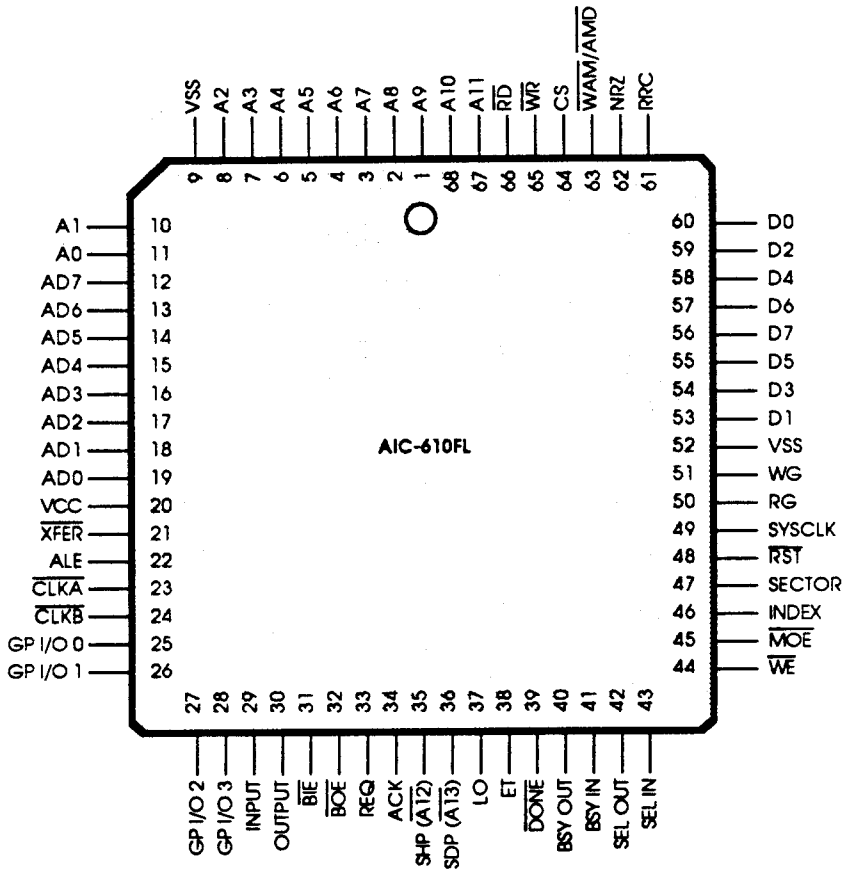
adaptec, inc.

AIC-610F

Integrated Programmable Storage Controller

PRELIMINARY



Peripheral Data Controller
- Controls Embedded, ST412/506, ESDI, And SMD Interface Hard Disk Drives
- Works With All Disk Encoding Schemes
- User-Modifiable RAM-Based Control Store
- User-Programmable Internal 32-Bit, 48-Bit ECC Polynomial, And 16-Bit CRC, Or Variable Length External Polynomial
- Transfer Rate Up To 15 Mbits/Sec
- Soft- Or Hard-Sector Drives
- Multiple Sector Transfer
- Sector-Level Defect Handling
- Noninterleaved Operation
- User-Programmable Sector Length Up To A Full Track

Buffer Data Management
- Buffer Sizes From 256 To 64K Bytes
- Buffer Size Of 16K Bytes Accessed Directly
- DMA Handshake Logic
- Supervises Data Transfers To Buffer With Overrun Control
- Dual-Port Circular FIFO Buffer Control
- Port Priority Resolver Resolves Host/Peripheral Requests

SCSI Interface
- Arbitration Logic Allows Stacking
- Programmable Request For Arbitration While SCSI Bus In Busy State
- Simplified SCSI Interface Implementation With AIC-500 SCSI Interface Adapter

Processor Interface
- Multiplexed Address/ Data Bus Interface
- Interrupt Driven Operation

Technologies
- CMOS
- Software Compatible With AIC-011/AIC-301
- 68 Pin PLCC

OVERVIEW

The Adaptec AIC-610F Integrated Programmable Storage Controller (IPSC) provides the major portion of the functions that are necessary to implement a high-performance disk or tape controller or an intelligent disk drive. The functions of the AIC-610F can be categorized as a serial data transfer controller responsible for data transfer to and from the peripheral, and a buffer management controller responsible for data transfer to and from the buffer memory. In addition, the AIC-610F has been designed for easy interface to support processors and the SCSI host bus.

The AIC-610F is capable of supporting most drive interfaces, including Embedded, ST412/506, ESDI, and SMD. In other words, the AIC-610F can be used with 3-1/2 inch, 5-1/4 inch, and 8 inch hard drives; and either floppy, optical, or tape drives.

The AIC-610F has a fully user-programmable RAM-based sequencer, which allows for a flexible mode of operation, full compatibility with various drives, and also allows user-defined specialized track formats.

The AIC-610F performs ECC/CRC generation, checking and correction. The error correction algorithm used by the AIC-610F is user-programmable with a 32-bit or 48-bit PRESET or RESET ECC polynomial, or an external variable length polynomial.

The IPSC also performs serialization and deserialization of peripheral data and has search and verify capabilities.

On the buffer management side, the AIC-610F allows low cost static RAM to be used as a dual-port circular FIFO. The AIC-610F supervises data transfers to the buffers, thereby reducing the possibility of overruns while allowing high-speed DMA transfers.

The chip also implements a two-wire arbitration circuit for resolving host peripheral requests by giving the priority to the peripheral and placing a hold on the host. Finally, the AIC-610F is interrupt-based, allowing for a more efficient data transfer mechanism.

Based on the functions discussed above, the AIC-610F forms the core of a three-chip set: the AIC-610F IPSC, the AIC-270 (or AIC-250) ENDEC, and the AIC-6225 Data Separator. These chips, combined with the necessary drivers and receivers (i.e., the AIC-500, providing the necessary SCSI interface) and a low-cost microprocessor, provide all that is required to implement a high-performance, full-featured controller.

The AIC-610F controls read/write functions for a mass storage device. The AIC-610F provides serialization/deserialization, formatting, function sequencing, and error processing. In addition, the chip provides a dual-ported buffer management control function in systems requiring the use of a buffer between the host bus and the controller due to available bus bandwidth.

The AIC-250 provides the write precompensation, write address mark/address mark detect, and NRZ to/from MFM conversion functions required in ST412/506-type drive interface applications. The AIC-270 differs in the encode/decode scheme, providing NRZ to/from 2,7 RLL code conversion functions. This method of encoding can increase the effective capacity and data transfer rate of a drive by as much as 100%.

The AIC-6225 Data Separator provides the complete data separator function, allowing the clock signal to be separated from the data. The AIC-6225 requires no adjustments and has selectable outputs of synchronized data for RLL codes or decoded MFM (NRZ).

The AIC-500 provides the necessary drivers and receivers to interface the AIC-610F directly to the SCSI bus.

Finally, the AIC-610F is designed to work with either a local processor or a host processor. The choice is up to the designer and is a function of the host system's available bus bandwidth and board space design considerations. Accordingly, the microcode for the control of the AIC-610F will be based in system RAM or on a local (EP)ROM.

Figure 1 illustrates a typical application of the AIC-610F controller used with an AIC-270 ENDEC, an AIC-6225 data separator, an AIC-500 SCSI interface adapter, a processor, ROM, and a 16K x 8 RAM buffer for a SCSI controller.

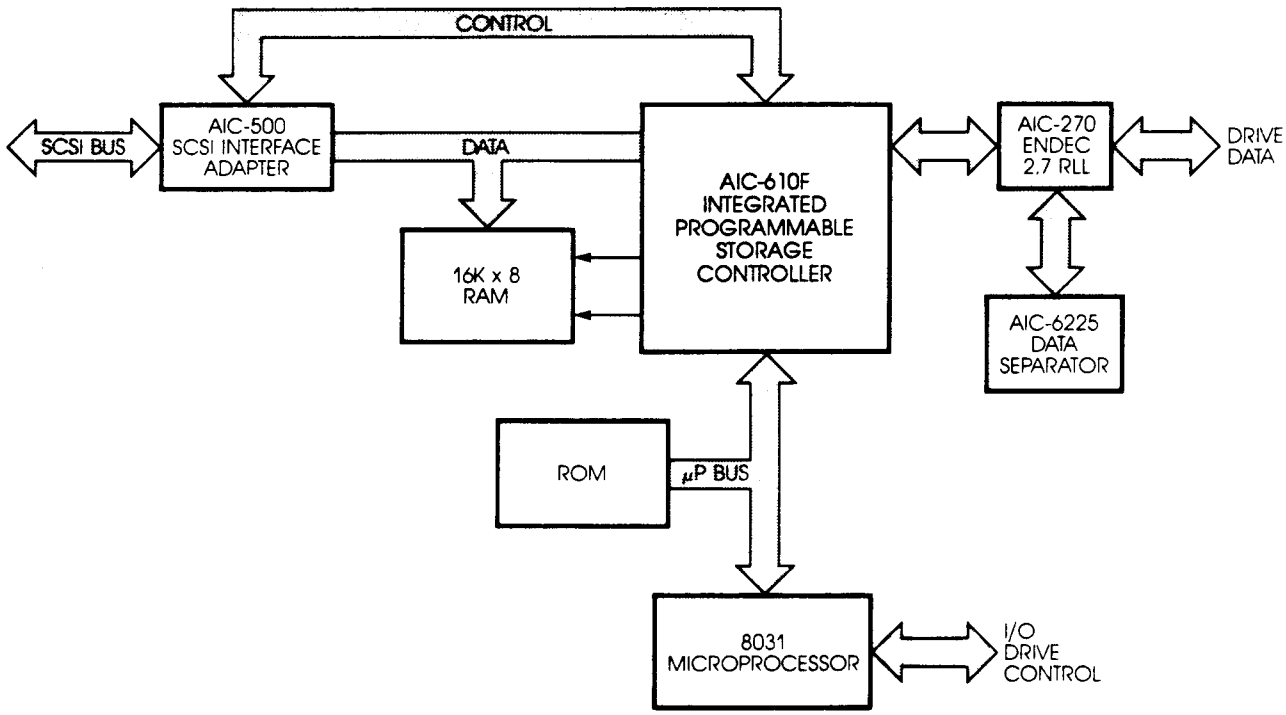


FIGURE 1. AIC-610F BASED SCSI CONTROLLER

TABLE 1. PIN DESCRIPTION

SYMBOL	PIN	TYPE	NAME AND FUNCTION
			MICROPROCESSOR INTERFACE SIGNALS
A/D0-7	12-19	I/O	MULTIPLEXED ADDRESS/DATA: These are three-state address/data lines which interface with a multiplexed microprocessor address/data bus.
ALE	22	IN	ADDRESS LATCH ENABLE: This control signal latches the address on the A/D0-A/D7 lines and identifies the bits as a register address.
$\overline{XFER}$	21	OUT	DATA TRANSFER/STOPPED INTERRUPT: Indicates the start of the data transfer to or from the buffer memory or a stopped condition.
$\overline{DONE}$	39	OUT	DMA DONE INTERRUPT: Occurs when the Read Access Pointer (RAP) = Stop Pointer (SP) during a read cycle, or when Write Address Pointer (WAP) = SP during a DMA write cycle. This indicates completion of the host data transfer and allows the microprocessor to do the necessary function of updating the SP if it is required.
CS	64	IN	CHIP SELECT: Active high input, used to select the chip during processor bus cycle access.
$\overline{WR}$	65	IN	WRITE: Signal from the microprocessor to enable data to be written from the A/D bus to a specified register.
$\overline{RD}$	66	IN	READ: Signal from the microprocessor to enable data from a specified register out onto the A/D bus.
$\overline{RST}$	48	IN	RESET: A low input sets an internal reset latch that stops all operations within the chip and deasserts RG, WG, WAM, and NRZ outputs. Registers 50_H through $7E_H$ are reset. All special outputs are set to the high state.
			BUFFER INTERFACE
A0-A11	1-8 10-11 67-68	OUT	BUFFER ADDRESS LINES: Bits 0-11 for addressing low-order address of buffer in applications with less than or equal to 14 bits of addressing. In applications with more than 14-bit addressing lines, A4-A7 are multiplexed for high-order addresses A12-A15.
SHP/$\overline{A12}$	35	OUT	STROBE HOST POINTER: Buffer address bit 12 in applications with buffer size of 14-bit or less addressing. This is the clocking signal for loading high-order address bits into an external host address register in applications using more than 14-bit addressing.
SDP/$\overline{A13}$	36	OUT	STROBE DEVICE POINTER: Buffer address bit 13 in applications with buffer size of 14-bit or less addressing. This is the clocking signal for loading high-order address bits into external device address register in applications using more than 14-bit addressing.
D0-D7	53-60	I/O	DATA BUS: Byte parallel data lines to and from the buffer. These lines are tri-stated at the rising edge of ALE.
$\overline{WE}$	44	OUT	WRITE ENABLE: Asserting $\overline{WE}$ enables data to be written into the RAM buffer.

TABLE 1. PIN DESCRIPTION (Continued)

SYMBOL	PIN	TYPE	NAME AND FUNCTION
			BUFFER INTERFACE (Continued)
$\overline{MOE}$	45	OUT	MEMORY OUTPUT ENABLE: Enables data to be read from RAM buffer.
$\overline{CLKA}$	23	I/O	CLOCK A: During a read or write operation, the output of this signal is derived from the input RRC. Otherwise, it is derived from the input SYSCLK. The relationship between the input clock and $\overline{CLKA}$ is controlled by the contents of Register $7F_H$.
$\overline{CLKB}$	24	I/O	CLOCK B: A pulse which overlaps the negative edge of $\overline{CLKA}$ and occurs whenever a byte is transferred to/from data bus pins D0-D7.
			SYSTEM BUS INTERFACE
$\overline{BIE}$	31	OUT	BUS IN ENABLE: Used to gate data out of external latches from the bus for writing into the buffer.
$\overline{BOE}$	32	OUT	BUS OUT ENABLE: Used to gate data out of external latches for transfer onto the bus. Asserted when arbitration latch is set.
REQ	33	OUT	REQUEST: The request for a data transfer to or from the buffer (DMA Handshake).
ACK	34	IN	ACKNOWLEDGE: Used to acknowledge data has been received from or sent to the buffer (DMA Handshake).
LO	37	OUT	LATCH OUT: Used to clock data into external latches after reading from buffer.
BSY OUT	40	OUT	BUSY OUT: Either set directly by the microprocessor or in an arbitration request mode the BSY OUT will be activated when BSY IN and SEL IN are inactive. The arbitration mode assures an arbitration phase.
BSY IN	41	IN	BUSY IN: Active when other devices are actively accessing the bus.
SEL OUT	42	OUT	SELECT OUT: This pin is used to request selection by the host bus. SEL OUT is set by the microprocessor setting Bit 6, Register 52 (Channel Control).
SEL IN	43	IN	SELECT IN: Active indicates a bus select status. SEL IN will reset the arbitration latch.
			PERIPHERAL INTERFACE SIGNALS
INDEX	46	IN	INDEX: Input for the index pulse received from the peripheral. Must be a minimum of nine bits wide.
SECTOR	47	IN	SECTOR: Input for the sector pulse received from drives that are hard-sectored. Must be minimum of nine bits wide.
RG	50	OUT	READ GATE: Enables the external phase-lock loop to lock onto the read data stream coming from the storage device.
WG	51	OUT	WRITE GATE: Is used to enable or gate the writing of NRZ data out to the storage device.

TABLE 1. PIN DESCRIPTION (Continued)

SYMBOL	PIN	TYPE	NAME AND FUNCTION
			PERIPHERAL INTERFACE SIGNALS (Continued)
RRC	61	IN	READ REFERENCE CLOCK: A multiplexed input sourced from the VFO oscillator during read gate. Otherwise from the write oscillator. This is the primary clock for the AIC-610 and must be present at all times, including during the reset operation.
NRZ	62	I/O	NRZ: Read data input from the storage device when RG is active. Write data to the storage device when WG is active.
$\overline{\text{WAM/AMD}}$	63	I/O	WRITE ADDRESS MARK/ADDRESS MARK DETECT: A one-bit wide pulse is output when write gate is active and an address mark is to be written. When read gate is active, a low level input to indicate address mark detect.
			MISCELLANEOUS SIGNALS
GP I/O 0 or $\overline{\text{WR}}$ 6E	25	I/O	GENERAL PURPOSE I/O LINE 0: A user-programmable I/O line for use as an input or an output. This pin can also be programmed as a decoded output for a write to address $6E_H$.
GP I/O 1 or $\overline{\text{RD}}$ 6E	26	I/O	GENERAL PURPOSE I/O LINE 1: A user-programmable I/O line for use as an input or an output. This pin can also be programmed as a decoded output for a read from address $6E_H$.
GP I/O 3 or $\overline{\text{WR}}$ 6F	27	I/O	GENERAL PURPOSE I/O LINE 2: A user-programmable I/O line for use as an input or an output. This pin can also be programmed as a decoded output for a write to address $6F_H$.
GP I/O 3 or $\overline{\text{RD}}$ 6F	28	I/O	GENERAL PURPOSE I/O LINE 3: A user-programmable I/O line for use as an input or an output. This pin can also be programmed as a decoded output for a read from address $6F_H$.
INPUT	29	IN	INPUT PIN: The state of this pin is sampled by reading Register $7E_H$, Bit 4. The input pin is also a branch input to the sequencer RAM.
OUTPUT	30	OUT	OUTPUT PIN: Controlled by bit 2 of the control block ($A0_H$ thru $B7_H$) of the sequencer RAM. It can be used to monitor the sequencer RAM or synchronize external circuitry to a particular state of the sequencer RAM.
$\overline{\text{ET}}$	38	OUT	ENABLE TARGET: A microprocessor settable signal to enable the target.
SYSCLK	49	IN	SYSCLK: A clock input used to derive the $\overline{\text{CLKA}}$ output when not reading or writing data. SYSCLK must be present whenever the AIC-610 is to be used in any way.
V_{SS}	9, 52	GND	GROUND.
V_{CC}	20	PWR	+5 Volts.

