granularity of 8 Kbyte boundaries. The offset into the selected Servo page will come directly from the Servo Pointer (reg. 12Ch, 12Dh, R/W) which was loaded earlier by the microprocessor.

The instruction which contains Load CDR FIFO decode also contains a byte count in the SEQCNT field. It is this value which determines how many bytes are to be read from the Servo segment. Each byte will be sequentially read from the buffer. If the Servo Pointer is incremented from the value in the Servo Segment End Of Segment Pointer (reg. 11E-11Fh, R/W), the Servo Pointer will wrap to the value in the Servo Segment Beginning Of Segment Pointer (reg. 11Ch-11Dh, R/W).

3.3.7 Host CIS Port Transfers

The Host CIS Port control block manages requests from the Host/PCMCIA block to read the PCMCIA Card Information Structure (CIS) from the Buffer. The lowest 256 bytes of the buffer RAM are used to store this information. This area of the buffer is not protected and care must be taken so as not to include this area as part of any other segment while operating the device in the PCMCIA environment. The Host CIS Port is enabled by setting the Host CIS Port Enable bit (reg. 109h, R/W, bit 5).

A CIS read is initiated by a request from the Host block accompanied by an 8-bit address that selects the byte to be read. When the CIS Port wins priority, it reads the appropriate byte from the buffer and returns it to the Host.

3.3.8 Buffer Access Priority

Access of the buffer is controlled by a prioritizer circuit. The prioritizer keeps track of which port is currently using the buffer and which ports have requested access. It resolves any simultaneous requests with the higher priority port receiving the access privilege.

The Servo port has the highest priority since it is involved with retrieving the Split Data values out of the buffer in a very small window and must get service quickly to ensure that the CDR FIFO have the correct values in it prior to the next Data Sync event.

The buffer priority ordering can be changed in several ways. The Enable High Priority For Host Port bit (reg. 100h, R/W, bit 1) can be used to dynamically alter the priority structure based upon how many bytes are in the Host FIFO. During certain times, the Host Port will be given higher priority. During transfers from the Host to the buffer, a higher priority will be established when the Host FIFO has more than 16 bytes of data. During transfers from the buffer to the Host, a higher priority will be established when the Host FIFO has less than 16 bytes of data.

In order for an access to be granted, the requesting Port must have the corresponding Port Enable bit set in the Buffer Port Enable register (reg. 109h, R/W).

Buffer Access priority is summarized in Table 3-19.

Table 3-19 Buffer RAM Access Priorities

Priority	Normal Order	Modified Order
HIGHEST	Servo	Servo
↑	Disk	Disk
	Refresh	Refresh
	Correction	Host High
	Microprocessor	Correction
	Host	Microprocessor
↓	Host CIS (PCMCIA)	Host Low
LOWEST		Host CIS (PCMCIA)

The amount of time the microprocessor is held up in the buffer read/write process is a function of when the access request occurs, what other higher priority accesses are pending, and the available buffer bandwidth as determined by BUFCLK and possibly RRCLK.

3.3.9 Buffer Automation

The buffer block is capable of automatically monitoring and servicing concurrent buffer RAM accesses from all the different Ports. In addition to scheduling these accesses, the buffer block takes care of automated tasks such as buffer full/empty throttling and pointer wrapping.

Segmentation:

The framework for buffer automation usually begins with the establishment of a buffer segmentation scheme. There are four formal segments that are used in the AIC-8375. They are the Auto-Write Host segment, the Host segment, the Disk segment, and the Servo segment.

Typically, the Auto-Write segment is dedicated for use as the default Auto-Write segment. This is done to create a safe area in the buffer which can immediately accept data from the Host during an Auto-Write command. A dedicated segment in the buffer protects any other data in the buffer from being overwritten. The location of the Auto-Write segment is determined at the time the Auto-Write command is received. Upon receipt of the Auto-Write command, the Auto-Write Select bit (reg. 10Bh, R/W, bit 0) is set indicating that this type of a command has been received. While this bit is set, control bits in the Host Segment Control register (reg. 133h, R/W) define how the Auto-Write segment is set up. The Auto-Write Segment Select bit (reg. 133h, R/W, bit 5) determines whether the Segment A Pointers or the Segment B Pointers are to be used for the Auto-Write segment. The BCTR used for the Auto-Write transfer (BCTRA or BCTRB) is automatically selected via the Auto-Write BCTR Select bit (reg. 133h, R/W, bit 4). The initial value of the Host Pointer (reg. 130h-132h, R/W) at the start of the transfer is determined by the Auto-Write Mode bits (reg. 133h, R/W, bits 7:6). Dependent upon the state of these bits at the beginning of the Auto-Write command execution, the Host Pointer can be loaded with the contents of the selected Beginning of Segment Pointer, the Write Cache Pointer (reg. 148h-14Ah, R/W), or can remain unchanged. The Enable Host Room Logic bit (reg. 133h, R/W, bit 0) must be set if automatic buffer full/empty throttling is desired.

The Host Pointer will increment after each byte or word is written into the buffer. The Write Cache Pointer is loaded with the contents of the Host Pointer after each sector has been written into the buffer. This prevents the Write Cache Pointer from ending up on a non-sector boundary in the event the transfer is aborted. If the Auto-Write Mode bits [1:0] = 10b, the Write Cache pointer will be automatically loaded back into the

Host Pointer at the start of an Auto-Write operation. Restoring the Host pointer on every Auto-Write operation allows sequential transfers into the buffer by multiple Auto-Write operations even if read transfers that modify the Host Pointer occur in between them.

The Disk segment is used for transfers to or from the disk. The location of the Disk segment is determined first by the setting of the control bits in the Disk Segment Control register (reg. 137h, R/W). The Disk Segment Select bit (reg. 137h, R/W, bit 2) determines if Segment A or Segment B is used for the disk transfer. The corresponding Beginning of Segment and End of Segment Pointers are selected based on the setting of this bit. The Disk BCTR Select bit (reg. 137h, R/W, bit 1) determines whether BCTRA or BCTRB is used for the transfer. The Enable Disk Room Logic bit (reg. 137h, R/W, bit 0) must be set if automatic buffer full/empty throttling is desired. The Disk Pointer (reg. 134h-136h, R/W) is used by the Disk side of the buffer to access this segment. The Disk Segment can be configured as a linear open ended or circular buffer.

The Host Segment is set up to service transfers between the Host and buffer (for non-Auto-Write commands). The Auto-Write Select bit (reg. 10Bh, R/W, bit 0) should be cleared by the microprocessor for transfers using the Host Segment indicating that the current command is not an Auto-Write command. This, together with control bits in the Host Segment Control register (reg. 133h, R/W) define how the Host segment is set up. The Host Segment Select bit (reg. 133h, R/W, bit 2) determines whether the Segment A Pointers or the Segment B Pointers are to be used for establishing the Host segment. The BCTR used for the Host transfer (BCTRA or BCTRB) is automatically selected via the Host BCTR Select bit (reg. 133h, R/W, bit 1). The Host Pointer (reg. 130h-132h, R/W) will not be modified at the start of the transfer as in the case of the Auto-Write command. The Enable Host Room Logic bit (reg. 133h, R/W, bit 0) must be set if automatic buffer full/empty throttling is desired. The Host Pointer (reg. 130h-132h, R/W) is used by the Host side of the buffer to access this segment. The Host segment can be configured as a linear open ended or circular buffer.

The Servo Segment is an area of the buffer that contains the Data Split values and sector skip information. As previously mentioned, the base 8K page location is established by the Servo Page registers (reg. 112h-113h, R/W). The Servo Beginning Of Segment registers (reg. 11Ch-11Dh, R/W), VBOS[12:0], provide the location within the 8K area where the Servo segment begins. The Servo End Of Segment registers (reg. 11Eh-11Fh, R/W), VEOS[12:0], establishes the end of the Servo segment within the 8K area.

The Servo Pointer register (reg. 12Ch, 12Dh, R/W) is used by the Disk Block to access the Servo segment. This process is explained in more detail in Section 3.5. The Servo Pointer is actually used as an offset pointer into a segment whose base address is determined by the Servo Page registers (reg. 112h, 113h, R/W). When the Servo Pointer reaches the End Of Servo Segment value, it wraps back to the Beginning Of Servo Segment value.

The various segments can be set up such that they are contiguous, separate, or overlapping in the buffer. To execute a read command, the Host and Disk Segments must overlap such that disk read data is being put into an area of the buffer RAM and it is taken by the Host via the overlapping Host Segment. To execute a write command, the Auto-Write Host or Host segments and the Disk Segment must overlap such that the Host write data is being put into an area of the buffer RAM and it is written to the disk via the overlapping Disk Segment.

BCTR Transfer Control:

The controlling factor in automatic transfers between the Host and the disk is the Buffer Counter register (BCTR) associated with that transfer. The BCTR is used to keep track of how many sectors or how much room is in the associated segment area. The transfers will be controlled by the value in the BCTR. The BCTR is always incremented by one whenever a sector has been written to the disk or whenever a sector has been read from the disk which either requires no ECC correction or was successfully corrected by the hardware. The BCTR is always decremented after any sector has been transferred to or from the Host. There are two BCTR registers that are available for use. They can be assigned to various specific tasks. For example, BCTR[A] can be used for write commands and BCTR[B] can be used for read commands.

The particular BCTR register to be used for current Disk transfer is determined by the Disk BCTR Select bit (reg. 137h, R/W, bit 1). When set, this bit selects BCTRB to be used for the Disk transfer. When cleared, BCTRA is used.

The particular BCTR register to be used for current Host (non Auto-Write) transfer is determined by the Host BCTR Select bit (reg. 133h, R/W, bit 1). When set, this bit selects BCTRB to be used for the non-Auto-Write Host transfer. When cleared, BCTRA is used.

The particular BCTR register to be used for current Host transfer (for Auto-Write commands) is determined by the Auto-Write BCTR Select bit (reg. 133h, R/W, bit 4). When set, this bit selects BCTRB to be used for the Auto-Write Host transfer. When cleared, BCTRA is used.

Associated with each BCTR register is a companion Maximum Buffer Count (BKMAX) register. BKMAX[A] (reg. 116h-117h, R/W), and BKMAX[B] (reg. 11Ah-11Bh, R/W) are used with the corresponding BCTR registers to define a revolving buffer within the segment. During write operations, the selected BKMAX is typically set to a value somewhat less than the maximum number of sectors (SECPERSEG) that can fit in the segment. In this case, the contents of a sector that has just been written to the disk cannot be over-written in the buffer until the next SECPERSEG-BKMAX number of sectors are written to the disk. This is useful during a write operation when recovering from a write fault error where it is required to still have the original data in the buffer so a retry operation can be attempted. During read operations, the selected BKMAX should also be set to a value equal to the physical segment size.

Before a disk read operation, the BCTR is initialized by the local microprocessor to zero. As data is being read from the disk, it is incremented by one whenever a good or corrected sector has been placed in the segment. It decrements whenever a sector is transferred to the Host. Whenever BCTR = BKMAX, the buffer segment is considered to be full and the disk transfer is stopped. Whenever BCTR = 0, the buffer segment is considered to be empty and the Host transfer is automatically paused.

Before a disk write operation, the BCTR is initialized by the local microprocessor to BKMAX. During the write operation, it is decremented whenever a sector is received from the Host. It is incremented after a sector has been written to the disk. When the BCTR = 0, the buffer segment is considered to be full and the Host transfer is automatically paused. When the BCTR = BKMAX, the buffer segment is considered to be empty and the disk transfer is stopped.

The Disk No Room bit (DNOROOM, reg. 10Bh, R/W, bit 7) is set whenever BCTR >= BKMAX, and can be used as an indication to the Disk Sequencer for stopping transfers to or from the buffer. This is a segment full condition during disk read operations and a buffer empty condition during disk write operations. DNOROOM is set during a read operation if BCTR = BKMAX-1 and ECC Busy is asserted indicating that the last sector read is currently in the process of being corrected. If the ECC error is determined to be uncorrectable, the DNOROOM bit will remain set and the BCTR will not increment to BKMAX. DNOROOM is forced to 0 when the Enable Disk Room Logic bit (reg. 137h, R/W, bit 0) is cleared. In this case, buffer segment pointer wrapping will occur but automatic pausing will not take place to prevent overwriting data or taking invalid data from the buffer.

The Enable Disk Room Logic bit (ENDROOM, reg. 137h, R/W, bit 0) must be set to enable incrementing of the BCTR associated with the Dick Buffer transfer.

The Enable Disk Room Logic On Wrap bit (ENDRMWRAP, reg. 102h, R/W, bit 3) is another bit which controls when the BCTR associated with the Disk/Buffer transfer is incremented. This bit can be used during Minimal Latency Read operations to hold off incrementing the BCTR until the Disk Pointer has wrapped to the Beginning of Segment and the first requested sector is available in the buffer.

The Host No Room bit (HNOROOM, reg. 10Bh, R/W, bit 6) is set whenever the BCTR <= 0 and is used to control data transfers between the Host and the buffer. In the case of an AT Read Multiple or Write Multiple command, where each transfer consists of blocks containing multiple sectors, HNOROOM is set whenever the BCTR is less than the number of sectors in a block as defined by the value in the Host Block Size register (reg. B6h, R/W).

The Enable Host Room Logic bit (ENHROOM, reg. 133h, R/W, bit 0) or the Enable Auto-Write Room Logic bit (ENAWROOM. reg. 133h, R/W, bit 3) must be set to enable decrementing of the BCTR associated with the Host/Buffer transfer.

The BCTR is a 16-bit two's complement counter and therefore represents a negative value when bit 15 is set. The ability to initialize the BCTR to a negative value is useful for certain Minimal Latency Read implementations. In this case, when the microprocessor has ascertained at which sector the MLR read will start, the BCTR is loaded with a negative number which represents the number of sectors required to read into the segment until the first sector of the track is encountered. Every time a sector is read into the segment, the BCTR will increment. During this time, transfers to the Host will be prevented since BCTR <= 0. Eventually, the BCTR, will increment to one after the first sector of the track has been read (and corrected) into the segment. At this point in time, the sector will automatically be sent to the Host. Eventually, the read operation will conclude by reading into the buffer the sector prior to the sector which was read to start the MLR read operation. At this point in time, the value of the BCTR is added with the number of sectors from this point until the end of the track which were loaded into the segment previously. This is done by first selecting either BCTRA or BCTRB via the Buffer Counter Select bit (reg. 108h, R/W, bit 0) and then writing the incremental value into the Buffer Counter Add/Subtract register (reg. 10Ah, W). This creates a positive value for BCTR and the Host will continue automatically.

As described above, the value in the associated BCTR can be directly manipulated using the BCTR Add/Subtract register (reg. 10Ah, R/W). The BCTR can also be manipulated using another set of registers. While the Enable Delayed Block Release bit (reg. 108h, R/W, bit 7) is set, a read of a "good" sector from the disk will increment the Disk Block Counter (reg. 10Eh, R/W) instead of the selected Disk transfer BCTR. At the appropriate time, the accumulated count in the Disk Block Counter can be added to the selected Disk transfer BCTR by setting the Release Blocks To Host bit (reg. 108h, R/W, bit 3). At that time, the value in the Disk Block Counter will automatically be added to the BCTR and the Disk Block Counter will be cleared. This feature is applicable for both Disk read and Disk write operations.

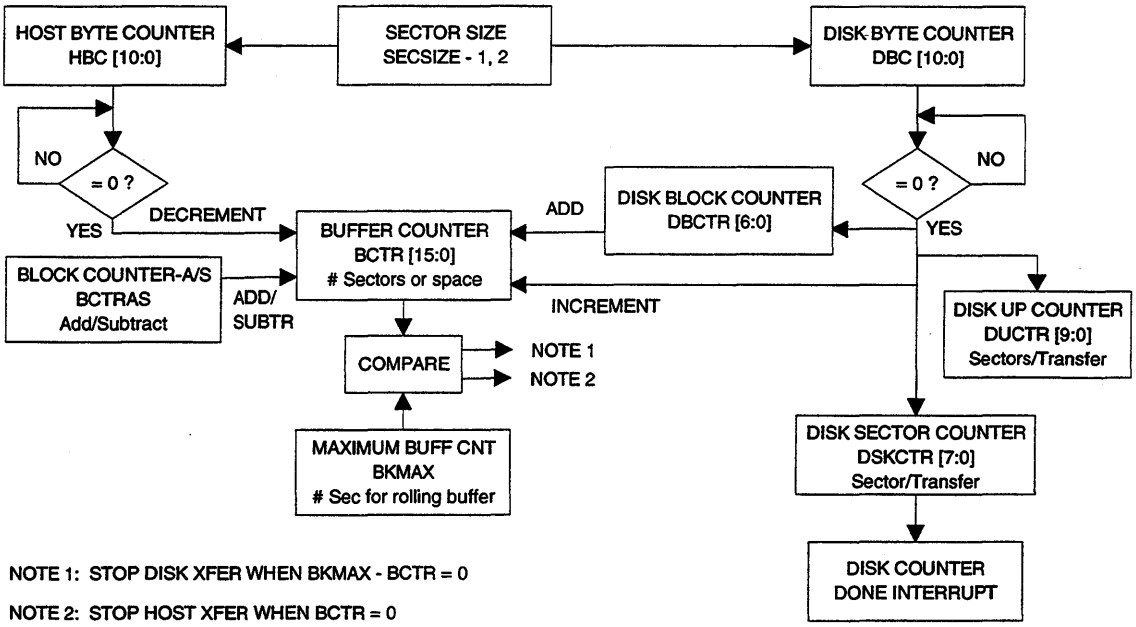

Figure 3-13 Buffer Empty/Full (BCTR) Automation Control Block Diagram

Figure 3-13 illustrates the buffer full/empty layer of buffer automation which relies heavily on the use of the BCTR registers. The structure, depicted in this figure, is the center of ADFM; Automatic Data Flow Management which is discussed in Section 3.8. Figure 3-14 illustrates a typical buffer segmentation scheme.

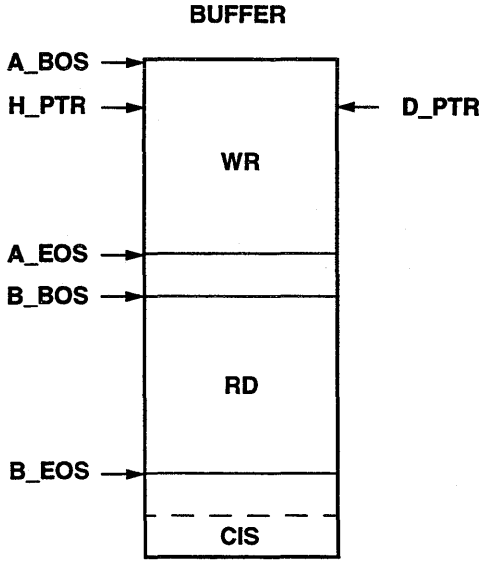

Figure 3-14 A Typical Buffer Segmentation Scheme

3.3.10 Manual Buffer Operation

The buffer can be used with very little automation turned on if so desired. This mode can be used for diagnostics, debugging, or vendor unique command execution.

Manual Host/Buffer Transfers:

Manual Host/buffer operations consist of reading or writing single sectors at a time into or out of the buffer RAM. The microprocessor must do specific actions in order to facilitate each sector transfer. As far as setting up the buffer, the following needs to take place:

- The Enable Multisector Transfer bit (reg. C2h, R/W, bit 7) must be cleared so that only one sector will be transferred across the ATA interface at a time.
- There is no need to select a BCTR, therefore the Auto-Write BCTR Select bit (reg. 133h, R/W, bit 4) and the Host BCTR Select bit (reg. 133h, R/W, bit 1) are not valid for the operation.
- The Enable Host Room Logic bit (reg. 133h, R/W, bit 0) should be cleared to disable the Host Room logic.
- The Host Pointer (reg. 130h-132h R/W) must be manually set to the desired starting location in the buffer for the data transfer.
- The Host Port Enable bit (reg. 109h, R/W, bit 0) must be set to allow the transfer to take place.

The local microprocessor starts each individual sector (or block) transfer between the buffer and the Host by setting the Start AT Transfer bit (reg. C3h, R/W, bit 2) or the Restart AT Transfer bit (reg. C3h, R/W, bit 1) to kick off the operation. Data will be read or written to the buffer with no buffer full/empty checking.

Manual Disk/Buffer Transfers:

Manual Disk/buffer operations consist of reading or writing a number of sectors at a time to or from the disk without the aid of automation. This situation is somewhat different from the host since the operation is very tightly coupled with the Disk Sequencer and the number of sectors read or written is dictated by the Sequencer. However, the most manual way of transferring involves no buffer full/empty checking as the transfer is taking place. As far as setting up the buffer, the following needs to take place:

- There is no need to select a BCTR, therefore the Disk BCTR Select bit (reg. 137h, R/W, bit 1) are not valid for the operation.
- The Enable Disk Room Logic bit (reg. 137h, R/W, bit 0) should be cleared to disable the Disk Room logic.
- The Disk Pointer (reg. 140h-142h R/W) must be set to the desired starting location in the buffer for the data transfer.
- The Disk Port Enable bit (reg. 109h, R/W, bit 1) must be set to allow the transfer to take place.

The local microprocessor starts the Disk Sequencer. Eventually the starting sector is encountered and the Disk Sequencer initiates the read or write of the indicated amount of sectors. The transfer will continue until the Sequencer is satisfied and has read or written all desired sectors on the track. This process takes place with no buffer full/empty checking.

3.3.11 Buffer Block Initialization

This section presents a sample initialization setup for the buffer block. The buffer block has six major features: SRAM/DRAM support, host/disk room logic with 2 BCTRs for ADFM, and automatic servo offset pointer increment.

The sample flow presented here supports the following commonly used features.

- 16-bit DRAM, runs at 6T wait state, with buffer clock of 50-MHz.
- Auto Write segment and BCTRA for Auto-Write commands.
- Host segment and BCTRB for read commands.
- CDR counts from servo segment.
- Intel based microprocessor.

In the flow table that follows, if any bits in a register are not specifically mentioned to be either set or cleared, their original value must be retained while setting or clearing the specified bits.

Table 3-20 Buffer Block Initialization

Initialization Function	Register Action
Reset buffer block.	Set reg. 50h bit 1, then clear this bit.
Set DRAM refresh period.	Set reg. 101h, R/W = 0Bh (or to an appropriate value for the selected DRAM).
Set buffer mode to 16-bit DRAM 6T.	Set reg. 100h, R/W = 3Ch.
Enable16-bit µP access, enable the Host and Disk No Room Occurred Status and enable Disk Room Logic On Wrap function.	Set reg. 102h, R/W = 3Ch.
Enable data transfer stop on errors.	Set reg. 105h, R/W = 0Ch.
Setup the Servo segment page and pointer. (May be required to be updated on a track basis.)	Set reg. 11Ch-11Dh, R/W = VBOS value. Set reg. 11Eh-11Fh, R/W = VEOS value. Set reg. 112h-113h, R/W = Servo page value. Set reg. 12Ch-12Dh, R/W = Servo pointer.
Setup the Auto-Write Host segment. This includes the WCP, BKMAXA and BCTRA registers. In this example, segment A and BCTRA is used for the Auto-Write segment. Enable Auto-Write Room logic.	Set reg. 133h, R/W = 8Fh. Set reg. 138h-13Ah, R/W = ABOS value. Set reg. 13Ch-13Dh, R/W = AEOS value. Set reg. 148h-14Ah, R/W = WCP value. Set reg. 116h-117h, R/W = BKMAXA (Segsize - 1) Set reg. 114h-115h, R/W = BCTRA (Segsize - 1)
Setup the non-Auto-Write Host segment. This includes the HP, BKMAXB and BCTRB registers. In this example, segment B and BCTRB is used for the non-Auto-Write segment. Enable Host Room logic.	Set reg. 133h, R/W = 8Fh. Set reg. 140h-142h, R/W = BBOS value. Set reg. 144h-146h, R/W = BEOS value. Set reg. 130h-132h, R/W = HP value. Set reg. 11Ah-11Bh, R/W = BKMAXB (Segsize - 1) Set reg. 118h-119h, R/W = BCTRB (Segsize - 1)
Setup the Disk segment. This includes the DP, BKMAXB and BCTRB registers. In this example, segment B, BKMAXB, and BCTRB are used for the Disk segment and have been already setup in a preceeding step. Enable Disk Room logic	Set reg. 137h, R/W = 07h. Set reg. 134h-136h, R/W = DP value.
Initialize buffer controls.	Write 70h to reg. 108h. R/W.
Enable buffer interrupts.	Set reg. 104h, R/W = 04h. Set reg. 107h, R/W = 17h. Set reg. 53h, R/W = 18h.
Enable Servo port, µP port, Correction port, and Host port.	Set register 109h bit 4, 3, 2, & 1.

3.4 Disk Interface and Control

The disk interface and control portion of the AIC-8375 incorporates automated features which allow headerless full track read and write operations to occur without microprocessor intervention. This provides the microprocessor with more bandwidth to use for other functions. The automated tasks includes finding the desired frame and sector, accessing it, updating the Current and Requested Sector values for the next sector, loading the CDR skip values for the next sector, skipping defective sectors and stopping after the appropriate number of sectors have been accessed. In addition, the disk block has knowledge of the empty/full status of the buffer and can make decisions on-the-fly based upon that information.

The data flowing between the disk interface logic and the buffer flows through the disk FIFO.

3.4.1 Disk Block Architecture

The architecture of the disk block is illustrated in Figure . This figure shows the general partitioning or major functions and data flow paths.

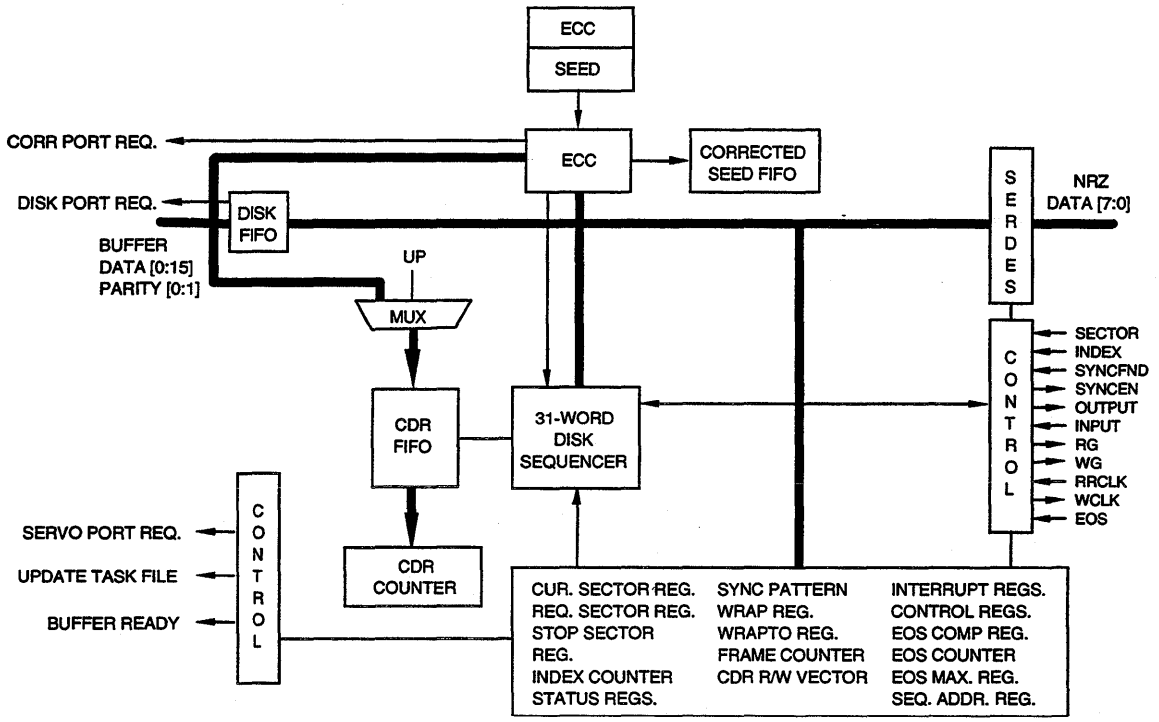

Figure 3-15 Disk Block Architecture

3.4.2 Disk Sequencer Operation

The Disk Sequencer is central to the disk interface and control logic; a programmable state machine that controls signals necessary for reading and writing data to and from the disk. The sequencer performs such operations as finding the target Frame and Sector, ECC verification, Sync byte detection, monitoring buffer full/empty status, automatic update of the Current and Requested sectors, loading or flushing the CDR FIFO at appropiate times, skipping defects, and transfer of data between the disk interface and the buffer.

The disk sequencer executes instructions which are contained in an internal control store RAM. The control store for the disk sequencer is called the sequencer map and consists of 93 bytes of RAM organized into 31 instructions of three bytes each. The various fields in the sequencer instruction indicate what set of conditions cause the sequencer to advance to the next instruction, the address of the next instruction to execute, and what control signals to assert or negate while executing the current instruction. The disk sequencer is clocked by a "Byte Clock" which is derived from the RRCLK on the disk interface. In 8-bit NRZ mode, the RRCLK is used directly, while in single-bit and dual-bit NRZ modes, RRCLK is divided down by eight and four respectively. Thus, a sequencer instruction will execute every disk byte time.

The disk sequencer controls the flow of data into or out of the disk FIFO from the disk interface.

The sections which follow provide a summary of the functions that the disk sequencer performs. Refer to Section 4 for details on how the various instruction decodes work.

3.4.2.1 Time-Out Functions

The AIC-8375 disk sequencer provides a timeout function. If the event being waited for does not occur within the number of byte times programmed in the SEQCNT Field, execution will either he halted or continue at the address specified in the NXTADDR field.

The timeout function is available while waiting for Index, Sector, Sector Valid, Sync, EOS Signal, EOS Compare Equal, or the Data Split value being loaded into the CDR FIFO. For all timeout conditions except waiting for the Data Split value to be loaded into the CDR FIFO, the Sequence Time-Out bit (reg. 66h, R, bit 6) will be set and can be programmed to generate an interrupt to the microprocessor. The "Wait For Defect Flag" timeout will set the Servo Overun bit (reg. 5Eh, R, bit 7).

In addition, if the target sector has not been found within two Index pulses, a timeout error is generated and the sequencer can be programmed to stop if the Stop On Two Index Time-Out bit (reg. 7Eh, R/W, bit 0) is set. As well, the Two Index Time-Out bit (reg. 66h, R, bit 7) can be programmed to generate an interrupt to the microprocessor.

3.4.2.2 Decision / Branching Functions

The AIC-8375 disk sequencer provides significant capabilities for making decisions and branching. These decision functions are programmed via the Primary and Alternate branch instructions. The instructions vary in how the branches are taken, but the address at which execution will take place after the decision has taken place will be either the Next Address Field, PC+1, or the Halt address (1Fh). Refer to Section 4 of this manual for more details on each of the branch instructions.

3.4.2.3 Counting Functions

There are several types of counting functions available in the disk sequencer.

The SEQCNT Field can be used in many instructions to determine how many byte times the instruction execution is to last. At the time the count expires, the control action or decision action takes place. The value used in the SEQCNT Field is n-1 where n is the desired number of execution byte times.