FUNCTIONAL DESCRIPTION

Internal to the AIC-610F Integrated Programmable Storage Controller chip are four main functional blocks:

- Microprocessor Interface
- Peripheral Data Control
  - Sector Format Sequencer
  - Data Flow
- Buffer Management
  - Buffer Control
  - Priority Resolver
  - DMA Control
- System Bus Interface
  - Arbitration

Microprocessor Interface

The microprocessor interface is an eight-bit multiplexed bus such as is found on the Intel 8085 family of processors. Other microprocessors (the Z80 or the 6800) can be utilized by multiplexing their address and data lines, and generating the necessary control lines. With some external circuitry, a 16-bit multiplexed or separate address and data bus can also be supported.

The AIC-610F decodes addresses from 50_H to FF_H, in order to prevent erroneous operation, no other addresses are used. The device architecture is structured to allow the firmware of the processor to determine what functions are to be incorporated in the control unit design.

The user has the ability to select the ECC or CRC for Error Detection and Correction (EDAC). Furthermore the user may select the ECC polynomial that is optimum for the media and the encoding scheme being used. The AIC-610F provides an internal 32-bit or 48-bit programmable ECC capability. The user can also use an external variable length ECC if required. The standard 16-bit CRC is used for error detection and is typically used on ID fields or with floppy and tape devices. Note: The CRC and ECC may be initialized with a reset or preset (Register 71_H, Bit 6).

Peripheral Data Control

SECTOR FORMAT SEQUENCER: The sector format sequencer performs the basic sequencing function for a mass storage device which includes:

- Read ID
- Read ID and Read Data (or Read Operation)
- Read ID and Write Data (or Write Operation)
- Write ID and Write Data (or Format Operation)

These functions can be modified to perform the search data and verify data functions.

The sequencer consists of 96 bytes of RAM, organized as a 24 x 4 byte matrix. These locations have to be set-up at initialization time for the proper operation of the chip. Under firmware control, the AIC-610F can be made to sequence through different types of operations. The user can control the timing relationships between various output signals and can monitor the different input lines to branch to various sequencer locations.

The controller chip also has other registers that can be used to control the definition of the track format. Using these registers, features such as gap length, sync characters, and ECC polynomials, can be controlled. The track layout (sector size and sector data fill character) can also be flexibly defined.

DATA FLOW: The data flow portion of the controller chip is composed of the ECC logic and a serializer/deserializer. Data to be written to the peripheral enters the device in 8-bit parallel format. The data is serialized and run through an ECC generator. The AIC-610F outputs NRZ serial bits. The bits include serialized constants required for address marks, gaps, and ID fields, as well as serialized data and ECC generator output.

Buffer Management

When used in a CPU environment, the AIC-610F will work well with the DMA control devices available to provide host processor memory addressing. The next three paragraphs describe its ability to support this function.

BUFFER CONTROL: The buffer control function provides read and write address registers, as well as Memory Output Enable (MOE) and Write Enable (WE) signals. These signals are used to read or write data from the RAM buffer.

PRIORITY RESOLVER: The priority resolver allows the typically synchronous peripheral to have priority over the host requests. This is crucial in peripheral controller applications where, in a 10 Mb/sec system, a data byte must be transferred exactly once every 800 nsecs.

DMA CONTROL: The DMA control generates a request to the host (REQ), gates the appropriate data into or out of the buffer, and waits for a correctly timed acknowledge (ACK).

System Bus Interface

The AIC-610F has the necessary signals to interface to a host or system bus. It also provides logic to arbitrate for the system bus.

ARBITRATION: The arbitration logic of the AIC-610F provides for a two-wire arbitration scheme where either Select In or Busy In indicate a system bus busy state. The device allows for a request to be stacked for arbitration when the bus is in the bus busy state. The arbitration logic will request the bus for arbitration when both Select In and Busy In are inactive for a minimum of one clock time.

Figure 2 is a block diagram of the AIC-610F which identifies the different functional blocks.

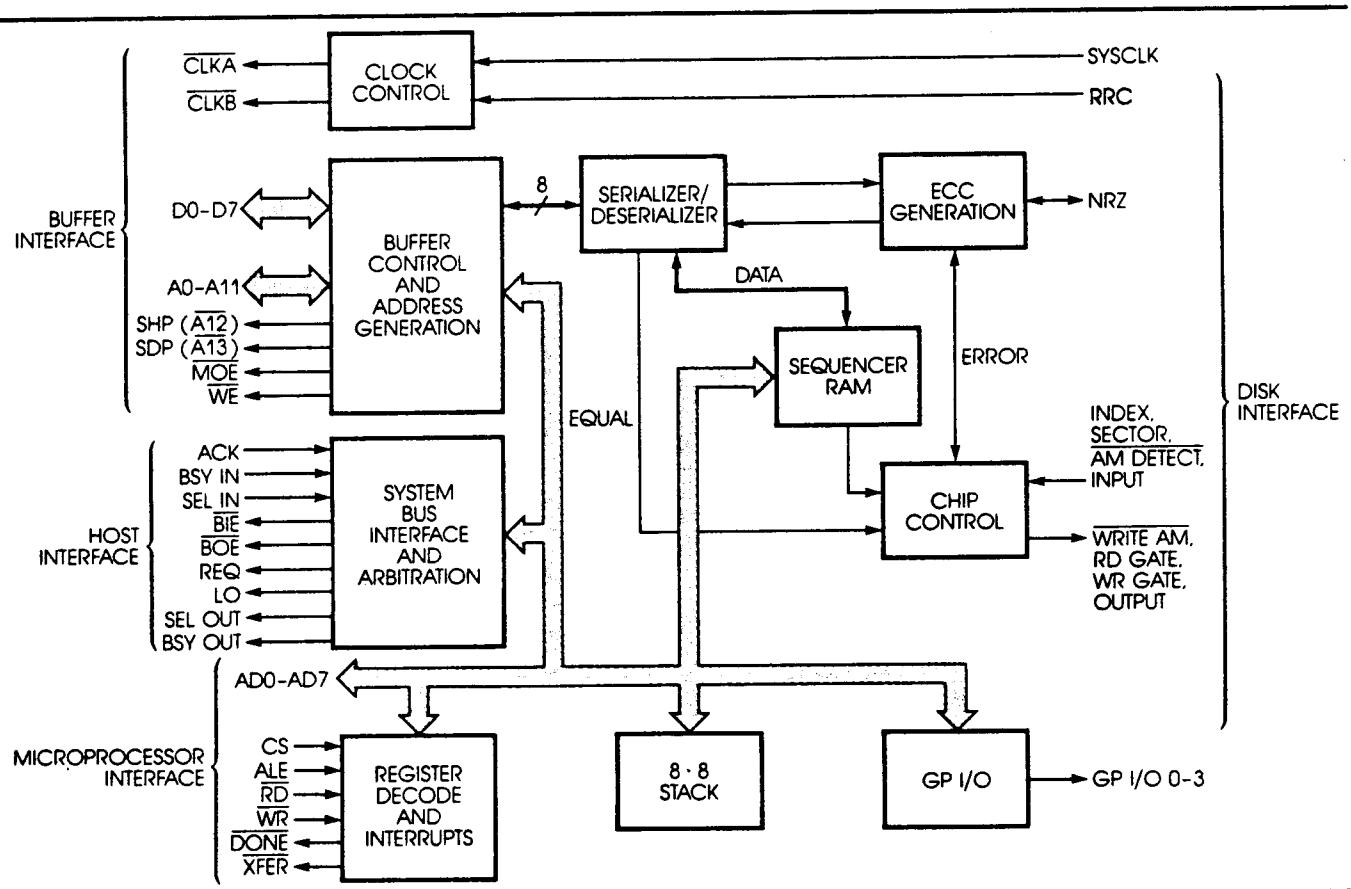


FIGURE 2. FUNCTIONAL BLOCK DIAGRAM

FUNCTIONAL OPERATION

The functional operation of the AIC-610F can be broken down into two parts. The first portion is the control of data to and from the peripheral. This involves the support processor, sequencer RAM, and various registers to exercise the necessary peripheral data control operation. The other portion is the buffer management of data to and from the buffer.

Peripheral Data Path Control

The control of data to and from the peripheral is controlled by the contents of the sequencer RAM. The sequencer RAM consists of 96 bytes, organized as 24-by-four words. The four bytes can be broken down into Data, Count, Control, and Next Address Fields. The operation of the AIC-610F revolves around the branch register (Register 78_H) and the status/start register (Register 79_H). Register 79_H is first loaded with the address where the sequencer is to begin execution. Thereafter, the AIC-610F sequences through the RAM and the next address is executed. If a successful branch condition occurs, the next address is based on the contents of Register 78_H. Otherwise, it is based on the contents of the next address field at that address. By setting different branch conditions, based on internal or external events, the chip can be made to sequence through different operations.

The AIC-610F also has a stack that is eight-bytes deep. By enabling the stack during a read process, information read from the drive can be pushed on to the stack. These bytes can then be popped by the microprocessor at a lower speed to examine ID fields and similar information.

During a read process from the peripheral, the AIC-610F has the ability to compare the data being received on a byte-for-byte basis with information found in other locations. When looking for the ID field, the data received is compared to the byte in the data field of the sequencer RAM. This feature is also used during a data field search operation where the information is compared to the data in the external buffer.

If an error is detected after a read data operation, the syndrome is saved in an ECC register and will not be reset until a new read operation is started. For EDAC, Bit 5 of the control field ($A0_H$-$B7_H$) determines if CRC or ECC is to be used. Registers 71_H-77_H control the ECC function. Using these registers, the processor can determine if the error is correctable and calculate the error pattern and displacement from the beginning of the sector. After the error pattern is determined, it is used to correct data bytes in the RAM buffer. The recommended 32-bit ECC polynomial is a computer selected pattern that will correct eight-bit single burst errors. A 48-bit ECC polynomial is available under a license agreement from Adaptec.

The internal operations of the AIC-610F are driven by the Read Reference Clock (RRC). The functions of the AIC-610F are based on the Bit-Ring oscillator. Once the AIC-610F synchronizes with the byte boundary of the incoming bit stream, Bit-Ring 0 will correspond to the first bit (LSB) in each byte.

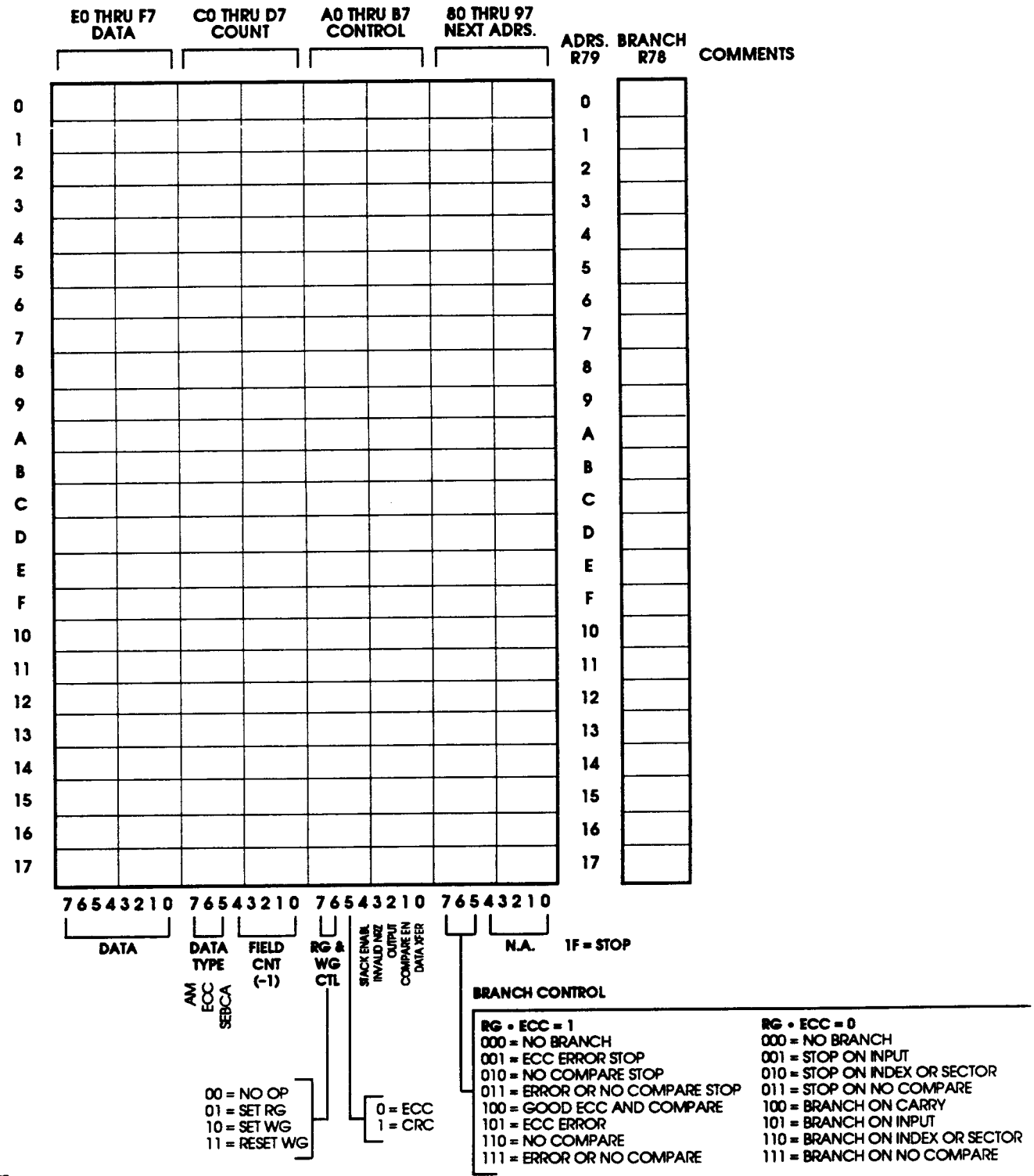
Based on the RRC and SYSCLK inputs, the AIC-610F generates and drives $\overline{CLKA}$. During actual data transfers between the AIC-610F and the storage device, $\overline{CLKA}$ is derived from the RRC. At all other times, $\overline{CLKA}$ is derived from the SYSCLK input. The relationship between $\overline{CLKA}$ and RRC or SYSCLK is based on the contents of the clock control register (Register $7F_H$). NOTE: The frequency of SYSCLK has no bearing on the processor interface.

$\overline{CLKA}$ and $\overline{CLKB}$ are used internally to generate the external RAM buffer address. $\overline{CLKB}$ should be interpreted as the beginning of a controller chip memory access with a $\overline{CLKA}$ period equal to RAM cycle access time. The data bus D(7-0) will contain valid data when $\overline{CLKA}$ is high. The following section on data buffering and transfers explains this operation in detail.

$\overline{CLKA}$ and $\overline{CLKB}$ can also be input signals, thus allowing the AIC-610F to function as a buffer controller. This is particularly applicable when there are two controllers on one board, such as an AIC-610F and an AIC-010 Controller. In this case, the AIC-010 clock outputs ($\overline{CLKA}$ and $\overline{CLKB}$) can drive the buffer management functions of the AIC-610F.

Table 2 shows the sequencer RAM which is located from address 80_H to FF_H. A copy of this table can be used by the user as a worksheet to modify the given sequencer maps or generate a new map based on specific requirements.

TABLE 2. SEQUENCE MEMORY BIT MAP



NOTE:
1. ALL SEQUENCER RAM ADDRESSES AND DATA VALUES ARE IN HEX.

Buffer Management

The AIC-610F can manage buffer sizes from 256 bytes to 64K bytes. The AIC-610F has the necessary registers for DMA control, buffer size, and stop pointers. It also provides the Memory Output Enable ($\overline{MOE}$) signal and Write Enable ($\overline{WE}$) signal. The AIC-610F has two modes of operation of up to 16K addressing and up to 64K addressing.

The ROP/$\overline{WOP}$ bit (Read Operation/Write Operation Register 53_H, Bit 4) controls the direction of data transfer. The WAP registers contain the Write Address Pointer. The RAP registers contain the Read Address Pointer. The SP registers contain the Stop Pointer. The relation of these pointers is described by explaining the actual mechanism of data transfers.

The effectiveness of a controller design is based on its ability to transfer data to and from the peripheral as fast and as accurately as physically possible through Error Detection and Correction (EDAC).

PORT A TRANSFER: The data byte transferred between the AIC-610F and the RAM buffer is called a Port A transfer. Since the drive is continuously spinning during a read or write operation, a byte has to be transferred from/to the AIC-610F controller chip every 800 ns (10 MHz operation). The AIC-610F chip either indicates the availability of a byte (during peripheral read) or requests a byte (during peripheral write), once every byte-time through the $\overline{CLKB}$ line. The appropriate address in the RAM is generated by the AIC-610F from a set of pointers present in the chip.

For a read operation from the peripheral, the ROP/$\overline{WOP}$ bit must be set. The contents of the WAP registers (Registers $5C_H$ and $5D_H$) are used to select the buffer address, and $\overline{WE}$ is used to write information into the buffer.

For a write operation to the peripheral, the ROP/$\overline{WOP}$ bit must be reset. The RAP registers are used to generate the buffer address, and data is read when $\overline{MOE}$ is active. The AIC-610F samples the data from the buffer RAM at the falling edge of $\overline{CLKA}$ following a Port A REQ ($\overline{CLKB}$).

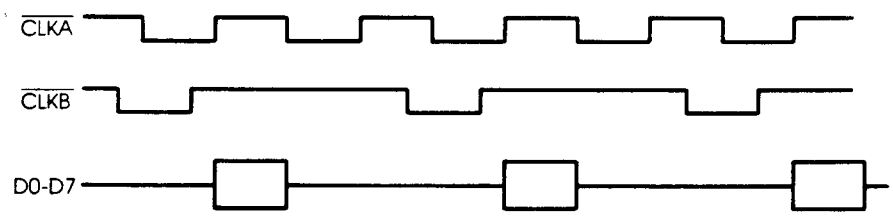


FIGURE 3. AIC-610F PORT A DATA BYTE TRANSFER TIMING

PORT B TRANSFER: The data transfer between the RAM buffer and the host is initiated and controlled by the AIC-610F and is referred to as Port B transfer. The AIC-610F generates the necessary request signal (REQ), RAM buffer address, and control signals. It also generates the host bus latch control signals. The block diagram is shown in Figure 4. The AIC-610F uses the Request signal (REQ) to initiate the transfer which is completed after an Acknowledge (ACK) is received.