The SEQCNT Field can also be used as a fetch count. When used as such, the instruction will execute for one byte time and the fetch of the indicated number of bytes (In the SEQCNT Field) will continue throughout the several instructions which follow. The Load CDR FIFO decode in DSA Mode (SEQCTLA != '11' and SEQCTLC = '001') is an example of where this function is used.

3.4.2.4 Compare Functions

Frame and Sector Detection are accomplished via two branch operations and two control decodes.

The Wait for EOS Compare Equal (alternate, BRSEL = '100') allows the sequencer to wait for the Target Frame. The EOSCMP register (reg. 5Ch, R/W) should be programmed to one less than the EOS count for the target frame. EOSCTR is updated automatically by the chip at 2-3 byte times after each EOS pulse.

The Wait for Sector Valid (alternate, BRSEL = '111') is a fairly complicated decode involving a number of factors. It will wait until the Sector is asserted. At this point, it will branch based on a number of conditions, the collection of which is termed Sector Valid. The conditions required for a sector to be valid are as follows:

- The internal BUFNRDY condition must be false (there must be space in the buffer, i.e., DNOROOM (reg. 10Bh, R, bit 7) must be deasserted)
- The CURRSECEQ bit (reg. 5Bh, R, bit 7) must be set (Current Sector (regs. 5Ah/5Bh, R/W) equal to Requested Sector (regs. 6Ch/6Dh, R/W))
- The internal CDRVALID condition must be true (a successful Data Split Word fetch must have concluded).
- The Defect Flag must not be set (the Data Split control words must not indicate the sector was invalid).

CURRSEC and REQSEC are updated by the sequencer via the Increment Current Sector (primary, SEQCTLB = '01') and the Increment Requested Sector (SEQCTLC = '010').

CURRSEC, REQSEC and EOSCMP should not be written while the sequencer is running. Unpredictable results may occur.

Individual terms of the Sector Valid function may be disabled through the clearing of the SECEQ, SECRDY and SECDEF bits in DCTL2 (reg. 62h, R/W, bits 2:0).

3.4.2.5 Input / Output Functions

Data can be input from or output to the Disk Sequencer on the NRZ[7:0] data lines. The Output pin can be directly controlled via the Output Pin decode (SEQCTLB = '01') while the Enable Output bit (ENOUTPUT, reg. 62h, R/W, bit 4) is set: In addition, the Input Pin and the Input 2 Pin can be monitored from the sequencer. Note that the Output and Input signals are multiplexed to the same pin. Which signal the pin is used for is determined by the value of ENOUTPUT.

3.4.2.6 Control Functions

There are numerous decodes available in the sequencer to control such things as setting and resetting Write Gate and Read Gate, initializing the ECC Seed, initializing the data field ECC logic, loading/flushing the CDR FIFO, resetting the CDR FIFO, loading the high byte of Sync, and enabling various external registers for use with the sequencer.

3.4.2.7 Frame Counter

The Frame Counter (reg. 74h, R/W) is a counter which can be used as a general purpose do-loop counter in the sequencer map. A count value is loaded into the Frame Count register by loading register 74h. This in turn is loaded into the actual Frame Counter. During sequencer map execution, the Primary '110' branch

instruction (Next Address If Frame Counter = 0) can be executed. This instruction will test for zero first, then decrement the Frame Counter. If it is zero, sequencer operation will continue at the address specified in the Next Address Field and the Frame Counter will then be reloaded with the original value that is in the Frame Counter register.

3.4.2.8 Starting and Stopping the Disk Sequencer

The disk sequencer is either in a stopped state or a running state. It is started by loading the address at which to start executing into the Sequencer Start Address bits (reg. 73h, R/W, bits 5:0) and then setting the Sequencer Run bit (reg. 73h, R/W, bit 7). These bits must all be set together in the same instruction if desired.

After a disk block reset or Power On reset condition, the sequencer is in its stopped state. When the sequencer is in this state, all of the control signals that it drives are in their inactive (negated) state and remain that way until the sequencer is started and they are deliberately changed. There are several ways that the sequencer may be stopped, they are as follows:

Stopping the Sequencer From the Sequencer Map:

The sequencer can be stopped by causing execution to take place at the "stop location" which is location 1Fh. This is accomplished whenever a decision/branch type instruction is executed with the value 1Fh in the Next Address Field and the decision branch turns out to be the Next Address Field. After the sequencer has stopped, the Sequencer Current Address bits (reg. 73h, R/W, bits 5:0) will contain the address of the instruction that was last executed before the sequencer stopped.

Stopping the Sequencer Via Selectable Events:

The sequencer may be automatically stopped when certain events take place. These events are selected in the Disk Auto Stop Control register (reg. 7Eh, R/W). Additional relevant bits not located within that register include the Servo Overun bit (reg. 5Eh, R, bit 7), the Stop on CDR Parity Error bit (reg. 5Dh, R/W, bit 3), the Stop on Seed Error bit (reg. 5Dh, R/W, bit 2), the Stop on Seed Overrun bit (reg. 5Dh, R/W, bit 6), the Stop on Auto-Write bit (reg. 5Dh, R/W, bit 5), and the Stop On Input Write Fault bit (reg. 61h, R/W, bit 3).

If either a Servo Overrun condition, CDR Parity Error, or a Write Fault condition occurs, the disk sequencer will immediately negate Write Gate and stop. This provides a safety feature for the user in situations where data corruption could occur. In general, if any of the stop conditions enabled by the bits in the Disk Auto Stop Control register (reg. 7Eh, R/W) occur, the sequencer will stop after Write Gate is negated. This may occur at the end of a sector or right before a servo split since Write Gate is negated at these times. However, during read operations, the sequencer will stop immediately when the condition is detected. Refer to the Disk Auto Stop Control register description for exceptions to this operation.

Stopping the Sequencer Directly Via the Microprocessor:

The disk sequencer can be asynchronously stopped by clearing the Sequencer Run bit (reg. 73h, R/W, bit 7). The Sequencer Current Address bits (reg. 73h, R/W, bits 5:0) will reflect the address of the instruction that was being executed at the time the sequencer was stopped.

Synchronous Stop Via the Microprocessor:

Setting the Stop On Sector Boundry bit (reg. 7Eh, R/W, bit 2) stops the Sequencer at the end of the sector after WG (or RG) has been deasserted. The Disk Sequencer is not stopped at Data Split boundries.

3.4.3 Automatic Frame and Sector Management (EDSA Mode)

The Disk Sequencer has fully automated the search for the target Frame and Sector. For each track, all the microprocessor must do is determine which frame contains the requested sector (EOSCMP, reg. 5Ch, R/W), the physical sector number of the first sector in that frame (CURRSEC, regs. 5Ah/5Bh, R/W), the physical sector number of the requested sector (REQSEC, regs. 6Ch/6Dh, R/W) and the location of the corresponding split information. Once these registers have been programmed and the sequencer started, the rest of the process is automated. The sequencer will load the CDR values, skip defective sectors, skip a disk revolution when out of buffer space, and skip sectors as indicated by the wrap registers.

The various automated sector management functions are summarized in the paragraphs that follow.

Target Frame Search Function

The first step in finding the requested sector is finding the frame containing it. Frames are identified by the number of the EOS pulse starting that frame. This value is maintained by the EOS Counter register (reg. 58h, R/W), which is incremented by the device at 2-3 byte times after every EOS pulse. The sequencer, using the Wait on EOS Compare Equal decode (alternate, BRSEL = '100') waits for the Target Frame. The EOS Compare register (reg. 5Ch, R/W) must be programmed to one less than the EOS count for the target frame. This operation (finding the target frame) need only be done once for each track, i.e. if a revolution is skipped because the buffer was not ready, the sequencer doesn't need to wait for EOS Compare Equal again.

Once the target Frame has been reached, the sequencer starts its search for the requested sector.

Target Sector Search Function

Searching for the target sectors requires that the Current Sector (reg. 5Ah, 5Bh; R/W) and Requested Sector (reg. 6Ch, 6Dh, R/W) be initialized, and that the Servo Pointer (reg. 12Ch, 12Dh, R/W), point to the first CDR entry for Current Sector. Current Sector should be initialized with the physical sector number of the first sector following a sector pulse in the target frame (any 'stumps' at the beginning of the frame should be ignored). Requested Sector should contain the physical sector number of the target sector.

The values below are used by the Wait for Sector Valid decode (alternate, BRSEL = '111'). It will wait until the Sector line is asserted and determine whether the following sector is valid. At this point, it will branch based on a number of conditions, the collection of which is termed Sector Valid. The conditions required for a sector to be valid are as follows:

- The internal BUFNRDY condition must be false (there must be space in the buffer, i.e., DNOROOM (reg. 10Bh, R, bit 7) must be deasserted)
- The CURRSECEQ bit (reg. 5Bh, R, bit 7) must be set (Current Sector (regs. 5Ah/5Bh, R/W) equal to Requested Sector (regs. 6Ch/6Dh, R/W))
- The internal CDRVALID condition must be true (a successful Data Split Word fetch must have concluded).
- The Defect Flag must not be set (the Data Split control words must not indicate the sector was invalid).

Stopping the Transfer Based on the Last ID:

The disk sequencer can automatically monitor what sector is currently being operated on and stop the sequencer (or do some other desired function) if it is the last sector to be read or written. This is accomplished using the Primary branch '111' instruction (NXTADDR if Secnum = Stop Sec Number). The Increment Requested Sector decode will increment the Requested Sector Number registers (reg. 6Ch, 6Dh, R/W) at a strategic location in the disk sequencer map and the "NXTADDR if Secnum = Stop Sec Number"

decode will compare the value of the Requested Sector Number register (reg. 6Ch, 6Dh, R/W) to the value in the Stop Sector register (reg. 6Eh, 6Fh, R/W). If the compare is equal, this indicates that the last sector has been read or written and execution will continue at Next Address. The Current Sector and Stop Sector registers must be loaded with the appropriate values prior to starting the sequencer after the microprocessor has constructed the framework and boundary condition of the impending read or write operation.

Wrapping Search ID For Non-Contiguous Transfers:

The disk sequencer can automatically interact with external registers to facilitate a non contiguous disk read or write operation. This type of operation is useful for implementing Minimal Latency reads.

For example, take the case where the current track has 50 sectors. Suppose the current command is a read command of 20 sectors starting at sector 10. Because this is a headerless architecture, a read of the disk after the seek has completed is not necessary to determine where the head is on the track. Examination of the EOS counter will suffice. Assume it is known that the first readable frame will contain sector 14. Instead of waiting for sector 10 to come around again and then start reading, the sequencer and supporting registers can be programmed to start the read at sector 14, read sector 15 through 30 and then wait until sector 10 comes around again to pick up the remaining sectors at sectors 10 through 14.

The operation just cited can be accomplished by first setting the Current Sector (regs. 5Ah, 5Bh, R/W) and Requested Sector (regs. 6Ch, 6Dh, R/W) registers to 14, the Stop Sector register (regs. 6Eh, 6Fh, R/W) to 13, the Wrap Sector register (regs. 68h, 69h, R/W) to 30, and the WrapTo Sector register (regs. 6Ah, 6Bh, R/W) to 10 before the sequencer is started. The read operation will begin at sector 14 and will continue until the Current Sector Number = Wrap Sector Number. When this occurs, the Requested Sector register will be loaded with the value in the WrapTo Sector register (which is 10). The sequencer will resume the read operation when sector 10 is encountered and will finally stop when the Requested Sector Number = Stop Sector Number.

3.4.3.1 Two-Byte Sector Number

With the AIC-8375, sector numbers are always 10 bits long.

3.4.4 Data Sync Generation and Detection

The AIC-8375 incorporates the ability to deal with internal or external sync generation and detection schemes. The data sync function provides a method for establishing the byte boundaries for the incoming or outgoing streams of data. It also tells the ECC logic when to start interpreting the data as valid data.

3.4.4.1 Single-Byte Internal Sync Generation and Detection

Using this method, a single sync byte originates in the disk sequencer and is written to the disk. Likewise, it is looked for by the disk sequencer during a read operation. This method can be used in single and dual NRZ modes but not in 8-bit NRZ mode. The Enable 2-Byte Fault Tolerant Sync bit (reg. 61h, R/W, bit 7) must be cleared while operating in this mode.

The value of the byte to be used for sync is determined by the SEQNADAT field of the last sequencer instruction which did not have the Next Address Enable bit asserted. Also, it is important to note that the INITDATA decode (SEQCTLC='011') must be used for only the first data sync event within the data/ECC field. It should not be used for the data sync events associated with data splits in the sector.

The sync bytes are written using a specific set of disk sequencer instructions. At sync pattern write time the INITDATA decode (SEQCTLC = '011') is set in the instruction prior to the first data byte. The SEQCNT

field must contain the value '0' so that execution of this instruction takes one byte time; the time required to write the byte sync pattern. During this instruction, the contents of the Sync byte will be written. The sync byte pattern will not be included in the ECC calculations.

The data sync byte is searched for by using the "Wait For Sync" disk sequencer instruction. During this instruction, the sequencer will wait for the amount of time specified in the SEQCNT field of the instruction. If the sync byte pattern is detected, execution will continue at PC + 1. The actual sync byte pattern to be searched for must be in the sync byte defined by the method described above. If the "Wait For Sync" instruction times out, a branch to the address in the Next Address Field will take place. The Sequencer Wait Time-Out bit (reg. 66h, R, bit 6) will be set and can be programmed to generate an interrupt to the microprocessor.

3.4.4.2 Fault Tolerant Sync Generation and Detection

A fault tolerant sync generation and detection system has been incorporated into the AIC-8375 to assist in situations where the media defect characteristics may prove problematic using a single byte sync.

Using this method, a selected two-byte sync pattern is specified by the data field of the sequencer. Byte 0 is determined in the same fashion used by the Signal Byte sync method. Byte 1 is loaded via the Load Sync1 decode (alternate, SEQCTLB = '11'). It is this two-byte pattern which will be used for the writing and searching of the fault tolerant byte sync. The logic is designed to detect the original two-byte pattern in the presence of any 6-bit error burst. There are some two-byte patterns which will not work. (Please contact Adaptec for a list of the patterns which will not work.) The Enable 2-Byte Fault Tolerant Sync bit (reg. 61h, R/W, bit 7) must be set while operating in this mode. While set, both the data sync and the split sync must use the fault tolerant method. This method can be used in single and dual NRZ modes but not in 8-bit NRZ mode.

The fault tolerant sync bytes are written using a specific set of disk sequencer instructions. At sync pattern write time the INITDATA decode (SEQCTLC = '011') is set in the instruction prior to the first data byte. The SEQCNT field must contain the value '01' so that execution of this instruction takes two byte times; the time required to write the byte sync pattern. During this instruction, the 2-byte sync pattern will be written, starting with the value of sync byte 1. The sync byte pattern will not be included in the ECC calculations.

The data sync byte is searched for by using the "Wait For Sync" disk sequencer instruction. During this instruction, the sequencer will wait for the amount of time specified in the SEQCNT field of the instruction. If the sync byte pattern is detected, and execution will continue at PC + 1. The actual sync byte pattern to be searched for must be specified in the manner described above. If the "Wait For Sync" instruction times out, a branch to the address in the Next Address Field will take place. The Sequencer Wait Time-Out bit (reg. 66h, R, bit 6) will be set and can be programmed to generate an interrupt to the microprocessor.

3.4.4.3 External Sync Byte Generation and Detection

External sync mode is intended to support those designs which use special sync techniques not compatible with a normal 8-bit or 16-bit pattern recognition scheme. It can be used in single, dual, or 8-bit NRZ mode. However, if operating in 8-bit NRZ mode, this method must be used.

The Enable External Sync Found bit (reg. 63h, R/W, bit 0) must be set and the Enable 2-Byte Fault Tolerant Sync bit (reg. 61h, R/W, bit 7) must be cleared while using this external byte sync method.

The "wait for sync" flow during a read operation is almost identical to the single byte sync method described above with a few notable exceptions. First, a signal indicating byte sync alignment must be presented to the AIC-8375 on the External Sync Found pin. The disk sequencer "Wait For Sync" instruction will wait until it sees that signal instead of waiting for a compare of incoming data.

The writing of the sync byte in external sync byte mode is very similar to the other methods. At sync byte write time, INITDATA decode (SEQCTLC = '011') is set in the instruction prior to the first data byte. The sync byte value will not be included in the ECC calculations.

Different read channels have different requirements for generating the sync mark. Some do not use any special sync byte while others utilize some type of recording code violation which can't be seen with a standard NRZ pattern recognition circuit. Some will wait for the first non-zero data after the VFO field a write a special byte(s) while others just write the 8-bit NRZ byte that is presented to them. The sync byte value will not be included in the ECC calculations.

In some cases, the actual SYNCFND signal may come a number of NRZ clock times earlier than the corresponding NRZ data. This offset can be accommodated by using the External Sync Byte Found Offset bits (reg. 61h, R/W, bit 2:0). These bits allow for an early offset of 0 -7 NRZ clock times. Offsets of 0 and 1 NRZ clock times can be used with any NRZ width, 2 and 3 in either single or dual mode, and 4-7 only in single mode.

3.4.5 Disk NRZ Data Operating Modes

The AIC-8375 is designed to interface to read channels incorporating single, dual, and byte wide NRZ data paths. For the most part, once the data has made its way into the disk block logic, operation is the same in all cases. The disk sequencer runs off of a byte clock. In single and dual NRZ modes, the data must be converted into a byte format and in addition, the NRZ clock (RRCLK) must be converted into a byte clock to run the sequencer.

3.4.5.1 Single-Bit NRZ Operation

The AIC-8375 is set up to operate in single-bit NRZ mode by clearing the NRZ Select bits (reg. 60h, R/W, bits 3:2) during device initialization. In this mode, the serial NRZ data will be written and read via the NRZ[0] pin. For proper operation, the data on this pin must meet the required setup and hold times. Read data is shifted into a de-serializer to form a byte. Write data is shifted out of the serializer circuit to form a one-bit serial data stream. RRCLK will be divided down to produce a byte clock to run the disk sequencer. This allows the disk sequencer to run synchronously with the data stream.

Internal or external byte sync mode may be used in single-bit NRZ mode. During write operations, the high order bit is always written first after the data sync function takes place. During a read operation, the high order bit is assumed to come first.

Running in single-bit NRZ mode can present a timing challenge with respect to NRZ data and RRCLK set-up and hold times. To help with this, a Write Clock is available on the NRZ[2] pin. The timing relationship between NRZ data and Write Clock will be improved over that of NRZ data and RRCLK.

3.4.5.2 Dual-Bit NRZ Operation

Dual-bit NRZ mode is established by setting the NRZ Select bits (reg. 60h, R/W, bits 3:2) to '01' during device initialization. In this mode, the 2-bit wide NRZ data will be written and read via the NRZ[0] and NRZ[1] pins. For proper operation, the data on these pins must meet the required setup and hold times. The 2-bit read data is loaded into a de-serializer to form a byte. Write data is shifted out of the serializer circuit two bits at a time to form a two-bit serial data stream. RRCLK will be divided down to produce a byte clock to run the disk sequencer. This allows the disk sequencer to run synchronously with the data stream.

Internal or external byte sync mode may be used in dual-bit NRZ mode. During write operations, the NRZ[1:0] pins will output a byte of data in the order [D7, D6], [D5, D4], [D3, D2], and [D1, D0]. During a read operation, the order of the bits on the NRZ[1:0] pins could be [D7, D6], [D5, D4], [D3, D2], and [D1, D0] or [D0, D7], [D6, D5], [D4, D3], and [D2, D1].

Since RRCLK will operate at one half the rate as in single-bit NRZ mode, more data can be transferred at an equivalent RRCLK frequency. Typically, dual-bit and 8-bit NRZ modes are used for the higher disk data rates. However, running even in dual-bit NRZ mode can present a timing challenge with respect to NRZ data and RRCLK set-up and hold times. To help with this, a Write Clock is available on the NRZ[2] pin. The timing relationship between NRZ data and Write Clock will be improved over that of NRZ data and RRCLK.

3.4.5.3 Eight-Bit NRZ Operation

Eight-bit NRZ mode is established by setting the NRZ Select bits (reg. 60h, R/W, bits 3:2) to '10' during device initialization. In this mode, the 8-bit wide NRZ data will be written and read via the NRZ[7:0] pins. For proper operation, the data on these pins must meet the required setup and hold times. The 8-bit read data is immediately available in byte form for use by the disk sequencer. Write data is output directly to the NRZ[7:0] pins. RRCLK will be used directly as the byte clock to run the disk sequencer. This allows the disk sequencer to run synchronously with the data stream.

External byte sync mode must be used in eight-bit NRZ mode.

Since RRCLK will operate at one eighth the rate as in single-bit NRZ mode, more data can be transferred at an equivalent RRCLK frequency. Typically, dual-bit and 8-bit NRZ modes are used for the higher disk data rates. Since the NRZ[2] pin is used in eight-bit NRZ mode, the Write Clock function, which is multiplexed on this pin, is not available for use in this mode.

3.4.6 Disk FIFO Operation

The Disk FIFO is used as an interface between the buffer and disk. It is a 32 byte x 8-bit FIFO organized as two 16 byte x 8-bit Dual Port RAMs. In addition to the actual RAM element, the Disk FIFO is controlled by logic which interfaces to the buffer logic and the Disk Sequencer. This logic automatically configures the FIFO to work properly with an 8-bit or 16-bit buffer path, and makes decisions based upon how full or empty it is. When Automatic Data Flow Management is enabled, the Disk FIFO will not request transfers to or from the buffer if a sector of data is not available or if space for a sector is not available.

The buffer is presented with a transfer request whenever the disk FIFO has 8 bytes (or words) to send to the buffer or whenever it has room for 8 bytes (or words) from the buffer. 8 bytes are used if the buffer is configured in 8-bit mode and 8-words are used if a 16-bit wide buffer is utilized. This transfer request will actually cause the buffer logic to initiate a transfer in the next available time slot.

The 8 byte (or word) value is used since the buffer logic is designed to interface to DRAMs using 8-byte page mode transfers. There are times when a transfer request to the buffer will be generated even if there are not 8 bytes (or words) ready to be transferred or if there is not space for 8 bytes (or words). These times include at the end of a sector or when a DRAM page boundary is encountered. Thus, at the end of a read operation, the Disk FIFO will be flushed of the remaining bytes to be transferred even if the number is less than eight.

For write operations, the first transfer request to the buffer will take place only after the disk sequencer has been started while the Disk Write bit (reg. 60h, R/W, bit 7) is set. In addition, a status signal is generated

which indicates that there is one sector of data in the buffer and that there are 8 bytes (or words) of this data in the Disk FIFO. This signal is used by the Alternate '111' branch instruction (Wait for Sector Valid) and the Alternate '110' branch instruction (NXTADDR If BUFNRDY) to alter flow within the sequencer map based on whether or not data is ready to be written.

During read operations, a status signal is generated which indicates that there is one sector of space available in the buffer. This signal is used by the Alternate '111' branch instruction (Wait for Sector Valid) and the Alternate '110' branch instruction (NXTADDR If BUFNRDY) to alter flow within the sequencer map based on whether or not data is ready to be read.

Further transfer requests to the buffer logic are prevented if certain conditions occur. If a Disk FIFO write is attempted while it is full or a disk FIFO read is attempted while it is empty, the Disk FIFO Error bit (reg. 7Dh, R, bit 7) will be set and no more buffer transfer requests will be generated. The Suppress Transfer bit (reg. 62h, R/W, bit 6), when set, will override the ENBUFFER decode (SEQCTLB = '10') in the Disk Sequencer and prevent buffer transfer requests from being generated.

The Disk FIFO Count bits (reg. 7Dh, R, bits 5:0) indicate how many bytes are in the Disk FIFO. A value of 00h indicates that the Disk FIFO is empty and a value of 20h indicates that it is full.

The Disk Byte Count register (reg. 126h, 127h, R) is a 10-bit register that counts the number of bytes that are transferred between the Disk FIFO and the buffer. When the first word of a sector is transferred between the disk FIFO and the buffer, the Disk Byte Counter wraps to a value that corresponds to the number of bytes remaining to be transferred for that sector. After the first word of a sector is transferred, the Disk Byte Counter decrements by 1 (for 8-bit buffer mode) or 2 (for 16-bit buffer mode) for every word transferred. It will return to the value zero after the entire sector has been transferred.

The disk FIFO is reset whenever the Reset Disk FIFO bit (reg. 60h, R/W, bit 6) is loaded with a '1', the Disk block is reset via the Disk Block Reset bit (reg. 50h, R/W, bit 2), or the Disk Byte Counter Clear bit (reg. 108h, R/W, bit 5) is loaded with a '1'.

3.4.7 Disk Block Automation

The Disk block automation features relieve the microprocessor of having to perform many time critical functions. The features include:

- Automatic management of frame and sector searches and compares.
- Automatic skipping of defective sectors.
- Automatic management of Sector wrapping/sequencing during disk read and write operations.
- Automatic checking of the buffer ready condition during disk read and write operations.
- Programmability of various functions to automatically stop the disk read or write operation in the event of an error condition.

3.4.8 Disk Block Initialization

Before the disk block can be used, it must be initialized. Initialization can be categorized into two categories. General initialization and specific initialization. General initialization takes care of all the various bits that must be initialized independent of the application. Specific initialization encompasses setting up various functions that are specific to the application. This would include, for example, 1-, 2-, or 8-bit NRZ transfers, the method of dealing with Data Splits, or 1- or 2-byte Sector number mode.

General initialization information is provided here along with a sample initialization which is used for specific examples which follow. The specific examples which follow use EDSA, Fault Tolerant Sync, 8-bit NRZ, and CDR splits. If any bits in a register are not specifically mentioned to be either set or cleared, their original value must be retained while setting or clearing the specified bits.

Table 3-21 Disk Block General Initialization

Action	Affected Registers and Bits
Clear Disk Block Reset during Power On Initialization.	Clear reg. 50h, R/W, bit 2.
Disable Disk Interrupts.	Clear reg. 53h, R/W, bits 6:5.

Table 3-22 Disk Block Specific Initialization

Action	Affected Registers and Bits
Reset Disk FIFO, Select 8-bit NRZ.	Set reg. 60h, R/W, bits 6, 3. Clear reg. 60h, R/W, bits.7, 5:4, 2:0.
Enable 2 Byte Sync, Stop On Input Write Fault, Enable Index to Sector Branch.	Set reg. 61h, R/W, bits 7,4,3.
Enable CDR, Enable Sector Eq branch, buffer not ready branch and defective sector branch. Note: Once you enable CDR, the servo FIFO will start fetching CDR values automatically if the servo port is enabled.	Set reg. 62h, R/W, bits 3:0.
Select all active high signals, select buffer CDR and enable external sync.	Set reg. 63h, R/W, bits 7:3,1:0.
Enable ECC Seeding, Stop on Seed Error, Stop on Seed Overrun and EDSA.	Set reg. 5Dh, R/W, bits 6,4,2,0.
Clear the Disk Sequencer Address register.	Set reg. 73h, R/W = 00h.
Not using Frame Counter and Index Counter.	Set reg. 74h-77h = 00h.
Initialize Maximum EOS.	Set reg. 59h to max EOS value.
Enable Stop On Uncorrectable ECC Error, Stop On Input, Stop On Disk/buffer Transfer Error, and Stop On 2-Index Timeout.	Set reg. 7Eh, R/W, bits 7:5, 0.
Initialize the EOS counter. Two basic methods may be used to accomplish the: 1) writing a known value to the register when the EOS line is low, and 2) Letting the chip reset EOSCTR at index.	Method #1: Set reg. 58h to current EOS count. Method #2: Set reg. 5Dh, bit 1.
Ensure all disk interrupts are cleared.	Set reg. 5Eh, W = 00h. Set reg. 66h, W = 00h.
Enable Disk Interrupts to microprocessor.	Set reg. 53h, R/W, bits 6:5.
Enable Disk Interrupts.	Set reg. 5Fh, R/W, bits 7,0. Set reg. 67h, R/W, bits 7:5, 0.

3.4.9 Disk Operation Execution

Any type of disk read or write command issued by the Host will require a specific disk operation to be performed. All of the common and required ATA disk related commands will use one of these specific disk operations. These disk operations are described in this section.

Register set flows and a sample sequencer Map is provided for each disk operation.

3.4.9.1 Format Operation

Format data is defined in the sequencer. This is the only thing that distinguises a headerless Format operation from a write operation. In the Format operation example that follows, the sequencer is identical to that of a write operation, except that the ENBUFFER (primary, SEQCTLB = '10') decode is not enabled during the data transfer. The data field for the data transfer instruction defines the pattern that will be written to the disk. The flow of a sample Format operation is presented below while the Sample Sequencer Map is found in Figure 3-16.

Assumptions:

- EDSA is being used.
- Split Data values are in the buffer.
- Disk write data comes from the sequencer.
- Sequencer will check for BUFNRDY, and Defective Sectors conditions.
- ECC block is assumed to set up properly.
- The ECC field is 21 bytes.
- EOS Counter has already been initialized.

Table 3-23 Format Operation Flow

Action	Affected Registers and Bits
Initiate seek to the target track. Modify Data Split Table to reflect defective sectors on track.	-
Update the default Sequencer Map to perform a Disk Format operation.	Set reg. 203h to 07h. Set reg. 243h to 84h. Set reg. 219h to 00h. Set reg. 259h to 00h. Set reg. 299h to 01h. Set reg. 209h to format pattern. Set reg. 249h to 06h.
Ensure the Disk Port is disabled, the ECC block is enabled. Disable the Disk Verify and Suppress Transfer functions.	Clear reg. 109h, R/W, bit 2. Set reg. 9Bh, W = 00h. Clear reg. 62h, R/W, bits 7:6.
Set up the correct CDR Write Vector address to be used when the data split occurs.	Set reg. 72h, R/W = 19h.
Clear the Disk Byte Counter and the Disk Sector Counter.	Set reg. 108h, R/W, bits 6:5.
Set the disk write direction and reset the Disk FIFO.	Set reg. 60h, R/W, bits 7:6.
Enable the Disk Port.	Set reg. 109h, R/W, bit 1.

Table 3-23 Format Operation Flow (Continued)

Determine the target frame and set the EOS Compare register accordingly.	Set reg. 5Ch to Target Frame - 1.
Determine the physical sector number of the sector following the first sector pulse in the target frame. Set the Current Sector register accordingly.	Set reg. 5Ah and 5Bh (bits 1:0) to Physical Sector Number
Disable CDR so that the CDR FIFO does not fetch table data automatically.	Clear reg 62h, bit 3.
Find the CDR table entry corresponding to Current Sector and set the Servo Pointer to its location	Set reg. 12Ch, 12Dh to the location of the first table entry for Current Sector.
Enable CDR to allow CDR FIFO fetch	Set reg. 62h, bit 3.
Determine the physical sector number of the Requested Sector and set the Requested Sector register accordingly	Set reg. 6Ch, 6Dh to physical sector number.
Set up the appropriate values in the Stop Sector Number register, the Wrap Sector Number register, and the WrapTo Sector Number register.	Set reg. 6Eh, 6Fh, R/W to the desired Stop Sector #. Set reg. 68h, 69h, R/W to the Wrap Sector Number. Set reg. 6Ah, 6Bh, R/W to the WrapTo Sector Number.
Set the Disk Counter register to the track transfer count.	Set reg. 10Dh, = Current Transfer Count.
Wait for the seek operation to complete. Start the disk sequencer at location 00h when the seek is complete.	Write reg. 73h, R/W = 80h.
Monitor the format operation for either good completion or an exception condition.	Registers 5Dh, 5Eh, 64h, 66h, and 73h all contain vital information that will indicate if the Write operation completed successfully or failed. If sequencer has stopped (reg. 66h, R, bit 0 set) with no error bits set and reg. 10Dh = 0, the Write operation completed OK.
Save any interrupt information and clear all disk interrupts.	Write a '1' to each bit set in reg. 5Eh and 66h after they have been examined and saved.
Save and clear all error status bits if the Format failed.	Clear reg. 64h, R/W, bits 6:5.
Restore sequencer to standard Write operation.	Set reg. 209h to 00h. Set reg. 249h to 16h.
Disable the Disk Port.	Clear reg. 109h, R/W, bit 1.

SEQ ADDR	ADDR LABEL	200h-21Fh R/W SEQNADAT	240h-25Fh R/W SEQCTL				280h-29Fh R/W SEQCNT	COMMENTS
00	start	82	0	11	11	000	00	Load high sync byte
01		81	1	11	00	000	FF	Wait for EOS Compare
02		7E	1	11	00	000	FF	Wair for Sector Pulse, cont. if Sector Valid
03		07	1	00	00	100	00	Initiate ECC seed, jump to Write path
04		03	0	01	00	000	00	Turn on RG, load low sync byte
05		5F	1	00	00	011	10	Wait for sync, start ECC at sync
06		0A	1	00	10	110	87	Transfer 512 bytes, enable buffer, goto ECC
07		00	0	01	00	000	07	VFO pat 0, WG on
08		03	0	00	00	011	01	Write Sync. Start ECC
09		00	0	00	00	110	87	Transfer 512 bytes
0A	ecc	82	0	11	11	111	14	ECC Field for 21 bytes, load high sync
0B		00	0	00	11	001	01	Pad, reset 2IDX, flush current CDR
0C		FF	1	10	00	000	00	Stop if EOT, reset RG/WG
0D		02	1	00	01	010	00	Jump to top of loop, inc CurrSec and ReqSec
0E		11	1	11	00	000	00	Jump if Defect Flag is set
0F		B4	1	11	00	000	00	Jump if CurrSec is not equal to ReqSec
10		14	1	00	11	000	00	Reset 2IDX because of bufnrdy
11		B4	1	11	00	000	00	Jump if CurSec not equal to ReqSec
12		00	0	00	11	000	00	Reset 2IDX because of defective sector
13		00	0	00	00	010	00	Inc ReqSec
14		02	1	00	01	001	00	Jump to top of loop. Inc CurSec, flush CDR
15	read vector	00	0	10	00	000	00	Reset RG
16		3F	0	11	00	000	10	Wait for EOS
17		7F	0	11	11	000	00	Skip Write Splice
18		FE	0	00	00	000	00	Set low sync byte
19	write vector	00	0	00	00	000	01	Skip pad
1A		00	0	10	00	000	00	Reset WG
1B		7F	0	11	11	000	00	Load high sync byte
1C		3F	1	11	00	000	10	Wait for EOS
1D		00	0	01	00	000	07	VFO field, WG on
1E		FE	0	00	00	101	00	Write sync & return

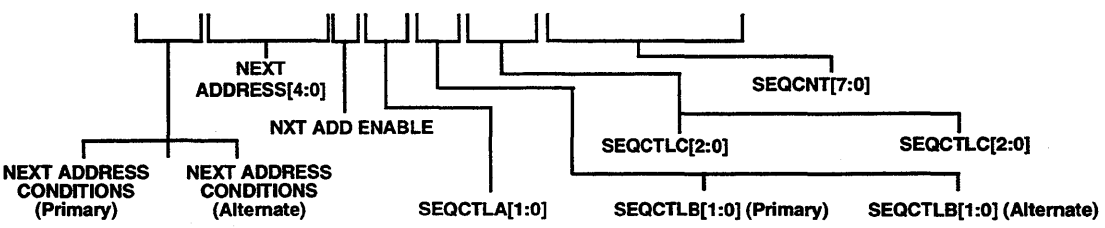

Figure 3-16 Format Disk Sequencer Map

3.4.9.2 Write Operation

The following is a sample flow for a disk Write operation accompanied by a Disk Sequencer Map in Figure 3-17 which supports this flow.

Assumptions:

- EDSA is being used.
- Split Data values are in the buffer.
- Disk write data comes from the buffer RAM.
- ADFM is assumed to be used with the disk filling a segment, BCTRB used.
- Sequencer will check for BUFNRDY, and Defective Sectors conditions.
- ECC block is assumed to set up properly.
- The ECC field is 21 bytes.
- EOS Counter has already been initialized.

Table 3-24 Write Operation Flow

Action	Affected Registers and Bits
Initiate seek to the target track. Modify Data Split Table to reflect defective sectors on track.	
Update the default Sequencer Map to perform a Disk Write operation.	Set reg. 203h to 07h. Set reg. 243h to 84h. Set reg. 219h to 00. Set reg. 259h to 00. Set reg. 299h to 01.
Ensure the Disk Port is disabled, the ECC block is enabled. Disable the Disk Verify and Suppress Transfer functions.	Clear reg. 109h, R/W, bit 2. Set reg. 9Bh, W = 00h. Clear reg. 62h, R/W, bits 7:6.
Set up the correct CDR Write Vector address to be used when the data split occurs.	Set reg. 72h, R/W = 19h.
Clear the Disk Byte Counter and the Disk Sector Counter.	Set reg. 108h, R/W, bits 6:5.
Select a disk segment and BCTR (in this example, segment B and BCTR B). Enable disk room logic.	Set reg. 137h = 07h.
Set the disk write direction and reset the Disk FIFO.	Set reg. 60h, R/W, bits 7:6.
Enable the Disk Port.	Set reg. 109h, R/W, bit 1.
Determine the target frame and set the EOS Compare register accordingly.	Set reg. 5Ch to Target Frame - 1.
Determine the Physical Sector Number of the sector following the first sector pulse in the target frame. Set the Current Sector register accordingly.	Set reg. 5Ah and 5Bh (bits 1:0) to Physical Sector Number.
Disable CDR so that the CDR FIFO does not fetch table data automatically.	Clear reg 62h, bit 3.

Table 3-24 Write Operation Flow (Continued)

Find the CDR table entry corresponding to Current Sector and set the Servo Pointer to its location.	Set reg. 12Ch, 12Dh to the location of the first table entry for Current Sector.
Enable CDR to allow CDR FIFO fetch.	Set reg. 62h, bit 3.
Determine the Physical Sector Number of the Requested Sector and set the Requested Sector register accordingly.	Set reg. 6Ch, 6Dh to physical sector number.
Set up the appropriate values in the Stop Sector Number register, the Wrap Sector Number register, and the WrapTo Sector Number register.	Set reg. 6Eh, 6Fh, R/W to the desired Stop Sector #. Set reg. 68h, 69h, R/W to the Wrap Sector Number. Set reg. 6Ah, 6Bh, R/W to the WrapTo Sector Number.
Set the Disk Counter register to the track transfer count.	Set reg. 10Dh, = Current Transfer Count.
Wait for the seek operation to complete. Start the disk sequencer at location 00h when the seek is complete.	Write reg. 73h, R/W = 80h.
Monitor the write operation for either good completion or an exception condition.	Registers 5Dh, 5Eh, 64h, 66h, and 73h all contain vital information that will indicate if the Write operation completed successfully or failed. If sequencer has stopped (reg. 66h, R, bit 0 set) with no error bits set and reg. 10Dh = 0, the Write operation completed OK.
Save any interrupt information and clear all disk interrupts.	Write a '1' to each bit set in reg. 5Eh and 66h after they have been examined and saved.
Save and clear all error status bits if the Write failed.	Clear reg. 64h, R/W, bits 6:5.
Disable the Disk Port.	Clear reg. 109h, R/W, bit 1.

SEQ ADDR	ADDR LABEL	200h-21Fh R/W SEQNADAT	240h-25Fh R/W SEQCTL				280h-29Fh R/W SEQCNT	COMMENTS
00	start	82	0	11	11	000	00	Load high sync byte
01		81	1	11	00	000	FF	Wait for EOS Compare
02		7E	1	11	00	000	FF	Wair for Sector Pulse, cont. if Sector Valid
03		07	1	00	00	100	00	Initiate ECC seed, jump to Write path
04		03	0	01	00	000	00	Turn on RG, load low sync byte
05		5F	1	00	00	011	10	Wait for sync, start ECC at sync
06		0A	1	00	10	110	87	Transfer 512 bytes, enable buffer, goto ECC
07		00	0	01	00	000	07	VFO pat 0, WG on
08		03	0	00	00	011	01	Write Sync. Start ECC
09		00	0	00	10	110	81	Transfer 512 bytes, enable buffer
0A	ecc	82	0	11	11	111	14	ECC Field for 21 bytes, load high sync
0B		00	0	00	11	001	01	Pad, reset 2IDX, flush current CDR
0C		FF	1	10	00	000	00	Stop if EOT, reset RG/WG
0D		02	1	00	01	010	00	Jump to top of loop, inc CurrSec and ReqSec
0E		11	1	11	00	000	00	Jump if Defect Flag is set
0F		B4	1	11	00	000	00	Jump if CurrSec is not equal to ReqSec
10		14	1	00	11	000	00	Reset 2IDX because of bufnrdy
11		B4	1	11	00	000	00	Jump if CurSec not equal to ReqSec
12		00	0	00	11	000	00	Reset 2IDX because of defective sector
13		00	0	00	00	010	00	Inc ReqSec
14		02	1	00	01	001	00	Jump to top of loop. Inc CurSec, flush CDR
15	read vector	00	0	10	00	000	00	Reset RG
16		3F	0	11	00	000	10	Wait for EOS
17		7F	0	11	11	000	00	Skip Write Splice
18		FE	0	00	00	000	00	Set low sync byte
19	write vector	00	0	00	00	000	01	Skip pad
1A		00	0	10	00	000	00	Reset WG
1B		7F	0	11	11	000	00	Load high sync byte
1C		3F	1	11	00	000	10	Wait for EOS
1D		00	0	01	00	000	07	VFO field, WG on
1E		FE	0	00	00	101	00	Write sync & return

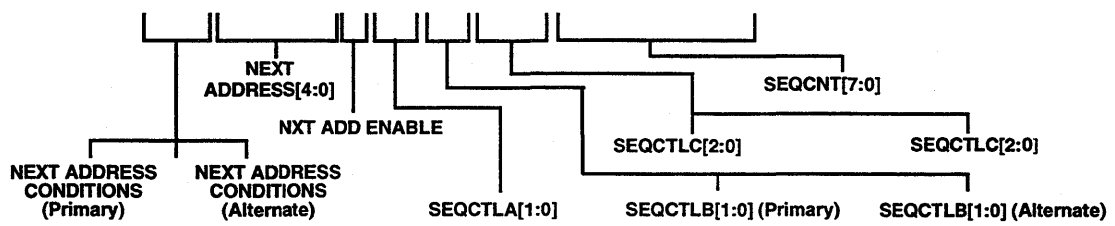

Figure 3-17 Write Disk Sequencer Map

3.4.9.3 Write Long Operation

The following is a sample flow for a Disk Write Long operation. Refer to Figure 3-18 for the sample Disk Sequencer Map which supports this flow.

Assumptions:

- EDSA is being used.
- Split Data values are in the buffer.
- Disk write data plus user defined ECC comes from the buffer RAM.
- ADFM is assumed to be used with the disk filling a segment, BCTRB used.
- Sequencer will check for BUFNRDY, and Defective Sectors conditions.
- ECC generation is disabled.
- The ECC field is 21 bytes.
- EOS Counter has already been initialized.

Table 3-25 Write Long Operation Flow

Action	Affected Registers and Bits
Initiate seek to the target track. Modify Data Split Table to reflect defective sectors on track.	-
Update the default Sequencer Map to perform a Disk Write Long operation.	Set reg. 203h to 07h. Set reg. 243h to 84h. Set reg. 219h to 00h. Set reg. 259h to 00h. Set reg. 299h to 01h. Set reg. 24Ah to 16h.
Ensure the Disk Port is disabled, the ECC block is disabled. Disable the Disk Verify and Suppress Transfer functions.	Clear reg. 109h, R/W, bit 2. Set reg. 9Bh, W = 0Bh. Clear reg. 62h, R/W, bits 7:6.
Set up the correct CDR Write Vector address to be used when the data split occurs.	Set reg. 72h, R/W = 19h.
Clear the Disk Byte Counter and the Disk Sector Counter.	Set reg. 108h, R/W, bits 6:5.
Select a disk segment and BCTR (in this example, segment B and BCTR B). Enable disk room logic.	Set reg. 137h = 07h.
Set the buffer sector size to include ECC length.	Set reg. 120h-121h = data sector size + ECC size.
Set the disk write direction and reset the Disk FIFO.	Set reg. 60h, R/W, bits 7:6.
Enable the Disk Port.	Set reg. 109h, R/W, bit 1.
Restore buffer sector size now that it has been loaded into the Disk Byte Counter.	Set reg. 120h-121h = data sector size only.
Determine the target frame and set the EOS Compare register accordingly.	Set reg. 5Ch to Target Frame - 1.

Table 3-25 Write Long Operation Flow (Continued)

Determine the Physical Sector Number of the sector following the first sector pulse in the target frame. Set the Current Sector register accordingly.	Set reg. 5Ah and 5Bh (bits 1:0) to Physical Sector Number.
Disable CDR so that the CDR FIFO does not fetch table data automatically.	Clear reg 62h, bit 3.
Find the CDR table entry corresponding to Current Sector and set the Servo Pointer to its location.	Set reg. 12Ch, 12Dh to the location of the first table entry for Current Sector.
Enable CDR to allow CDR FIFO fetch.	Set reg. 62h, bit 3.
Determine the Physical Sector Number of the Requested Sector and set the Requested Sector register accordingly.	Set reg. 6Ch, 6Dh to physical sector number.
Set up the appropriate values in the Stop Sector Number register, the Wrap Sector Number register, and the WrapTo Sector Number register.	Set reg. 6Eh, 6Fh, R/W to the desired Stop Sector #. Set reg. 68h, 69h, R/W to the Wrap Sector Number. Set reg. 6Ah, 6Bh, R/W to the WrapTo Sector Number.
Set the Disk Counter register to the track transfer count.	Set reg. 10Dh, = Current Transfer Count.
Wait for the seek operation to complete. Start the disk sequencer at location 00h when the seek is complete.	Write reg. 73h, R/W = 80h.
Monitor the Write Long operation for either good completion or an exception condition.	Registers 5Dh, 5Eh, 64h, 66h, and 73h all contain vital information that will indicate if the Write Long operation completed successfully or failed. If sequencer has stopped (reg. 66h, R, bit 0 set) with no error bits set and reg. 10Dh = 0, the Write Long operation completed OK.
Save any interrupt information and clear all disk interrupts.	Write a '1' to each bit set in reg. 5Eh and 66h after they have been examined and saved.
Save and clear all error status bits if the Write Long failed.	Clear reg. 64h, R/W, bits 6:5.
Disable the Disk Port.	Clear reg. 109h, R/W, bit 1.
Re-enable ECC logic.	Set reg. 9Bh, W = 00h.

SEQ ADDR	ADDR LABEL	200h-21Fh R/W SEQNADAT	240h-25Fh R/W SEQCTL				280h-29Fh R/W SEQCNT	COMMENTS
00	start	82	0	11	11	000	00	Load high sync byte
01		81	1	11	00	000	FF	Wait for EOS Compare
02		7E	1	11	00	000	FF	Wair for Sector Pulse, cont. if Sector Valid
03		07	1	00	00	100	00	Initiate ECC seed, jump to Write path
04		03	0	01	00	000	00	Turn on RG, load low sync byte
05		5F	1	00	00	011	10	Wait for sync, start ECC at sync
06		0A	1	00	10	110	87	Transfer 512 bytes, enable buffer, goto ECC
07		00	0	01	00	000	07	VFO pat 0, WG on
08		03	0	00	00	011	01	Write Sync. Start ECC
09		00	0	00	10	110	81	Transfer 512 bytes, enable buffer
0A	ecc	82	0	00	10	110	14	ECC field for 21 bytes, ld high sync, en buffer
0B		00	0	00	11	001	01	Pad, reset 2IDX, flush current CDR
0C		FF	1	10	00	000	00	Stop if EOT, reset RG/WG
0D		02	1	00	01	010	00	Jump to top of loop, inc CurrSec and ReqSec
0E		11	1	11	00	000	00	Jump if Defect Flag is set
0F		B4	1	11	00	000	00	Jump if CurrSec is not equal to ReqSec
10		14	1	00	11	000	00	Reset 2IDX because of bufnrdy
11		B4	1	11	00	000	00	Jump if CurSec not equal to ReqSec
12		00	0	00	11	000	00	Reset 2IDX because of defective sector
13		00	0	00	00	010	00	Inc ReqSec
14		02	1	00	01	001	00	Jump to top of loop. Inc CurSec, flush CDR
15	read vector	00	0	10	00	000	00	Reset RG
16		3F	0	11	00	000	10	Wait for EOS
17		7F	0	11	11	000	00	Skip Write Splice
18		FE	0	00	00	000	00	Set low sync byte
19	write vector	00	0	00	00	000	01	Skip pad
1A		00	0	10	00	000	00	Reset WG
1B		7F	0	11	11	000	00	Load high sync byte
1C		3F	1	11	00	000	10	Wait for EOS
1D		00	0	01	00	000	07	VFO field, WG on
1E		FE	0	00	00	101	00	Write sync & return

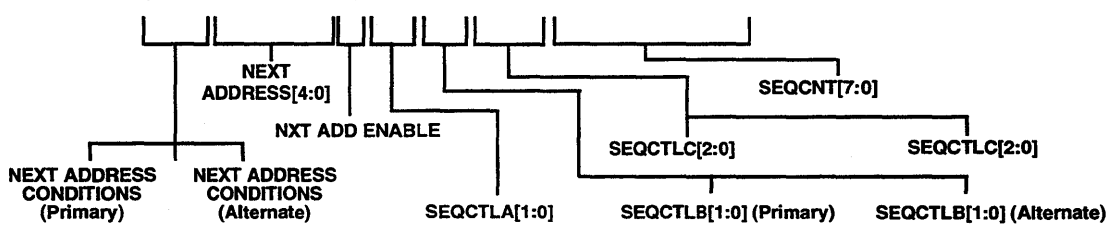

Figure 3-18 Write Long Disk Sequencer Map

3.4.9.4 Read Operation

The following is a sample flow for a basic disk Read operation accompanied by a Disk Sequencer Map in Figure 3-19 which supports this flow.

Assumptions:

- EDSA is being used.
- Split Data values are in the buffer.
- Disk read data will be transferred into the buffer RAM.
- ADFM is assumed to be used with the disk reading a segment, BCTRB used.
- Sequencer will check for BUFNRDY, and Defective Sectors conditions.
- ECC block is assumed to set up properly.
- The ECC field is 21 bytes.
- EOS Counter has already been initialized.

Table 3-26 Read Operation Flow

Action	Affected Registers and Bits
Initiate seek to the target track. Modify Data Split Table to reflect defective sectors on track.	-
Update the default Sequencer Map to perform a Disk Read operation.	Set reg. 203h to 00h. Set reg. 243h to 04h. Set reg. 219h to 5Fh. Set reg. 259h to A5h. Set reg. 299h to 10h.
Ensure the Disk Port is disabled, the ECC block is enabled. Disable the Disk Verify and Suppress Transfer functions.	Clear reg. 109h, R/W, bit 2. Set reg. 9Bh, W = 00h. Clear reg. 62h, R/W, bits 7:6.
Set up the correct CDR Vector address to be used when the data split occurs.	Set reg. 72h, R/W = 15h.
Clear the Disk Byte Counter and the Disk Sector Counter.	Set reg. 108h, R/W, bits 6:5.
Select a disk segment and BCTR (in this example, segment B and BCTR B). Enable disk room logic.	Set reg. 137h = 07h.
Set the disk read direction and reset the Disk FIFO.	Set reg. 60h, R/W, bit 6.
Enable the Disk Port.	Set reg. 109h, R/W, bit 1.
Determine the target frame and set the EOS Compare register accordingly.	Set reg. 5Ch to Target Frame - 1.
Determine the Physical Sector Number of the sector following the first sector pulse in the target frame. Set the Current Sector register accordingly.	Set reg. 5Ah and 5Bh (bits 1:0) to Physical Sector Number.
Disable CDR so that the CDR FIFO does not fetch table data automatically.	Clear reg 62h, bit 3.

Table 3-26 Read Operation Flow (Continued)

Find the CDR table entry corresponding to Current Sector and set the Servo Pointer to its location.	Set reg. 12Ch, 12Dh to the location of the first table entry for Current Sector.
Enable CDR to allow CDR FIFO fetch.	Set reg. 62h, bit 3.
Determine the Physical Sector Number of the Requested Sector and set the Requested Sector register accordingly.	Set reg. 6Ch, 6Dh to physical sector number.
Set up the appropriate values in the Stop Sector Number register, the Wrap Sector Number register, and the WrapTo Sector Number register.	Set reg. 6Eh, 6Fh, R/W to the desired Stop Sector #. Set reg. 68h, 69h, R/W to the Wrap Sector Number. Set reg. 6Ah, 6Bh, R/W to the WrapTo Sector Number.
Set the Disk Counter register to the track transfer count.	Set reg. 10Dh, = Current Transfer Count.
Wait for the seek operation to complete. Start the disk sequencer at location 00h when the seek is complete.	Write reg. 73h, R/W = 80h.
Monitor the Read operation for either good completion or an exception condition.	Registers 5Dh, 5Eh, 64h, 66h, and 73h all contain vital information that will indicate if the Read operation completed successfully or failed. If sequencer has stopped (reg. 66h, R, bit 0 set) with no error bits set and reg. 10Dh = 0, the Read operation completed OK.
If ECC is still busy after sequencer has stopped, examine the sector OK status and other error status bits for the last sector status.	If reg. 9Ch, bit 5 is set, wait until reg. 66h, bit 1 is set or an error status bit is set.
Save any interrupt information and clear all disk interrupts.	Write a '1' to each bit set in reg. 5Eh and 66h after they have been examined and saved.
Save and clear all error status bits if the Read failed.	Clear reg. 64h, R/W, bits 6:5.
Disable the Disk Port.	Clear reg. 109h, R/W, bit 1.

SEQ ADDR	ADDR LABEL	200h-21Fh R/W SEQNADAT	240h-25Fh R/W SEQCTL				280h-29Fh R/W SEQCNT	COMMENTS
00	start	82	0	11	11	000	00	Load high sync byte
01		81	1	11	00	000	FF	Wait for EOS Compare
02		7E	1	11	00	000	FF	Wair for Sector Pulse, cont. if Sector Valid
03		00	0	00	00	100	00	Initiate ECC seed
04		03	0	01	00	000	00	Turn on RG, load low sync byte
05		5F	1	00	00	011	10	Wait for sync, start ECC at sync
06		0A	1	00	10	110	87	Transfer 512 bytes, enable buffer, goto ECC
07		00	0	01	00	000	07	VFO pat 0, WG on
08		03	0	00	00	011	01	Write Sync. Start ECC
09		00	0	00	10	110	81	Transfer 512 bytes, enable buffer
0A	ecc	82	0	11	11	111	14	ECC Field for 21 bytes, load high sync
0B		00	0	00	11	001	01	Pad, reset 2IDX, flush current CDR
0C		FF	1	10	00	000	00	Stop if EOT, reset RG/WG
0D		02	1	00	01	010	00	Jump to top of loop, inc CurrSec and ReqSe
0E		11	1	11	00	000	00	Jump if Defect Flag is set
0F		B4	1	11	00	000	00	Jump if CurrSec is not equal to ReqSec
10		14	1	00	11	000	00	Reset 2IDX because of bufnrdy
11		B4	1	11	00	000	00	Jump if CurSec not equal to ReqSec
12		00	0	00	11	000	00	Reset 2IDX because of defective sector
13		00	0	00	00	010	00	Inc ReqSec
14		02	1	00	01	001	00	Jump to top of loop. Inc CurSec, flush CDR
15	read vector	00	0	10	00	000	00	Reset RG
16		3F	0	11	00	000	10	Wait for EOS
17		7F	0	11	11	000	00	Skip Write Splice
18		FE	0	00	00	000	00	Set low sync byte
19	write vector	5F	1	01	00	101	10	Wait for sync, RG on, return
1A		00	0	10	00	000	00	Reset WG
1B		7F	0	11	11	000	00	Load high sync byte
1C		3F	1	11	00	000	10	Wait for EOS
1D		00	0	01	00	000	07	VFO field, WG on
1E		FE	0	00	00	101	00	Write sync & return

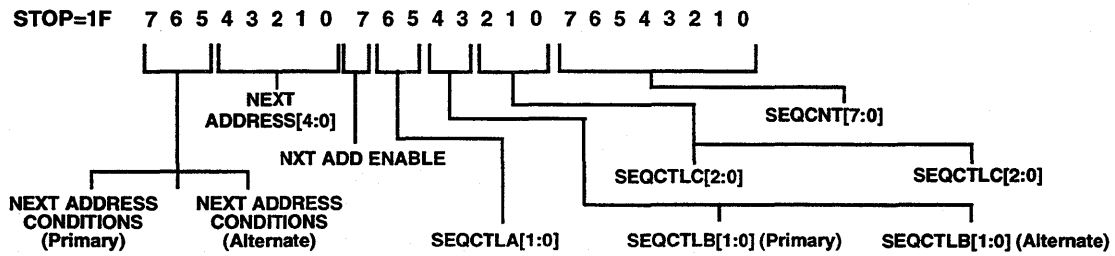

Figure 3-19 Read Disk Sequencer Map

3.4.9.5 Read Long Operation

The following is a sample flow for a disk Read Long operation. Refer to Figure 3-20 for the sample Disk Sequencer Map which supports this flow.

Assumptions:

- EDSA is being used.
- Split Data values are in the buffer.
- Disk read data will be transferred into the buffer RAM.
- ADFM is assumed to be used with the disk reading a segment, BCTRB used.
- Sequencer will check for BUFNRDY, and Defective Sectors conditions.
- ECC block is disabled.
- The ECC field is 21 bytes.
- EOS Counter has already been initialized.

Table 3-27 Read Long Operation Flow

Action	Affected Registers and Bits
Initiate seek to the target track. Modify Data Split Table to reflect defective sectors on track.	-
Update the default Sequencer Map to perform a Disk Read Long operation.	Set reg. 203h to 00h. Set reg. 243h to 04h. Set reg. 219h to 5Fh. Set reg. 259h to A5h. Set reg. 299h to 10h. Set reg. 24Ah to 17h.
Ensure the Disk Port is disabled, the ECC block is disabled. Disable the Disk Verify and Suppress Transfer functions.	Clear reg. 109h, R/W, bit 2. Set reg. 9Bh, W = 0Bh. Clear reg. 62h, R/W, bits 7:6.
Set up the correct CDR Vector address to be used when the data split occurs.	Set reg. 72h, R/W = 15h.
Clear the Disk Byte Counter and the Disk Sector Counter.	Set reg. 108h, R/W, bits 6:5.
Select a disk segment and BCTR (in this example, segment B and BCTR B). Enable disk room logic.	Set reg. 137h = 07h.
Set the disk read direction and reset the Disk FIFO.	Set reg. 60h, R/W, bit 6.
Enable the Disk Port.	Set reg. 109h, R/W, bit 1.
Determine the target frame and set the EOS Compare register accordingly.	Set reg. 5Ch to Target Frame - 1.
Determine the physical sector number of the sector following the first sector pulse in the target frame. Set the Current Sector register accordingly.	Set reg. 5Ah and 5Bh (bits 1:0) to Physical Sector Number.

Table 3-27 Read Long Operation Flow (Continued)

Disable CDR so that the CDR FIFO does not fetch table data automatically.	Clear reg 62h, bit 3.
Find the CDR table entry corresponding to Current Sector and set the Servo Pointer to its location.	Set reg. 12Ch, 12Dh to the location of the first table entry for Current Sector.
Enable CDR to allow CDR FIFO fetch	Set reg. 62h, bit 3
Determine the Physical Sector Number of the Requested Sector and set the Requested Sector register accordingly.	Set reg. 6Ch, 6Dh to Physical Sector Number.
Set the Disk Counter register to the track transfer count.	Set reg. 10Dh, = Current Transfer Count.
Wait for the seek operation to complete. Start the disk sequencer at location 00h when the seek is complete.	Write reg. 73h, R/W = 80h.
Monitor the Read Long operation for either good completion or an exception condition.	Registers 5Dh, 5Eh, 64h, 66h, and 73h all contain vital information that will indicate if the Read Long operation completed successfully or failed. If sequencer has stopped (reg. 66h, R, bit 0 set) with no error bits set and reg. 10Dh = 0, the Read Long operation completed OK.
Save any interrupt information and clear all disk interrupts.	Write a '1' to each bit set in reg. 5Eh and 66h after they have been examined and saved.
Save and clear all error status bits if the Read Long failed.	Clear reg. 64h, R/W, bits 6:5.
Disable the Disk Port.	Clear reg. 109h, R/W, bit 1.
Re-enable ECC logic.	Set reg. 9Bh, W = 00h.
Restore sequencer to standard read.	Set reg. 24Ah to 7Fh.