If the ROP/$\overline{WOP}$ bit (Register 53_H, Bit 4) is set and the Read Latch is on (Register 53_H, Bit 3), then data is transferred from the buffer to the host. The contents of the RAP registers (Registers $5A_H$ and $5B_H$) are used to generate the addresses. The data is latched out into an external latch with the LO signal and then $\overline{BOE}$ signal enables the data to the host bus. A Request (REQ) is sent to the host. After an Acknowledge (ACK) is received, the $\overline{BOE}$ is deasserted.

If the ROP/$\overline{WOP}$ bit is reset and the Write Latch is on (Register 53_H, Bit 2), then data is transferred from the host bus to the buffer. The contents of the WAP registers (Registers $5C_H$ and $5D_H$) are used to generate the buffer address. The $\overline{BIE}$ line is asserted to enable an external receiver.

The relationships among the different control signals during Port B Host/Buffer transfers are shown in Figures 5 and 6.

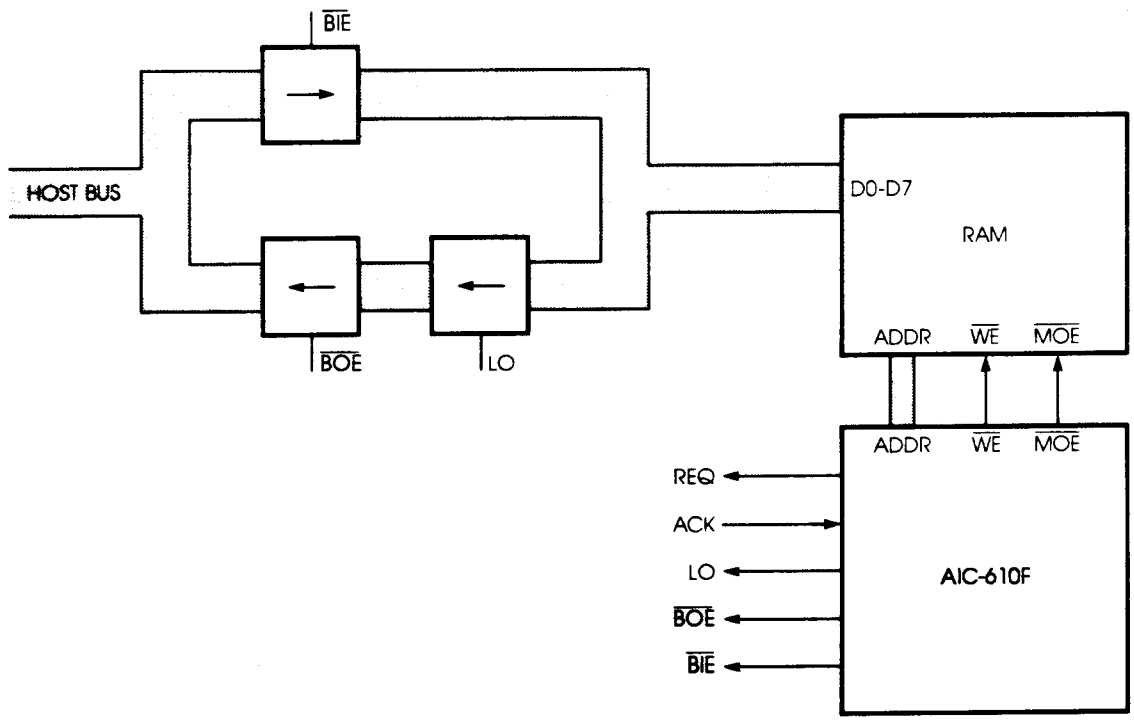


FIGURE 4. BLOCK DIAGRAM OF THE PORT B TRANSFER

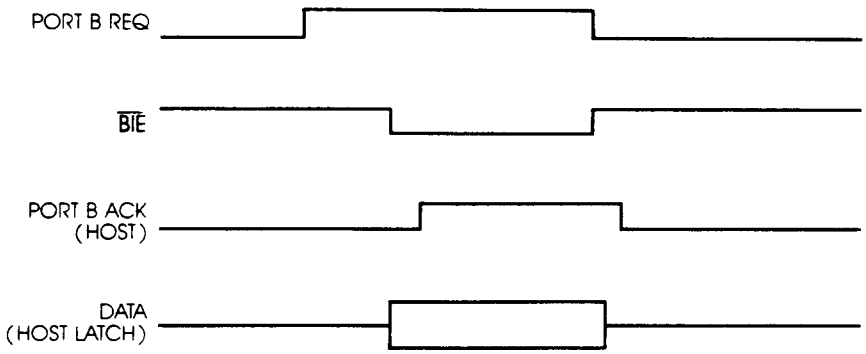


FIGURE 5. HOST TO BUFFER TRANSFER

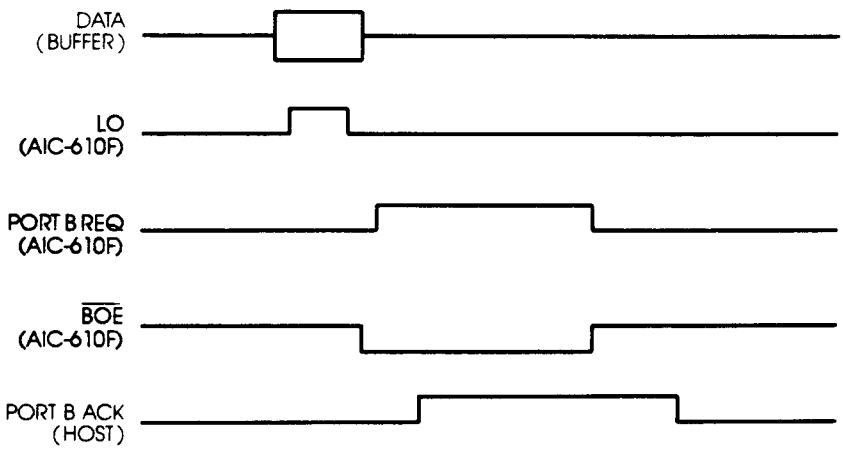


FIGURE 6. BUFFER TO HOST TRANSFER

DATA TRANSFER OVERVIEW: In a controller design, the AIC-610F provides all the necessary signals to interface to the host and the peripheral. The controller also generates the necessary control signals to access the buffer, alternating between Port A and Port B transfers. While the Port A transfer is synchronous in nature (at the data transfer frequency), the Port B transfer is asynchronous and is based on a REQ/ACK handshake with the host.

Thus during any data transfer operation, even though the access to the buffer is interleaved between Port A and Port B, the AIC-610F controller chip should stay at least one sector ahead of the host. This is necessary to keep error detection and correction transparent to the host. The AIC-610F has an internal stop pointer (Register $5E_H$ and $5F_H$) used to prevent overruns. An overview of the buffer operation is shown in Figure 9.

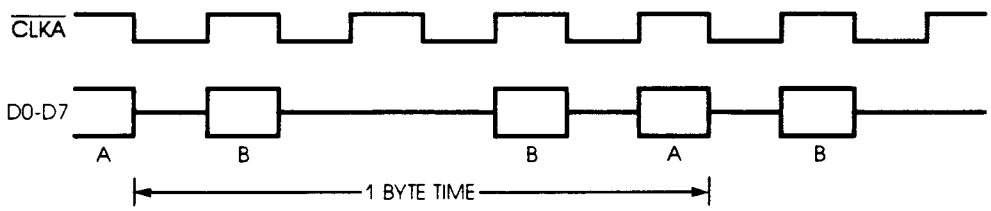


FIGURE 7. DUAL PORT BUFFER TIMING OVERVIEW (FREQUENCY $\overline{CLKA}$ = FREQUENCY RRC/2)

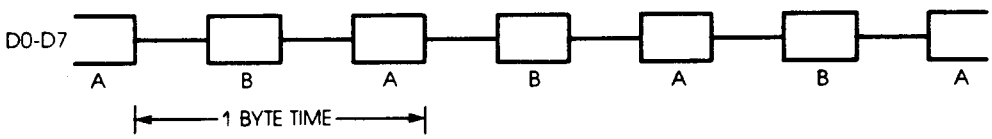


FIGURE 8. DUAL PORT BUFFER TIMING OVERVIEW (FREQUENCY $\overline{CLKA}$ = FREQUENCY RRC/4)

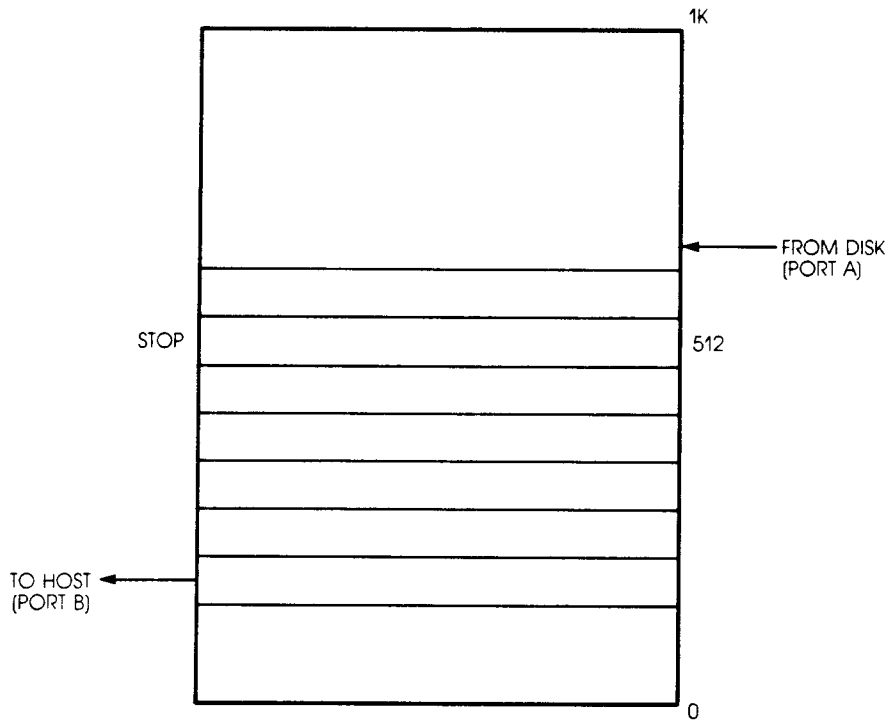


FIGURE 9. BUFFER OPERATION OVERVIEW

During a read operation while data is being transferred into the buffer from the drive (Port A transfer), the previous sector can be transferred to the host (Port B transfer). The stop pointer is set to the end of the sector being transferred to the host, preventing an overrun. At the conclusion of a successful sector read, the stop pointer can be updated to point to the end of this sector. Now this sector can also be transferred to the host, while yet another sector is read in.

The AIC-610F has a built-in priority resolver circuit which allows the synchronous peripheral to have priority over a host request. This is particularly crucial in disk controller applications. For example, in 10 Mb/Sec systems, a data byte has to be transferred once every 800 ns.

TABLE 3. AIC-610F REGISTER SUMMARY

50_H	DATA BUS ACCESS (0-7)
	7 HOST
	6 HOST
	5 HOST
	4 HOST
	3 HOST
	2 HOST
	1 HOST
	0 HOST

51_H	DATA BUS ACCESS (8-15)
	7 HOST
	6 HOST
	5 HOST
	4 HOST
	3 HOST
	2 HOST
	1 HOST
	0 HOST

52_H	HOST INTF CTL
	7 BSY OUT
	6 SEL OUT
	5 BSY IN
	4 SEL IN
	3 BOE
	2 BIE
	1 NOT USED
	0 ARBITRATION

53_H	DMA CTL
	7 ENBL TARGET
	6 NOT USED
	5 DMA DONE
	4 ROP / $\overline{WOP}$
	3 READ LATCH
	2 WRITE LATCH
	1 ACK
	0 REQ

54_H	BUFFER SIZE
	7 2^{15}
	6 2^{14}
	5 2^{13}
	4 2^{12}
	3 2^{11}
	2 2^{10}
	1 2^{9}
	0 2^{8}

59_H	RESET CTL
(WR)	7 NOT USED
	6 NOT USED
	5 NOT USED
	4 NOT USED
	3 NOT USED
	2 NOT USED
	1 NOT USED
	0 REG RST (52-5F)

$5A_H$	RAP (0-7)
(R/W)	7 RAP 7
	6 RAP 6
	5 RAP 5
	4 RAP 4
	3 RAP 3
	2 RAP 2
	1 RAP 1
	0 RAP 0

$5B_H$	RAP (8-15)
(R/W)	7 RAP 15
	6 RAP 14
	5 RAP 13
	4 RAP 12
	3 RAP 11
	2 RAP 10
	1 RAP 9
	0 RAP 8

$5C_H$	WAP (0-7)
(R/W)	7 WAP 7
	6 WAP 6
	5 WAP 5
	4 WAP 4
	3 WAP 3
	2 WAP 2
	1 WAP 1
	0 WAP 0

$5D_H$	WAP (8-15)
(R/W)	7 WAP 15
	6 WAP 14
	5 WAP 13
	4 WAP 12
	3 WAP 11
	2 WAP 10
	1 WAP 9
	0 WAP 8

$5E_H$	STOP (0-7)
	7 STOP 7
	6 STOP 6
	5 STOP 5
	4 STOP 4
	3 STOP 3
	2 STOP 2
	1 STOP 1
	0 STOP 0

$5F_H$	STOP (8-15)
	7 STOP 15
	6 STOP 14
	5 STOP 13
	4 STOP 12
	3 STOP 11
	2 STOP 10
	1 STOP 9
	0 STOP 8

$6E_H$	EXT ADDRS DECODE
	7 NOT USED
	6 NOT USED
	5 NOT USED
	4 NOT USED
	3 NOT USED
	2 NOT USED
	1 NOT USED
	0 NOT USED

$6F_H$	EXT ADDRS DECODE
	7 NOT USED
	6 NOT USED
	5 NOT USED
	4 NOT USED
	3 NOT USED
	2 NOT USED
	1 NOT USED
	0 NOT USED

NOTE: $6E_H$ and $6F_H$ are external address decodes that can be addressed through the AIC-610F. The actual bits in the register are not used.

TABLE 3. AIC-610F REGISTER SUMMARY (Continued)

70_H	BUFFER DATA
R/W	7 BUFFER
	6 BUFFER
	5 BUFFER
	4 BUFFER
	3 BUFFER
	2 BUFFER
	1 BUFFER
	0 BUFFER

71_H	ECC CONTROL
W	7 SEL 32 /48 BIT ECC
	6 RESET / PRESET ECC
	5 CHIP RESET (78-7F)
	4 EN SECTOR BRCH
	3 CLEAR ECC
	2 DISABL FEEDBACK
	1 SHIFT ECC
	0 SERIAL ECC IN

72_H	ECC (32-39) ECC (16-23)
R	7 ECC 23 / 39
	6 ECC 22 / 38
	5 ECC 21 / 37
	4 ECC 20 / 36
	3 ECC 19 / 35
	2 ECC 18 / 34
	1 ECC 17 / 33
	0 ECC 0-16 / 0-32

72_H	POLY (1-8)
W	7 POLY 8
	6 POLY 7
	5 POLY 6
	4 POLY 5
	3 POLY 4
	2 POLY 3
	1 POLY 2
	0 POLY 1

73_H	ECC (40-47) ECC (24-31)
R	7 ECC 31 / 47
	6 ECC 30 / 46
	5 ECC 29 / 45
	4 ECC 28 / 44
	3 ECC 27 / 43
	2 ECC 26 / 42
	1 ECC 25 / 41
	0 ECC 24 / 46

73_H	POLY (9-16)
W	7 POLY 16
	6 POLY 15
	5 POLY 14
	4 POLY 13
	3 POLY 12
	2 POLY 11
	1 POLY 10
	0 POLY 9

74_H	POLY (1-8)/(17-24)
W	7 POLY 8 / 24
	6 POLY 7 / 23
	5 POLY 6 / 22
	4 POLY 5 / 21
	3 POLY 4 / 20
	2 POLY 3 / 19
	1 POLY 2 / 18
	0 POLY 1 / 17

75_H	POLY (9-16)/(25-32)
W	7 POLY 16 / 32
	6 POLY 15 / 31
	5 POLY 14 / 30
	4 POLY 13 / 29
	3 POLY 12 / 28
	2 POLY 11 / 27
	1 POLY 10 / 26
	0 POLY 9 / 25

76_H	POLY(17-24)/(33-40)
W	7 POLY 24 / 40
	6 POLY 23 / 39
	5 POLY 22 / 38
	4 POLY 21 / 37
	3 POLY 20 / 36
	2 POLY 19 / 35
	1 POLY 18 / 34
	0 POLY 17 / 33

77_H	POLY(25-31)/(41-47)
W	7 NOT USED
	6 POLY 31 /47
	5 POLY 30 / 46
	4 POLY 29 / 45
	3 POLY 28 / 44
	2 POLY 27 / 43
	1 POLY 26 / 42
	0 POLY 25 / 41

78_H	BRANCH/NA
R	7 NOT USED
	6 NOT USED
	5 NOT USED
W/R	4 BRCH/NA 4
	3 BRCH/NA 3
	2 BRCH/NA 2
	1 BRCH/NA 1
	0 BRCH/NA 0

79_H	W START ADR/ R READ STATUS
R	7 AM ACTIVE
	6 DATA XFER
	5 BRCH ACTIVE
W/R	4 STOPPED
	3 NOT USED
	2 ECC ERR
	1 COMPARE LOW
	0 COMPARE EQUAL

$7A_H$	OP CTL
W/R	7 INHIBIT CARRY
	6 NOT USED
	5 SUPRES XFER
	4 SRCH OP
	3 NOT USED
R	2 NRZ DATA IN
	1 SECTOR PAST
	0 INDEX PAST

$7B_H$	WAM CTL
W	7 WAM AT BR7
	6 WAM AT BR6
	5 WAM AT BR5
	4 WAM AT BR4
	3 WAM AT BR3
	2 WAM AT BR2
	1 WAM AT BR1
	0 WAM AT BR0

$7C_H$	SYNC CTL
W	7 SYNC MATCH
	6 SYNC MATCH
	5 SYNC MATCH
	4 SYNC MATCH
	3 SYNC MATCH
	2 SYNC MATCH
	1 SYNC MATCH
	0 SYNC MATCH

$7D_H$	GPI/O CTL
W	7 ENABL RD R6F
	6 ENABL SET R6F
	5 ENABL RD R6E
	4 ENABL SET R6E
	3 ENABL GP3 OUT
	2 ENABL GP2 OUT
	1 ENABL GP1 OUT
	0 ENABL GP0 OUT

$7E_H$	GP I/O
	7 NOT USED
	6 NOT USED
R	5 OUTPUT
	4 BRCH IN
R/W	3 GP I/O 3
	2 GP I/O 2
	1 GP I/O 1
	0 GP I/O 0

$7F_H$	CLK CTL
W	7 SEL CLKA FREQ
	6 SEL CLKA FREQ
	5 NOT USED
	4 SEL CLKA FREQ
	3 SEL CLKA, CLKB DIR
	2 SYNC CMPR CTL
	1 SYNC CMPR CTL
	0 SYNC CMPR CTL

$7F_H$	STACK READ/ POP TOP OF STACK
R	7 STACK
	6 STACK
	5 STACK
	4 STACK
	3 STACK
	2 STACK
	1 STACK
	0 STACK

Register Description

50_H PROCESSOR/HOST BUS ACCESS

A Register 50_H decode is used to allow the support processor to access the host data bus. In either the read or write process, the host data bus is bridged across the AIC-610F to the support processor.

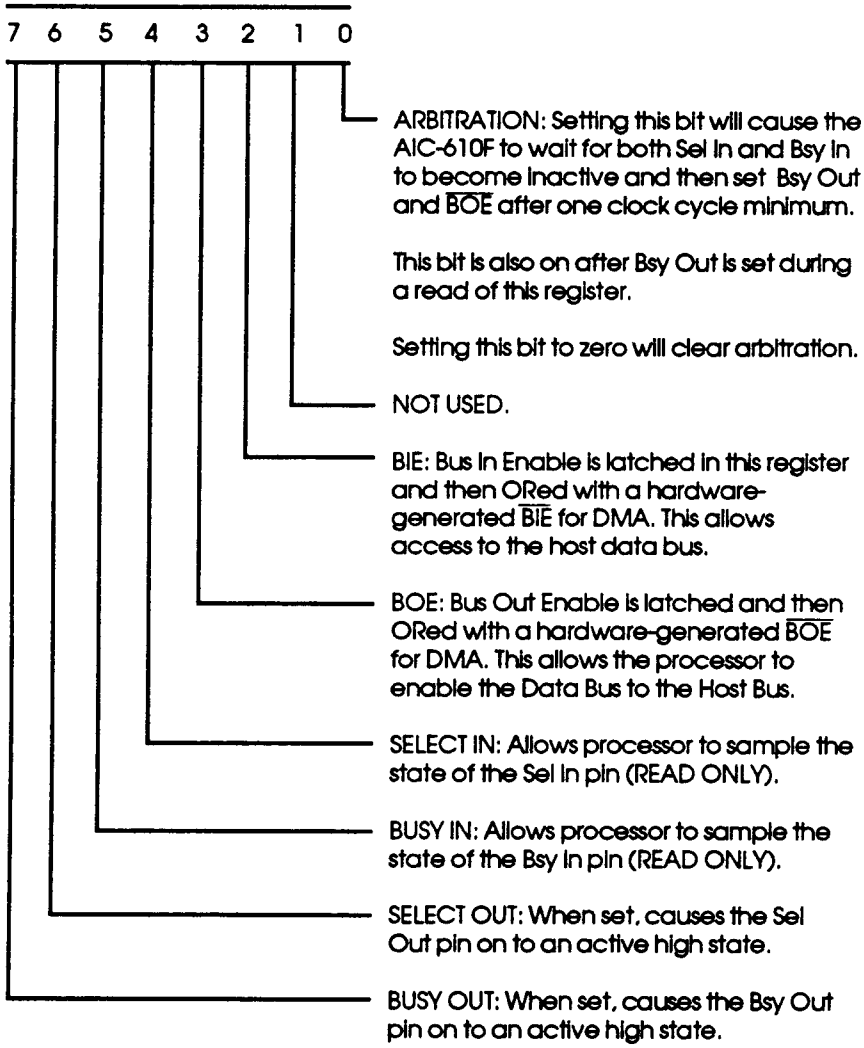
WRITE: A Register 50_H decode and a write from the microprocessor ($\overline{WR}$ asserted) causes LO to be asserted, followed by the $\overline{BOE}$ line being asserted. This allows the data passed through the AIC-610F from the support processor to be latched first before being enabled to the host data bus.

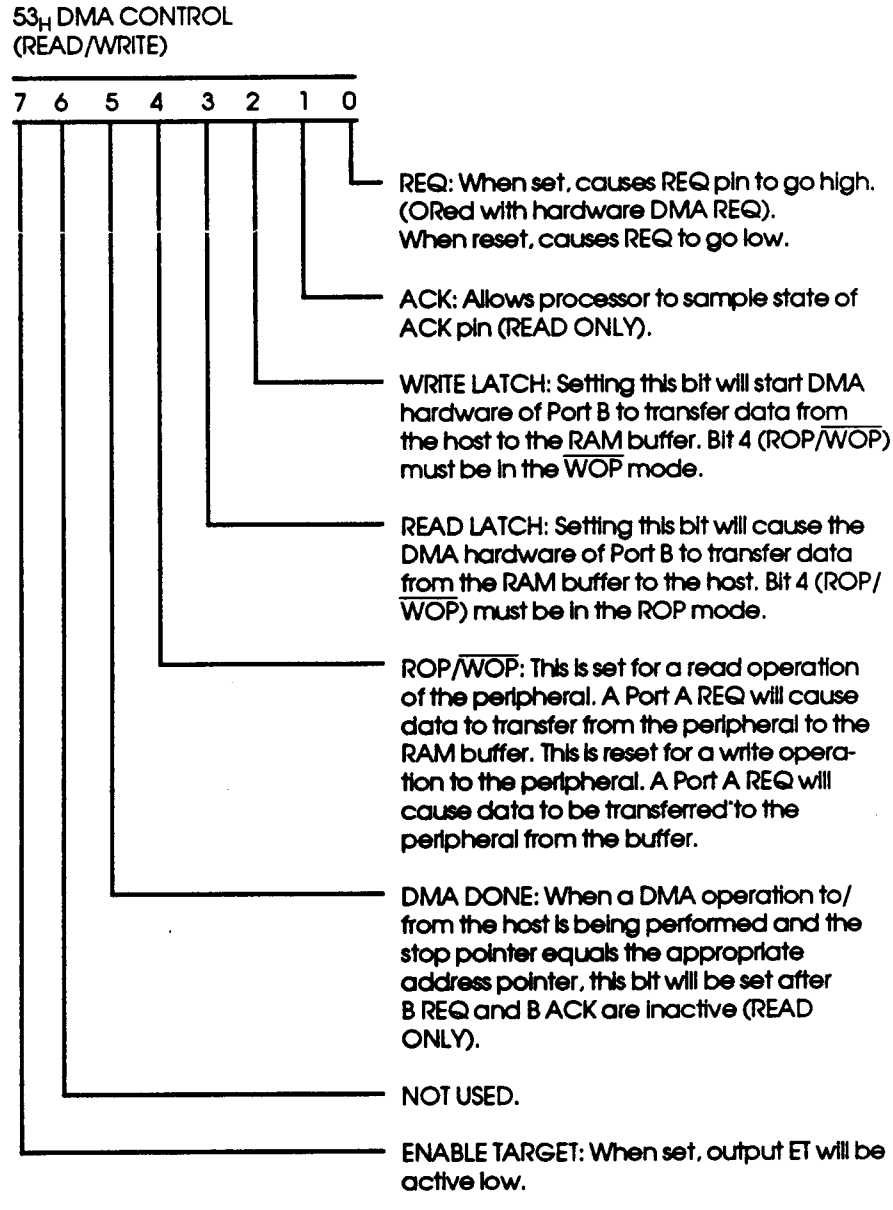
READ: During a Register 50_H decode and a read from the microprocessor ($\overline{RD}$ asserted), the $\overline{BIE}$ line is enabled. This allows the processor to read the available host data through the AIC-610F.

51_H PROCESSOR/HOST BUS ACCESS (HIGH ORDER BYTE)

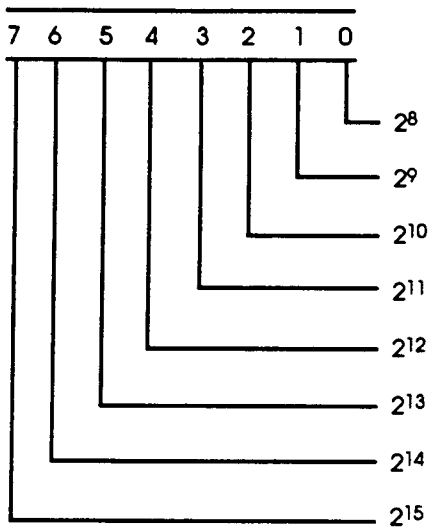
A Register 51_H decode is used to allow the support processor to access the high order byte of the data bus in 16-bit applications. Read and write operations are similar to Register 50_H.

52_H HOST INTERFACE
CONTROL (READ/WRITE)




53_H DMA CONTROL
(READ/WRITE)
7 6 5 4 3 2 1 0
REQ: When set, causes REQ pin to go high. (ORed with hardware DMA REQ). When reset, causes REQ to go low.
ACK: Allows processor to sample state of ACK pin (READ ONLY).
WRITE LATCH: Setting this bit will start DMA hardware of Port B to transfer data from the host to the RAM buffer. Bit 4 (ROP/$\overline{WOP}$) must be in the $\overline{WOP}$ mode.
READ LATCH: Setting this bit will cause the DMA hardware of Port B to transfer data from the RAM buffer to the host. Bit 4 (ROP/$\overline{WOP}$) must be in the ROP mode.
ROP/$\overline{WOP}$: This is set for a read operation of the peripheral. A Port A REQ will cause data to transfer from the peripheral to the RAM buffer. This is reset for a write operation to the peripheral. A Port A REQ will cause data to be transferred to the peripheral from the buffer.
DMA DONE: When a DMA operation to/from the host is being performed and the stop pointer equals the appropriate address pointer, this bit will be set after B REQ and B ACK are inactive (READ ONLY).
NOT USED.
ENABLE TARGET: When set, output ET will be active low.

54_H BUFFER SIZE (WRITE)

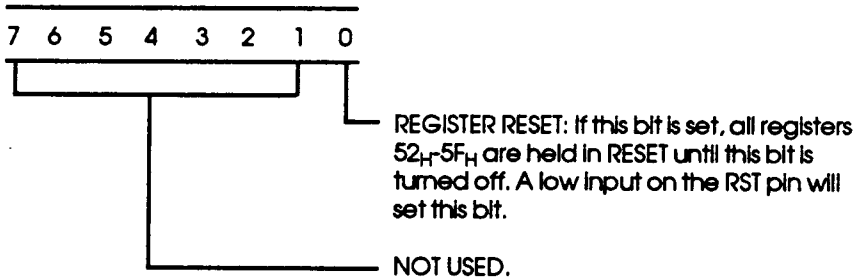


Examples:

- Set Register 54_H to 03_H for a 1K byte buffer.
- Set Register 54_H to $1F_H$ for an 8K byte buffer.

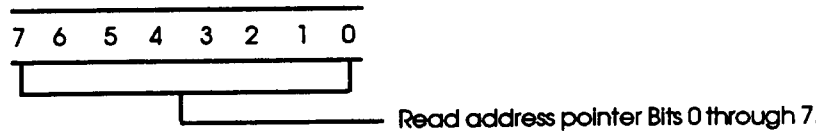
NOTE: When a buffer size greater than 16K bytes is set, address pins $\overline{A12}$ and $\overline{A13}$ function as Set Host Pointer (SHP) and Set Device Pointer (SDP), respectively.

59_H RESET CONTROL (WRITE)



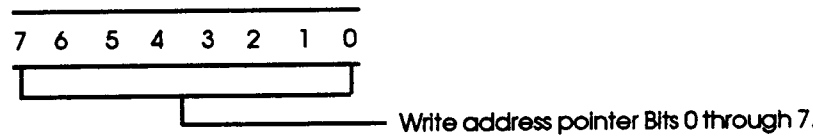
NOTE: Any write to this register will reset WAP, RAP, and SP. If external high-order address latches are used they will not be reset. Therefore, it will be necessary to execute a set Register $5B_H$ and Register $5D_H$ to zero. This will generate the required SHP and SDP strobes.

$5A_H$ RAP (0-7) (READ/WRITE)

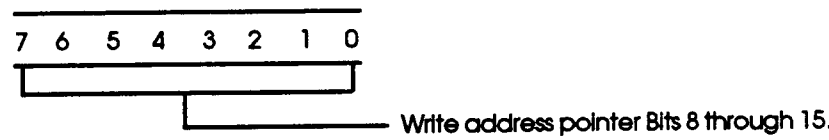


$5B_H$ RAP (8-15) (READ/WRITE)

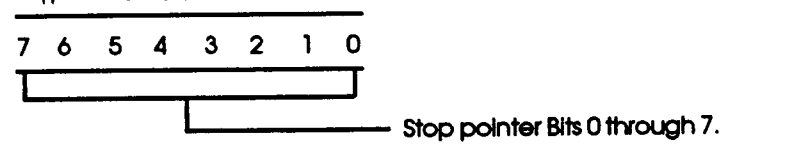
$5C_H$ WAP (0-7) (READ/WRITE)



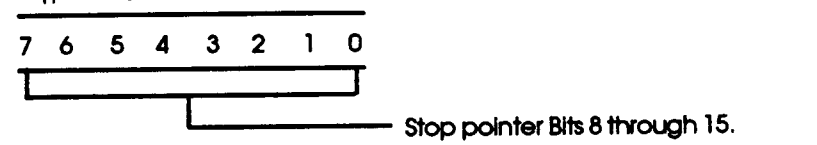
$5D_H$ WAP (8-15) (READ/WRITE)



$5E_H$ STOP (0-7) (READ/WRITE)



$5F_H$ STOP (8-15) (READ/WRITE)



$6E_H$ EXTERNAL REGISTER ACCESS/EXTERNAL BUFFER ENABLE

A read or write to this register can be made to generate a negative pulse on two (of the four) GP I/O pins. Thus Register $6E_H$ is addressed through the AIC-610F, yet its function is to enable an external buffer or clock an external latch. This pulse can be used to simplify the interface to various drive interfaces such as ESDI. Figure 10 shows how Register $6E_H$ may be used.

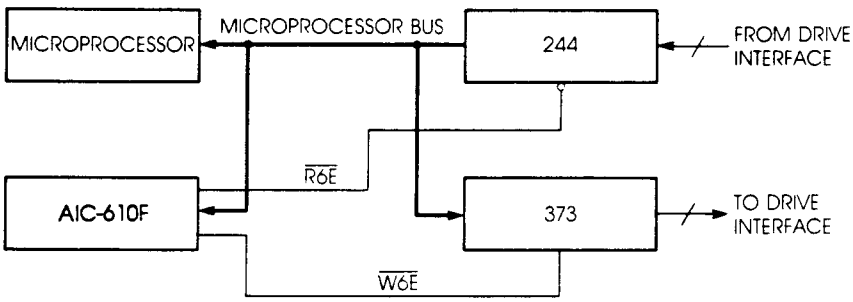


FIGURE 10. REGISTERS $6E_H$ AND $6F_H$ USAGE

$6F_H$ EXTERNAL REGISTER ACCESS/EXTERNAL BUFFER ENABLE

A read or write to this register can be made to generate a negative pulse on two of the four GP I/O pins. Thus Register $6F_H$ is addressed through the AIC-610F, yet its function is to enable an external buffer or clock an external latch. This pulse can be used to simplify the interface to various drive interfaces.

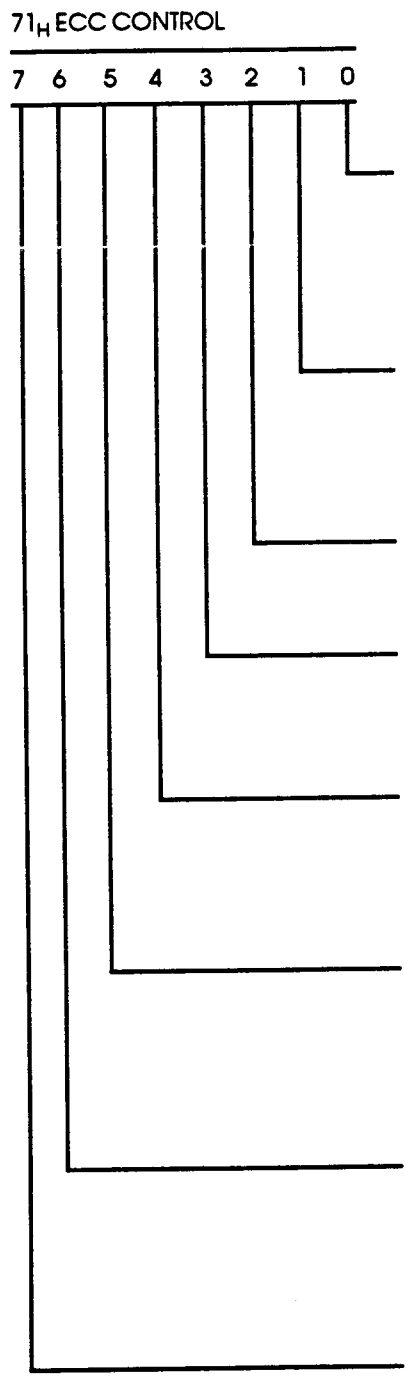
NOTE: Before using Registers $6E_H$ and $6F_H$, the GP I/O control register (Register $7D_H$) must be set up for required function.

70_H PROCESSOR/RAM BUFFER ACCESS

Register 70_H decode is used to allow the support processor to access the buffer. A read (on Register 70_H) causes the processor data bus to be bridged to the RAM data bus via the AIC-610F. $\overline{MOE}$ is asserted and $\overline{WE}$ is set up for read/write operations.