SEQ ADDR	ADDR LABEL	200h-21Fh R/W SEQNADAT	240h-25Fh R/W SEQCTL				280h-29Fh R/W SEQCNT	COMMENTS
00	start	82	0	11	11	000	00	Load high sync byte
01		81	1	11	00	000	FF	Wait for EOS Compare
02		7E	1	11	00	000	FF	Wair for Sector Pulse, cont. if Sector Valid
03		00	0	00	00	100	00	Initiate ECC seed
04		03	0	01	00	000	00	Turn on RG, load low sync byte
05		5F	1	00	00	011	10	Wait for sync, start ECC at sync
06		0A	1	00	10	110	87	Transfer 512 bytes, enable buffer, goto ECC
07		00	0	01	00	000	07	VFO pat 0, WG on
08		03	0	00	00	011	01	Write Sync. Start ECC
09		00	0	00	10	110	81	Transfer 512 bytes, enable buffer
0A	ecc	82	0	00	10	111	14	ECC field for 21 bytes, ld high sync, en buffe
0B		00	0	00	11	001	01	Pad, reset 2IDX, flush current CDR
0C		FF	1	10	00	000	00	Stop if EOT, reset RG/WG
0D		02	1	00	01	010	00	Jump to top of loop, inc CurrSec and ReqSe
0E		11	1	11	00	000	00	Jump if Defect Flag is set
0F		B4	1	11	00	000	00	Jump if CurrSec is not equal to ReqSec
10		14	1	00	11	000	00	Reset 2IDX because of bufnrdy
11		B4	1	11	00	000	00	Jump if CurSec not equal to ReqSec
12		00	0	00	11	000	00	Reset 2IDX because of defective sector
13		00	0	00	00	010	00	Inc ReqSec
14		02	1	00	01	001	00	Jump to top of loop. Inc CurSec, flush CDR
15	read vector	00	0	10	00	000	00	Reset RG
16		3F	0	11	00	000	10	Wait for EOS
17		7F	0	11	11	000	00	Skip Write Splice
18		FE	0	00	00	000	00	Set low sync byte
19	write vector	5F	1	01	00	101	10	Wait for sync, RG on, return
1A		00	0	10	00	000	00	Reset WG
1B		7F	0	11	11	000	00	Load high sync byte
1C		3F	1	11	00	000	10	Wait for EOS
1D		00	0	01	00	000	07	VFO field, WG on
1E		FE	0	00	00	101	00	Write sync & return

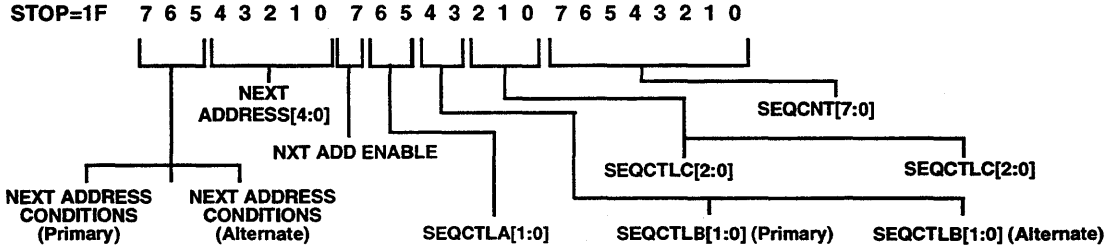

Figure 3-20 Read Long Disk Sequencer Map

3.4.9.6 Read Verify Operation

The following is a sample flow for a Disk Read Verify operation accompanied by a Disk Sequencer Map in Figure 3-19 which supports this flow.

Assumptions:

- The Read Verify operation is an ECC check only. No byte by byte compare will take place.
- ECC hardware correction is turned off.
- EDSA is being used.
- Split Data values are in the buffer.
- Sequencer will check for Defective Sectors conditions.
- ECC block is assumed to set up properly.
- The ECC field is 21 bytes.
- EOS Counter has already been initialized.

Table 3-28 Read Verify Operation Flow

Action	Affected Registers and Bits
Initiate seek to the target track. Modify Data Split Table to reflect defective sectors on track.	-
Update the default Sequencer Map to perform a Disk Read operation.	Set reg. 203h to 00h. Set reg. 243h to 04h. Set reg. 219h to 5Fh. Set reg. 259h to A5h. Set reg. 299h to 10h.
Ensure the Disk Port is disabled, the ECC block is enabled. Enable the Disk Verify and Suppress Transfer functions.	Clear reg. 109h, R/W, bit 2. Set reg. 9Bh, W = 00h. Set reg. 62h, R/W, bits 7:6.
Set up the correct CDR Vector address to be used when the data split occurs.	Set reg. 72h, R/W = 15h.
Clear the Disk Byte Counter and the Disk Sector Counter.	Set reg. 108h, R/W, bits 6:5.
Select a disk segment and BCTR (in this example, segment B and BCTR B). Disable disk room logic.	Set reg. 137h = 06h.
Disable hardware ECC correction.	Clear reg. 98h, R/W, bits 4:3.
Set the disk read direction and reset the Disk FIFO.	Set reg. 60h, R/W, bit 6.
Enable the Disk Port.	Set reg. 109h, R/W, bit 1.
Determine the target frame and set the EOS Compare register accordingly.	Set reg. 5Ch to Target Frame - 1.
Determine the physical sector number of the sector following the first sector pulse in the target frame. Set the Current Sector register accordingly.	Set reg. 5Ah and 5Bh (bits 1:0) to Physical Sector Number.

Table 3-28 Read Verify Operation Flow (Continued)

Disable CDR so that the CDR FIFO does not fetch table data automatically.	Clear reg 62h, bit 3.
Find the CDR table entry corresponding to Current Sector and set the Servo Pointer to its location.	Set reg. 12Ch, 12Dh to the location of the first table entry for Current Sector.
Enable CDR to allow CDR FIFO fetch	Set reg. 62h, bit 3.
Determine the Physical Sector Number of the Requested Sector and set the Requested Sector register accordingly.	Set reg. 6Ch, 6Dh to Physical Sector Number.
Set up the appropriate values in the Stop Sector Number register, the Wrap Sector Number register, and the WrapTo Sector Number register.	Set reg. 6Eh, 6Fh, R/W to the desired Stop Sector #. Set reg. 68h, 69h, R/W to the Wrap Sector Number. Set reg. 6Ah, 6Bh, R/W to the WrapTo Sector Number.
Set the Disk Counter register to the track transfer count.	Set reg. 10Dh, = Current Transfer Count.
Wait for the seek operation to complete. Start the disk sequencer at location 00h when the seek is complete.	Write reg. 73h, R/W = 80h.
Monitor the read verify operation for either good completion or an exception condition.	Registers 5Dh, 5Eh, 64h, 66h, and 73h all contain vital information that will indicate if the Read Verify operation completed successfully or failed. If sequencer has stopped (reg. 66h, R, bit 0 set) with no error bits set and reg. 10Dh = 0, the Read Verify operation completed OK.
Save any interrupt information and clear all disk interrupts.	Write a '1' to each bit set in reg. 5Eh and 66h after they have been examined and saved.
Save and clear all error status bits if the Read Verify failed.	Clear reg. 64h, R/W, bits 6:5.
Disable the Disk Port.	Clear reg. 109h, R/W, bit 1.
Re-enable ECC hardware correction.	Restore reg. 98h, R/W, bits 4:3.

3.5 Split Data Field/Constant Density Recording (CDR) Support

The AIC-8375 disk controller supports track formats with embedded servo bursts. It provides great flexibility in split data field/constant density recording (CDR) implementations by allowing the data and ECC fields in the track format to be split by a servo burst.

The hardware which controls the Data Split / CDR function consists of a 16-bit CDR counter and a 4-word CDR FIFO which feeds the CDR counter. The CDR FIFO is loaded with Data Split values. The Data Split/ CDR hardware and the format of the Data Split word is illustrated in Figures 3-21 and 3-22. The Data Split values are 16-bit entities which contain four fields of information. Bits 0 through 12 contain the actual Data Split count; bit 13 is the Last bit and is used to identify the last split for a sector, bit 14 is the Skip bit and is used by disk sequencer to skip the current sector if set; and bit 15 is the even parity bit. The Data Split values can originate in the Servo Segment or from the local microprocessor.

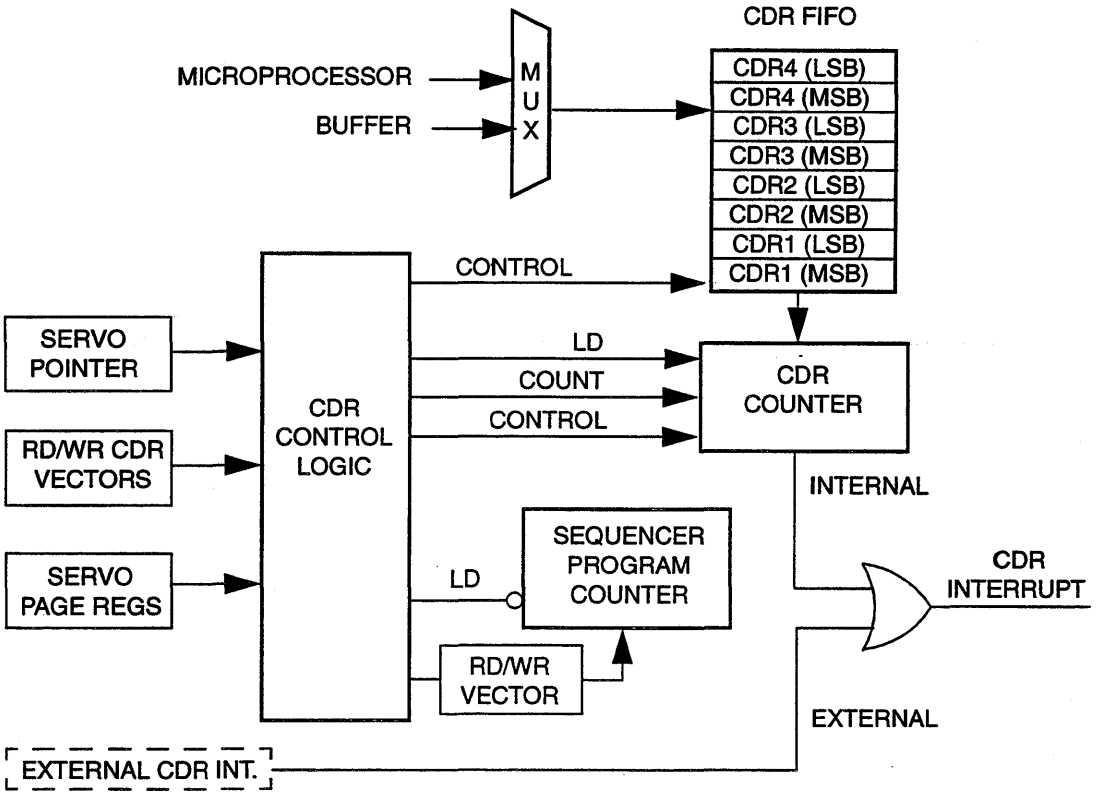

Figure 3-21 CDR Circuitry

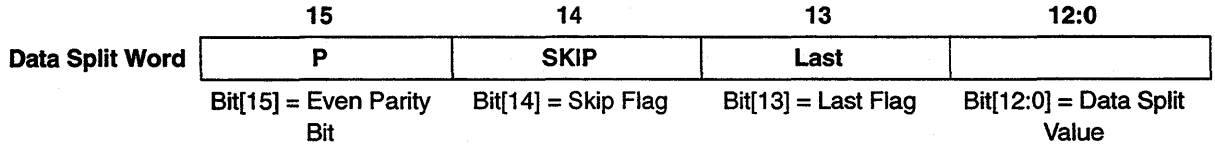

Figure 3-22 Data Split Word Format

As the Data Split values are loaded into the CDR counter from the CDR FIFO, their parity is checked. The parity checking function will occur independent of the method in which the Data Split values are loaded. If a parity error is encountered and Stop on CDR Parity Error (reg. 5Dh, bit 3) is enabled, the CDR Parity Error bit (reg. 64h, R, bit 6) will be set and the disk sequencer will be immediately forced to stop. The CDR counter is always loaded from the CDR FIFO after it has counted down to zero or if it is empty and a new Split Data value has been put into the CDR FIFO, unless the count which just expired had the Last bit set. In this case, the CDR counter will be loaded with 1FFFh. When the CDR counter counts down to zero, a CDR interrupt is generated. The data split will occur at the location equal to the Data Split value plus one. For example, a value of zero will cause a data split to occur after the first byte following the data sync. A value of FFh will cause a data split to occur after the 256th byte following the data sync.

The split data operation is controlled by the microprogram in the Disk Sequencer RAM. An internal interrupt to the disk sequencer (CDR interrupt) determines when to skip over a servo area. When the CDR interrupt occurs, sequencer program control is transferred to the CDR interrupt handling routine within the sequencer map. The CDR interrupt routine performs the steps necessary for skipping over the servo burst and then returns control to the process that was executing when the interrupt occurred.

There are two methods available to deal with split data fields and CDR. These are called the Microprocessor Load method and the Buffer Load method. The Buffer Load method is fully automated, requiring no microprocessor intervention other than initially setting the servo pointer and setting the Skip bit(s). The Microprocessor Load method requires microprocessor intervention. Both methods allow for unlimited data splits per sector.

The paragraphs that follow explain how the internal CDR interrupt is generated and how the servo burst is skipped. The term "data sync event" refers to the generation and detection of the data sync byte(s) in internal byte sync mode or the generation and detection of the Sync Enable and Sync Found signals in external sync mode.

3.5.1 Generating the CDR Interrupt

An internal CDR interrupt will be generated when the 16-bit CDR Counter underflows. The value that is loaded into the CDR Counter is a byte count representing the number of bytes from any Data Field SYNC Byte to the servo burst. This counter is loaded from the 4-word CDR FIFO when the current CDR count is zero and there is a CDR word ready in the CDR FIFO. The CDR FIFO may be loaded by the microprocessor (Microprocessor Load method) or from a protected area of the buffer RAM (Buffer Load method). The generation of the CDR interrupt will take place in the same way no matter which method of loading the CDR FIFO takes place. The Enable CDR bit (reg. 62h, R/W, bit 3) must be set to allow the CDR interrupt to be generated.

The CDR Counter will only count during the times when the Disk Sequencer Data Field decode (SEQCTLC = '110') or ECC Field decode (SEQCTLC = '111') is selected.

3.5.2 Loading Servo Split Locations From the Microprocessor

When the CDR FIFO is loaded from the microprocessor, any number of splits within a sector are possible, as long as the microprocessor keeps the CDR FIFO from becoming empty.

The CDR FIFO is loaded by writing to the CDR FIFO Data registers (reg. 78h, 79h, R/W). CDR FIFO 0 register (reg. 78h, R/W) is first loaded with the low order byte of the split data word. The upper byte of the split data word can then be loaded into the CDR FIFO 1 register (reg. 79h, R/W). Parity will be checked on every load of the CDR Counter from the CDR FIFO, unless Stop on CDR Parity Error bit (reg. 5Dh, R/W, bit 3) has been cleared. The microprocessor must make sure the Data Split word has a valid parity bit.

The CDR FIFO feeds the CDR counter and works in conjunction with the data sync events to generate a CDR interrupt as explained above. This process continues as long as the microprocessor keeps loading values into the CDR FIFO.

The Enable Reading Servo Counts From Buffer bit (reg. 63h, W, bit 1) must be cleared when operating in this mode.

3.5.3 Loading Servo Split Locations From the Buffer

The CDR FIFO can also be loaded from the buffer RAM. Before reading, writing, or formatting of the disk takes place, the servo split values for the disk are loaded into the Servo Segment in the buffer RAM. The Servo End of Segment Pointer and Beginning of Segment Pointer are adjusted to ensure they frame the proper split values. At the appropriate time, a Servo Segment fetch is initiated from the disk sequencer map. The specific data split word that is read from the Servo Segment and put into the CDR FIFO is determined by where the Servo Pointer is pointing to in the Servo Segment. The microprocessor must set the value of the Servo Pointer.

Parity is checked for every word that is loaded into the CDR Counter from the CDR FIFO. In addition, the "Skip" bit is used by the "Wait for Sector Valid" decode, and can be checked via the "Wait For Defect Flag" branch instruction. The "Last" bit is used by the CDR FIFO decode to determine if an entry is the last for a particular sector.

The Servo Overun bit (reg. 5Eh, R, bit 7) will be set if the Servo Segment fetch logic did not transfer the Data Split bytes into the CDR FIFO before the next data sync event. The Enable Reading Servo Counts From Buffer bit (reg. 63h, R/W, bit 1) must be set when operating in this mode.

When the Servo Port is enabled and buffer CDR is selected, the CDR FIFO will start attempting to fetch control words as soon as CDR is enabled. It will continue to fetch until the FIFO is full. After the transfer of the first sector, and every sector thereafter, the sequencer will execute a "Load CDR FIFO" instruction. This instruction will flush all remaining control words remaining in the CDR FIFO up to and including the next one with the "Last" bit set. This mechanism allows automatic support of a variable number of splits per sector, with the number of splits limited only by the buffer bandwidth. When the CDR FIFO is flushed, it will fetch control words referencing subsequent sectors. This fetch may take more than one byte time to conclude. The SEQCNT field for this instruction indicates the number of byte times the sequencer will wait before continuing to the next instruction. Fetching will continue until the CDR FIFO is full, regardless of whether the sequencer continued.

3.5.4 Skipping the Servo Burst During Read/Write Operations

The Disk Sequencer typically utilizes two separate CDR interrupt handling routines within the sequencer RAM. The base address of the CDR interrupt routine is indicated by the value loaded into the CDR Vector register (reg. 72h, R/W). The microprocessor must change this value according to whether a read or a write operation will be performed.

Before the sequencer branches to the CDR interrupt routine, it saves the return address to be used following completion of the CDR interrupt routine. The return address may be the address of the instruction executing (Current Address) or the current address plus one (PC + 1). If the return address is the current instruction, the sequencer saves the current Count Field value which will be restored upon the return and based upon conditions specified below.

The amount of time (bytes) spent in the CDR interrupt routine will be determined by the duration of the servo burst. The sequencer map may be programmed to either delay out the appropriate number of bytes or to wait for an End Of Servo signal indicating the servo burst has been passed. The EOS function is always active and may be sensed by the "Wait For EOS" branch instruction.

The following paragraphs explain when each of the CDR vectors are taken and the details performed within each CDR interrupt handling routine, as well as the End Of Servo Signal.

3.5.4.1 CDR Read Vector

The code in a typical CDR read interrupt routine causes Read Gate to turn off, the servo burst to be skipped, a Re-SYNC operation to occur, and then execution of a return. This CDR interrupt routine returns either to Current Address or to PC + 1 based on the Count Field Carry at the time of the CDR interrupt.

3.5.4.2 CDR Write Vector

A typical CDR write interrupt routine turns Write Gate off, skips the servo burst, writes a Re-SYNC Field and then returns. The CDR write interrupt routine returns to Current Address or PC + 1 based on the state of the Count Field Carry at the time of the CDR interrupt.

3.5.4.3 End Of Servo Input

An externally generated End Of Servo signal can be utilized by the sequencer to determine when to return from the CDR interrupt routine. It is available as an input on the EOS pin. The Branch on EOS instruction (Secondary Branch on EOS instruction '001') can then be used to detect the presence of this signal. The EOS signal will be sampled by the sequencer on a particular byte boundary during instruction execution.

3.5.5 Skipping the Servo Burst During Format Operations

During Format operations, the same data split functions are applied to allow splitting of data at the appropriate times.

Formatting When Using the Buffer Load Method:

Before formatting of the disk takes place, the servo split values for the disk are loaded into the Servo Segment in the buffer RAM. The Servo End of Segment Pointer and Beginning of Segment Pointer are adjusted to ensure they frame the proper split values.

The microprocessor manually adjusts the Servo Pointer (reg. 12Ch, 12Dh, R/W) to point to the first data split entry for that track. During the format operation, the Load CDR FIFO decode (SEQXCTLB = '11') is executed after the first and every subsequent sector (the control words for the first sector are prefetched as soon as CDR is enabled) and a Servo Segment fetch is requested. By the time the sector pulse occurs, the first data split value should be in the CDR FIFO ready to be loaded into the CDR counter.

The Enable Reading Servo Counts From Buffer bit (reg. 63h, R/W, bit 1) must be set when operating in this mode.

Just as in the case for regular write operations, the access of servo split values from the buffer RAM will be initiated as soon as CDR is enabled (as long as buffer CDR and the Servo Port are also enabled). The split locations will be loaded into the CDR FIFO and upon each subsequent data sync event, their countdown will be initiated.

Formatting When Using the Microprocessor Method:

During formatting using this method, the Data Split values are loaded into the CDR FIFO and subsequently the CDR counter by the microprocessor. The microprocessor must keep the CDR FIFO fed and always make sure that there is one Data Split value sitting in the CDR FIFO before the next sector pulse arrives.

The Enable Reading Servo Counts From Buffer bit (reg. 63h, R/W, bit 1) must be cleared when formatting in this mode.

3.5.6 External CDR Interrupt

A CDR interrupt can also be generated if the CDRINT signal on the ALE/IN2/CDRINT pin is asserted while the device is configured for external CDR mode. This mode is extablished by setting the Enable External CDR Interrupt bit (reg. 61h, R/W, bit 5) and clearing the Enable CDR bit (reg. 62h, R/W, bit 3).

Caution must be exercised while using the feature. A CDR interrupt will be generated at the same byte boundary on different external CDR occurances only if the External CDR Interrupt signal pin is asserted consistently with the same setup and hold times with respect to that byte time.

3.6 Error Detection and Correction

The AIC-8375 provides error correction and detection for the data field. The data field is protected by a user programmable Reed Solomon code. Error correction on the data field is performed automatically by the hardware.

The data ECC (EDAC) block performs the following functions:

- Generates check symbols for the data field during a data write operation.
- Generates the syndromes for the data and check fields during a data read operation.
- Calculates the error locations and patterns (if enabled and error occurs in a data field read) and handshakes with the buffer manager to perform corrections.
- Compares data and confirms check bytes during a data verification operation.

This block can perform automatic on-the-fly error correction, i.e., if a sector is in error, error correction can occur automatically while the next sector is being read in.

The primary features of the EDAC block are:

- 168-bit Reed-Solomon code:
  - Incorporates three interleaves.
  - Seven 8-bit check symbols per interleave.
  - Corrects up to three error symbols per interleave (up to nine error symbols total) in hardware before the next sector has been fully read into the buffer.
  - Programmable early warning mechanism.
  - Status kept on per-interleave basis.
  - Syndromes available to the microprocessor.
  - ECC seeding available.
- Fault-tolerant data verification (compare) operations supported.
- Clocks stopped during idle time to reduce power.

The capability and use of the EDAC block is described in the sections that follow.

3.6.1 Data Field EDAC

The data field is protected by a three way interleaved, programmable, Reed Solomon ECC. The ECC check bytes are automatically calculated by the hardware and appended to the end of the data field during a write operation. During a read operation, the data field and appended check bytes are read and checked by the hardware. Any errors that are correctable will be automatically corrected by the hardware. Upon encountering an uncorrectable error, the status will be indicated and appropriate action as programmed by the user will take place.

3.6.1.1 Data Field EDAC Specification and Capabilities

The following information summarizes the specification and capabilities of the EDAC.

General Information:

- The implementation uses a three way interleaved, Reed Solomon code. Each interleave is considered a separate subcode and is independently corrected.
- A maximum of 744 data bytes can be handled by the ECC (the maximum under the current ATA specification is 512 data bytes).

Symbols:

- 8-bit symbols are used. Each 8-bit data field byte and each ECC check byte is a symbol.

Interleaves:

- Three way interleave is implemented. The interleaves are labeled '0', '1', and '2'. Every third data byte belongs to the same interleave.
- The order of writing and reading the interleaves on the disk depends upon the size of the data field. The last byte of the user data is in interleave 0 and the next to last byte in interleave 1. In the case of an ATA compliant 512 byte sector, the first data byte is in interleave 1.

Figure 3-23 illustrates the how the interleaves are organized while Figure 3-24 shows the layout of a sector. Note that if any error falls into the unused data area, the error is uncorrectable and will NOT be corrected. If any error falls into the phantom seed region, a seed error will be generated; errors in the 512-byte sector data area are corrected in buffer (except for data verification). Errors in check symbols will not be corrected in the buffer.

---------- Patching '0's in MSBs ----------

Location in Interleave	Interleave 2	Interleave 1	Interleave 0
α^{254}	0 (743)	0 (742)	0 (741)
α^{253}	0 (740)	0 (739)	0 (738)
α^{252}	0 (737)	0 (736)	0 (735)
α^{251}	0 (734)	0 (733)	0 (732)
	:	:	:
α^{181}	0 (524)	0 (523)	0 (522)
α^{180}	0 (521)	0 (520)	0 (519)
α^{179}	0 (518)	0 (517)	0 (516)
α^{178}	0 (515)	0 (514)	0 (513)
α^{177}	0 (512)	0 0 (511)	0 1 (510)
α^{176}	0 2 (509)	0 3 (508)	0 4 (507)
α^{175}	0 5 (506)	0 6 (505)	0 7 (504)
α^{174}	0 8 (503)	0 9 (502)	0 10 (501)
α^{173}	0 11 (500)	0 12 (499)	0 13 (498)
	:	:	:
α^{13}	0 491 (20)	0 492 (19)	0 493 (18)
α^{12}	0 494 (17)	0 495 (16)	0 496 (15)
α^{11}	0 497 (14)	0 498 (13)	0 499 (12)
α^{10}	0 500 (11)	0 501 (10)	0 502 (9)
α^{9}	0 503 (8)	0 504 (7)	0 505 (6)
α^{8}	0 506 (5)	0 507 (4)	0 508 (3)
α^{7}	0 509 (2)	0 510 (1)	0 511 (0)
α^{6}	Check Symbol 0	Check Symbol 1	Check Symbol 2
α^{5}	Check Symbol 3	Check Symbol 4	Check Symbol 5
α^{4}	Check Symbol 6	Check Symbol 7	Check Symbol 8
α^{3}	Check Symbol 9	Check Symbol 10	Check Symbol 11
α^{2}	Check Symbol 12	Check Symbol 13	Check Symbol 14
α^{1}	Check Symbol 15	Check Symbol 16	Check Symbol 17
α^{0}	Check Symbol 18	Check Symbol 19	Check Symbol 20

LEGEND

(1499) Number in parentheses ( ) is the error
7 (504) location from end of data field, which is generated by the EDAC block for error correction in the buffer. The number without parentheses is data byte counting from start of data field.

Unused data area - uncorrectable region; error falling in this region will NOT be corrected.

Phantom seed bytes (if error occurs here, a seed error is generated. These bytes are NOT stored in sector.

512-byte sector data symbols - MSBs are patched with 0s. Correction will occur for data verification.

Check symbols - correctable region, but no corrections will occur in buffer.

Figure 3-23 Sector Interleave Organization

The layout of a sector with 512-byte data is shown below. Note that the phantom seed bytes are not stored regardless of whether seed injection is enabled.

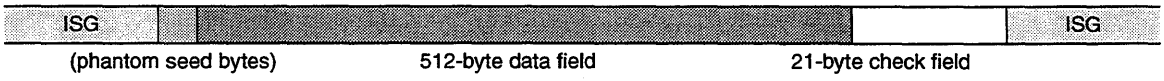

Figure 3-24 Sector Layout

Check Bits:

- 21 check bytes are used: 7 per interleave. This results in a range of 56 check bits per interleave or 168 total check bits in the ECC field.
- The 8-bit check symbols are appended after the data field bytes - each is in a specific interleave. This is illustrated in Figure 3-23.

Automatic Hardware Correction:

- A maximum of any three symbols per interleave can be corrected automatically by the hardware. The symbols may be any combination of data bytes and check bytes. Any ECC check symbols which are in error are not corrected in the buffer since the ECC check field is not put in the buffer during a read operation. A summary of the automatic hardware error correction and detection capability is shown in Table 3-29.
- Automatic hardware correction will be completed before the start of ECC check field for the sector following the sector in error. The intrinsic minimum correction time required for a sector in error is about 320 NRZ byte times.
- Should the memory bandwidth be so limited that correction is not done within a sector time, a correction overrun flag will be asserted. In this case, correction of the previous sector will continue, but only error detection can be performed on the sector just read in.

This code can correct up to three errors in each interleave while detecting four (for simplicity, seed injection is assumed to be disabled). The guaranteed correction and detection spans are as follows:

Table 3-29 Guaranteed Correctable and Detectable Error Burst Spans (#bits)

Guaranteed Error Burst Span Type	Burst Span for 1 Burst Error	Burst Span for 2 Burst Errors	Burst Span for 3 Burst Errors	Burst Span for 4 Burst Errors
Guaranteed correction error burst span (#bits) #1	65	One 17-bit error and One 41-bit error	17	N/A
Guaranteed detection error burst span (#bits)	89	Two 41-bit errors or One 65-bit error and One 17-bit error	One 41-bit error and Two 17-bit errors	Four 17-bit errors

Note #1 The above guaranteed correctable error burst spans are guaranteed for the code.

The following should be noted:

- Certain errors longer than the guaranteed correctable burst lengths are correctable. Certain errors longer than the guaranteed detectable burst length are detectable.
- The guaranteed correction capability is derived from the number of correctable error symbols in each interleave. The guaranteed detection capability is derived from the number of detectable error symbols, given the maximum number of correctable error symbols, in each interleave.
- Software correction with burst limiting can be applied to reduce the miscorrection probabilities. The burst limits need not be set to the values in Table 3-29. Better protection against miscorrection can be achieved by shorter burst limits.

3.6.1.2 ECC seed injection theory of operation

In disk drive formats using headers, the cylinder, head, and sector number for each sector written to the disk are stored in the header. The header is read before the data, assuring that the correct track and sector is being accessed. In headerless implementations, a different way is needed to store the sector address with its data. Seed injection has been developed to do this. Four seed bytes [normally cylinder, head, and sector] are added to the ECC calculation. For the purposes of the ECC calculation they are located immediately before the data, but the seed bytes are not stored on the disk - they are called phantom seed bytes (see Figures 3-23 and 3-24).

The four seed bytes are defined as follows. ECC SEED registers 80h and 81h contain the two most significant bytes, bits 31:16 (nominally cylinder). ECC SEED register 83h(82h) [bits 7:2] contains bits 15:10 of the seed [nominally head]. Bits 9:0 of the seed are not stored in a separate register, but are taken from the REQSEC registers (regs. 6Ch, 6Dh, R/W) and are automatically updated when REQSEC is updated after each sector. The ECC SEED registers (regs. 80h-83h, R/W) are not updated automatically, but are loaded by the local microprocessor whenever it is desired that they change.

Seed injection is enabled for both reading or verifying and writing by setting ENSEED (reg. 5Dh, R/W, bit 4). The sequencer decode INITECC (SEQCTLC[2:0] = '100') is issued at least 4 byte times before the sync byte is detected or written. When the INITECC decode is issued, the four seed bytes are output on the next four byte clocks. Then the sequencer decode INITDATA (SEQCTLC bits 2-0 = 011) is issued as normal for ECC block initialization before the sync and data fields. See the included sequencer maps (refer to Figures 3-16 through 3-20) for examples. When seed injection is enabled, ECC bytes are generated or corrected using the four seed bytes as the first part of the data field. The size of the data field stored on the disk is not changed.

If seed injection is disabled, the INITECC decode must not be used; only the INITDATA decode is used for sync byte and the ECC block initialization.

Note: With seed injection the same data on a different sector will have different ECC bytes.

Status when reading or verifying with seed injection:

If there is no ECC error, the data is good and the track and sector are correct. Register 66h, R, bits 5 (Uncorrectable ECC Interrupt) and 3 (Corrected ECC Interrupt) are clear, and bit 1 [Sector OK Interrupt] is set. Register 5Eh, R/W, bit 4 (Software Correctable ECC Interrupt) is clear. Register 9Ch, R, bits 7 (Current Sector In Error), 6 (Correction Overrun), and 4 (Uncorrectable ECC Error) are clear. Register 9Bh, R/W, bits 4 (Seed Error) and 5 (Error Threshold Exceeded) are clear. Register 9Dh, R, bits 1 (Software Correction Required) and 0 (Hardware Correctable Error) are clear.

If uncorrectable ECC errors are found, the sector data read is not usable, and the Uncorrectable Error bits (regs. 66h, R, bit 5 and 9Ch, R, bit 4) are set.

If errors are correctable and not in the phantom seed bytes, the seed field is confirmed and the sector data is usable after correction. The Corrected ECC Error bit (reg. 66h, R, bit 3) and the Hardware Correctable Error bit (reg. 9D, R, bit 0) are set.