The address selected is the contents of the WAP register if the ROP/$\overline{WOP}$ (Register 53_H, Bit 4) is set (read disk). Otherwise, it is the contents of the RAP register if the ROP/$\overline{WOP}$ bit is reset.

NOTE: See special requirements, described in the hardware section, for a buffer greater than 16K.



SERIAL ECC INPUT: ECC Bit 0 will be loaded with the contents of this bit when the shift control bit is set. Note that the RRC pin must be cycling. RG and WG must be inactive and feedback must be disabled.

ECC SHIFT CONTROL: Each time this bit is set, a single shift pulse will be sent to the ECC register. This bit is automatically cleared after the shift pulse occurs.

DISABLE ECC FEEDBACK: When set, causes the ECC polynomial to function as a simple shift register.

CLEAR ECC: While this bit is on, the ECC shift register will be reset or preset, based on the state of Bit 6 in this register.

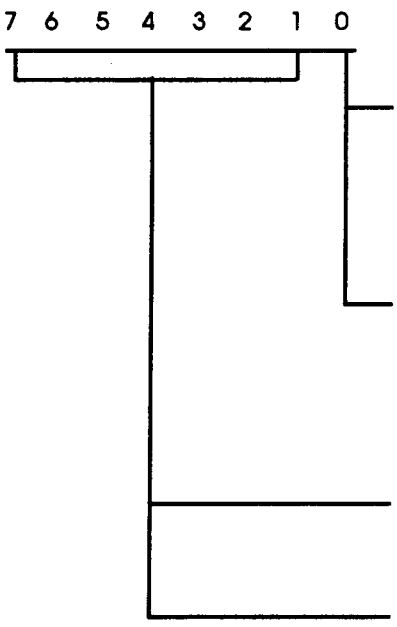
ENABLE SECTOR BRANCH: When set, will cause the sector input to be ORed with the index so that an operation may begin at index or sector.

CHIP RESET: When an external reset occurs, this bit will be set. A constant reset will be generated to the chip while it is set. The reset condition is cleared by setting this bit to zero from the microprocessor.

ECC RESET/PRESET: This bit defines the initial state of the ECC and CRC shift register during read or write operations.
0 = Preset the registers.
1 = Reset the registers.

POLYNOMIAL LENGTH: Controls the length of the ECC polynomial.
0 = 48-bit ECC polynomial enabled.
1 = 32-bit ECC polynomial enabled.

72_H ECC SYNDROME (16-23)/ ECC SYNDROME (32-39) (READ)



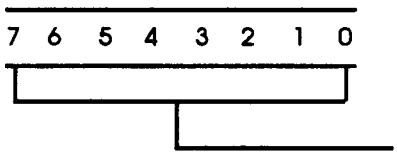
48-BIT ECC SYNDROME: This bit is an OR of the ECC syndrome, Bits 0 through 32. Whenever one of these bits is set, this bit will also be set. This OR is gated by the appropriate length selected.

32-BIT ECC SYNDROME: This bit is an OR of the ECC syndrome, Bits 0 through 16. Whenever one of these bits is set, this bit will also be set. This OR is gated by the appropriate length selected.

48-BIT ECC SYNDROME—ECC BITS 33 THROUGH 39: Bit 39 is in Bit 7 of Register 72_H.

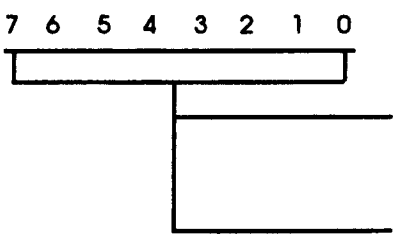
32-BIT ECC SYNDROME—ECC BITS 17 THROUGH 23: Bit 23 is in Bit 7 of Register 72_H.

72_H ECC POLY (1-8) (WRITE)



Each bit corresponds to a feedback path being enabled; i.e., if Bit 0 (i.e., Poly Bit 1) in the register is on, the output (ECC 47) will be XORed with the data in and then XORed with ECC Bit 0 and the result is the input to ECC Bit 1. This register represents Bits 1 to 8 for the 48-bit ECC polynomial. Register 72_H, Bit 7, represents Bit 8 of the 48-bit ECC polynomial.

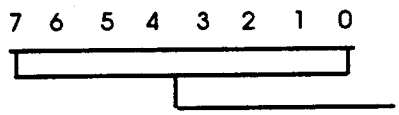
73_H ECC SYNDROME (24-31)/ ECC SYNDROME (40-47) (READ)



48-BIT ECC SYNDROME—ECC BITS 40 THROUGH 47: Bit 47 is in Bit 7 of Register 73_H.

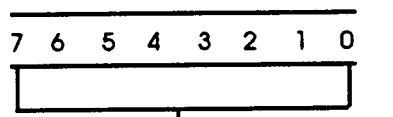
32-BIT ECC SYNDROME–ECC BITS 24 THROUGH 31: Bit 31 is in Bit 7 of Register 73_H.

73_H ECC POLY (9-16) (WRITE)



Each bit corresponds to a feedback path being enabled; i.e., if Bit 0 (i.e., Poly Bit 9) in the register is on, the output (ECC 47) will be XORed with the data in and then XORed with ECC Bit 8 and the result is the input to ECC Bit 9. This register represents Bits 9 to 16 for the 48-bit ECC polynomial. Bit 7 represents Bit 16 of the 48-bit ECC polynomial.

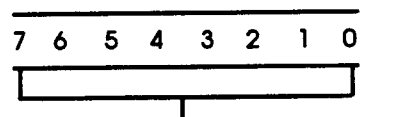
74_H ECC POLY (1-8)/
ECC POLY (17-24) (WRITE)



48-BIT ECC POLYNOMIAL: This register represents polynomial from 17 to 24 for the 48-bit ECC polynomial. Bit 7 represents Bit 24 of the polynomial.

32-BIT ECC POLYNOMIAL: This register represents polynomial from 1 to 8 for the 32-bit ECC polynomial. Bit 7 represents Bit 8 of the polynomial.

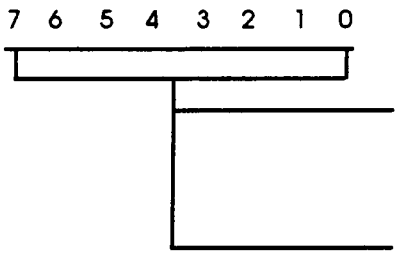
75_H ECC POLY (9-16)/
ECC POLY (25-32) (WRITE)



48-BIT ECC POLYNOMIAL: This register represents polynomial from 25 to 32 for the 48-bit ECC polynomial. Bit 7 represents Bit 32 of the polynomial.

32-BIT ECC POLYNOMIAL: This register represents polynomial from 9 to 16 for the 32-bit ECC polynomial. Bit 7 represents Bit 16 of the polynomial.

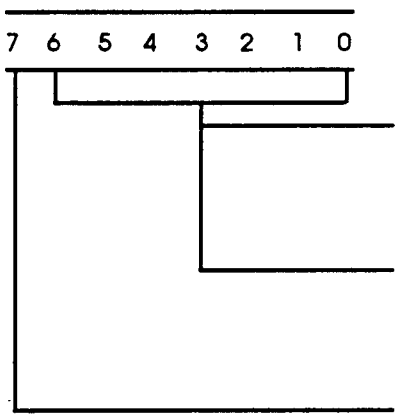
76_H ECC POLY (17-24)/
ECC POLY (33-40) (WRITE)



48-BIT ECC POLYNOMIAL: This register represents polynomial from 33 to 40 for the 48-bit ECC polynomial. Bit 7 represents Bit 40 of the polynomial.

32-BIT ECC POLYNOMIAL: This register represents polynomial from 17 to 24 for the 32-bit ECC polynomial. Bit 7 represents Bit 24 of the polynomial.

77_H ECC POLY (25-31)/
ECC POLY (41-47) (WRITE)



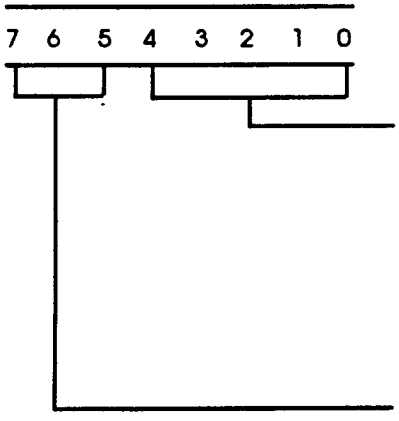
48-BIT ECC POLYNOMIAL: This register represents polynomial from 41 to 47 for the 48-bit ECC polynomial. Bit 7 represents Bit 47 of the polynomial.

32-BIT ECC POLYNOMIAL: This register represents polynomial from 25 to 31 for the 32-bit ECC polynomial. Bit 7 represents Bit 31 of the polynomial.

NOT USED.

NOTE: Registers 74_H through 77_H are not reset by the reset latch.

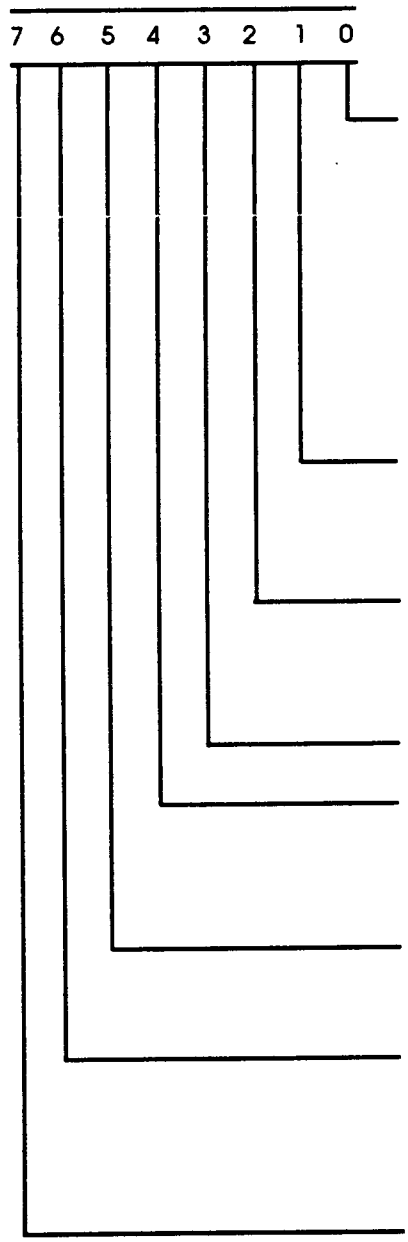
78_H BRANCH/NEXT ADDRESS
(READ/WRITE)



BRANCH/NEXT ADDRESS: Writing Bits 0-4 sets the sequencer branch address register. When a sequencer branch condition is met, the sequencer will jump to this address. A read of this register (Bits 0-4) gives the next address the sequencer will execute, except during a search for AM.

NOT USED.

79_H CONTROLLER STATUS (READ)



All bits reset by external reset.

COMPARE EQUAL: The state of the compare operation, as a result of all bytes where comparison was enabled. The comparison is done between the data buffer or sequencer RAM and the read data register. The final result is based on whether all enabled bytes compared. The value of this bit is valid only after the ECC bytes have been read in.

COMPARE LOW: Same as above, except that the data buffer or sequencer RAM was greater than the read data register.

ECC ERROR: After the last bit of ECC data is read, this bit is either set or reset depending upon whether all bits in the ECC are zero.

NOT USED: Will be zero.

STOPPED: The sequencer RAM is at address $1F_H$. The ECC contents have not been reset and RG and WG are reset. The bit ring is running.

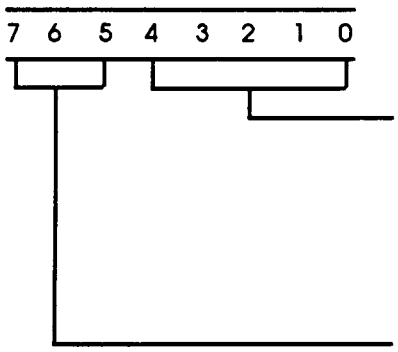
BRANCH ACTIVE: This bit is set whenever a branch condition is met. The bit is reset by a read of this register.

DATA TRANSFER: This bit is asserted whenever data is being transferred either to or from the buffer memory; i.e., it is the data XFER enable bit of the sequencer RAM.

AM ACTIVE: Is set by reading or writing an AM or SYNC byte and is reset by reading or writing the ECC bytes. The bit is also reset by a stopped condition.

NOTE: The COMPARE bits and ECC ERROR bit remain valid for the last read until the next sequencer word which reads or writes an ECC is executed.

79_H SEQUENCER START (WRITE)



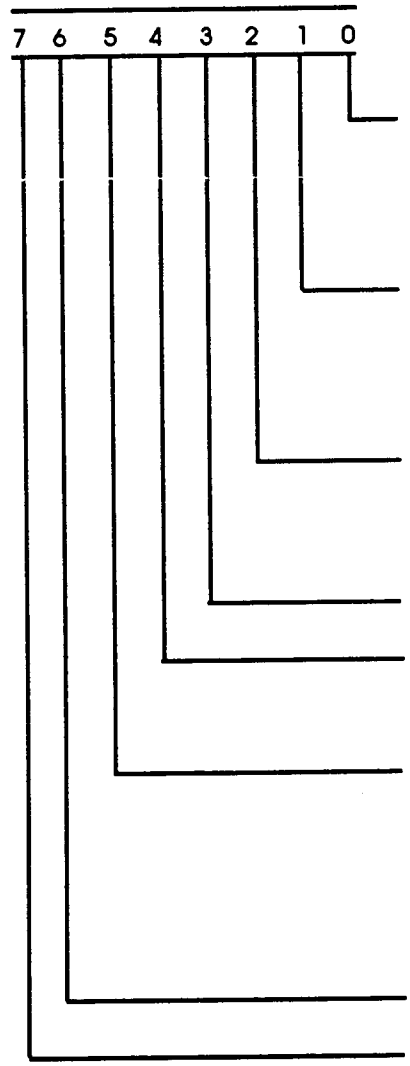
START ADDRESS: A write to Bits 0-4 will start the sequencer at the appropriate address. This register may only be set when Register 79_H, Bit 4 (STOPPED), indicates the sequencer is in the correct initial stopped state.

NOT USED.

NOTE: In normal operation, it should not be necessary to stop the sequencer by setting a $1F_H$ to Register 79_H. However, when a sync character is missed and the microprocessor timer expires, the AIC-610F should be stopped and restarted to retry the operation. The recommended way to do this is in a loop that sets $1F_H$ to Register 79_H and then examines the stopped bit. If not stopped, repeat the $1F_H$ to 79_H. The AIC-610F will typically stop the first time, but occasionally two or three loops may be required.

The start or stop of the sequencer may take from 0 to 8 RRC cycles.

$7A_H$ OPERATION CONTROL
(READ/WRITE)



INDEX PAST: Index pulse from the device has been detected since the last time this register was read. Reading this bit while index is present does not reset the bit.

SECTOR PAST: Sector pulse has been received from the device since the last read of this register. Reading this bit while sector is present does not reset the bit.

NRZ DATA IN: An input of "1" on the NRZ data pin while read gate was on has occurred since the last time this register was accessed.

NOT USED: Will be a one.

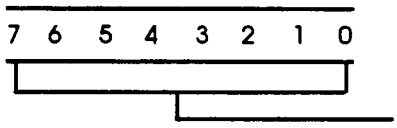
SEARCH OPERATION: This bit must be set whenever a data field compare is required.

SUPPRESS TRANSFER: $\overline{\text{CLKB}}$ will not be generated when this bit is on. Also, during WG, the data field will be written with the contents of the sequencer RAM data or during RG compared with the contents of sequencer RAM data.

NOT USED.

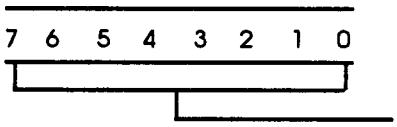
INHIBIT DATA FIELD CARRY: When set, the carry/load of the sequencer for the data field will be inhibited. After a carry has occurred, this bit will be reset. This is used to cause the sequencer to execute its present address again. For this time, however, the count field will start at 00 and count down (256 cycles will be executed).

$7B_H$ WAM CONTROL
(WRITE ONLY)



WRITE ADDRESS MARK CONTROL: The WAM/AMD pin will go active for each bit cell time corresponding to the bits set in this register during a write address mark operation. This is used to indicate to external logic when the clock pulse should be deleted from the outgoing data stream after encoding. This, then, creates an illegal pattern which becomes the Address Mark.

$7C_H$ SYNC DETECT CONTROL
(WRITE ONLY)

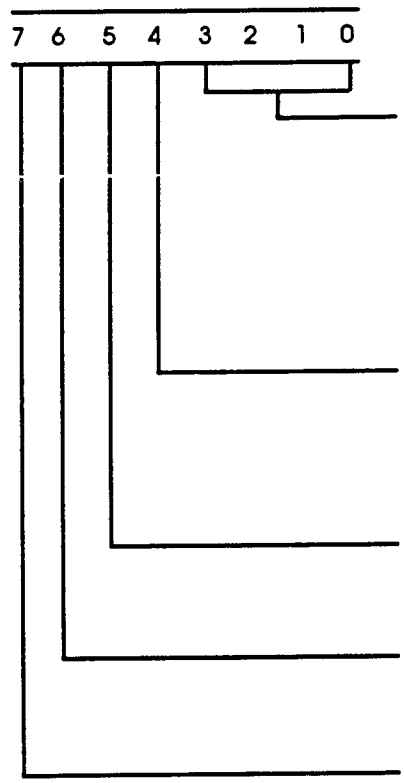


SYNC DETECT CONTROL: A match between this register and the serial NRZ read data input will cause a sync detect (if AMD input is active). It will also cause the bit ring to start at zero and data to be gated into the ECC. Only those bits enabled by Register $7F_H$ can be set for comparison.