If errors are correctable and at least one is in the phantom seed region, the seed is inconsistant with the sector. The Seed Error bit (reg. 9Bh, R/W, bit 4), the Corrected ECC Error bit (reg. 66h, R, bit 3), and the Hardware Correctable Error bit (reg. 9Dh, R, bit 0) are all set in this case.

If the sector was written with seed injection but read without seed injection, a seed error will result.

Error logging for correctable errors:

To assist the firmware in deciding which sectors it wishes to reassign, error logging is provided for up to two correctable sectors, but only if seed injection is enabled. While ECC Busy (reg. 9Ch, R, bit 5) is not asserted the error log threshold (ELTH[3:0], reg. 96h, R/W, bits 3:0) should be preloaded with the allowed number of correctable byte errors per sector. The maximum number of bytes that can be corrected by hardware or software is nine, three per interleave. Writing 0Fh to ELTH[3:0] disables error logging.

At the start of correction or at a correction state machine reset, the error log counter (ELOGCTR[3:0], reg. 97h, R, bits 3:0) is reset to 0. During a read or verify operation, for each sector the number of error bytes is counted in the error log counter. If error logging is enabled, and the sector is correctable and the number of bytes in error exceeds the error threshold, the 10-bit REQSEC value is saved in the corrected seed FIFO (CSEED0, CSEED1, regs. 84h, 85h, R), and bit 7 of CSEED1 (CSEEDVALID) is set. Up to two sector numbers can be saved. If the CSEED FIFO is full and another error sector is pushed, the SEEDOVRN bit (reg. 64h, R/W, bit 7) is set, but the previous CSEED contents are not overwritten. The Disk Sequencer can optionally be stopped on a seed overrun condition if the Stop On Seed Overrun bit (register 5Dh, R/W, bit 6) is set.

The local microprocessor can read the CSEED FIFO to retrieve the one or two sectors in error. When there are no more valid sector numbers in the FIFO, the CSEEDVALID bit (reg. 85h, R, bit 7) is reset.

3.6.1.3 Using Interleaves When Analyzing Or Testing ECC

Adjacent bytes in the data buffer are not in the same interleave. ECC correction is done by interleave. If data bytes are randomly corrupted, either accidentally or intentionally, it is useful to determine in which interleave they fall. Table 3-30 is a convenient way to do this for a 512 byte sector.

Table 3-30 Interleave Locations for 512-Byte Sector

Hex Offset into Sector	Interleave Number *				Byte Type
	0123	4567	89AB	CDEF	
0000	1021	0210	2102	1021	Data bytes
0010	0210	2102	1021	0210	
0020	2102	1021	0210	2102	
0030	1021	0210	2102	1021	
0040	0210	2102	1021	0210	
0050	2102	1021	0210	2102	
0060	1021	0210	2102	1021	
0070	0210	2102	1021	0210	
0080	2102	1021	0210	2102	
0090	1021	0210	2102	1021	
00A0	0210	2102	1021	0210	
00B0	2102	1021	0210	2102	
00C0	1021	0210	2102	1021	
00D0	0210	2102	1021	0210	
00E0	2102	1021	0210	2102	
00F0	1021	0210	2102	1021	
0100	0210	2102	1021	0210	Data bytes
0110	2102	1021	0210	2102	
0120	1021	0210	2102	1021	
0130	0210	2102	1021	0210	
0140	2102	1021	0210	2102	
0150	1021	0210	2102	1021	
0160	0210	2102	1021	0210	
0170	2102	1021	0210	2102	
0180	1021	0210	2102	1021	
0190	0210	2102	1021	0210	
01A0	2102	1021	0210	2102	
01B0	1021	0210	2102	1021	
01C0	0210	2102	1021	0210	
01D0	2102	1021	0210	2102	
01E0	1021	0210	2102	1021	
01F0	0210	2102	1021	0210	
0200	2102	1021	0210	2102	ECC bytes
0210	1021	0210			

* The numbers 0-F are the byte offsets for the Interleave Numbers in the columns below.

For example, assume that data bytes 54h, DBh, F0h, 18Fh, and 1C2h are corrupted. At first glance, these are random locations. But using the table will show that these bytes are all in interleave 1, and therefore this combination is uncorrectable.

However, if bytes 54h, DCh, F2h, 18Fh, and 1C3h are corrupted, the table shows that two bytes are in interleave 1, two bytes are in interleave 0, and one byte is in interleave 2. This is correctable even with 2 correctable bytes per interleave selected.

3.6.1.4 Read and Write Operation Using Data Field Hardware EDAC

The information which follows summarizes how the EDAC block is set up and used during data field EDAC write operations.

Initialization:

Before a write, read, or verify operation using data field ECC can be performed, the EDAC block must be properly configured as follows.

- The Error Log Threshold (reg. 96h, R/W) should be set as desired.
- The ECC Configuration options (reg. 98h, R/W) should be set as desired.
- The various reset and control bits in the ECC Control register (reg. 9Bh, W, bits 3:0) must be cleared.
- If seed injection is used, the Enable Seed bit (reg. 5Dh, R/W, bit 4) must be set.
- The Seed registers (reg. 08h-83h, W) should be initialized.

The Write Operation:

The EDAC write operation is controlled by the disk sequencer. As data is taken from the buffer and sent to the disk, it is also sent to the EDAC block where it is used to calculate the ECC check bytes. After the last byte in the data field is written, the EDAC block outputs the calculated ECC check symbols and they are appended to the data field. The following items must be performed to successfully write out the correct ECC check symbols.

- If seed injection is used, the sequencer decode INITECC (SEQCTLC = '100') must be issued at least 4 byte times before the sync byte is written.
- The Initialize Data Field decode (SEQCTLC = '011') must be set at the first (only) data sync event. This will initialize the ECC circuit in preparation for the data field which follows. The actual byte sync value is not included in the ECC calculation.
- The ECC Field decode (SEQCTLC = '111') must be set during the time the actual ECC check bytes are being written to the disk.
- While the ECC check symbols are being written to the disk, the ECC Busy bit (reg. 9Ch, R, bit 5) will be set.
- The Sector OK bit (reg. 66h, R, bit 1) will be set after the ECC check symbols have been written to the disk.
- After the ECC check bytes have been written to the disk, Write Gate should remain asserted a sufficient amount of time to ensure that it is not turned off before the last of the ECC check bits have made there way through any delays in the write channel path.

Ending Status and Error Conditions:

There are no ECC specific error conditions that can occur during a write operation. However, other errors unrelated to ECC may occur such as Write Fault.

The Read Operation:

The EDAC read operation is controlled by the Disk Sequencer. As the data field and ECC check symbols are being read from the disk and put into the buffer, they are also sent to the EDAC block where they are used to calculate the syndromes for each interleave. After the last check symbols are read from the disk, various status bits will be set if the sector contains any errors.

- If seed injection is used, the sequencer decode INITECC (SEQCTLC = '100') must be issued at least 4 byte times before the sync byte is detected.
- The Initialize Data Field decode (SEQCTLC = '011') must be set in the same instruction as the Wait For Sync branch instruction. This will initialize the ECC circuit in preparation for the data field which follows. If internal byte sync mode is used, the actual byte sync value will not be included in the ECC checking operation.
- The ECC Field decode (SEQCTLA = '01') must be set during the time the actual ECC check bytes are being read from the disk.
- While the ECC check symbols are being read from the disk, the ECC Busy bit (reg. 9Ch, R, bit 5) will be set.
- After a sector without error has been received in the buffer the Sector OK bit (reg. 66h, R, bit 1) will be set.

Ending Status and Error Conditions:

If the sector is good and without errors, the appropriate BCTR registers will be incremented. This action will release the sector.

Many things will happen if the sector is in error. They are summarized below.

- If the sector has an ECC error, the Current Read Sector In Error bit (reg. 9Ch, R, bit 7) will get set sometime within the last four bytes of the ECC field for the sector. The next INITID decode (SEQCTL[2:0] = '010') will clear this status.
- The Hardware Correctable Error bit (reg. 9Dh, R, bit 0) will be set if the error is correctable via the hardware. Refer to the Register description for complete information on what clears this bit.
- While the ECC error is being corrected, the ECC Busy bit (reg. 9Ch, R, bit 5) will remain set.
- The amount of time required for the automatic hardware correction to complete the correction will vary depending upon the total number of errors in the sector. It will be corrected before the ECC field of the next sector.
- The correction process for every byte in error will consist of a read buffer/modify/write buffer operation and is performed automatically by the EDAC block. The maximum number of these operations for the worse case is nine.
- After a sector with ECC errors has been corrected in the buffer the Sector OK bit (reg. 66h, R, bit 1) will be set along with the Corrected ECC Error bit (reg. 66h, R/W, bit 3).
- The Stop On Corrected ECC Error bit (reg. 7Eh, R/W, bit 3) can be used to stop the sequencer and prevent the BCTR from being incremented after a sector having an ECC error has been corrected in the buffer.

- The Software Correction Required bit (reg. 9Dh, R, bit 1) and the Software Correctable ECC Error Detected bit (reg. 5Eh, R, bit 4) will be set if the hardware can't correct the error (because the ECC Correction Threshold bits (reg. 98h, R/W, bits 4:3) have been set to less than the maximum) but it can be corrected in software. Refer to the Register Description for these bits for more information.
- During correction, the Current Interleave bits (reg. 9Dh, R, bits 7:6) will reflect the interleave that is currently being operated on. As such, these bits will be constantly changing. During the first sector read while no correction is taking place, these bits will be '11'. Corrections are done one interleave at a time.
- During correction, the Number of Errors In Current Interleave bits (reg. 9Dh, R, bits 3:2) will indicate the number of errors in the interleave currently being corrected. These bits will be changing during the correction process.
- The SWCORR bit (reg. 5Eh, R, bit 4), the UCORR bit (reg. 66h, R, bit 5), the CORRECCERR bit (reg. 66h, R, bit 3), and the SECTOROK bit (reg. 66h, R, bit 1) can all be optionally programmed as an interrupt from the disk block to the local microprocessor.

If the sector is uncorrectable, various status bits are set and actions occur as follows.

- If a sector is found to be uncorrectable in hardware, the Uncorrectable ECC Error bit (reg. 9Ch, R, bit 4) will be set.
- If a sector is found to be uncorrectable in hardware, the Uncorrectable ECC Error bit (reg. 66h, R, bit 5) will be set. In addition, if this bit gets set while the Stop On Uncorrectable ECC bit (reg. 7Eh, R/W, bit 7) is set, the disk sequencer will automatically stop.
- A sector is be determined to be uncorrectable as early as seven byte clocks after the check field.
- If a sector is found to be uncorrectable, the BCTR register associated with this transfer is not incremented.

3.6.1.5 Verify Operation

The AIC-8375 incorporates the logic necessary to perform a Read Verify operation which includes the assistance of the ECC circuitry. There are several ways of performing a Read Verify operation; the method described below involves a byte by byte comparison between data on the disk and data in the buffer with error correction turned on. This is usually termed fault tolerant verification.

To begin, the buffer must contain the data that the read data will be compared with. The Disk Verify bit (reg. 62h, R/W, bit 7) must be set to put the ECC block and disk block into a special mode to perform the verify operation. The Disk Write bit (reg. 60h, R/W, bit 7) must be cleared.

The Correction Threshold bits (reg. 98h, R/W, bits 4:3) can be used to establish a level at which mis-compares (or number of symbols per interleave that are in error) will be allowed. For example, if only one symbol needing correction per interleave is the requirement, the Correction Threshold bit field should be set to '01'.

As the read verify operation progresses, data from the disk will be compared to data from the buffer inside the EDAC block. After both the data and the check fields are compared, if there are any errors, the correction state machine will be enabled and the number of errors will be calculated. The hardware automatically keeps track of how many symbols mis-compared and the location of the symbol. If the errors are correctable and if the number of errors calculated is larger than or equal to the number internally logged for that interleave, the data is considered to be verified, otherwise, data verification has failed and the DCMPNEQ bit (reg. 5Eh, R, bit 1) will be set. If the error is determined to be uncorrectable, the UNCORR bit (reg. 66h, R, bit 5) will be set.

In the case where errors are not correctable in hardware but the number of mis-compared data bytes is still within the correction capability of the code, mis-compare status will be reported in hardware. However, the microprocessor can determine that the error is correctable via firmware and the number of ECC errors found is greater than or equal to the number of mis-compared data bytes in the corresponding interleaves.

The Verify Compare Not Equal bit (reg. 9Ch, R, bit 3) will be set when the number of defective symbols per interleave, as indicated by the Correction threshold bits (reg. 98h, R/W, bits 4:3), has been exceeded.

3.6.1.6 Read Long Operation

The Read Long operation can be implemented such that after the data field and ECC check bytes have been read into the buffer, the EDAC block will post the ECC status for the read just as if it were a regular read. To accomplish this, the read of the data field and ECC check bytes is performed with the Reset Correction State Machine bit (reg. 9Bh, W, bit 1) set. This will prevent the EDAC block from performing correction after the sector is read. The syndromes for all three interleaves are now available. They can be compared against the syndromes obtained from another read long of the same sector to check for channel noise or other unstable errors.

3.6.2 Probabilities of Uncorrectable Errors and Miscorrection

As with all error detection/correction schemes there is a probability that certain errors are detected that cannot be corrected as shown by the P_{UE} value in the tables below. These errors will be flagged as uncorrectable errors. There is also a probability that certain errors cannot be detected by the code. The error will not be reported (miscorrection would probably occur as a result) as shown by P_{UED} in the tables below.

Tables 3-31 through 3-32 use the probability values listed below:

P_E = raw burst error probability, per bit
P_{UE} = probability of uncorrectable sector
P_{MC} = miscorrection probability if uncorrectable error has occurred
P_{UED} = probability of sending host an undetected errorneous sector

Tables 3-31 and 3-32 assume there are uncorrectable errors in all three interleaves (the most likely case).

Table 3-31 Triple Error Correction Probability (3 Interleaves)

Redundancy	P_E	P_{UE}	P_{MC}	P_{UED}
56 bits / interleave	10^{-4}	8.46×10^{-4}	6.93×10^{-12}	5.86×10^{-15}
56 bits / interleave	10^{-5}	1.13×10^{-7}	6.93×10^{-12}	7.83×10^{-19}
56 bits / interleave	10^{-6}	1.17×10^{-11}	6.93×10^{-12}	8.11×10^{-23}
56 bits / interleave	10^{-7}	1.18×10^{-15}	6.93×10^{-12}	8.18×10^{-27}
56 bits / interleave	10^{-8}	1.18×10^{-19}	6.93×10^{-12}	8.18×10^{-31}

Table 3-32 Double Error Correction Probability (3 Interleaves)

Redundancy	P_E	P_{UE}	P_{MC}	P_{UED}
56 bits / interleave	10^{-4}	8.45×10^{-3}	2.31×10^{-24}	1.95×10^{-26}
56 bits / interleave	10^{-5}	1.11×10^{-5}	2.31×10^{-24}	2.56×10^{-29}
56 bits / interleave	10^{-6}	1.14×10^{-8}	2.31×10^{-24}	2.63×10^{-32}
56 bits / interleave	10^{-7}	1.14×10^{-11}	2.31×10^{-24}	2.63×10^{-35}
56 bits / interleave	10^{-8}	1.14×10^{-14}	2.31×10^{-24}	2.63×10^{-38}

Tables 3-31 and 3-32 assume there are uncorrectable errors in only one interleave.

Table 3-33 Triple Error Correction Probability (1 Interleave)

Redundancy	P_E	P_{UE}	P_{MC}	P_{UED}
56 bits / interleave	10^{-4}	8.46×10^{-4}	1.91×10^{-4}	1.62×10^{-7}
56 bits / interleave	10^{-5}	1.13×10^{-7}	1.91×10^{-4}	2.16×10^{-11}
56 bits / interleave	10^{-6}	1.17×10^{-11}	1.91×10^{-4}	2.23×10^{-15}
56 bits / interleave	10^{-7}	1.18×10^{-15}	1.91×10^{-4}	2.25×10^{-19}
56 bits / interleave	10^{-8}	1.18×10^{-19}	1.91×10^{-4}	2.25×10^{-23}

Table 3-34 Double Error Correction Probability (1 Interleave)

Redundancy	P_E	P_{UE}	P_{MC}	P_{UED}
56 bits / interleave	10^{-4}	8.45×10^{-3}	1.32×10^{-8}	1.11×10^{-10}
56 bits / interleave	10^{-5}	1.11×10^{-5}	1.32×10^{-8}	1.47×10^{-13}
56 bits / interleave	10^{-6}	1.14×10^{-8}	1.32×10^{-8}	1.50×10^{-16}
56 bits / interleave	10^{-7}	1.14×10^{-11}	1.32×10^{-8}	1.50×10^{-19}
56 bits / interleave	10^{-8}	1.14×10^{-14}	1.32×10^{-8}	1.50×10^{-22}

Software is available to perform software correction with burst limiting which will further reduce the probability of miscorrectng. Contact Adaptec for more information on using software correction.

3.7 EDSA Headerless Format Support

The EDSA headerless architecture, a method used to increase disk drive capacity, involves writing data onto the disk without the use of a traditional header. In a format that does use headers, the Disk Sequencer plays a key role in verifying and making sure that data is being written or read at the correct disk location. However, using a headerless format, the function of location verification shifts somewhat and is absorbed into other parts of the system. The microprocessor is responsible for determining the desired frame and sector based on a mapping function and a defect table.

EDSA Headerless supports Minimal Latency Read (MLR) operations and also has the ability to deal with spared and defective sectors.

This method requires an EOS (End Of Servo) pulse, used by the AIC-8375 to keep track of rotational position at all times. Once the head has arrived at the target track, the Disk Sequencer can be started. It will use the rotational position information to automatically search for the target Frame and Sector, and start the transfer when these are found. At each sector, before the read or write operation is started, the Disk Sequencer fetches Data Split words and control bits out of a special table in the buffer. This information provides Data Split values of the split locations for the current sector, and control bits pertaining to that sector. The "Last Flag" bit is used by CDR FIFO logic to determine when to stop fetching split table entries for a particular sector. The "Skip Flag" bit is used by the Disk Sequencer to decide whether the sector referenced by the control word should by transferred or skipped. This process will continue, coupled with other automated features within the Disk Sequencer and disk block, until the desired sectors on the track are all accessed.

Figure 3-25 shows the major blocks and signals used to perform EDSA Headerless. The details and requirements for each block will be explained along with several examples.

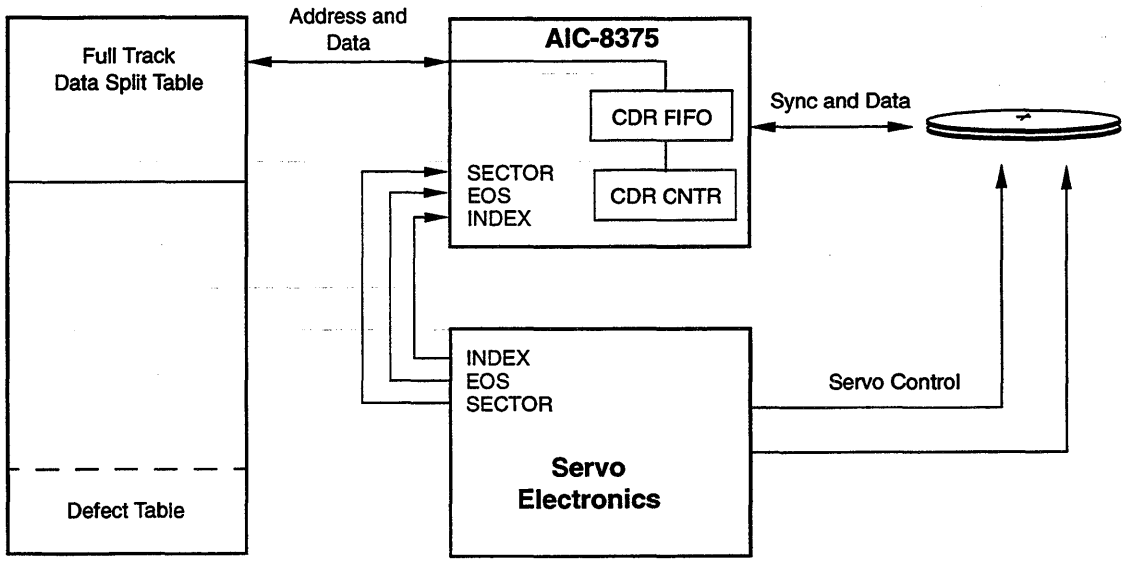

Figure 3-25 Major Blocks and Signals for EDSA Headerless

3.7.1 Details of EDSA Implementation

To implement EDSA Headerless, various bits and registers must be used and certain control structures must be in place. These will be discussed in this section. It is assumed that the reader is familiar with header based formats, CDR/Data Split techniques, and general drive controller signals.

3.7.1.1 The EOS Counter Function

The EDSA headerless method supports fully automated EOS counting in hardware. The chip will increment the EOS Counter in the EOSMAX register (reg. 58h, R/W) (2 to 3 BUFCLKs + 3 to 4 RCLKs) after each EOS pulse, wrapping when it reaches the maximum EOS count (EOSMAX, reg. 59h). The microprocessor is responsible for the initial synchronization of the counter, accomplished by writing a value to the EOS Counter register at a time the count is known. (The EOS Counter should not be written while the EOS pulse is asserted, as unpredictable results may occur.) Alternatively, on architectures that provide an Index pulse, the Enable Clear EOS Counter On Index bit (reg. 5Dh, R/W, bit 1) may be used to synchronize the EOS Counter to zero at Index time.

3.7.1.2 The Current Sector Register (Reg. 5Ah, 5Bh)

This 10-bit register is used by the Disk Sequencer and indicates which sector within a target frame the head is currently over. It is initialized by the microprocessor and can be incremented by the Disk Sequencer.

3.7.1.3 The Requested Sector Register (Reg. 6Ch, 6Dh)

This 10-bit register is used by the Disk Sequencer to determine which sector within a target frame is the first to be transferred. It is compared to the Current Sector by the Disk Sequencer. It is initialized by the microprocessor and can be incremented by the Disk Sequencer.

3.7.1.4 The EOS Signal

An EOS signal must be provided to the AIC-8375 device on the EOS pin. This signal indicates to the device that the servo burst has just been passed. The chip, through the EOS Counter, uses this signal to determine the rotational position of the disk. After the Disk Sequencer has been started for a read or write operation, it will wait for this signal to get synchronized to the rotational position of the disk.

3.7.1.5 Sector Mark

The Sector Mark is required for performing EDSA Headerless. There is no requirement that it be present at Index time if there is an index pulse.

3.7.1.6 Index Mark

The Index Mark is optional since the rotational information obtained from the EOS Counter function can provide this information.

3.7.1.7 Servo Segment

The Servo Segment is a user-specified 8-Kbyte area of buffer RAM space that is used to hold Data Split Table entries (see example Table 3-36) for use by the Disk Sequencer. Similar to other buffer Segments, the Servo Segment Pointer can wrap using Beginning of Segment (VBOS, regs. 11Ch, 11Dh) and End of Segment (VEOS, regs 11Eh, 11Fh) registers. The wrapping points will usually be used to delimit a track's worth of Data Split table entries.

3.7.1.8 Servo Page Pointer

The Servo Page Pointer determines the base address (upper 9 bits) of the Servo Segment. It is a 9-bit pointer accessible to the microprocessor via registers 112h and 113h.

3.7.2 Servo Pointer

The Servo Pointer is a 13-bit pointer into the Servo Segment used to point to the address at which the next Data Split Table entry will be read for use by the Disk Sequencer. It is a R/W register which is loaded by the microprocessor to point to the location in the Data Split Table that corresponds to the current sector.

3.7.2.1 Data Split Table

The Data Split Table is a special table which must reside in the buffer Servo Segment. It will be automatically accessed on a sector-by-sector basis by the Disk Sequencer. Each entry in the table is a 16-bit word and has a specific format. This format is shown in Table 3-35. Bits [12:0] contain the Data Split count, bit 13 the Last Split Flag, bit 14 is the Skip Flag, and bit 15 is the Parity bit.

The Data Split Table must correspond to the current zone that is being written or read. Since there are likely to be a multiple number of zones on the disk (each one with its own attributes), a Data Split Table must be available for each zone. How exactly they are stored in the Servo Segment is not covered in the User's Guide.

3.7.2.2 Data Split Word

The Data Split words are read from the Servo Segment and placed into the CDR FIFO. They are sequentially loaded into the CDR Counter where they are used to generate an internal CDR interrupt to the Disk Sequencer. Loading will continue until the CDR FIFO is full. Subsequently, each time a word is popped off the CDR FIFO, a new Data Split word will be fetched. The Load CDR FIFO Sequencer Map decode (Primary, SEQCTLC = '001') is used to initiate a fetch of the Data Split Word for each sector, flushing all entries up to and including the next word with the Last Split Flag bit set. Loading of the CDR FIFO will not take place unless the Enable CDR bit (reg. 62h, R/W, bit 3), the Servo Port Enable bit (reg. 109h, R/W, bit 4), and the Enable Reading Servo Counts From Buffer bit (reg. 63h, R/W, bit 1) are set. The Data Split Words (which make up the Data Split Table) consist of four fields as shown in Table 3-35.

Table 3-35 The Data Split Word Format

Data Split Word Field	Description
Bits [12:0]	Data Split Count Value
Bit [13]	Last Split Flag
Bit [14]	Skip Flag
Bit [15]	Parity

The Data Split Count is the byte count from a Data Sync byte until the Data Split occurs. For every data split in the sector, there should be one Data Split value in the Data Split Table. They should be ordered sequentially. If there is no split in a sector, the Data Split value should be set to 1FFFh which forces a long count such that the CDR Counter will never generate an interrupt in that sector.

The Skip Flag is used to mark any sectors to be skipped. The skipped sector may be defective or it may just be a sector that is not required for the current read or write operation. The Skip Flags in the Data Split Table entries are set or reset by the microprocessor after it ascertains which sectors on the current track are defec-

tive (by analyzing the Defect Table) and which ones will not be part of the pending read or write operation. This is done before the Disk Sequencer is started. The Disk Sequencer will fetch the Data Split Table entries in sequential order from the Servo Segment in the manner described above. The Wait for Sector Valid branch (Alternate, BRSEL = '111') is used in the Disk Sequencer to test for this Skip Flag, among other things (a full description can be found in Section 3.7.2.8). The instruction will wait for the amount of time programmed in the SEQCNT field. If the Data Split word has not yet been fetched and loaded into the CDR FIFO by this time, an error condition is flagged and the Disk Sequencer is stopped. If the sector is not valid, the Disk Sequencer must determine the reason through a series of tests, one of which uses the Wait for Defect Flag (Alternate, BRSEL = '000'). The instruction will branch to a different location based on whether or not the Skip Flag is set or reset.

The Parity bit is used to help reduce the chances of reading a corrupt Data Split word from the buffer. It is checked before the Data Split value is put into the CDR FIFO. If a Parity error is encountered while the Stop on CDR Parity Error bit (reg. 5Dh, bit 3) is enabled, an error bit will be set and the Disk Sequencer will stop.

3.7.2.3 Defect Table

The microprocessor must have access to a Defect Table which indicates the media defect locations. The data in this table will be used by the microprocessor to determine which Skip Flags must be set in the Data Split Table.

3.7.2.4 The Enable Reading Servo Counts From Buffer Bit (Reg. 63h, R/W, bit 1)

This bit must be set to allow the Disk Sequencer to fetch the buffer's Data Split Table entries.

3.7.2.5 The Wait for EOS Compare Equal Decode (Alternate, BRSEL = '100')

This decode is used to find the target Frame. While this instruction decode is being executed and EOS is asserted, it will compare the EOS Counter (reg. 58h) with the EOS Compare register (reg. 5Ch), and branch if they are equal. The value written into the EOS Compare register should be 1 less than the actual target frame number.

3.7.2.6 The Wait for Sector Valid Decode (Alternate, BRSEL = '111')

This decode is used to determine whether the Disk Sequencer has found a sector which it should transfer. The branch takes into account a number of factors. For a sector to be considered valid, the following conditions must be valid at this time:

- The internal BUFNRDY condition must by false (there must be buffer space for the transfer).
- The CURRSECEQ bit (reg. 5Bh, R/W, bit 7 (R)) must be set (Current Sector equal to Requested Sector).
- The internal CDRVALID condition must be true (a successful Data Split Word fetch must have concluded).
- The Defect Flag must not be set (the Data Split control words must not indicate the sector was invalid).

Note that these branches are enabled or disabled by clearing the SECDEF, SECRDY, and SECEQ bits in the Disk Control 2 register (reg. 62h, R/W).

If the CDR FIFO was not valid (a load of the CDR FIFO was not accomplished by the time indicated in the SEQCNT field), then the Disk Sequencer will stop, and the Servo Overrun bit (reg. 5Eh, bit 7) will be set indicating that a Servo Overrun condition has occurred.

If any of the other conditions failed, the decode will branch to NXTADR. The Disk Sequencer must then use other functions to determine the exact cause of failure. If the test does not fail, the Disk Sequencer will continue to the next instruction.

3.7.2.7 The LOAD CDR FIFO Decode (Primary, SEQCTLC = '001')

This decode is used to flush the CDR FIFO of any remaining control words referencing the last sector and CDR FIFO until a control word with the "Last" flag set is popped. As it pops, more words are transferred in from the Servo segment, so that at the conclusion of the operation the CDR FIFO contains the control words for the next sector. The CDR FIFO is not valid until at least one control word is fetched.

The CDR FIFO decode will execute for only the number of byte-times indicated in the SEQCNT field. After that, it goes on to the next instruction. However, the actual loading of the CDR FIFO may not be complete and will continue during the subsequent instruction(s). This operation will be ignored if it is attempted while the CDR FIFO is not valid. The amount of time it takes to complete the fetch and load depends on many variables and will take place as subsequent instructions are being executed.

3.7.2.8 The Wait for Defect Flag Decode (Alternate, BRSEL = '000')

This instruction will typically follow very closely after a Wait For Sector Valid decode (Alternate, BRSEL='111') has been executed in the Sequencer Map. It will test the Skip Flag in the Data Split word. If the Skip Flag is not set, this is interpreted as a sector to be read or written and execution will continue at PC + 1. If the Skip Flag is set, the sector is to be skipped and execution will continue at the address specified in the NXTADR field.

3.7.2.9 The CDR Parity Error Bit (Reg. 64h, R, bit 6)

This bit indicates a Parity error has occurred while reading the Data Split word from the Servo Segment.

3.7.2.10 The Wait for EOS Decode (Alternate, BRSEL = '001')

This instruction is used to wait for the EOS signal to be asserted before continuing execution at PC + 1. If EOS is not seen by the time-out value specified in the SEQCNT field, execution will continue at the address specified in the NXTADR field. This instruction may be executed at any time; there is no requirement that it must occur within a CDR subroutine.

3.7.2.11 The Increment Current Sector Decode (Primary, SEQCTLB = '01')

This decode is used to increment the Current Sector register (reg. 5Ah, 5Bh, R/W), indicating a physical sector has been passed.

3.7.2.12 The Increment Requested Sector Decode (Primary, SEQCTLC = '010')

This decode is used to increment the Requested Sector register, indicating a physical sector has been transferred.

3.7.3 Example EDSA Headerless Implementation

The following is an example to illustrate how the EDSA headerless method can be implemented. Note that this example is for illustration purposes only and some aspects of the track layout will be unrealistic (ie., greatly varying number of splits). Also note that this example does not support splits in the ECC field.

3.7.3.1 Example - Simple Five-Sector Read

This example describes the mechanics of EDSA Headerless via a simple five-sector read operation illustrating the search for a Target Frame, the search for a Target Sector, variable number of servo splits and skipping defects. Figure 3-26 shows the track layout for the five-sector read example. Table 3-36 shows the Data Split Table for the example. Table 3-37 lists the overall summary of events that occur during the example. A more detailed description of events are described below Table 3-37.

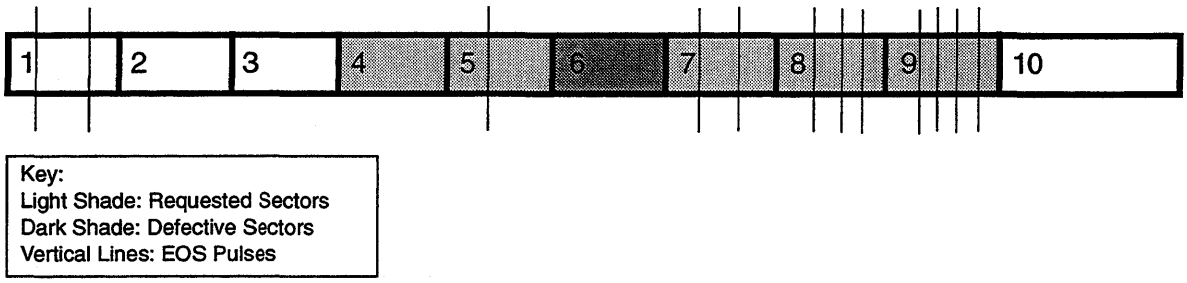

Figure 3-26 Track Layout for the Example

Table 3-36 Data Split Table for the Example

Sector	Parity Bit	Skip Flag	Last Flag	Count Field
1	1 0	0 0	0 1	0040 0020
2	0	0	1	1FFF
3	0	0	1	1FFF
4	1	0	1	1FFF
5	1	0	1	0061
6	1	1	1	1FFF
7	1 0	0 0	0 1	0040 0020
8	0 1 0	0 0 0	0 0 1	0101 0031 0001
9	0 1 0 0	0 0 0 0	0 0 0 1	0030 0043 00FF 0040

Table 3-37 Summary of Events for Example

Summary of Events in Example Five-Sector Read
Five sector read operation, with the target sector = Sector 4, the third sector of Frame 22 on the target track.
The Microprocessor determines the target frame, target sector and the physical number of the sector following the first sector pulse in the target frame.
EOS Counter function is used.
Microprocessor sets any Skip bits in the Data Skip Table for any sectors not to be read.
Start the Disk Sequencer.
Requested Sector register and Sector Stop Number register are used to stop the transfer.
Data is put contiguously into the buffer.

Detail of Events for Example Five-Sector Read:

1. The device receives the Read command for five sectors. Data is not in the buffer and a media access is required. There is enough space in the buffer for the data, as indicated by the selected BCTR.
2. The microprocessor converts the received CHS or LBA command address into a real physical address using translation information and defect tables. The real physical address consists of a Zone #, a Cylinder #, a Head #, and a Starting Sector #. In this example, assume these are Zone 1, Cylinder 100, Head 0, and Starting Sector 4, starting Frame 2.
3. The microprocessor must have a copy of the applicable Servo Skip Table in the selected Zone in the Buffer Servo Segment. The Skip bits that indicate defective sectors can be set before, during, or after the seek operation has taken place depending upon the particular application. In this example, the Skip bit for Sector 6 is set since Sector 6 is defective.
4. The microprocessor initializes the Requested Sector to the physical sector number of the desired sector (4). It initializes the EOS Compare register to one less than the EOS count for the target frame (1, because the target frame is 2). Finally, it initializes the Current Sector register to the physical sector number of the sector following the first sector mark of the target Frame (2). The stop sector is set to a value of 9.
5. The servo pointer is positioned to point to the first Data Split entry corresponding to the first sector after the first sector pulse in Frame 2. In Table 3-36 (above) that would be the second entry. The Servo Port and CDR are enabled. At this point the CDR FIFO starts fetching CDR control words.
6. Upon reaching the Target Track, the Disk Sequencer is started.
7. Assuming the head is located over sector 1, (before the first EOS pulse) at the time the Disk Sequencer is started, the EOS Counter will be 0. The Disk Sequencer will begin to Wait for EOS Compare Equal.
8. At the next EOS pulse, the EOS Counter will increment to 1. Although it is equal to the EOS Compare register at this point, the Disk Sequencer will not branch.
9. At the next EOS pulse, the EOS Counter will increment to 2. The Disk Sequencer compares the EOS Compare register against its internally latched EOS Counter register, concludes they are equal, and proceeds to the next instruction.
10. The Disk Sequencer starts a Wait for Sector Valid. At the next sector pulse, it checks the conditions for a valid sector and determines that it is not valid (because Current Sector (2) is not equal to Requested Sector (4)). The Disk Sequencer then increments Current Sector, Loads the CDR FIFO, and starts waiting for the next sector. This sequence repeats once again at the next sector pulse, when Current Sector is 3 and Requested Sector is 4.

11. At the next sector pulse, the Wait for Sector Valid decode determines that this is a valid sector (Current Sector (4) is equal to the Requested Sector (4), there is space in the buffer, the CDR FIFO is valid and the control words for this sector do not indicate a defective sector). The Disk Sequencer then transfers the sector.

12. At the end of each sector, the Disk Sequencer increments both the Requested and Current Sector registers, and loads the Data Split words for the next sector. In this case, Current Sector and Requested Sector will be 5. The Disk Sequencer loops back to waiting for Sector Valid, and as soon as it sees the next sector pulse, starts transferring the next sector.

13. After transferring sector 5, the Disk Sequencer increments both sector registers, loads the Data Split control words for sector 6, and loops back to waiting for Sector Valid. Sector 6 is defective, as indicated by its first control word. At the next sector pulse, the Wait for Sector Valid will determine it is not a valid sector. The Disk Sequencer then checks whether this is due to a defect, and discovers that it is. Because Current Sector and Requested Sector are equal (indicating that the defective sector was part of the requested transfer), it then increments both sector registers and loads the Data Split control words for the next sector.

14. From this point on, the Disk Sequencer continues transferring valid sectors until the Requester Sector is equal to the Stop Sector.

Figures and 3-27 show the example Five-Sector Read Operation Flow Charts and Figure 3-19 shows the Example Disk Sequencer Map.

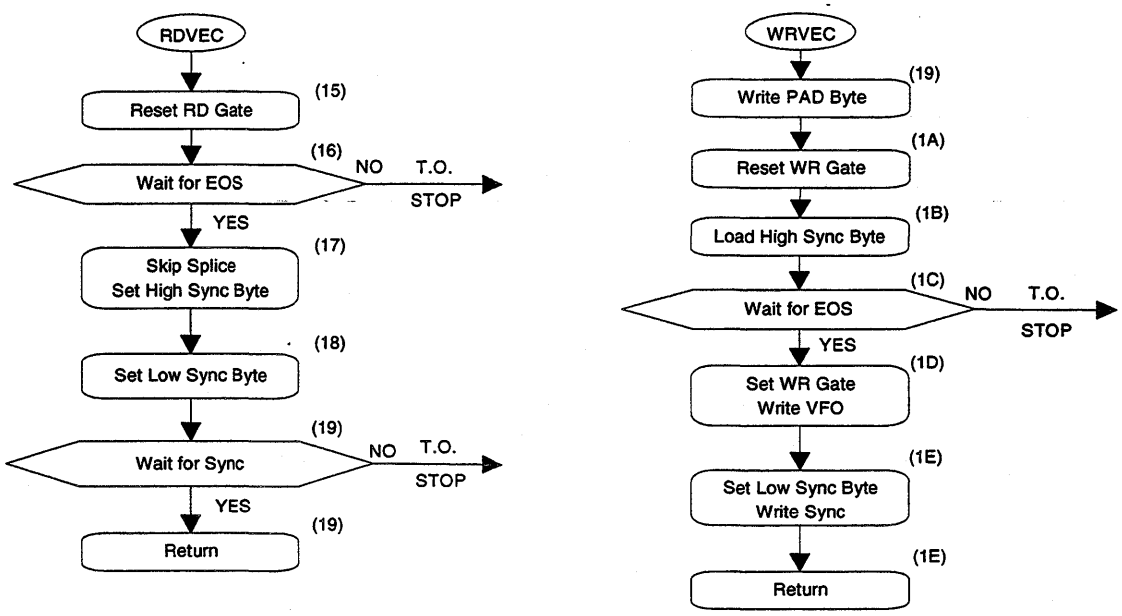

Figure 3-27 Read Operation Flow for Five-Sector Read Example

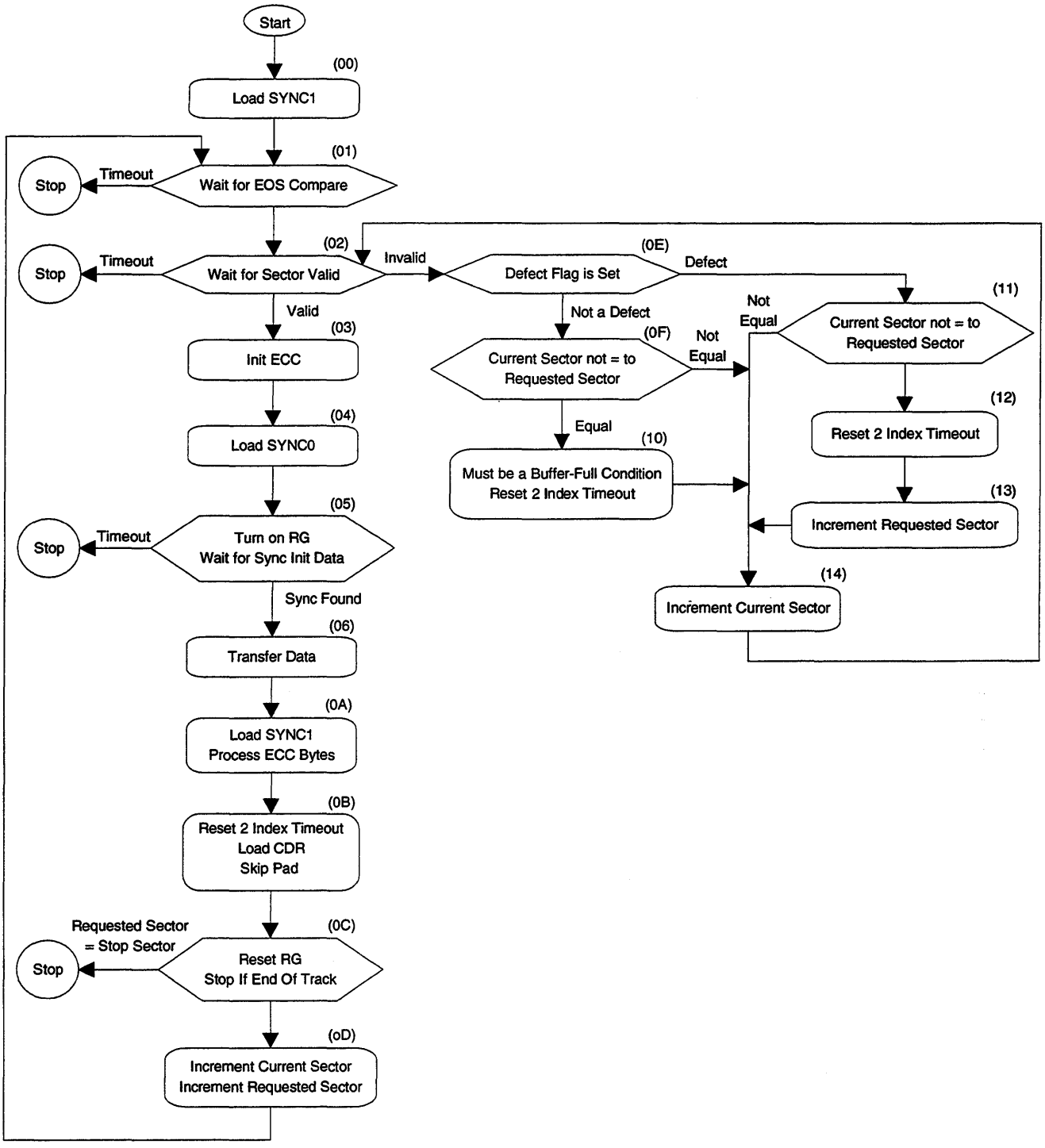

Figure 3-27 Read Operation Flow for Five-Sector Read Example (Continued)

SEQ ADDR	ADDR LABEL	200h-21Fh R/W SEQNADAT	240h-25Fh R/W SEQCTL				280h-29Fh R/W SEQCNT	COMMENTS
00	start	82	0	11	11	000	00	Load high sync byte
01		81	1	11	00	000	FF	Wait for EOS Compare
02		7E	1	11	00	000	FF	Wait for Sector Pulse, cont. if Sector Valid
03		00	0	00	00	100	00	Initiate ECC seed
04		03	0	01	00	000	00	Turn on RG, load low sync byte
05		5F	1	00	00	011	10	Wait for sync, start ECC at sync
06		0A	1	00	10	110	87	Transfer 512 bytes, enable buffer, goto ECC
07		00	0	01	00	000	07	VFO pat 0, WG on
08		03	0	00	00	011	01	Write Sync. Start ECC
09		00	0	00	10	110	81	Transfer 512 bytes, enable buffer
0A	ecc	82	0	11	11	111	14	ECC Field for 21 bytes, load high sync
0B		00	0	00	11	001	01	Pad, reset 2IDX, flush current CDR
0C		FF	1	10	00	000	00	Stop if EOT, reset RG/WG
0D		02	1	00	01	010	00	Jump to top of loop, incr CurrSec and ReqSec
0E		11	1	11	00	000	00	Jump if Defect Flag is set
0F		B4	1	11	00	000	00	Jump if CurrSec is not equal to ReqSec
10		14	1	00	11	000	00	Reset 2IDX because of Bufnrdy
11		B4	1	11	00	000	00	Jump if CurrSec not equal to ReqSec
12		00	0	00	11	000	00	Reset 2IDX because of defective sector
13		00	0	00	00	010	00	Encl ReqSec
14		02	1	00	01	001	00	Jump to top of loop. Incr CurrSec, flush CDR
15	read vector	00	0	10	00	000	00	Reset RG
16		3F	0	11	00	000	10	Wait for EOS
17		7F	0	11	11	000	00	Skip Write Splice
18		FE	0	00	00	000	00	Set low sync byte
19	write vector	5F	1	01	00	101	10	Wait for sync, RG on, return
1A		00	0	10	00	000	00	Reset WG
1B		7F	0	11	11	000	00	Load high sync byte
1C		3F	1	11	00	000	10	Wait for EOS
1D		00	0	01	00	000	07	VFO field, WG on
1E		FE	0	00	00	101	00	Write sync & return

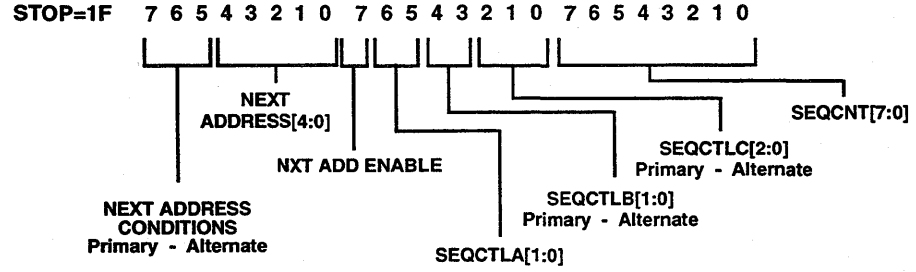

Figure 3-28 Disk Sequencer Map for Five-Sector Read Example

3.8 Automatic Data Flow Management

There are many ways of utilizing the buffer RAM for data transfers between the host and the disk. The AIC-8375 allows buffer management configurations which are completely automated, semi-automated, or manual.

Tables 3-38 and 3-39 illustrate these various configurations along with information as to how the transfer takes place.

Automated Data Flow Management (ADFM) allows for concurrent host/buffer and buffer/disk transfers out of the same segment while utilizing automation from the host, disk, and buffer blocks in order to support automated read/write operations without microprocessor intervention. This is one important method of using buffer automation and is indicated in Tables 3-38 and 3-39.

It is important that the user set up all required registers and ensure that space is available in the buffer prior to enabling automation. If enabled, the logic will automatically transfer the first sector of data (for a write operation) into the buffer. If the BCTR, Host Pointer, Disk Pointer, and other registers are not set up properly prior to enabling automation, data in the buffer may be corrupted.

The sections which follow describe how the various automated elements within the host, disk, and buffer blocks work together to provide this feature.

Table 3-38 Buffer Management Configurations for Host/Buffer Write Operations

Type Of Xfer	Host/Buffer Xfer Method	Buffer/Disk Xfer Method	Comments
$H \rightarrow B_A\ B_D \rightarrow D$	Auto Write Cmds Enabled Host Room Lo gic Enabled Auto. Multi-Sector XFERs	Disk Room Logic Enabled	Fully Automated Write Using ADFM
		Disk Room Logic Disabled Manual Buffer/Disk XFER	Semi-Automated Write
$H \rightarrow B_H\ B_D \rightarrow D$	Manual (Micro. Assisted) Host/Buffer XFER	Disk Room Logic Enabled	Semi-Automated Write
		Disk Room Logic Disabled Manual Buffer/Disk XFER	Manual Write operation
$H \rightarrow B_A\ B_D \rightarrow D$	Auto Write Cmds Enabled Host Room Logic Enabled Auto Multi-Sector XFERs	Disk Room Logic Disabled Manual Buffer/Disk XFER	Auto Write Operation While Write Cache Operation To Disk
$H \rightarrow B_H\ B_D \rightarrow D$	Manual (Micro, Assisted) Host/Buffer XFER	Disk Room Logic Enabled	Write operation while write cache operation
		Disk Room Logic Disabled Manual Buffer/Disk XFER	(Same)
$H \rightarrow B_A\ B_D \leftarrow D$	Auto Write Cmds Enabled Host Room Logic Enabled Auto Multi-Sector XFERs	Disk Room Logic Disabled Manual Buffer/Disk XFER	Auto Write Operation While Read Cache Operation From Disk
$H \rightarrow B_H\ B_D \rightarrow D$	Manual (Micro, Assisted) Host/Buffer XFER	Disk Room Logic Enabled	Write operation while read cache operation
		Disk Room Logic Disabled Manual Buffer/Disk XFER	(Same)

NOTE: B_A is the Auto-Write Segment.
B_D is the Disk Segment.
B_H is the Host Segment.

Table 3-39 Buffer Management Configurations for Host/Buffer Read Operations

Type Of Xfer	Host/Buffer Xfer Method	Buffer/Disk Xfer Method	Comments
$H \leftarrow B_H \leftarrow D$	Auto Read Cmds Enabled Host Room Logic Enabled Auto. Multi-Sector XFERs	Disk Room Logic Enabled	Fully automated Read Using ADFM
		Disk Room Logic Disabled Manual Disk/Buffer XFER	Semi-automated Read
	Manual (Micro. Assisted) Host/Buffer XFER	Disk Room Logic Enabled	Semi-automated Read
		Disk Room Logic Disabled Manual Disk/Buffer XFER	Manual Read operation
$H \leftarrow B_H \ B_D \leftarrow D$	Auto Read Cmds Enabled Host Room Logic Enabled Auto Multi-Sector XFERs	Disk Room Logic Disabled Manual Disk/Buffer XFER	Auto Read operation while Read Cache operation from disk
	Manual (Micro Assisted) Host/Buffer XFER	Disk Room Logic Enabled	Read operation while Read Cache operation from disk
		Disk Room Logic Disabled Manual Disk/Buffer XFER	(Same)
$H \leftarrow B_H \ B_D \rightarrow D$	Auto Read Cmds Enabled Host Room Logic Enabled Auto Multi-Sector XFERs	Disk Room Logic Disabled Manual Disk/Buffer XFER	Auto Read Cache operation while Write Cache operation to disk
	Manual (Micro Assisted) Host/Buffer XFER	Disk Room Logic Enabled	Read operation while Write Cache operation to disk
		Disk Room Logic Disabled Manual Disk/Buffer XFER	(Same)

NOTE: B_A is the Auto-Write Segment.
B_D is the Disk Segment.
B_H is the Host Segment.

3.8.1 All the Blocks Work Together

Analysis of a fully automated read sector(s) ATA command and a fully automated write sector(s) command utilizing ADFM will illustrate how elements of automation within the various blocks work together.

3.8.2 Boundary Conditions

There are several parameters used during ADFM transfers and general buffer automation which need to be carefully set in order to prevent erroneous operation.

- For write operations, the Maximum Buffer Count registers (BKMAXA, BKMAXB) require loading with a value which is <= [Sectors/Segment] - 1. This prevents problems at times when write problems occur. This method prevents a sector from being over written in the buffer before it can be re-written to the disk during an attempted retry algorithm.
- The BCTR is initialized to 0 to support disk read operations. It is initialized to BKMAX to support disk write operations.
- During disk read operations, when BCTR = BKMAX, the buffer is full and when BCTR = 0, the buffer is empty. During disk write operations, when BCTR = 0, the buffer is full and when BCTR = BKMAX, the buffer is empty.

3.8.3 Write Command Flow Example

The example presented in this section provides a detailed look into how an ATA Write Sectors command is executed using ADFM. Initialization, details of flow, and a Disk Sequencer Map is provided to illustrate the command execution.

Assumptions:

- ADFM is implemented using BCTRA.
- The command is enabled for Auto-Write execution.
- The Host Segment is used as a circular buffer.
- The Disk Segment is used as a circular buffer.
- 8-bit NRZ.
- 21 bytes of ECC.
- Data Split values stored in the Servo Segment.
- 16-bit wide DRAM buffer without wait states.

Table 3-40 Write Command Host Block Setup

Action	Affected Registers and Bits
Enable Auto-Write execution for the desired commands. Enable Automatic Multi-Sector Transfers.	Set reg. C2h, R/W = D5h.
Enable LBA Mode. Enable Auto Error Set and Auto Pause on Error.	Set reg. C3h, R/W = 38h.
Enable 16-Bit Host Data Transfer. Enable Auto IRQ Mode. Enable PIO Transfer Mode and Automatic Wait States. Set Data Transfer Mode for early IOCHRDY to support faster ATA Mode transfers.	Set reg. C5h, R/W, bits 6, 4, 3, 2, 0.
Set up the IRQ mode functions.	Set reg. C4h, R/W = Desired IRQ Mode functions.
Enable Automatic Task File Updates.	Clear reg. C0h, R/W, bit 6.
Set up the Host IRQ Time and Host Busy Time registers.	Set reg. BBh, W = IRQ Timeout Value. Set reg. BAh, W = Busy Time Value.
Set up the Host Block Size register for R/W Multiple Sector Per Block Size.	Set reg. B6h, R/W = Desired Sec/Blk Size.
Set up the Maximum Sector Number register. Set up the Maximum Head Number register.	Set reg. B1h, B2h, R/W = Max. Sec. # for Task File. Set reg. B3h, R/W = Max. Head # for Task File.
Set up Master/Slave Mode as required.	Set reg. C0h, R/W, bits 2:1 = Required Mode.
Enable Multiword DMA. Enable Host Data Bus to be driven early to support faster ATA mode transfers.	Set reg. C6h, R/W, bits 1:0.
Do not allow Host transfer to stop after one sector.	Clear reg. C1h, R/W, bit 2.
Enable Host Interrupts to the Microprocessor.	Set reg. 53h, R/W, bits 1:0.
Enable Host Interrupts.	Set reg. C9h, CBh, R/W = FFh.

Table 3-41 Write Command Buffer Block Setup

Action	Affected Registers and Bits
Enable Host and Disk transfer to stop on buffer error. Enable buffer parity checking.	Set reg. 105h, R/W = 01h.
Enable Servo Port, Disk Port, and Host Port check error interrupts.	Set reg. 107h, R/W = 13h.
Set BCTRA to the maximum allowable size of the Auto-Write segment. Set BKMAXA to a value smaller by two sectors.	Set reg. 114h-115h, R/W = Max. Seg. size. Set reg. 116h-117h, R/W = BCTRA-2.
Set up the Sector size.	Set reg. 120h, R/W = 00h. Set reg. 121h, R/W = 02h.
Set up the buffer RAM parameters to reflect DRAM, no wait states, 16-bit path. Disable high priority for Host port.	Set reg. 100h, R/W = 3Ch. Set reg. 101h, W = Appropriate DRAM refresh Value.
Setup miscellaneous buffer control functions as desired.	Set reg. 102h, R/W = Desired Value.
Set up the segment A Beginning Of Segment and End Of Segment.	Set reg. 138h - 13Ah = Auto-Write segment BOS. Set reg. 13Ch - 13Eh = Auto-Write segment EOS.
Select Segment A for Auto-WRITE transfers. Enable Auto-Write room logic.	Set reg. 133h, bits 7:6 = desired Auto-Write mode. Clear bits 5,4; Set bit 3.
Set the Write Cache Pointer to the beginning of the Auto-Write segment.	Set reg. 148h-14Ah, R/W = Auto-Write BOS.
Enable the Host, Microprocessor, Correction, and Servo Ports.	Set reg. 109h, R/W, bits 4, 3, 2, 0.
Enable Buffer Interrupts to the Microprocessor.	Set reg. 53h, R/W, bits 4:3.
Enable Buffer Interrupts.	Set reg. 104h, 107h, R/W = FFh.

Table 3-42 Disk Block Specific Initialization

Action	Affected Registers and Bits
Set up the Servo Segment with the Data Split Table. Set the Servo Base Pointer.	Set reg. 112h-113h = Servo Segment location.
Reset Disk FIFO, Select 8-bit NRZ.	Set reg. 60h, R/W, bits 6, 3. Clear reg. 60h, R/W, bits 7, 5:4, 2:0.
Enable 2 Byte Sync, Stop On Input Write Fault, Enable Index to Sector Branch.	Set reg. 61h, R/W, bits 7,4,3.
Enable CDR, Enable Sector Eq branch, buffer not ready branch and defective sector branch. Enable output. **Note**: Once you enable CDR, the servo FIFO will start fetching CDR values automatically if the servo port is enabled.	Set reg. 62h, R/W, bits 4:0.
Select all active high signals, select buffer CDR and enable external sync.	Set reg. 63h, R/W, bits 7:3,1:0.
Enable ECC Seeding, Stop on Seed Error, Stop on Seed Overrun and EDSA.	Set reg. 5Dh, R/W, bits 6,4,2,0.
Clear the Disk Sequencer Address register.	Set reg. 73h, R/W = 00h.
Not using Frame Counter and Index Counter.	Set reg. 74h-77h = 00h.
Enable Stop On Uncorrectable ECC Error, Stop On Input, Stop On Disk/buffer Transfer Error, and Stop On 2-Index Timeout.	Set reg. 7Eh, R/W, bits 7:5, 0.
Initialize the EOS counter. Two basic methods may be used to accomplish the: 1) writing a known value to the register when the EOS line is low, and 2) Letting the chip reset EOSCTR at index.	Method #1: Set reg. 58h to current EOS count. Method #2: Set reg. 5Dh, bit 1.
Ensure all disk interrupts are cleared.	Set reg. 5Eh, W = 00h. Set reg. 66h, W = 00h.
Enable Disk Interrupts to microprocessor.	Set reg. 53h, R/W, bits 6:5.
Enable Disk Interrupts.	Set reg. 5Fh, R/W, bits 7,0. Set reg. 67h, R/W, bits 7:5, 0.

Command Execution:

The drive receives a Write Sector(s) command from the host.

- The Auto-Write Command Started bit (reg. C8h, R/W, bit 6), the Selection Phase Detected bit (reg. C8h, R/W, bit 2), and the Auto-Write Command bit (reg. CCh, R, bit 0) will automatically be set.
- The contents of the ATA Task File registers are automatically loaded into the Image registers (reg. A8h-ACh).
- DRQ (1F7h, R, bit 7) is cleared. If the device was powered down, the host and Buffer blocks are powered up automatically.
- AT Busy (reg. 1F7h, R, bit 7) and Internal Busy (reg. C1h, R/W, bit 7) are set.
- The Host Write Operation bit (reg. C3h, R/W, bit 7) is automatically set.
- The Host Transfer Started bit (reg. CAh, R, bit 0) is cleared when the new command is received.

A SELPHDET and AWRCMD interrupt is generated via the Host 0 Interrupt register (reg. C8h, R, bits 6, 2) and the following events will occur:

- The Microprocessor clears the SELPHDET and AWRCMD interrupts (reg. C8h, W, bits 6, 2).
- The contents of the Command register (reg. A7h, R/W) and the Image registers (reg. A8h-ACh) are saved.
- The microprocessor calculates the target disk address and initiates a seek to that address.
- The microprocessor will set up the buffer to disk part of the transfer as shown Table 3-43.

Table 3-43 Write Command Flow

Action	Affected Registers and Bits
Initiate seek to the target track. Modify CDR data to reflect defective sectors on track.	-
Update the default Sequencer Map to perform a Disk Write operation.	Set reg. 203h to 07h. Set reg. 243h to 84h. Set reg. 219h to 00h. Set reg. 259h to 00h. Set reg. 299h to 01h.
Ensure the Disk Port is disabled, the ECC block is enabled. Disable the Disk Verify and Suppress Transfer functions.	Clear reg. 109h, R/W, bit 2. Set reg. 9Bh, W = 00h. Clear reg. 62h, R/W, bits 7:6.
Set up the correct CDR Write Vector address to be used when the data split occurs.	Set reg. 72h, R/W = 19h.
Clear the Disk Byte Counter and the Disk Sector Counter.	Set reg. 108h, R/W, bits 6:5.
Select a Disk segment and BCTR (in this example, segment A and BCTR A). Enable disk room logic.	Set reg. 137h = 01h.
Set the Disk Pointer to the Start of write data.	Set reg. 140h-142h = Start of write data.
Set the disk write direction and reset the Disk FIFO.	Set reg. 60h, R/W, bits 7:6.