NOTE: Only those bits enabled by Register $7F_H$ can be set for comparison. Any don't-care bits must be set to zero. Whenever Read Gate is turned on, the Bit-Ring oscillator stops within two byte times. For this reason, the sequencer word, which turns on RG, must have a count of 01_H. At this point, the AIC-610F starts to shift in the data on the NRZ pin. The bit stream is compared with the contents of this register in order to obtain byte sync. In addition to byte sync, an AM detect must also be observed on pin 63 (of the 68 pin PLCC) for the Bit Ring oscillator to start up. This comparison is independent of the Compare Enable function in the sequencer RAM which is normally used for additional qualification, such as differentiating between ID Address Mark and Data Address Mark.

Some sync characters like FE_H can cause problems if the chip is not set up properly. This is because a floating NRZ line (read and write gates not active) shifts 1's into the sync comparison register. When read gate is turned on over the preamble, an FE_H is framed, causing improper alignment. This can be avoided by first turning on the invalid NRZ bit (Bit 3 in control field) along with read gate for a few byte times. After this, the next word should have a count of 01_H which is then followed by the sync word.

$7D_H$ GP I/O CONTROL
(WRITE ONLY)



GP I/O DIRECTION CONTROL: When set, these bits enable the corresponding bits of the GP I/O register to the output pins. When these bits are zero, the pins are the source of the GP I/O register; i.e., inputs are gated to the bus when a read of GP I/O is done.

W6E CONTROL: When set along with Bit 0, this bit will disable GP I/O register Bit 0 as an output and enable a set Register $6E_H$ output pulse. When zero, the GP I/O register is the output.

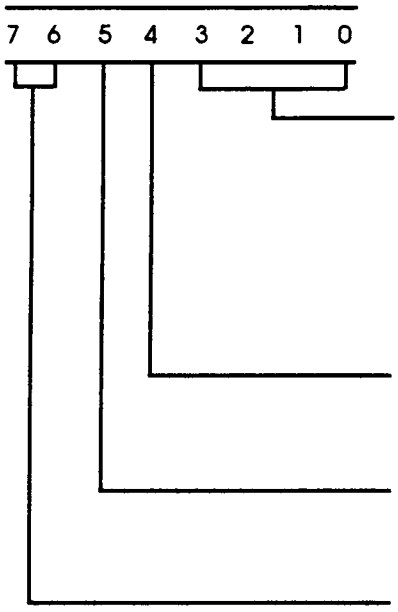
R6E CONTROL: Same function as above, except a read pulse for Register $6E_H$ will be output from pin 1.

W6F CONTROL: Same function as above, except a write pulse for Register $6F_H$ will be output from pin 2.

R6F CONTROL: Same function as above, except a read pulse for Register $6F_H$ will be output from pin 3.

NOTE: To set up the AIC-610F for $6E_H$ or $6F_H$ decode, first set Bits 0 to 3 before setting Bits 4 to 7 (i.e., write a $0F_H$, followed by FF_H).

$7E_H$ GP I/O (READ/WRITE)



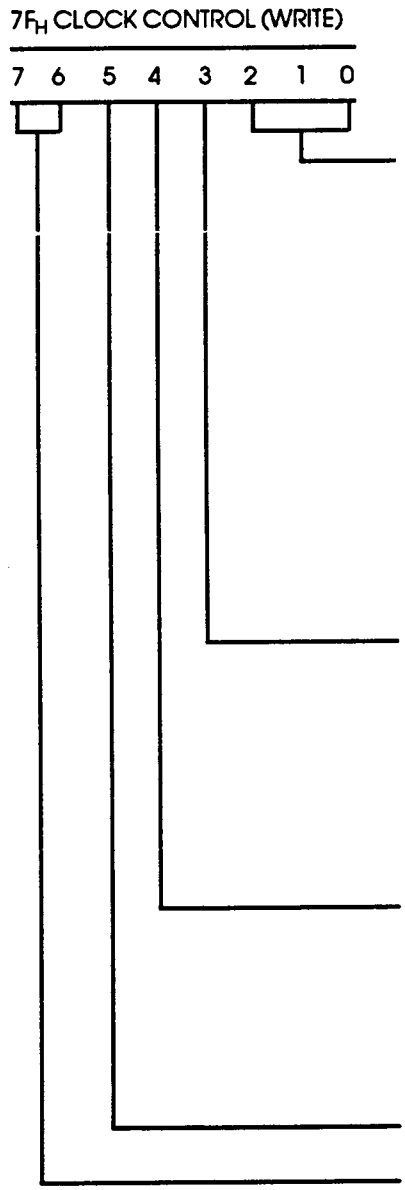
GP I/O BITS 0-3: General purpose input/output bits (Bits 0-3) are independently programmable for direction of data on the four pins and each bit has a gated latch that holds data for output purposes only. A read of this register will gate the contents of the input pin.

INPUT PIN: An external input that may be used as a branch condition for the sequencer RAM.

OUTPUT PIN: This bit indicates the state of the output bit in the sequencer RAM.

NOT USED.

NOTE: Register $7D_H$ must be set up prior to accessing Register $7E_H$.



SYNC COMPARE CONTROL:
Specifies the number of bits to be used in the compare for the sync byte.
000 = No compare.
001 = Only Bit 7 compared.
010 = Only Bits 7 and 6 compared.
011 = Only Bits 7, 6, and 5 compared.
100 = Only Bits 7, 6, 5, and 4 compared.
101 = Only Bits 7, 6, 5, 4, and 3 compared.
110 = Only Bits 7, 6, 5, 4, 3, and 2 compared.
111 = All Bits compared.

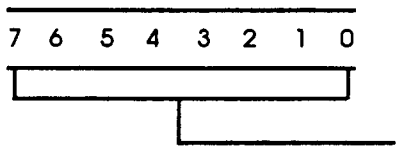
$\overline{CLKA}$, $\overline{CLKB}$ DIRECTION CONTROL:
0 = $\overline{CLKA}$, $\overline{CLKB}$ are outputs to control external memory and are generated by the AIC-610F.
1 = $\overline{CLKA}$, $\overline{CLKB}$ are inputs allowing an external controller to control the AIC-610F's buffer management functions.

0 = $\overline{CLKA}$ during sequencer data transfer will be 1/2 of RD/REF clock frequency.
1 = $\overline{CLKA}$ during sequencer data transfer will be 1/4 of RD/REF clock frequency.

NOT USED.

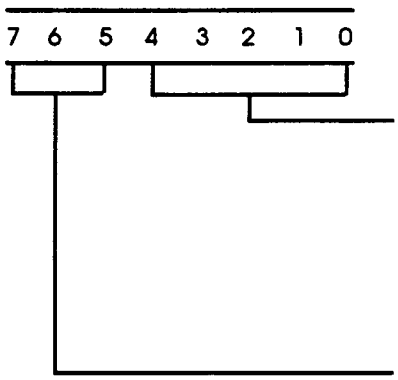
00 = $\overline{CLKA}$ during nonsequencer data transfer will be 1/4 of SYSCLK frequency.
01 = $\overline{CLKA}$ during nonsequencer data transfer will be 1/2 of SYSCLK frequency.
10 = $\overline{CLKA}$ during nonsequencer data transfer will be equal to SYSCLK frequency.
11 = $\overline{CLKA}$ will not provide valid information.

$7F_H$ STACK (READ)



STACK: A read of this register will read and pop the top of the 8-byte stack. The stack wraps around on the 8th pop.

80_H THRU 97_H NEXT ADDRESS FIELD (READ/WRITE)

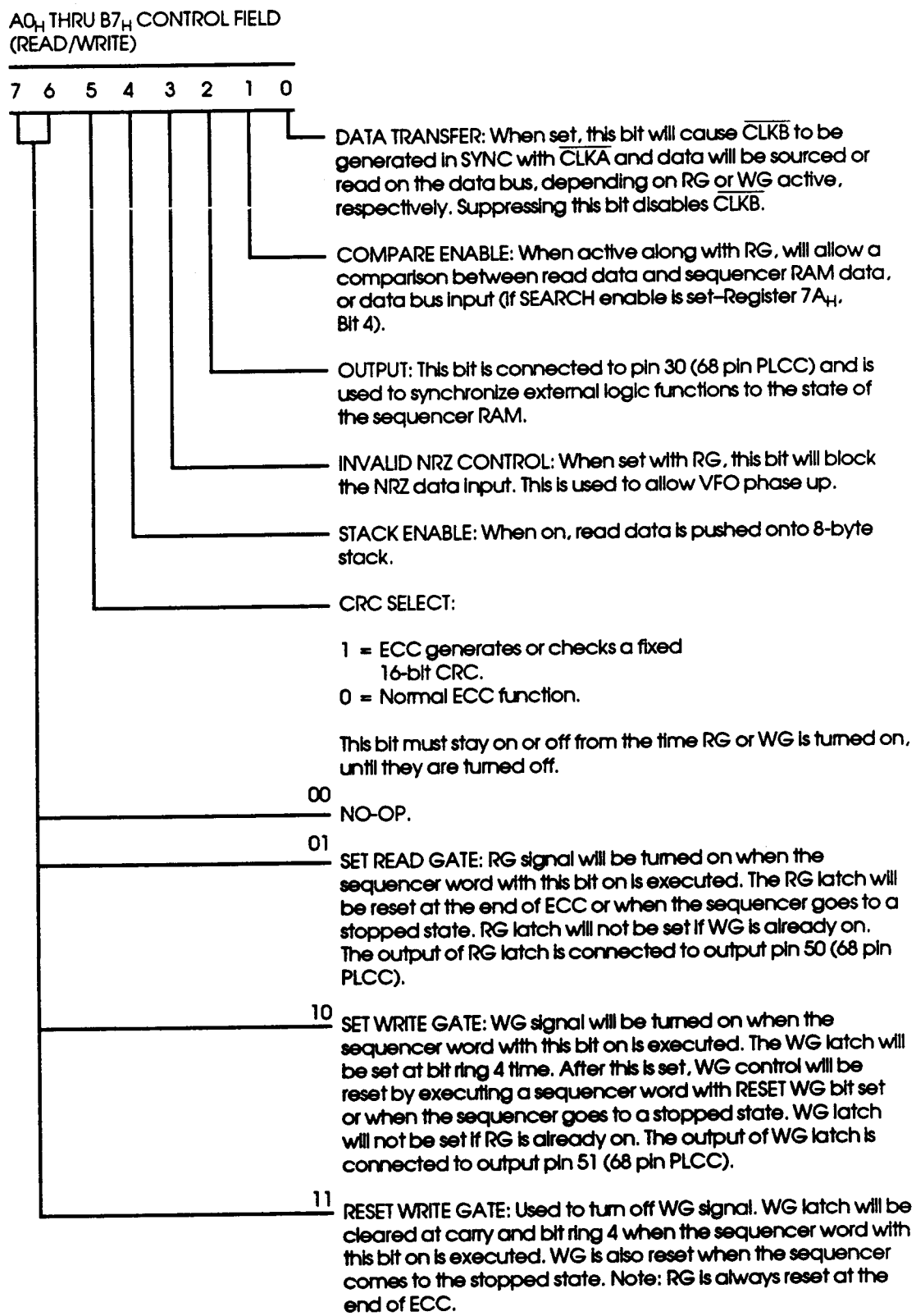


NEXT ADDRESS: This is the address the sequencer will go to after the down counter has reached zero and a branch has not been taken. There are 24 possible next-address locations (00 to 17). Addresses from 18 to 1F establish a stopped condition.

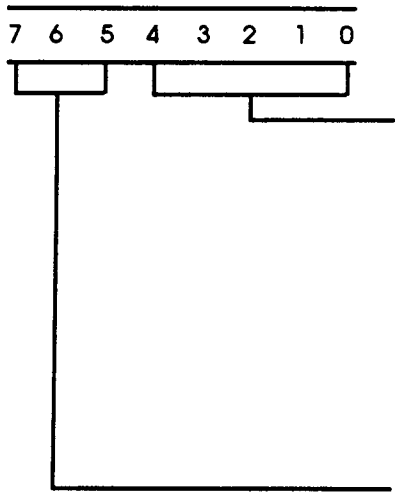
BRANCH CONDITIONS: Branch conditions when *both ECC and read gate are active.* These branches are taken at the end of ECC time.
000 = Continue, next address used.
001 = Stop on ECC error.
010 = Stop on no compare equal.
011 = Stop on no compare equal or ECC error.
100 = Branch on good ECC and compare equal.
101 = Branch on ECC error.
110 = Branch on no compare equal.
111 = Branch on no compare equal or ECC error.

Branch condition *at all other times.* These branches are taken when the operations specified by that word are complete.
000 = Continue, next address used.
001 = Stop if INPUT is active.
010 = Stop if INDEX or SECTOR is active.
011 = Stop on no compare equal.
100 = Branch on carry (expiration of count).
101 = Branch on INPUT active.
110 = Branch on INDEX or SECTOR active.
111 = Branch on no compare equal.

NOTE: The compare equal flag is checked after the ECC bytes have been read in. The compare equal flag is reinitialized when read gate is turned on.


A0H THRU B7H CONTROL FIELD
(READ/WRITE)
7 6 5 4 3 2 1 0
DATA TRANSFER: When set, this bit will cause CLKB to be generated in SYNC with CLKA and data will be sourced or read on the data bus, depending on RG or WG active, respectively. Suppressing this bit disables CLKB.
COMPARE ENABLE: When active along with RG, will allow a comparison between read data and sequencer RAM data, or data bus input (if SEARCH enable is set–Register 7AH, Bit 4).
OUTPUT: This bit is connected to pin 30 (68 pin PLCC) and is used to synchronize external logic functions to the state of the sequencer RAM.
INVALID NRZ CONTROL: When set with RG, this bit will block the NRZ data input. This is used to allow VFO phase up.
STACK ENABLE: When on, read data is pushed onto 8-byte stack.
CRC SELECT:
1 = ECC generates or checks a fixed 16-bit CRC.
0 = Normal ECC function.
This bit must stay on or off from the time RG or WG is turned on, until they are turned off.
00
NO-OP.
01
SET READ GATE: RG signal will be turned on when the sequencer word with this bit on is executed. The RG latch will be reset at the end of ECC or when the sequencer goes to a stopped state. RG latch will not be set if WG is already on. The output of RG latch is connected to output pin 50 (68 pin PLCC).
10
SET WRITE GATE: WG signal will be turned on when the sequencer word with this bit on is executed. The WG latch will be set at bit ring 4 time. After this is set, WG control will be reset by executing a sequencer word with RESET WG bit set or when the sequencer goes to a stopped state. WG latch will not be set if RG is already on. The output of WG latch is connected to output pin 51 (68 pin PLCC).
11
RESET WRITE GATE: Used to turn off WG signal. WG latch will be cleared at carry and bit ring 4 when the sequencer word with this bit on is executed. WG is also reset when the sequencer comes to the stopped state. Note: RG is always reset at the end of ECC.

$C0_H$ THRU $D7_H$ COUNT FIELD
(READ/WRITE)



COUNT: These bits are the initial value of the sequencer counter when a new state is entered. Bits 0 thru 4 of the counter are set to Bits 0 thru 4 of the counter field, respectively. The counter is decremented on bit ring 7. When it reaches zero, a new state will be accessed from the sequencer RAM. The value specified here must be one less than the count to be executed (for a count of 256, set FF_H).

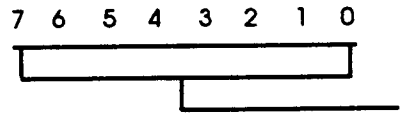
DATA TYPE: When the DATA TRANSFER bit (Control Byte, Bit 0) of the sequencer RAM is off, these bits are decoded for data type as indicated below:

000 = Normal.
100 = Address Mark.
010 = ECC. During an ECC Read/Write, one additional byte of data is read/written due to the one byte delay in ECC computation.
001 = Set enable bit ring to $\overline{CLKA}$ (SEBCA). Reset at end of ECC. Used to synchronize $\overline{CLKB}$. Turning on this bit causes the sequencer to switch from SYSCLK to RRC as the source for $\overline{CLKA}$. This is done prior to a data transfer occurring between the device and the AIC-610F (READ OP) or the buffer and the AIC-610F (WRITE OP). *This bit should be turned on at least two-byte times before the data field.*

When the DATA TRANSFER bit is OFF, Bits 5, 6, and 7 of the counter will be initialized to zero.

When the DATA TRANSFER bit is set, Bits 5, 6, and 7 of the counter will be initialized with Bits 5, 6, and 7 of this register.

$E0_H$ THRU $F7_H$ DATA FIELD
(READ/WRITE)



DATA: This register is the source for all overhead bytes of data used by the device during write operations. During read operations, it is one of the operands to the comparison logic.

When DATA TRANSFER is on with WG, the source for write data will be the external data bus. However, when SUPPRESS TRANSFER is on with WG, this register will again be the source for write data.

NOTE: The sequencer RAM (addresses 80_H-97_H, $A0_H$-$B7_H$, $C0_H$-$D7_H$, and $E0_H$-$F7_H$) may only be set by the support microprocessor when there is no risk of the contents being accessed by the sequencer. This is normally true only during long data transfers or when the sequencer is stopped.

In order to shift the data into the ECC registers during a disk read operation, the following events have to occur:

1. RG must be on.
2. AM detect must take place (on pin 63 for 68 pin PLCC).
3. Sync detect must take place (match between the contents of Register $7C_H$ and NRZ read data).

AIC-610F HARDWARE OVERVIEW

The next few sections explain the various hardware interfaces of the AIC-610F to ensure a successful operation. The sections are as follows:

- Support Processor Interface
- RAM Buffer Interface
- Peripheral Interface
- System/Host Bus Interface

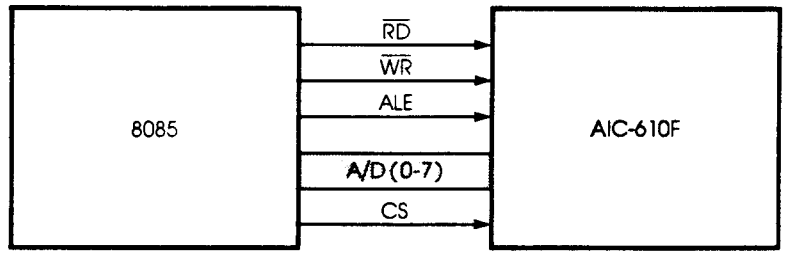


FIGURE 11. BASIC INTERFACE BETWEEN AN 8085 AND AN AIC-610F

Support Processor Interface

The AIC-610F is set up and monitored by a support processor. The interface to the support microprocessor is through a multiplexed address/data bus as found in the Intel 8085 family of processors. This, however, can easily be adapted to other processors with nonmultiplexed address and data buses or with 16-bit buses using minimal external logic.