Table 3-43 Write Command Flow (Continued)

Enable the Disk Port.	Set reg. 109h, R/W, bit 1.
Determine the target frame and set the EOS Compare register accordingly.	Set reg. 5Ch to Target Frame - 1.
Determine the physical sector number of the sector following the first sector pulse in the target frame. Set the Current Sector register accordingly.	Set reg. 5Ah and 5Bh (bits 1:0) to Physical Sector Number.
Disable CDR so that the CDR FIFO does not fetch table data automatically.	Clear reg 62h, bit 3.
Find the CDR table entry corresponding to Current Sector and set the Servo Pointer to its location.	Set reg. 12Ch, 12Dh to the location of the first table entry for Current Sector.
Enable CDR to allow CDR FIFO fetch.	Set reg. 62h, bit 3.
Determine the physical sector number of the Requested Sector and set the Requested Sector register accordingly.	Set reg. 6Ch, 6Dh to physical sector number.
Set up the appropriate values in the Stop Sector Number register, the Wrap Sector Number register, and the WrapTo Sector Number register.	Set reg. 6Eh, 6Fh, R/W to the desired Stop Sector #. Set reg. 68h, 69h, R/W to the Wrap Sector Number. Set reg. 6Ah, 6Bh, R/W to the WrapTo Sector Number.
Set the Disk Counter register to the track transfer count.	Set reg. 10Dh, R/W = Current Transfer Count.
Wait for the seek operation to complete. Start the disk sequencer at location 00h when the seek is complete.	Write reg. 73h, R/W = 80h.
Monitor the write operation for either good completion or an exception condition.	Registers 5Dh, 5Eh, 64h, 66h, and 73h all contain vital information that will indicate if the Write operation completed successfully or failed. If sequencer has stopped (reg. 66h, R, bit 0 set) with no error bits set and reg. 10Dh = 0, the Write operation completed OK.
Save any interrupt information and clear all disk interrupts.	Write a '1' to each bit set in reg. 5Eh and 66h after they have been examined and saved.
Save and clear all error status bits if the Write failed.	Clear reg. 64h, R/W, bits 6:5.
Disable the Disk Port.	Clear reg. 109h, R/W, bit 1.

Meanwhile, the host has started the data transfer and data is being written into the Auto-Write Segment that was set up. Since the Enable Automatic Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) is set and the Stop After One Block bit (reg. C1h, R/W, bit 2) is cleared, the transfer will not stop after the first sector is transferred. The transfer into the buffer will continue until completion with possible pauses along the way if the buffer ever becomes full at any time or if the Host FIFO has not kept up with the transfer.

As the data is being transferred from the host to the buffer, BCTRA will be decrementing down from its initially assigned value of BKMAXA. Host to buffer transfers will continue as long as BCTRA > 0. If BCTRA = 0, host automation will pause and resume whenever BCTRA > 0.

The disk write operation will be started and progress as follows:

- As soon as the seek is complete, the disk sequencer will be started at location 00h by writing a value of 80h to the Sequencer Address register (reg. 73h, R/W).
- The Disk Sequencer will search for the starting target Sector and when found it will take data out of the Disk Segment and write it to the data field of the current Sector.
- The sequencer will request control words from the Servo segment. The Data Split Counts will be placed in the CDR counter and used for determining where the splits in the Sector will occur.
- Before the Sequencer transfers each sector from the buffer, the Buffer status is checked via the BUFNRDY branch instruction. If the buffer is found to be ready (not empty), the sector will be transferred to the disk. If the buffer is found to be not ready (empty), the Sequencer will not start the transfer and will skip a revolution to try it again.
- The sequencer will not transfer any sectors marked as defective by the CDR table data.
- The Disk Sequencer will continue transferring sectors until finally the Requested Sector number register (reg. 6Ch-6Dh, R/W) = Stop Sector Number register (reg. 6Eh-6Fh, R/W). When this occurs, the write operation for the current track is complete and the Sequencer is halted.
- If the command requires more data to be written on subsequent tracks or heads, the sequencer needs to be updated with new track/head parameters and restarted.

As the data is transferred into and out of the buffer:

- If the command transfer length is long enough, concurrent disk/host transfers will be occurring in the buffer. Data from the host will decrement BCTRA and data to the disk will increment BCTRA.
- If the transfer length is larger than any of the segment sizes, the Disk or Host Pointers will automatically wrap from the End Of Segment Pointer value to the Beginning Of Segment value.
- During the concurrent transfer, the HNOROOM (reg. 10Bh, R, bit 6) and the DNOROOM (reg. 10Bh, R, bit 7) bits will be set and cleared as buffer full/empty conditions occur.
- Various conditions are monitored while the transfer is taking place and appropriate error bits will be set as required.
- Host block automation will automatically set and clear DRQ, AT Busy, and IRQ and update the Task File registers between each sector that is transferred.

End Of Execution:

After the last byte of data has been transferred into the host FIFO from the host, the following will occur:

- AT Busy will be set, while DRQ and IRQ will be cleared.
- Transfer Done (reg. C8h, R, bit 0) will get set (when the last byte is in the buffer).
- ATA automation will pause at this point in time.

If write caching is implemented, the microprocessor can go ahead and post the ending status before the data has been completely written to the disk. If write caching is not implemented, it must wait until the last sector has been written on the disk and then post the ending status. Assuming write caching is implemented, the microprocessor will do the following (refer to Section 3.2.5.1):

- The Task File registers will be updated. The Sector Number register (A3h, R/W) will be pointing to the last sector written.
- Clear AT Busy and Internal Busy by setting the RSTBSY bit (reg. C1h, R/W, bit 6).
- Set IRQ by setting the Host Interrupt Request bit (reg. C1h, R/W, bit 4).
- Wait for the Disk transfer to finish, check status, and clear interrupts.

After the Write command is complete, BCTRA will = BKMAXA, and the Disk and Host Pointers will have advanced by the quantity of sectors in the transfer.

3.8.4 Read Command Flow Example

The example presented in this section provides a detailed look into how an ATA Read Multiple command is executed using ADFM. Initialization, details of flow, and a Disk Sequencer Map is provided to illustrate the command execution.

Assumptions:

- ADFM is implemented using BCTRA.
- The Host Segment is used as a circular buffer.
- The Disk segment is used as a circular buffer.
- 1-bit NRZ.
- 21 bytes of ECC.
- 8-bit wide SRAM buffer using 3T mode.

Table 3-44 Read Command Host Block Setup

Action	Affected Registers and Bits
Enable LBA Mode. Enable Auto Error Set and Auto Pause on Error.	Set reg. C3h, R/W = 38h.
Enable 16-Bit Host Data Transfer. Enable Auto IRQ Mode. Enable PIO Transfer Mode and Automatic Wait States. Set Data Transfer Mode for early IOCHRDY to support faster ATA Mode transfers.	Set reg. C5h, R/W, bits 6, 4, 3, 2, 0.
Set up the IRQ mode functions.	Set reg. C4h, R/W = Desired IRQ Mode functions.
Enable Automatic Task File Updates.	Clear reg. C0h, R/W, bit 6.
Set up the Host IRQ Time and Host Busy Time registers.	Set reg. BBh, W = IRQ Timeout Value. Set reg. BAh, W = Busy Time Value.
Set up the Host Block Size register for R/W Multiple Sector Per Block Size.	Set reg. B6h, R/W = Desired Sec/Blk Size.
Set up the Maximum Sector Number register. Set up the Maximum Head Number register.	Set reg. B1h, B2h, R/W = Max. Sec. # for Task File. Set reg. B3h, R/W = Max. Head # for Task File.
Set up Master/Slave Mode as required.	Set reg. C0h, R/W, bits 2:1 = Required Mode.
Enable Multiword DMA. Enable Host Data Bus to be driven early to support faster ATA mode transfers.	Set reg. C6h, R/W, bits 1:0.
Do not allow Host transfer to stop after one sector.	Clear reg. C1h, R/W, bit 2.
Enable Host Interrupts to the Microprocessor.	Set reg. 53h, R/W, bits 1:0.
Enable Host Interrupts.	Set reg. C9h, CBh, R/W = FFh.

Table 3-45 Read Command Buffer Block Setup

Action	Affected Registers and Bits
Enable Host and Disk transfer to stop on buffer error. Enable buffer parity checking.	Set reg. 105h, R/W = 01h.
Enable Servo Port, Disk Port, and Host Port check error interrupts.	Set reg. 107h, R/W = 13h.
Set BCTRA to 0 Set BKMAXA to Segment Size - 2.	Set reg. 114h-115h, R/W = 0. Set reg. 116h-117h, R/W = BCTRA-2.
Set up the Sector size.	Set reg. 120h, R/W = 00h. Set reg. 121h, R/W = 02h.
Set up the buffer RAM parameters to reflect DRAM, no wait states, 16-bit path. Disable high priority for Host port.	Set reg. 100h, R/W = 3Ch. Set reg. 101h, W = Appropriate DRAM refresh Value.
Setup miscellaneous buffer control functions as desired.	Set reg. 102h, R/W = Desired Value.
Enable the Host, Microprocessor, Correction, and Servo Ports.	Set reg. 109h, R/W, bits 4, 3, 2, 0.
Set up the Segment A Beginning Of Segment and End Of Segment.	Set reg. 138h - 13Ah = Read segment BOS. Set reg. 13Ch - 13Eh = Read segment EOS.
Select Segment A for host transfers. Enable Host room logic.	Set reg. 133h bit 0, Clear bits 1, 2.
Set the Host Pointer to the beginning of the Read segment.	Set reg. 130h-132h, R/W = Host BOS.
Enable Buffer Interrupts to the Microprocessor.	Set reg. 53h, R/W, bits 4:3.
Enable Buffer Interrupts.	Set reg. 104h, 107h, R/W = FFh.

Table 3-46 Read Command Disk Block Setup

Action	Affected Registers and Bits
Set up the Servo Segment with the Data Split Table. Set the Servo Base Pointer.	Set reg. 112h-113h = Servo Segment location.
Reset Disk FIFO, Select 1-bit NRZ.	Set reg. 60h, R/W, bit 6. Clear reg. 60h, R/W, bits 7, 5:0.
Enable 2 Byte Sync, Stop On Input Write Fault, Enable Index to Sector Branch.	Set reg. 61h, R/W, bits 7,4,3.
Enable CDR, Enable Sector Eq branch, buffer not ready branch and defective sector branch. Enable output. Note: Once you enable CDR, the servo FIFO will start fetching CDR values automatically if the servo port is enabled.	Set reg. 62h, R/W, bits 4:0.
Select all active high signals, select buffer CDR and enable external sync.	Set reg. 63h, R/W, bits 7:3,1:0.
Enable ECC Seeding, Stop on Seed Error, Stop on Seed Overrun and EDSA.	Set reg. 5Dh, R/W, bits 6, 4, 2, 0.
Clear the Disk Sequencer Address register.	Set reg. 73h, R/W = 00h.
Not using Frame Counter and Index Counter.	Set reg. 74h-77h = 00h.
Enable Stop On Uncorrectable ECC Error, Stop On Input, Stop On Disk/buffer Transfer Error, and Stop On 2-Index Timeout.	Set reg. 7Eh, R/W, bits 7:5, 0.
Initialize the EOS counter. Two basic methods may be used to accomplish the: 1) writing a known value to the register when the EOS line is low, and 2) Letting the chip reset EOSCTR at index.	Method #1: Set reg. 58h to current EOS count. Method #2: Set reg. 5Dh, bit 1.
Ensure all disk interrupts are cleared.	Set reg. 5Eh, W = 00h. Set reg. 66h, W = 00h.
Enable Disk Interrupts to microprocessor.	Set reg. 53h, R/W, bits 6:5.
Enable Disk Interrupts.	Set reg. 5Fh, R/W, bits 7,0. Set reg. 67h, R/W, bits 7:5, 0.

Command Execution:

The drive receives a Read Multiple command from the host.

- The Read/Write Multiple Command bit (reg. CCh, R, bit 3) and the Selection Phase Detected bit (reg. C8h, R/W, bit 2) will automatically be set.
- The contents of the ATA Task File registers are automatically loaded into the Image registers (reg. A8h-ACh).
- DRQ (1F7h, R, bit 7) is cleared. If the device was powered down, the host and Buffer blocks are powered up automatically.
- AT Busy (reg. 1F7h, R, bit 7) and Internal Busy (reg. C1h, R/W, bit 7) are set.
- The Host Write Operation bit (reg. C3h, R/W, bit 7) is automatically cleared if it were set.
- The Host Transfer Started bit (reg. CAh, R, bit 0) is cleared when the new command is received.

A SELPHDET interrupt is generated via the Host 0 Interrupt register (reg. C8h, R, bits 2) and the following events will occur:

- The Microprocessor clears the SELPHDET interrupt (reg. C8h, W, bits 2).
- The contents of the Command register (reg. A7h, R/W) and the Image registers (reg. A8h-ACh) are saved.
- The microprocessor calculates the target disk address and initiates a seek to that address.

The microprocessor will set up the buffer to disk part of the transfer as shown in Table 3-47.

Table 3-47 Read Command Flow

Action	Affected Registers and Bits
Initiate seek to the target track. Modify CDR data to reflect defective sectors on track.	-
Update the default Sequencer Map to perform a Disk Read operation.	Set reg. 203h to 00h. Set reg. 243h to 04h. Set reg. 219h to 5Fh. Set reg. 259h to A5h. Set reg. 299h to 10h.
Ensure the Disk Port is disabled, the ECC block is enabled. Disable the Disk Verify and Suppress Transfer functions.	Clear reg. 109h, R/W, bit 2. Set reg. 9Bh, W = 00h. Clear reg. 62h, R/W, bits 7:6.
Set up the correct CDR Write Vector address to be used when the data split occurs.	Set reg. 72h, R/W = 19h.
Clear the Disk Byte Counter and the Disk Sector Counter.	Set reg. 108h, R/W, bits 6:5.
Select a Disk segment and BCTR (in this example, Segment A and BCTR A). Enable disk room logic.	Set reg. 137h = 01h.
Set the Disk Pointer to the same location as the Host Pointer.	Set reg. 140h-142h = Host Pointer.
Clear the disk write direction and reset the Disk FIFO.	Set reg. 60h, R/W, bits 7:6.

Table 3-47 Read Command Flow

Enable the Disk Port.	Set reg. 109h, R/W, bit 1.
Determine the target frame and set the EOS Compare register accordingly.	Set reg. 5Ch to Target Frame - 1.
Determine the Physical Sector Number of the sector following the first sector pulse in the target frame. Set the Current Sector register accordingly.	Set reg. 5Ah and 5Bh (bits 1:0) to Physical Sector Number.
Disable CDR so that the CDR FIFO does not fetch table data automatically.	Clear reg 62h, R/W, bit 3.
Find the CDR table entry corresponding to Current Sector and set the Servo Pointer to its location.	Set regs. 12Ch, 12Dh to the location of the first table entry for Current Sector
Enable CDR to allow CDR FIFO fetch.	Set reg. 62h, R/W, bit 3
Determine the physical sector number of the Requested Sector and set the Requested Sector register accordingly.	Set reg. 6Ch, 6Dh to physical sector number.
Set up the appropriate values in the Stop Sector Number register, the Wrap Sector Number register, and the WrapTo Sector Number register.	Set reg. 6Eh, 6Fh, R/W to the desired Stop Sector #. Set reg. 68h, 69h, R/W to the Wrap Sector Number. Set reg. 6Ah, 6Bh, R/W to the WrapTo Sector Number.
Set the Disk Counter register to the track transfer count.	Set reg. 10Dh, = Current Transfer Count.
Wait for the seek operation to complete. Start the disk sequencer at location 00h when the seek is complete.	Write reg. 73h, R/W = 80h.
Monitor the read operation for either good completion or an exception condition.	Registers 5Dh, 5Eh, 64h, 66h, and 73h all contain vital information that will indicate if the Read operation completed successfully or failed. If sequencer has stopped (reg. 66h, R, bit 0 set) with no error bits set and reg. 10Dh = 0, the Read operation completed OK.
Save any interrupt information and clear all disk interrupts.	Write a '1' to each bit set in reg. 5Eh and 66h after they have been examined and saved.
Save and clear all error status bits if the Read failed.	Clear reg. 64h, R/W, bits 6:5.
Disable the Disk Port.	Clear reg. 109h, R/W, bit 1.

The disk read operation will be started and progress as follows:

- As soon as the seek is complete, the disk sequencer will be started at location 00h by writing a value of 80h to the Sequencer Address register (reg. 73h, R/W).
- The Disk Sequencer will search for the starting target Sector and when found it will read the sector data field and send it to the Disk Segment.
- Before the Sequencer transfers each sector to the buffer, the Buffer status is checked via the BUFNRDY branch instruction. If the buffer is found to be ready (not full), the sector will be transferred to the buffer. If the buffer is found to be not ready (full), the Sequencer will not start the transfer and will waste a revolution and try it again.
- The Disk Sequencer will continue transferring sectors until finally the Current Sector number register (reg. 6Ch-6Dh, R/W) = Stop Sector Number register (reg. 6Eh-6Fh, R/W). When this occurs, the read operation for the current track is complete and the Sequencer is halted.
- If the command requires more data to be read on subsequent tracks or heads, the sequencer needs to be updated with new track/head parameters and restarted.

Meanwhile, the host has started taking data from the host segment. Since the Enable Automatic Multi-Sector Transfer bit (reg. C2h, R/W, bit 7) is set, the transfer will not stop after the first sector is transferred. The transfer from the buffer will continue until completion with possible pauses along the way if the buffer ever becomes empty at any time or if the Host FIFO has not kept up with the transfer.

As the data is being transferred to the host from the buffer, BCTRA will be decrementing. Buffer to host transfers will continue as long as BCTRA > HBLKSIZE. If BCTRA = HBLKSIZE, host automation will pause and resume whenever BCTRA > HBLKSIZE.

As the data is transferred into and out of the buffer:

- If the command transfer length is long enough, concurrent disk/host transfers will be occurring in the buffer. Data to the host will decrement BCTRA and data from the disk will increment BCTRA.
- If the transfer length is larger than any of the segment sizes, the Disk or Host Pointers will automatically wrap from the End Of Segment Pointer value to the Beginning Of Segment value.
- During the concurrent transfer, the HNOROOM (reg. 10Bh, R, bit 6) and the DNOROOM (reg. 10Bh, R, bit 7) bits will be set and cleared as buffer full/empty conditions occur.
- During the read operation, BCTRA will not be incremented until a sector requiring correction has been corrected in the buffer. This may take until the end of the next sector data field.
- Various conditions are monitored while the transfer is taking place and appropriate error bits will be set as required.
- Host block automation will automatically set and clear DRQ, AT Busy, and IRQ and update the Task File registers between each sector that is transferred.

End Of Execution:

After the last byte of data has been transferred to the host, the following will occur:

- AT Busy is reset and DRQ and IRQ will be cleared.
- Transfer Done (reg. C8h, R, bit 0) will get set.
- ATA automation will consider the command done.
- The Sector Number register (A3h, R/W) will be pointing to the last sector written.
- The microprocessor clears Internal Busy by setting the RSTBSY bit (reg. C1h, R/W, bit 6).
- The microprocessor checks status, and clears any interrupts.

After the Read command is complete, BCTRA will = 0, and the Disk and host Pointers will have advanced by the quantity of sectors in the transfer.

3.9 Power Management

The AIC-8375 has an assortment of features that can be used to reduce power consumption. Some aspects of power management are controlled by the microprocessor, some are automatically controlled by the internal hardware, and others are utilized as required on the printed circuit board. From the power management perspective, the chip is divided into three blocks; the Disk block, the Buffer Block, and the Host block. When automatic power-down is enabled, appropriate internal clocks in the affected blocks are turned off when various circuitry is not being used, thus significantly reducing the current consumption of that block. Additionally, the chip features logic which enables it to power up automatically upon receipt of a Host command or a Host Reset.

3.9.1 Power Down Modes

Chip power-down modes are controlled via the Chip Reset & Power-down Control register (reg. 50h, R/W). Within this register, each of the three blocks has a corresponding bit, which may be set independently to enable automatic power-down of that block. Clearing the bit will disable the automatic power-down function for that block and the internal clocks will continue to run. Each bit is internally synchronized so that the internal clocks do not glitch when being powered up or down.

The Host block and the Buffer block are automatically powered-up upon receipt of a Host Command, a Host Reset, or Power On Reset.

The bits which control power management are listed in Table 3-48.

Table 3-48 Summary of Available Power Management Control Bits

Bit Name	Location
Host Block Power-Down Enable	Reg. 50h, R/W, bit 3
Buffer Block Power-Down Enable	Reg. 50h, R/W, bit 4
Disk Block Power-Down Enable	Reg. 50h, R/W, bit 5
Disable Output and I/O Pullups	Reg. 50h, R/W, bit 7
Disable Buffer Data Bus Pullups	Reg. 51h, R/W, bit 4
Disable Host Control Pullups	Reg. 51h, R/W, bit 6
Disable Host Address Pullups	Reg. 51h, R/W, bit 7

3.9.1.1 Disk Block Power Management

The Disk block power down mode is directly controlled by the Disk Block Power-Down Enable bit (reg. 50h, R/W, bit 5). While the Disk Block Power-Down Enable bit is set, any clocks in the Disk block not required for the current operation are automatically turned off. The disk block consumes the least power in this mode.

The Disk block is automatically powered up when any one of the following activities occurs:

- The disk sequencer is started via the Sequencer Run bit (reg. 73h, R/W, bit 7).
- ECC (H/W or S/W) correction is occurring.

When the microprocessor sets the SEQRUN bit, the internal setting of that signal is delayed by 8 clock cycles so that the clock to the sequencer is turned back on before the sequencer is actually started. It will take approximately 2 to 3 internal byte clocks after this bit is set before the sequencer starts running.

The Disk block is automatically powered down when the disk sequencer is stopped by the microprocessor or by program execution at address 1Fh.

When the Sequencer Run bit is negated (either via the microprocessor or sequencer stopped), clocks are maintained for 8 cycles before the clocks are disabled.

Clocks in the Disk and ECC blocks are enabled when the sequencer is running. There is no delay for ECC correction or microprocessor accesses when clocks are enabled for these activities.

Some Disk Block input signal pins incorporate pullup resistors. These pullups eliminate excessive current consumption during times when the output signal pins are not driven. This prevents the CMOS output structure from consuming current if the pin is floating. The internal pullup structures on these pins can be turned off if the Disable All Pull-Ups bit (reg. 50h, R/W, bit 7) is set. This allows the user to use an external pullup scheme if desired.

3.9.1.2 Buffer Block Power Management

The Buffer block can be enabled for automatic power-down independently of the Disk and Host blocks. Automatic power-down is invoked by setting the Buffer Block Power Down Enable bit (reg. 50h, R/W, bit 4). This bit is synchronized such that internal clock glitches will not occur when this bit changes states.

When the Buffer Block Power Down Enable bit (reg. 50h, R/W, bit 4) is set, the Buffer block will be powered down whenever it is idle, and will automatically be powered up whenever the Disk or Host blocks are active or whenever the microprocessor accesses the buffer.

If operating in DRAM mode, the DRAM refresh counter will not be powered down while the device is idle. However, when the refresh counter counts down to zero, the refresh ciruitry will be powered up and a refresh cycle will occur. After the refresh cycle has occured, the refresh ciruictry will be powered down until the next refresh time.

The Buffer Block BD[15:0] signal pins incorporate pullup resistors. These pullups eliminate excessive current consumption during times when the BD[15:0] signal pins are not driven. This prevents the CMOS output structure from consuming current if the pin is floating. The internal pullup structures on these pins can be turned off if the Disable Buffer Data Bus Pull-Ups bit (reg. 51h, R/W, bit 4) is set. This allows the user to use an external pullup scheme if desired.

3.9.1.3 Host Block Power Management

The Host block can be enabled for automatic power-down independently of the Disk and Buffer blocks. This is done by setting the Host Block Power-Down Enable bit (reg. 50h, R/W, bit 3). This bit is synchronized such that internal clock glitches will not occur when this bit changes states.

In order to support Auto-Write and Auto-Read commands, the Host block will be powered up automatically whenever the AIC-8375 receives either a Host Reset or any Read/Write command which has been enabled for Auto operation. In addition, the Host block will be powered up whenever any command is received from the Host.

The Host block will also power up when the microprocessor writes to a synchronous register or whenever there is a data transfer, i.e., the Host transfer state machine on, which include Auto Read/Writes and Read/Write data transfers that are started manually.

The Host block *PDIAG and *DASP I/O signal pins incorporate pullup resistors. These pullups eliminate excessive current consumption during times when the signal pins are not driven. This prevents the CMOS output structure from consuming current if the pin is floating. The internal pullup structures on these pins can be turned off if the Disable Host Control Pull-Ups bit (reg. 51h, R/W, bit 6) is set. In addition, the HA[2:0] signal input pins have pullup structures also which can be turned off if the Disable Host Address Pull-Ups bit (reg. 51h, R/W, bit 7) is set. Both of these bits allow the user to use an external pullup scheme if desired.

The other host interface signal pins incorporate internal weak pullups. Thus, when the Host interface is floating, for example, during system standby, the CMOS inputs on the AIC-8375 will be held high, restricting current drawn by the CMOS input transistors.

This page intentionally left blank.

4.1 Overview Of The Disk Sequencer Map

The AIC-8375 Disk Sequencer Map (DSM) controls the flow of Data between the external ENDEC and the Buffer Memory. The DSM is a programmable memory which stores user specific instructions to control the data flow. It is organized as an array of 31 x 3 byte instructions with each instruction being 3 bytes wide. The DSM can be programmed to perform the three main disk drive functions of Read, Write, and Format. The DSM does this by providing means for: Index detection, Sector detection, Sync Byte detection and synchronization, external Sync detection and synchronization, Read Gate and Write Gate control, skips of embedded servo fields, and other specific functions as described in this section.

The Disk Sequencer Map must be loaded with the appropriate user instructions after the power on process and before it is started (SEQRUN, reg. 73h, R/W, bit 7 = 1). A DSM instruction is executed once every byte time. The number of Read Reference clocks (RRCLKs) per byte time depends on which NRZ mode (single, double, or byte wide) is enabled. The relationship between the byte time and RRCLK is listed in Table 4-1. After each instruction is executed the DSM will either execute the same instruction again, or the next instruction, or branch to a different instruction, or stop.

The Disk Sequencer should not be restarted until a poll of the Sequencer Run Status bit (SEQRUN, reg. 73h, R/W, bit 7) shows a zero.

Table 4-1 Relationship Between Byte Time and Read Reference Clock (RRCLK)

NRZ Mode	Byte Time in # of RRCLK
Single	8
Double	4
Byte-wide	1

Figure 4-1 is an example of a single instruction in the Disk Sequencer Map.

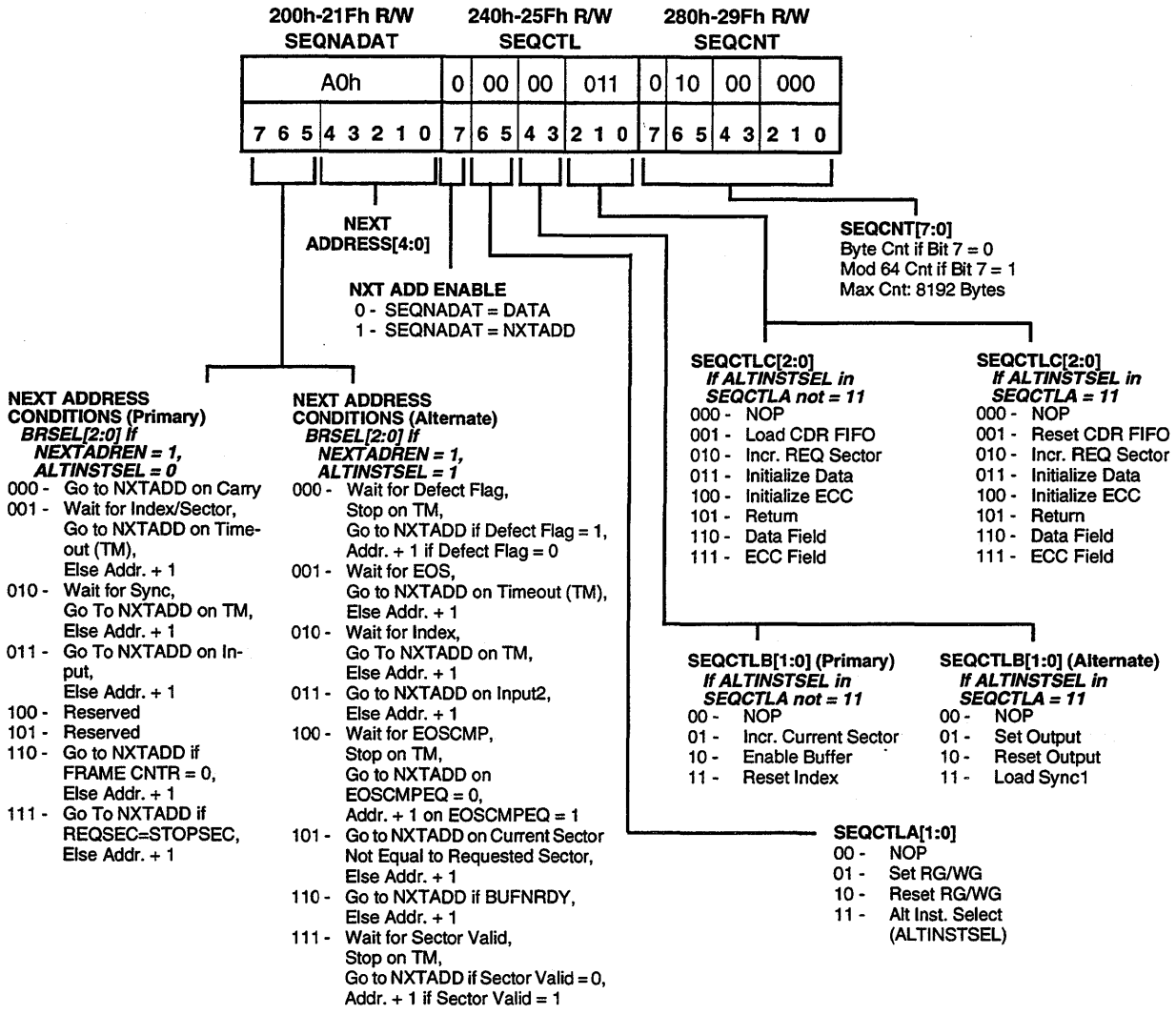

Figure 4-1 Sequencer Map Format

The sections that follow define the usage of each bit within the 3-byte instructions.

4.2 Sequencer Data/Next Address Byte (200h-21Fh, R/W, SEQNADAT)

Each instruction in the Disk Sequencer Map (DSM) may use this first byte as either a Data Field (SEQDATA) or a Next Address Field (SEQNADR) of a branch condition which must also be specified in the same byte. If used as Data Field, bits [7:0] define one byte of Sequencer data. If used as Next Address Field, bits [7:5] define a branch condition and bits [4:0] specify the five least significant bits of the Next Address to be executed if the branch condition is satisfied. The field type for the current instruction is determined by the state of the NXTADDEN bit of the Sequencer Control Field (second instruction byte, SEQCTL, bit 7). If the NXTADDEN bit is cleared to 0, the 'Data Field' type is selected. If it is set to 1, the 'Next Address field' type will be selected.

4.2.1 Data Field

In total, there are two sources for data which can be written to the disk. These are from:

1. the buffer memory, or
2. the Data field (SEQDATA) which is also referred to as Sequencer Data.

Table 4-2 shows when each of these sources is enabled. Data specified in the 'Data Field' (SEQDATA) can either be written to the disk.