Figure 11 shows the basic interface between an 8085 and AIC-610F.

The support processor is used to maintain "loose" synchronization with the disk through Branch Control and Sequencer Status registers (Register 78_H and 79_H). Based on the operation, the processor also sets-up registers to control the data transfer to and from the buffer.

PROCESSOR HOST BUS ACCESS (REGISTERS 50_H AND 51_H): Although the AIC-610F can handle data transfers between the host bus and buffer, sometimes it may be necessary to handle data transfers between the support processor and host bus. Such is the case in a SCSI implementation during the transfer of the Command Descriptor Block (CDB) and message bytes. This can be accomplished by the support processor through an access to Register 50_H or 51_H.

An access to either register is decoded internally by the AIC-610F. The AIC-610F internally bridges the support processor data bus and the RAM buffer data bus, thus offering a data path to host latch and receivers. The chip also asserts the control signals to access the host bus.

During a read operation, $\overline{BIE}$ is asserted. During a write, LO line will be first asserted, followed by the $\overline{BOE}$ line. This allows the data to be latched before being enabled onto the host bus. See Figure 12 illustrating the data path.

RAM BUFFER ACCESS (REGISTER 70_H): During the process of transferring data between the disk and the host, the support processor has to sometimes have access to the sector data. This is especially necessary during a read operation if a correctable ECC error is encountered. Using the error syndrome information in the AIC-610F, the support processor calculates the error mask and displacement.

After that, the actual data stored in the buffer has to be read, modified, and written back. This is done through an access to Register 70_H.

Here, again, the AIC-610F decodes an access to Register 70_H and bridges the processor data bus and the buffer data bus. This allows the transfer of data between the support processor and RAM.

The AIC-610F decodes an access to Register 70_H and asserts memory select and read/write to the RAM. The address selected is the contents of the WAP registers if the ROP/$\overline{WOP}$ bit (Register 53_H, Bit 4) is set (Read Disk) and the contents of the RAP registers if the ROP/$\overline{WOP}$ bit is reset (Write Disk).

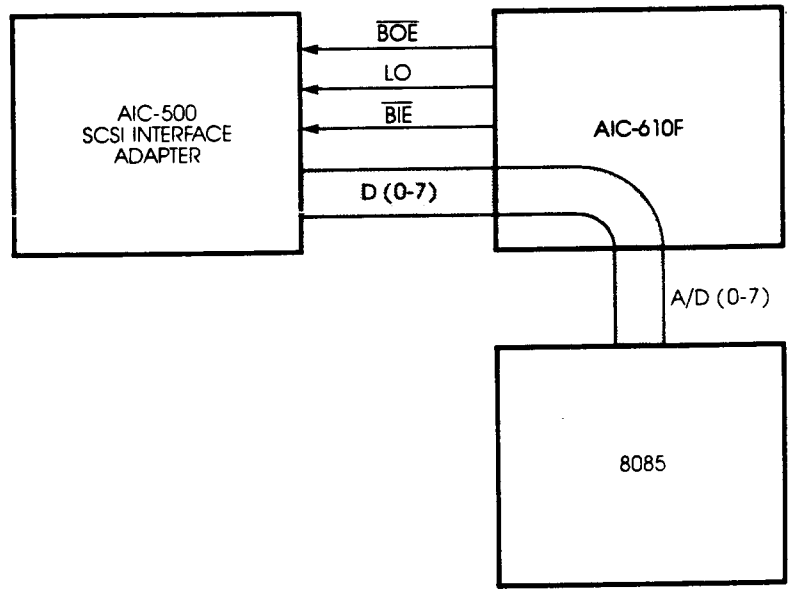


FIGURE 12. DATA PATH BETWEEN THE PROCESSOR AND THE HOST

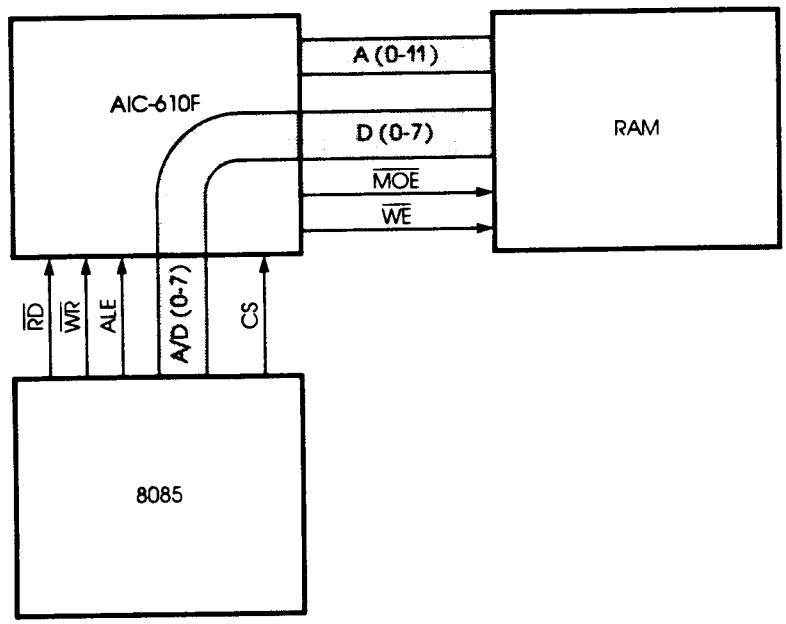


FIGURE 13. RAM BUFFER ACCESS

RAM Buffer Interface

The AIC-610F is capable of managing up to 64K byte buffer. The chip provides the address, memory select and read/write lines necessary to perform this function. There are two distinct modes of RAM interface: equal to or less than 16K bytes (up to 14 address lines), and greater than 16K bytes up to 64K bytes (up to 16 address lines).

The figure below illustrates a simple 14-bit addressing design. Note: $\overline{CLKA}$ and $\overline{CLKB}$ outputs are not used when the AIC-610F is used in buffer management applications with 16K or less buffer addressing.

In the 16-bit addressing mode, the higher-order address lines (A12-A15) and the lower-order address lines (A4-A7) are multiplexed on pins A4 to A7. Two external TRI-STATE registers are required for the higher-order address lines (A12-A15). These registers hold a copy of Bits 12-15 of the internal counters registers $5B_H$ and $5D_H$. These registers are updated on the CLK cycle following the increment of the internal counters ($5B_H$ or $5D_H$), if a Port A cycle is not required or on the CLK cycle following a write of $5B_H$ or $5D_H$ by the microprocessor.

A word of caution: The microprocessor update of these external registers is not prioritized and should only be done when Port A and Port B operations have quiesed. The AIC-610F updates the external registers by emitting the appropriate A12 through A15 on address lines A4 through A7 and then pulses SHP or SDP appropriately. In normal operation, these updates will occur after every 4K bytes transferred by either Port. When slow microprocessors are used, double pulsing of SHP and SDP will occur because of the internal synchronization circuit. However, the address for both pulses is valid.

The $\overline{CLKA}$ cycle determines the access time requirement for the buffer RAMs, along with the address valid time for the AIC-610F. For a 400 ns clock period, 150 ns access-time RAM is needed for proper operation.

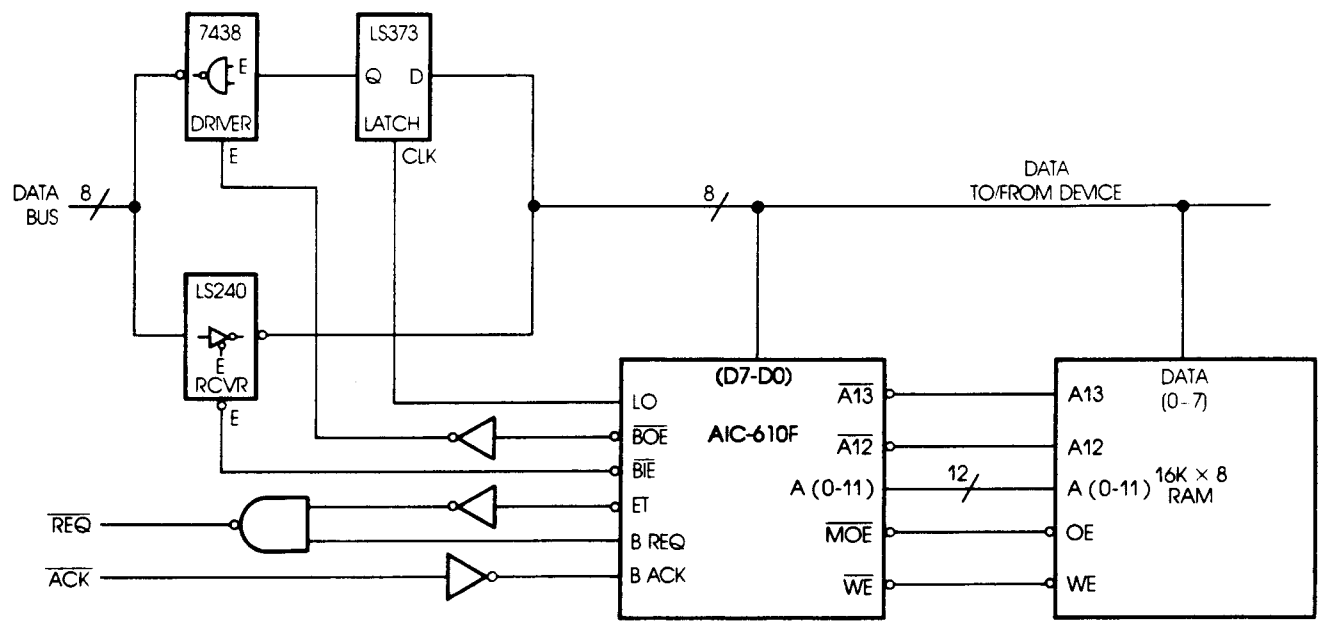


FIGURE 14. 14-BIT ADDRESSING APPLICATION EXAMPLE

For proper interleaved Port A-Port B operation, the maximum rate at which Port A requests should occur is once for every two CLK cycles. This will allow the REQ/ACK cycles to occur at one-half the CLK rate. Note that Port A cycles have a priority over Port B and high-order register updates. With a very fast DMA interface and a CLK period of 333 ns and 120 ns access buffer RAMs, a 1.5 Mbyte transfer rate is possible in this mode.

An example of a 16-bit addressing mode is shown in Figure 15. There are two programming requirements for this mode that are not obvious:

1. Since the LS173s do not have a reset pin, a reset of the pointers using Register 59_H is not adequate. A microprocessor load of 00_H to the higher-order pointer registers ($5B_H$ and $5D_H$) is also required.

2. Any time the higher-order pointer registers are set up, they have to be set up twice: once to set the internal register, and once to update the external LS173s.

Internal to the AIC-610F are two sets of pointer registers used to generate the RAM addresses. These are the Read Address Pointer (RAP) and the Write Address Pointer (WAP). The actual register used depends on which port is selected and the direction of the data transfer. This is controlled by the value of the ROP/$\overline{\text{WOP}}$ (Read Operation/$\overline{\text{Write Operation}}$) bit in the DMA control register (Register 53_H, Bit 4). In addition to this, another pair of registers called the Stop Pointer (SP) is used to control data transfers between the host and the buffer. The different possibilities are shown in Table 4.

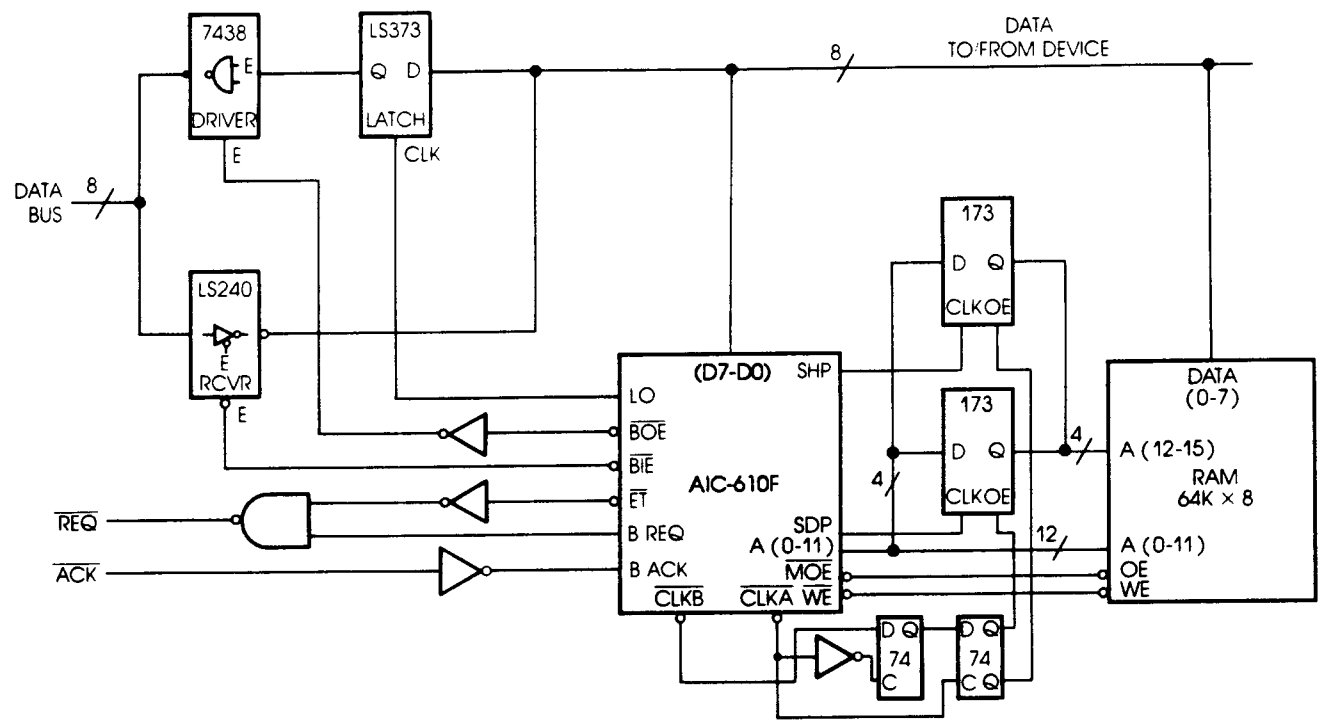


FIGURE 15. 16-BIT ADDRESSING APPLICATION EXAMPLE

TABLE 4. BUFFER ADDRESS GENERATION

ROP/$\overline{WOP}$	SOURCE FOR ADDRESS	DIRECTION OF TRANSFER
Read Disk		
1	WAP	Disk to Buffer
1	RAP	Buffer to Host (RAP ≠ SP)
Write Disk		
0	RAP	Buffer to Disk
0	WAP	Host to Buffer (WAP ≠ SP)

Peripheral Interface

The AIC-610F handles all the data formatting and sequencing necessary to interface with the peripheral. Consider the interface to a drive:

During a write operation, parallel data is transferred into the chip from the RAM buffer. This data is passed through a 48-bit shift register before it is output as serial data in the NRZ format. During this time, a four or six-byte ECC is also computed. After the entire sector is written to the drive, the ECC bytes are also output in a serial fashion.

During a read operation, NRZ data is read into the controller chip and byte parallel data is transferred to the RAM buffer. Error checking is also performed at this time, using the ECC bytes stored at the end of the sector. If there is an error, the support processor usually initiates retries to determine if the error is correctable. If the error is determined to be correctable, the error syndrome information found in the AIC-610F chip is used to correct the data in the buffer. This process is transparent to the host.

The serial data is synchronized through a read reference clock that is input to the AIC-610F chip. The disk controller chip has the necessary logic to look for index mark and sector mark (on hard-sectored drives), in addition to generating the Read Gate and Write Gate signals. The disk controller also controls the writing of the address mark during a write operation and looks at when the address mark is detected during a read operation. Figure 16 shows the necessary drive interface lines.

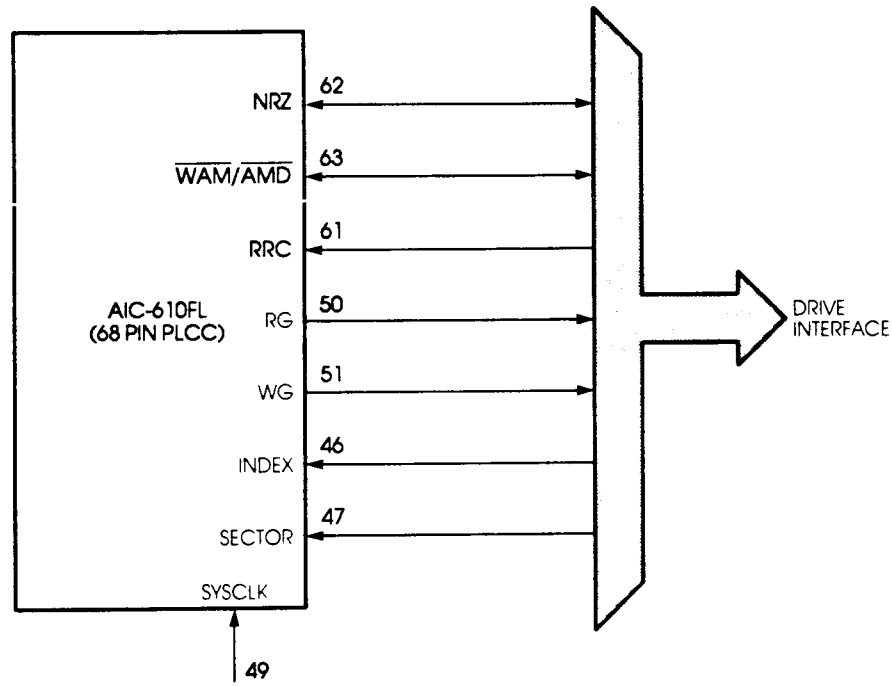


FIGURE 16. AIC-610F DRIVE INTERFACE SIGNALS

System/Host Bus Interface

In the design of intelligent controllers, SCSI (Small Computers Systems Interface) is a popular host interface. The Adaptec AIC-500 SCSI Interface Adapter chip provides the necessary drivers and receivers, while the AIC-610F provides the major functionality to ease implementation of a SCSI based controller.