Table 4-2 Disk Sequencer Data Sources

En. Buffer Xfer SEQCTLB (10b)	Suppress Xfer reg. 62h, Bit 6	Data Source
0	X	Sequencer Data Field (SEQDATA)
1	0	Buffer Memory
1	1	Sequencer Data Field (SEQDATA)

The Data field value (SEQDATA) is stored in a holding register during the time the field is being used as the Next Address field. The loading of the holding register occurs on every instruction in which the NXTADDREN decode in the SEQCTL byte is equal to "0". This specifically allows for a Sync-Byte to be loaded into a holding register in preparation for a 'Wait for Sync' branch condition.

When this field is used as a Data field, the default Sequencer operation is to execute the next instruction on Sequencer Count Field Carry (third instruction byte, SEQCNT). The Sequencer will execute the current instruction until the Count Field (SEQCNT) of the current instruction has decremented pass zero (Count Field Carry). On Count Field Carry, the Sequencer will execute the next instruction (PC=PC+1) in the Map.

4.2.2 Next Address Field

The DSM logic uses a program counter (PC) which points to the instruction currently being executed. When the Sequencer is started, the value contained in the Sequencer Address register (reg. 73h, W, bits 4:0) is loaded into the PC. Every byte time thereafter, the PC is loaded with the address of the next instruction to be executed. There are a total of six sequencing operations that determine which Sequencer instruction is to be executed next. These sequencing operations are listed in Table 4-3.

Table 4-3 Sequencer Program Flow Instructions

DSM Operation	Effect on PC	Action
Hold	PC = PC	Execute the same instruction again.
Continue	PC = PC+1	Execute the following instruction.
Jump	PC = Next Addr. Field	Branch to the address contained in the Next Address Field of the current instruction (SEQCTL[7], SEQNADAT[4:0]).
CDR Interrupt	PC = CDR Vector	Vector to the address pointed to by the CDR Vector register (reg. 72h, R/W).
Return	PC = Stored Address	Return to the address stored when a CDR interrupt was taken.
Stop	PC = 1Fh	Sequencer stop condition or when Sequencer jumps to location 1Fh.

The Next Address field is divided into two sub-fields. The three most significant bits [7:5] select one of eight possible primary branch conditions, and when used with SEQCTLA='11' (Alternate Instruction Select) define eight alternate branch conditions. The five least significant bits [4:0] form the address of the instruction to which the sequencer will jump to next. A jump will occur if the specified conditions or condition within the selected branch parameters are satisfied. The following describes the eight primary branch conditions that are implemented for the Sequencer.

NOTE: When using a "Wait For" instruction in the primary or alternate condition fields, always use a non-zero value for the count field value in any instruction.

7:5 (R/W) NEXT ADDRESS *Primary* Condition Field

Bit 7 6 5	Description
0 0 0	(NA_CARRY) - NEXT ADDRESS ON CARRY: The Sequencer will execute the current instruction until Count Field Carry occurs. On Count Field Carry, the Sequencer will execute a Jump operation as specified by Next Address (PC=SEQNADR).
0 0 1	(W_SECTOR) - WAIT FOR SECTOR: The Sequencer will execute the current instruction until Count Field Carry occurs or the SECTOR input is asserted. If the Enable Index To Sector Branch bit (ENINDEX, reg. 61h, R/W, bit 4) is set, the INDEX pin is OR'd with SECTOR. If the Sequencer leaves its current state because of a Count Field Carry, then a Branch instruction to Next Address will be executed (PC=SEQNADR) and the Sequencer Wait Time-Out status bit (reg. 66h, R, bit 6) is set. Otherwise, execution will continue with the next instruction (PC=PC+1).
0 1 0	(W_SYNC) - WAIT FOR SYNC: The Sequencer will execute the current instruction until Count Field Carry occurs or the Sync-Byte is detected. If the Sequencer leaves its current state because of a Count Field Carry (Sync Time Out), then a branch instruction to Next Address will be executed (PC=SEQNADR) and the Sequencer Wait Time-Out bit (SEQWAITTO, reg. 66h, R, bit 6) is set. Otherwise, when the Sync-Byte is detected, the Sequencer will execute the next instruction (PC=PC+1). If External Sync is enabled, then the SYNCEN pin will be asserted during this instruction.
0 1 1	(NA_INPUT) - NEXT ADDRESS IF INPUT: The Sequencer will execute the current instruction until Count Field Carry occurs. When the Count Field Carry occurs the status of the INPUT pin is tested. If INPUT=1 the Next Address branch is executed (PC=SEQNADR). If INPUT=0 the Continue is executed (PC=PC+1).
1 0 0	Reserved
1 0 1	Reserved

7:5 (R/W) NEXT ADDRESS *Primary* Condition Field (continued)

Bit 7 6 5	Description
1 1 0	(NA_FRAME) - NEXT ADDRESS IF FRAME COUNTER IS ZERO: The Sequencer will execute the current instruction until Count Field Carry occurs. When Count Field Carry occurs, the Sequencer examines the value in the Frame Counter (reg. 74h, R). If the Frame Counter contains 0, the Sequencer jumps to the Next Address (PC=SEQNADR) and the Frame Counter (reg. 74h, R) is reloaded with the value from the Frame Count register (reg. 74h, W). If the Frame Counter is not 0, then it is decremented and execution continues with next instruction (PC=PC+1).
1 1 1	(NA_ENDXFR) - NEXT ADDRESS IF END OF TRANSFER: The Sequencer will execute the current instruction until Count Field Carry occurs. Upon Count Field Carry, if the contents of the Request Sector Number register (reg. 6Ch-6Dh, R/W) equal the contents of the Stop Sector Number register (reg. 6Eh-6Fh, R/W), the Sequencer will execute a branch instruction to Next Address (PC=SEQNADR). Otherwise, the next instruction in the Map will be executed (PC=PC+1).

The following describes the eight possible alternate branch conditions.

7:5 (R/W) NEXT ADDRESS *Alternate* Condition Field

Bit 7 6 5	Description
0 0 0	(W_DEFECT) - WAIT FOR DEFECT FLAG: The Sequencer will hold on the current instruction until Count Field Carry occurs or the first CDR Split count is loaded into the CDR Counter. On Count Field Carry, the SRVOVRN Status is set and the disk sequencer will halt. If the CDR Counter is loaded before the carry occurs and the Defect Flag (CDR bit 14) is set then a branch to the Next Address Field is executed (PC=SEQNADR). If the Defect Flag is not set a Continue is executed (PC=PC+1). **NOTE:** In DSA Mode the Defect Flag signal requires a minimum of 12 byte times following the "Load CDR FIFO" instruction to become effective, i.e., loaded and activated. EDSA Mode requires 1 to n byte times where n is the remaining number of splits for the previous sector.
0 0 1	(W_EOS) - WAIT FOR EOS: The Sequencer will hold on the current instruction until Count Field Carry occurs or the EOS input (pin 38) signal is asserted. If the Sequencer leaves its current state because of a Count Field Carry then a branch to the Next Address Field is executed (PC=SEQNADR), and the Sequencer Wait Time-Out status bit (reg. 66h, R, bit 6) is set. If the EOS branch is taken a Continue is executed (PC=PC+1).
0 1 0	(W_INDEX) - WAIT FOR INDEX: The Sequencer will hold on the current instruction until the Count Field Carry occurs or the INDEX input is asserted. If the Sequencer leaves its current state because of a Count Field Carry then a branch to the Next Address Field is executed (PC=SEQNADR), and the Sequencer Wait Time-Out status bit (reg. 66h, R, bit 6) is set. If the INDEX branch is taken a Continue is executed (PC=PC+1).

7:5 (R/W) NEXT ADDRESS *Alternate* Condition Field (continued)

Bit 7 6 5	Description
0 1 1	(NA_INPUT2) - NEXT ADDRESS IF INPUT2 ASSERTED: The Sequencer will hold on the current instruction until the Count Field Carry occurs. When the Count Field Carry occurs the INPUT2 (ALE/IN2, pin 104) status is tested. If INPUT2 is false, a Continue is executed (PC=PC+1). If INPUT2 is true, the Next Address Branch is executed (PC= SEQNADR). INPUT2 is driven by the ALE pin and can only be used as a Sequencer input branch while using a non-multiplexed processor.
1 0 0	(W_EOSCMP) - WAIT FOR EOSCMP: The Sequencer will hold on the current instruction until the Count Field Carry occurs or the EOS input is asserted. If the Sequencer leaves its current state because of a Count Field Carry, the Disk Sequencer stops and the EOSTIMEOUT status bit (reg. 65h, W, bit 1) is set. When EOS is detected, the compare of EOSCTR to EOSCMP is tested (EOSCMPEQ status). If EOSCMPEQ=1, a Continue is executed (PC=PC+1). If EOSCMPEQ=0, the Next Address Branch is executed (PC=SEQNADR).
1 0 1	(NA_CURRSENCEQ) - NEXT ADDRESS CURRSEC NOT EQUAL REQSEC: The Sequencer will hold on the current instruction until the Count Field Carry occurs. When the Count Field Carry occurs, the CURRSECEQ status is tested (reg. 5Bh, R/W, bit 7). If CURRSECEQ is true, a Continue is executed (PC=PC+1). If CURRSECEQ is false, the Next Address Branch is executed (PC=SEQNADR).
1 1 0	(NA_BUFNRDY) - NEXT ADDRESS IF BUFFER NOT READY: The Sequencer will hold on the current instruction until the Count Field Carry occurs. When the Count Field Carry occurs, the Buffer Not Ready status is tested. If BUFNRDY is false, a Continue is executed (PC=PC+1). If BUFNRDY is true, the Next Address Branch is executed (PC=SEQNADR).
1 1 1	(W_SECVALID) - WAIT FOR SECTOR VALID: The Sequencer will hold on the current instruction until the Count Field Carry occurs or the SECTOR input is asserted. If the Sequencer leaves its current state because of a Count Field Carry, the Sequencer stops, and the SECTIMEOUT status bit (reg. 65h, R/W, bit 1) is set. If the Sequencer leaves its current state because of SECTOR asserted, the BUFNRDY if SECRDY=1 (reg. 62h, R/W, bit 1), the CURRSECEQ if SECEQ=1 (reg. 62h, R/W, bit 2), and/or the CDRVALID/DEFSEC (bit 13 of CDR Split Count) statuses are checked. If CDRVALID=0, the Sequencer stops and the SRVOVRN status bit (reg. 5Eh, W, bit 7) is set. If CDRVALID=1 and DEFSEC=1, or BUFNRDY=1 or CURRSECEQ=0, a branch to the Next Address Field is executed (PC=SEQNADR). If CDRVALID=1 and DEFSEC=0 and CURRSECEQ=1 and BUFNRDY=0, a Continue is executed (PC=PC+1). If the ENINDEX bit (reg. 61h, R/W, bit 4) is set, the INDEX pin is OR'd with SECTOR.

4:0 (R/W) NEXT ADDRESS[4:0] Field: These bits form bits 4:0 of the Sequencer Program Counter (PC) when a branch is taken to the Next Address (PC=SEQNADR).

4.3 Sequencer Control Byte (240h-25Fh, R/W, SEQCTL)

This is the second byte of a Sequencer instruction. This byte contains four fields. These are: Next Address Enable (bit 7), SEQCTLA (bits 6:5), SEQCTLB (bits 4:3), and SEQCTLC (bits 2:0). Each of these fields are described in detail below.

7 (R/W) **(NEXTADREN) - NEXT ADDRESS ENABLE:** This bit is used to select whether the SEQNADAT field contains data or Next Address values. If NEXTADREN=0 the Data Field is selected; if NEXTADREN=1 the Next Address Field is selected.

6:5 (R/W) SEQCTLA Field

Bit 6 5	Description
0 0	(NOP) - NO OPERATION: No action is taken. The state of the RG (Read Gate, pin 44) and WG (Write Gate, pin 45) outputs are not affected by this decode. In other words, this decode does not reset any decode in this field (SEQCTLA) that had been previously set.
0 1	(SETRGWG) - SET READ GATE / WRITE GATE: If DWRITE=0 (reg. 60h, R/W, bit 7) this decode asserts the Read Gate output at the beginning of the instruction cycle. If DWRITE=1, this decode asserts the Write Gate output at the beginning of the instruction cycle. Note that Read Gate and Write Gate are mutually exclusive signals.
1 0	(RSTRGWG) - RESET READ GATE AND WRITE GATE: This decode deasserts both the Write Gate and Read Gate outputs at the beginning of the instruction cycle.
1 1	(ALTINSTSEL) - ALTERNATE INSTRUCTION SELECT: This decode selects between the Alternate Branches and Instructions and the Primary Branches and Instructions. Alternate Branch instructions are selected when this decode is true.

4:3 (R/W) SEQCTLB Field

Bit 4 3	Description
0 0	(NOP) - NO OPERATION: No action is taken by this decode. This decode resets any previously set decode of this field to 'NOP.'
0 1	(INCURSEC/SETOUTPUT) - *If ALTINSTSEL=0* INCREMENT CURRENT SECTOR COUNTER: This decode will increment the CURRSEC Counter (regs. 5Ah/5Bh, R/W). *If ALTINSTSEL=1* SET OUTPUT PIN: This decode will set the OUTPUT Signal, and it stays set until it is reset with the RSTOUTPUT decode (SEQCTLB='10').

4:3 (R/W) SEQCTLB Field (continued)

Bit 4 3	Description
1 0	(ENBUFFER/RSTOUTPUT) -

If ALTINSTSEL=0

DATA TRANSFER FROM BUFFER MEMORY: This decode enables the data buffer as the source or destination of the disk data transfers. This bit is overridden by the Suppress Transfer Control bit (SUPXFR, reg. 62h, R/W, bit 6) which when set disables data transfer to/from the Buffer Memory.

If ALTINSTSEL=1

RESET OUTPUT PIN: This decode will reset the OUTPUT Signal. OUTPUT is set with the SETOUTPUT decode.

1 1 (RSTIDX/LOADSYNC1) -

If ALTINSTSEL=0

RESET TWO INDEX TIMER: This decode resets the Two Index Timer.

If ALTINSTSEL=1

LOAD SYNC BYTE 1: This decode will load the internal SYNCPAT1 Register from the SEQDATA field. NEXTADREN must be 0. This is used to load the most significant byte of the Sync Byte during Internal Fault Tolerant Sync Mode.

2:0 (R/W) SEQCTLC Field

Bit 2 1 0	Description
0 0 0	(NOP) - NO OPERATION: No action is taken by this decode. This decode resets any previously set decode of this field to 'NOP.'
0 0 1	(LOADCDR/RSTCDR) -

If ALTINSTSEL=0

LOAD CDR FIFO: This decode is used to load the Servo Split information into the CDR FIFO and/or flush old CDR counts from the CDR FIFO and Counter. The CDR FIFO can be loaded with data from either the Buffer Memory or Local MPU.

When ENBUFCDR=0 (reg. 63h, R/W, bit 1) and ENHANCEDSA=0 (reg. 5Dh, R, bit 0) this decode must not be used.

When ENBUFCDR=1 and ENHANCEDSA=0, the CDR FIFO will be loaded with data from the Buffer. The number of bytes fetched is specified in the Sequencer Count Field when the Load CDR FIFO decode is set. The Sequencer count is forced to 00h for this instruction. The fetch of information is done in parallel with the succeeding Sequencer instructions. When the Load CDR FIFO decode is executed, the CDR FIFO is reset before the load occurs throwing away any split counts in the CDR FIFO and Counter. When specifying the CDR Byte Count, the actual number of bytes required minus 1 must be used.

2:0 (R/W) SEQCTLC Field (continued)

Bit 2 1 0	Description
	While the ENHANCEDSA bit (reg. 5Dh, R/W, bit 0) and the ENBUFCDR bit (reg. 63h, R/W, bit 1) are set, issuing this decode will not cause a fetch of bytes from the Buffer Memory. It will flush all of the CDR Split counts for the current Sector by flushing all counts up to and including the count with the LASTSPLIT bit (CDR Split Count Word bit 13) set. The count field of this instruction is valid, but this decode is only counted as one occurrence regardless of the count. This decode cannot be issued in two consecutive instructions with the expectation that two flushes will occur. Flushes can be consecutive only after the CDR is valid. After LOADCDR is issued the CDR is not valid until the flush is complete. *If ALTINSTSEL=1* RESET CDR FIFO: When this decode is issued the CDR FIFO, CDR Counter, and associated logic are reset.
0 1 0	(INCREQSEC) - INCREMENT REQUEST SECTOR NUMBER: This decode is used to increment the Request Sector Number Register by one. If the REQSEC Register equals the Wrap Sector Number Register (reg. 68h/69h, R/W), the REQSEC Register will wrap to the value specified in the Wrap To Sector Number Register (reg. 6Ah/6Bh, R/W). If this decode is used in the same instruction as a "Next Address on End of Transfer", the increment will take place after the compare.
0 1 1	(INITDATA) - INITIALIZE DATA FIELD: This decode initializes and starts ECC logic after the data Sync byte is detected or written. The Sync-Byte is not included in the ECC calculation. This decode must be used in a 'Wait for Sync' instruction. This decode should also be set when writing the first data Sync byte and not for the subsequent data Sync bytes used in Servo splits. When writing a 2-byte fault tolerant byte sync, this decode must be executed coincident with the second of the two sync bytes.
1 0 0	(INITECC) - INITIALIZE DATA ECC: If the ENSEED bit (reg. 5Dh, R/W, bit 4) is set, this decode initializes the Data ECC and triggers a four byte Seed write to the EDAC. If ENSEED=0, this decode is not used. This operation should be executed 4 byte times before the sync byte is written.
1 0 1	(RETURN) - RETURN FROM CDR VECTOR: This decode returns the program address counter to the address saved during a CDR Interrupt Vector. The ECC circuitry is automatically enabled when the RETURN is executed.
1 1 0	(DATAFIELD) - DATA FIELD: This decode is set while the Data Field is being transferred.
1 1 1	(ECCFIELD) - ECC FIELD: This decode is set while the Data ECC field is being transferred to/from the disk if the ECC circuitry is to be used.

4.4 Sequencer Count Byte (280h-29Fh, R/W, SEQCNT)

This is the third byte of a DSM instruction. This byte can function as a fetch count, or as a Sequencer Count byte. If used as a fetch count the Sequencer Count is forced to zero in order to execute the instruction in one byte time.

If the 'Load CDR FIFO' decode (SEQCTLC[2:0] = '011' with ALTINSTEL not ='11') is enabled, and the Enable CDR from Buffer bit (reg. 63h, R/W, bit 1) is set to 1, and the Enable EDSA Mode bit is cleared (reg. 5Dh, R/W, bit 0), this byte represents the number of CDR bytes (minus 1) to be fetched from the buffer memory and loaded into the CDR FIFO. In this mode, the Sequencer Count is forced to zero.

If the 'Load CDR FIFO' decode (SEQCTLC='001') is not set, this byte represents, in byte times, the number of times plus one that the current instruction is to be executed. Of course, the execution of the current instruction can be interrupted for a number of reasons. Every byte time the Sequencer Count is decremented. The DSM will execute a new instruction upon a Count Field Carry which occurs if the Sequencer Count counts down from zero. The following describes how the bits in this byte are used.

7 (R/W) **(MODCNTEN) - MODULO 64 COUNT ENABLE**: When this bit is set to 1, the Sequencer Count is Modulo 64. For every 64 bytes, the Sequencer Count[6:0] is decremented by one. When this bit is cleared to 0, the Sequence Count[6:0] is a normal 7-bit down counter and is decremented on every byte time.

6:0 (R/W) **(SEQCNT[6:0]) - SEQUENCER COUNT[6:0]**: This field determines the value of the Sequencer counter. These bits are loaded into the Sequencer Counter at the start of the Sequencer instruction. The counter will decrement by one for every 1 or 64 bytes that are transferred, depending on the state of the MODULO 64 COUNT ENABLE bit. When the counter generates a Carry, a new instruction is accessed from the Sequencer Map.

NOTE: In either normal mode or Mod 64 mode, the counter should be loaded with the desired count minus one.

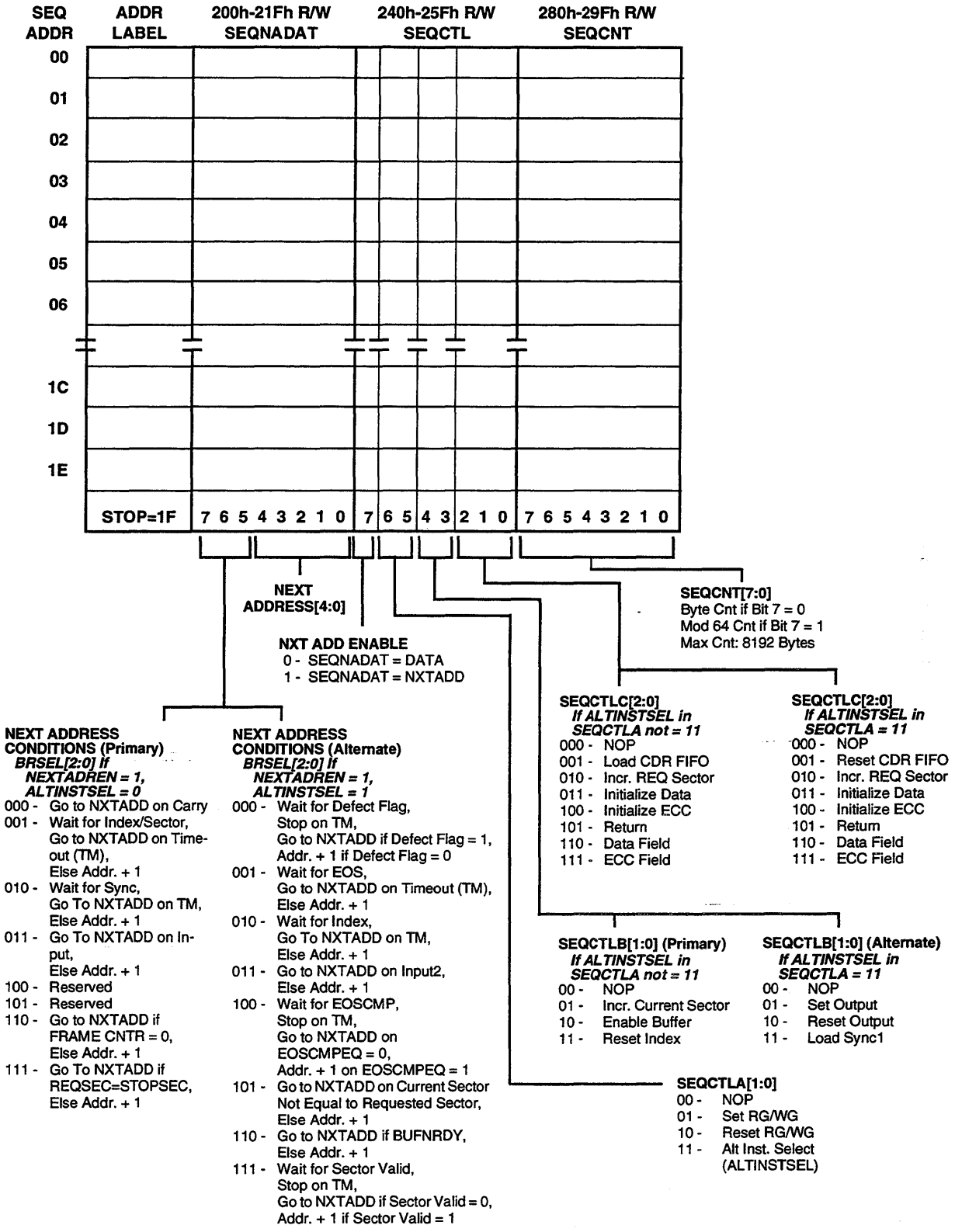

Figure 4-2 Blank Sequencer Map

4.5 Disk Sequencer Examples

4.5.1 EDSA Read/Write/Format With CDR Example

The following sequencer map, EDSA Read/Write With CDR, is designed to be loaded after initialization. Because the Read, Write and Format sequencers are very similar, but not identical, minor changes (detailed below) are required when switching between these operations. No other fields in the sequencer map need to be updated.

The sector format for this sequencer is:

- From Index or Sector, user specified Inter-Sector Gap (ISG)
- VFO Field of 8 bytes
- Data Sync-Byte of 8003h
- 512 bytes of data
- 21 bytes of data ECC
- 2 pad bytes
- User Specified Inter-Sector Gap to Next Sector

CDR gaps have the following format:

- 2 byte write pad
- Servo Field (user defined length)
- VFO Field of 17 bytes
- CDR Sync-Bytes of FFFEh

The following registers should be programmed as follows:

Read/Write Vector Address Register
Reg. 72h = 15h for a Read Operation
Reg. 72h = 19h for a Write or Format Operation

The following four registers determine the exact location of the desired sector. These registers should not be written while the sequencer is running. Also, care should be taken that the EOS counter is updated during a valid timeframe. Current Sector and Requested Sector are updated by the sequencer. The EOS counter is updated by the AIC-8375, while EOS Compare is not updated at all.

Current Sector Number
Reg. 5Ah, 5Bh = User Defined, physical sector number of first sector in target frame.

Requested Sector Number
Reg. 6Ch, 6Dh - User Defined, physical sector number of first desired sector.

EOS Compare Register
Reg. 5C = User Defined, should be set to 1 less than the EOS count for the target frame.

EOS Counter Register
Reg. 58h = User Defined, should be written once during initialization, the AIC-8375 will then update it automatically.

The Stop Sector Number will be used to determine which sector will be the last read by the Disk sequencer. It can be programmed to the last sector on a track, or to any intermediate sector.

Stop Sector Number Register
Reg. 6Eh, 6Fh = User Defined, actual last sector number on track.
(6Eh is low byte - 6Fh is high byte and must be set last for the Stop Sector value to be latched)

The seed registers need to be initialized only if ECC Seeding is being used.

ECC Seed Registers
Reg. 83h, 80h, 81h = User Defined. Usually set to the upper 22 bits of the CHS, any value uniquely identifying the track will suffice.

To start the operation:
Sequencer Address Register
Reg. 73h = 80h, starts the Sequencer running at location 00h.

4.5.2 Write Sequencer Modification

The allowable number of instructions do not allow for both read and write maps to fit in the same sequencer. However, two instructions can be modified to convert the read map to the write map:

- Instruction 03 should be changed to 07/84/00.
- Instruction 19 should be changed to 00/00/01.

4.5.3 Format Sequencer Modification

The format sequencer is identical to the write sequencer, except for the data field instruction, which needs to be changed to load data from the sequencer, rather than the buffer.

SEQ ADDR	ADDR LABEL	200h-21Fh R/W SEQNADAT	240h-25Fh R/W SEQCTL				280h-29Fh R/W SEQCNT	COMMENTS
00	start	82	0	11	11	000	00	Load high sync byte
01		81	1	11	00	000	FF	Wait for EOS Compare
02		7E	1	11	00	000	FF	Wair for Sector Pulse, cont. if Sector Vali
03		00	0	00	00	100	00	Initiate ECC seed
04		03	0	01	00	000	00	Turn on RG, load low sync byte
05		5F	1	00	00	011	10	Wait for sync, start ECC at sync
06		0A	1	00	10	110	87	Transfer 512 bytes, enable buffer, goto E
07		00	0	01	00	000	07	VFO pat 0, WG on
08		03	0	00	00	011	01	Write Sync. Start ECC
09		00	0	00	10	110	81	Transfer 512 bytes, enable buffer
0A	ecc	82	0	11	11	111	14	ECC Field for 21 bytes, load high sync
0B		00	0	00	11	001	01	Pad, reset 2IDX, flush current CDR
0C		FF	1	10	00	000	00	Stop if EOT, reset RG/WG
0D		02	1	00	01	010	00	Jump to top of loop, inc CurrSec and Req
0E		11	1	11	00	000	00	Jump if Defect Flag is set
0F		B4	1	11	00	000	00	Jump if CurrSec is not equal to ReqSec
10		14	1	00	11	000	00	Reset 2IDX because of bufnrdy
11		B4	1	11	00	000	00	Jump if CurSec not equal to ReqSec
12		00	0	00	11	000	00	Reset 2IDX because of defective sector
13		00	0	00	00	010	00	Inc ReqSec
14		02	1	00	01	001	00	Jump to top of loop. Inc CurSec, flush CI
15	read vector	00	0	10	00	000	00	Reset RG
16		3F	0	11	00	000	10	Wait for EOS
17		7F	0	11	11	000	00	Skip Write Splice
18		FE	0	00	00	000	00	Set low sync byte
19	write vector	5F	1	01	00	101	10	Wait for sync, RG on, return
1A		00	0	10	00	000	00	Reset WG
1B		7F	0	11	11	000	00	Load high sync byte
1C		3F	1	11	00	000	10	Wait for EOS
1D		00	0	01	00	000	07	VFO field, WG on
1E		FE	0	00	00	101	00	Write sync & return

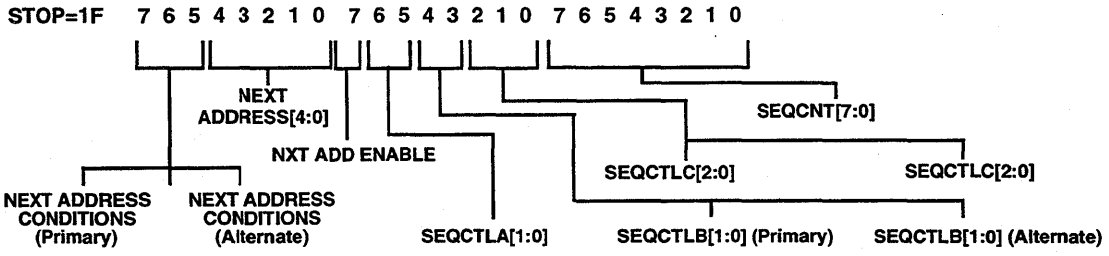

Figure 4-3 Read with CDR Sequencer Map

ADDENDUM
AIC-8375B Operational Characteristics
(8/28/95) Revision B.0

This addendum details characteristics of the AIC-8375's (Rev B) operation that designers should be especially aware.

A1.1 Host Block - No operational characteristics noted.

A1.2 Buffer Block - No operational characteristics noted.

A1.3 Disk Block

A1.3.1 Reset 2 Index Timeout Operational Characteristic

Condition: There is a possibility of getting a two-index timeout during a normal disk operation. This occurs on systems where the last EOS pulse is raised before the Index pulse.

If the sequencer is started after the last EOS pulse but before the Index pulse and the target Frame is the first on the track, the Index counter will increment but EOSCMPEQ (reg. 5Bh, R, bit 6) will not be true. This will cause the sequencer to wait until the it encounters the last EOS pulse again (one disk revolution later). At this point EOSCMPEQ will be true, allowing the sequencer to start searching for the requested sector. However, the Index counter will increment once again at the Index pulse, causing a Two Index Timeout.

Workaround: This situation is avoided by executing a Reset Two Index Timer decode (primary, SEQCTLB = '11'b) after the Wait for EOS Compare Equal (alternate, BRSEL = '100'b) has found the correct frame, but before the Index pulse occurs (usually on the instruction).

A1.4 Microprocessor Block - No operational characteristics noted.

A1.5 ECC Block - No operational characteristics noted.

This page intentionally left blank.

Adaptec, Inc.
691 South Milpitas Boulevard
Milpitas, CA 95035
Tel: (408) 945-8600
Fax: (408) 262-2533

P/N: 700191-011A Rev 3
Printed in U.S.A. DM 9/12/95