The AIC-610F has several interface lines dedicated to serve the host/system bus. The $\overline{BIE}$ line clocks data from the system bus into external latches. The LO signal latches the data from the buffer into external latches. This data can then be enabled to the system bus by $\overline{BOE}$.

Particularly in SCSI applications, the AIC-610F must request arbitration for the system bus through an arbitration sequence. The AIC-610F is set up to provide the necessary arbitration logic, which may be used for other applications to arbitrate for the system bus.

ARBITRATION LOGIC: The arbitration logic is responsible for providing a bus arbitration between the IPSC and the system bus. The circuit in the AIC-610F has been designed to allow rapid arbitration in two wire arbitration systems. Setting Register 52_H, Bit 0, sets a request for arbitration.

The circuit monitors the Busy In and Sel In pins, waiting for both to be deactivated. After a minimum of one clock cycle time of a bus free condition, the AIC-610F will begin arbitration support. The Bsy Out signal will be activated until either Sel Out, Sel In, or Bsy In is active, or Bsy Out is reset by the microprocessor.

For SCSI applications, the external circuits required to support the arbitration logic are provided by the AIC-500.

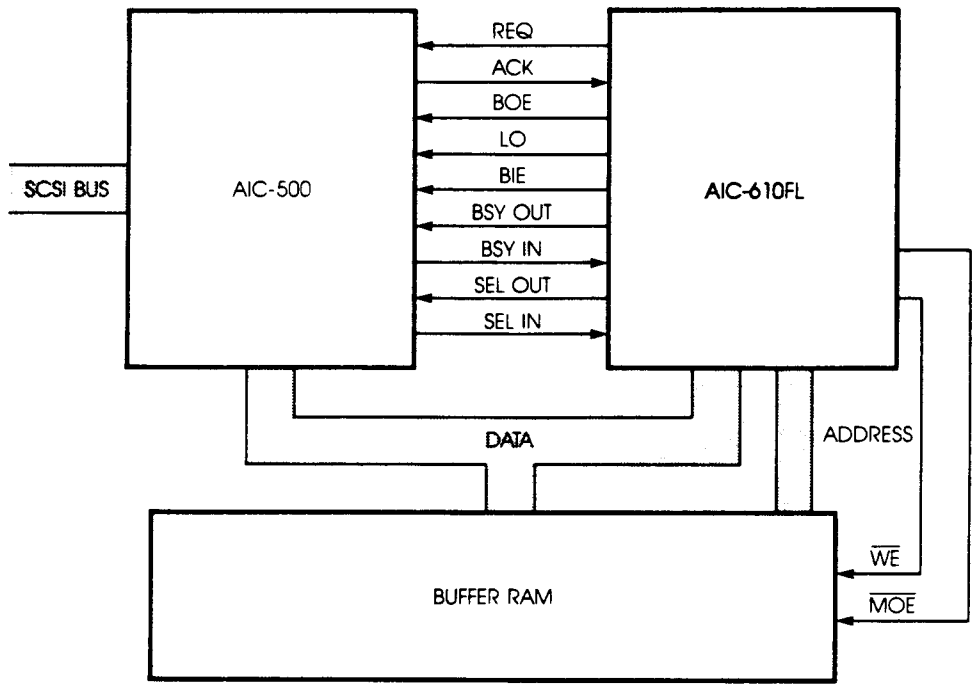


FIGURE 17. INTERFACING THE AIC-610F TO THE AIC-500

AIC-610F SOFTWARE OVERVIEW

Programming Example

The operation of the AIC-610F is based on the following sequence of events:

1. Initialization of the AIC-610F. This is done once after every power-on reset and is not required thereafter.

2. The software of the AIC-610F can now be divided into two separate functions. Each of these processes must be running concurrently to ensure proper operation. The first can be referred to as Serial Data Transfer operation. This function is controlled by the sequencer map and the status and branch registers, and ensures the correct sequence of events and transfer of data to and from the disk device. The second process can be referred as the Buffer Data Transfer operation, and is the data transfer to and from the RAM buffer.

SERIAL DATA TRANSFER

- Set up the sequencer RAM, loading appropriate values.
- Set up Registers 70_H-$7F_H$ to handle control of data.
- Start sequencing the chip at the appropriate RAM location by loading Register 79_H with the starting address.
- Monitor the status using Register 79_H and monitor the branching, when appropriate, using the branch register (Register 78_H).

In the section of Serial Data Transfer, each of these steps is explained using register set up requirements and then demonstrated with flowcharts for read, write, and format operations.

BUFFER DATA TRANSFER

- Set up Registers 52_H-$5F_H$ for a read or write operation.

In the section of Buffer Data Transfer, register set up is explained through the use of flowcharts for buffer read and write operations.

Initialization

Upon a hardware reset, the following software initialization is necessary to ensure the AIC-610F's proper operation:

1. Set Bit 5 in Register 71_H.
2. Reset Bit 5 in Register 71_H.
3. Set Bit 0 in Register 59_H.
4. Reset Bit 0 in Register 59_H.
5. Load buffer size into Register 54_H.

Serial Data Transfer Programming Example

On initialization, the following registers have to be set up as shown in Table 5a.

Table 8 shows the sequencer RAM of the AIC-610F programmed as a ST412/506 controller. The contents of RAM are loaded by the support processor after power up. In addition, the ECC polynomial has to be set up. The polynomial used is as follows:

Forward:

$x^{30}+x^{24}+x^{18}+x^{14}+x^{8}+x^{7}+x^{2}+1$

Reverse:

$x^{30}+x^{25}+x^{24}+x^{18}+x^{14}+x^{8}+x^{2}+1$

The forward polynomial is set up during power-on initialization. The reverse polynomial is only used during error correction time. After this, the forward polynomial must be restored.

The given polynomial is loaded in the registers as shown in Table 5b.

TABLE 5a. REGISTER SET UP ON INITIALIZATION

REGISTER	DESCRIPTION	FUNCTION
71_H	Error Control	Define polynomial size and type.
72_H - 77_H	ECC Polynomial	Define ECC polynomial.
$7B_H$	WAM Control	Define WAM signal timing.
$7C_H$	Sync Control	Define sync byte to be used for compare during sync detect.
$7D_H$	GP I/O Control	Define use of GP I/O pins.
$7E_H$	GP I/O Data	Set up or monitor external control or status.
$7F_H$	Clock Control	Define $\overline{CLKA}$–SYSCLK relationship. Define $\overline{CLKA}$–RRC relationship. Define start bit of NRZ shift register compare.
80_H-FF_H	Sequencer RAM	Set up sequencer RAM.

TABLE 5b. REGISTER SET UP FOR A 32-BIT ECC POLYNOMIAL

REGISTER	VALUE	DESCRIPTION
71_H	80_H	32-bit ECC enabled.
72_H	FF_H	Forward/reverse polynomial.
73_H	FF_H	Forward/reverse polynomial.
74_H	$C2_H$	Forward polynomial.
	82_H	Reverse polynomial.
75_H	20_H	Forward/reverse polynomial.
76_H	82_H	Forward/reverse polynomial.
77_H	20_H	Forward polynomial.
	21_H	Reverse polynomial.
$7B_H$	10_H	WAM timing control.
$7C_H$	$A1_H$	Sync byte.
$7D_H$	FF_H	Set up for address decode.
$7E_H$	00_H	Don't care.
$7F_H$	87_H	Clock and sync control.

NOTE: For proper 32-bit ECC operation, Registers 72_H and 73_H should be loaded with FF_H.

The values to be set up in the controller for 48-bit ECC are shown in Table 5c.

Information on the 48-bit ECC polynomial is available under a license agreement with Adaptec, Inc.

At this point, there are four fundamental Serial Data Transfer operations that should be looked at. These are as follows:

- Soft Sector Format
- Soft Sector Read/Write
- Hard Sector Format
- Hard Sector Read/Write

Tables 6 and 7 show the sequencer maps corresponding to the soft-sector flowcharts for MFM and 2,7 RLL encoding schemes. Table 8 shows the sequencer map corresponding to the hard-sector flowcharts for a ST412/506 controller.

Three separate sequencer maps are given to illustrate various techniques that can be used to program the AIC-610F. The designer should select the sequencer map closest to the required application and modify it to meet the specific design requirements.

TABLE 5c. REGISTER SET UP FOR A 48-BIT ECC POLYNOMIAL

REGISTER	VALUE	DESCRIPTION
71	00	48-bit ECC enabled.
72	XX	Forward/reverse polynomial.
73	XX	Forward/reverse polynomial.
74	XX	Forward/reverse polynomial.
75	XX	Forward/reverse polynomial.
76	XX	Forward/reverse polynomial.
77	XX	Forward/reverse polynomial.
7B	40	WAM timing control.
7C	A1	Sync byte.
7D	FF	Set up for address decode.
7E	00	Don't care.
7F	87	Clock and sync control.

TABLE 6. SEQUENCE MEMORY BIT MAP FOR MFM VERSION

	E0 THRU F7 DATA		C0 THRU D7 COUNT		A0 THRU B7 CONTROL		80 THRU 97 NEXT ADRS.		ADRS. R79	BRANCH R78	COMMENTS
0	CYLINDER				1	2	0	1	0		CYLINDER
1	HEAD				1	2	0	2	1		HEAD
2	SECTOR				1	2	0	3	2		SECTOR
3	FLAG				1	0	0	C	3		FLAG
4	6	C	F	F	0	5	0	D	4		DATA FIELD 6C = FILL CHARACTER
5			0	3	C	0	0	F	5		START FOR R/W DATA & WG ↓
6	A	1	8	0	0	2	0	7	6		ID AM
7	F	E			0	2			7		ID SYNC
8			0	1			1	7	8		PAD ID TO DATA RG ↑
9			2	B	8	0	0	A	9		WG ↑ VFO LOCK ON FOR DATA
A	A	1	A	0	0	2	0	B	A		DATA AM & SEBCA
B	F	8			0	2	0	4	B		DATA SYNC & SEBCA
C			4	3			R=68 W=69	F=11	C		ID ECC BRANCH TO STOP ON ERROR
D			4	3			R=74 W=14	F=14	D		DATA ECC BRANCH TO STOP ON ERROR
E			0	A	8	0	0	6	E		WG ↑ VFO LOCK ON FOR ID
F			0	1	4	0	0	6	F		RG ↑ FOR ID
10					8	0	0	E	10		WG ↓ FOR 27 VFO LOCK ON
11			0	5	C	0	0	9	11		PAD DATA FIELD TO GAP 3 (FORMAT)
12	F	8			0	2	0	4	12		DATA FIELD IDENTIFIER
13	4	E					5	3	13		WAIT FOR INDEX & STOP
14			0	1			R=05 W=05	F=16 FLAST=13	14		POST DATA FIELD
15					R=48 W=48	F=00	R=0F W=0F	F=D5	15	16	WAIT FOR INDEX START FORMAT
16	4	E	0	A	8	0	1	0	16		WG ↑ GAP 1 AFTER INDEX & GAP 3
17			0	1	4	0	0	A	17		RG ↑ FOR DATA

7 6 5 4 3 2 1 0 — DATA

7 6 5 4 3 2 1 0 — DATA TYPE: AM, ECC, SEBCA — FIELD CNT (-1)

7 6 5 4 3 2 1 0 — RG & WG CTL: 00 = NO OP, 01 = SET RG, 10 = SET WG, 11 = RESET WG — STACK ENABL, INVALID NRZ, OUTPUT, COMPARE EN, DATA XFER: 0 = ECC, 1 = CRC

7 6 5 4 3 2 1 0 — N.A. — 1F = STOP

BRANCH CONTROL

RG • ECC = 1	RG • ECC = 0
000 = NO BRANCH	000 = NO BRANCH
001 = ECC ERROR STOP	001 = STOP ON INPUT
010 = NO COMPARE STOP	010 = STOP ON INDEX OR SECTOR
011 = ERROR OR NO COMPARE STOP	011 = STOP ON NO COMPARE
100 = GOOD ECC AND COMPARE	100 = BRANCH ON CARRY
101 = ECC ERROR	101 = BRANCH ON INPUT
110 = NO COMPARE	110 = BRANCH ON INDEX OR SECTOR
111 = ERROR OR NO COMPARE	111 = BRANCH ON NO COMPARE

NOTES:
1. ALL SEQUENCER RAM ADDRESSES AND DATA VALUES ARE IN HEX.
2. BLANKS = 00_H.
3. RD & WR DATA DONT USE THE BRANCH REGISTER.

TABLE 7. SEQUENCE MEMORY BIT MAP FOR 2,7 RLL VERSION

	E0 THRU F7 DATA		C0 THRU D7 COUNT		A0 THRU B7 CONTROL		80 THRU 97 NEXT ADRS.		ADRS. R79	BRANCH R78	COMMENTS
0	CYLINDER				1	2	0	1	0		CYLINDER
1	HEAD				1	2	0	2	1		HEAD
2	SECTOR				1	2	0	3	2		SECTOR
3	FLAG				1	0	0	C	3		FLAG
4	6	C	F	F	0	5	0	D	4		DATA FIELD 6C = FILL CHARACTER
5			0	3	C	0	0	F	5		START FOR R/W DATA & WG ↓
6	5	E	8	0	0	2	0	7	6		ID AM
7	A	1	8	0	0	2			7		ID SYNC
8			0	1			1	7	8		PAD ID TO DATA RG ↑
9			2	F	8	0	0	A	9		WG ↑ VFO LOCK ON FOR DATA
A	5	E	A	0	0	2	0	B	A		DATA AM & SEBCA
B	A	0	A	0	0	2	1	2	B		DATA SYNC & SEBCA
C			4	3			R=68 W=69	F=11	C		ID ECC BRANCH TO STOP ON ERROR
D			4	3			R=74 W=14	F=14	D		DATA ECC BRANCH TO STOP ON ERROR
E			0	F	8	0	0	6	E		WG ↑ VFO LOCK ON FOR ID
F			0	1	4	0	0	7	F		RG ↑ FOR ID
10	A	A			C	0	0	E	10		WG ↓ FOR 27 VFO LOCK ON
11			0	5	C	0	0	9	11		PAD DATA FIELD TO GAP 3 (FORMAT)
12	F	8			0	2	0	4	12		DATA FIELD IDENTIFIER
13	A	A					5	3	13		WAIT FOR INDEX & STOP
14			0	1			R=05 W=05	F=16 FLAST=13	14		POST DATA FIELD
15					R=48 W=48	F=00	R=0F W=0F	F=D5	15	16	WAIT FOR INDEX START FORMAT
16	A	A	0	A	8	0	1	0	16		WG ↑ GAP 1 AFTER INDEX & GAP 3
17			0	1	4	0	0	B	17		RG ↑ FOR DATA
	7 6 5 4 3 2 1 0		7 6 5 4 3 2 1 0		7 6 5 4 3 2 1 0		7 6 5 4 3 2 1 0				
	DATA		DATA TYPE: AM, ECC, SEBCA	FIELD CNT (-1)	RG & WG CTL	STACK ENABL, INVALID NRZ, OUTPUT, COMPARE EN, DATA XFER	N.A.				1F = STOP

00 = NO OP
01 = SET RG
10 = SET WG
11 = RESET WG

0 = ECC
1 = CRC

BRANCH CONTROL

RG • ECC = 1	RG • ECC = 0
000 = NO BRANCH	000 = NO BRANCH
001 = ECC ERROR STOP	001 = STOP ON INPUT
010 = NO COMPARE STOP	010 = STOP ON INDEX OR SECTOR
011 = ERROR OR NO COMPARE STOP	011 = STOP ON NO COMPARE
100 = GOOD ECC AND COMPARE	100 = BRANCH ON CARRY
101 = ECC ERROR	101 = BRANCH ON INPUT
110 = NO COMPARE	110 = BRANCH ON INDEX OR SECTOR
111 = ERROR OR NO COMPARE	111 = BRANCH ON NO COMPARE

NOTES:
1. ALL SEQUENCER RAM ADDRESSES AND DATA VALUES ARE IN HEX.
2. BLANKS = 00_H.
3. RD & WR DATA DON'T USE THE BRANCH REGISTER.

TABLE 8. SEQUENCE MEMORY BIT MAP FOR ST412/506 CONTROLLER

	E0 THRU F7 DATA		C0 THRU D7 COUNT		A0 THRU B7 CONTROL		80 THRU 97 NEXT ADRS.		ADRS. R79	BRANCH R78	COMMENTS
0	CYLINDER				1	2	0	1	0		CYLINDER
1	HEAD				1	2	0	2	1		HEAD
2	SECTOR				1	2	0	3	2		SECTOR
3	FLAG				1	0	0	C	3		FLAG
4	6	C	n	n	0 0	5 7	0	D	4		DATA FIELD
5			0	3	C	0	0	F	5		RD ID
6	A	1	8	0	0	2	0	7	6		ID AM
7	F	E			0	2	0	0	7		ID SYNC
8							1	7	8		READ DATA DELAY
9			0	8	8	0	0	A	9		WG ↑ FOR DATA
A	A	1	A	0	0	2	0	B	A		DATA AM
B	F	8			0	2	0	4	B		DATA SYNC
C			4	3			1001 9	4	C	11	ID ECC BRANCH ON GOOD ID
D			4	3			9	4	D		DATA ECC BRANCH ON CARRY
E			0	B	8	0	0	6	E		WG ↑ FORMAT ID
F			0	1	4	0	0	6	F		RG ↑ FOR ID
10	4	E	00 - 1F		8	0	0	E	10		WG ↑ GAP 1 & 4
11			0	5			0	9	11		FORMAT DATA FIELD
12			0	1			1	0	12		END DATA FIELD FOR FORMAT
13			0	1			1	6	13		END DATA FIELD FOR LAST SECTOR
14			0	3			1	F	14		END DATA FIELD FOR RD/WR
15					R=48 W=48	F=00	R=0F W=0F	F=D5	15	10	BRANCH ON INDEX ↑
16	4	E					5	6	16		STOP ON INDEX ↑
17			0	1	4	0	0	A	17		RG ↑ FOR DATA

7 6 5 4 3 2 1 0 — DATA

7 6 5 4 3 2 1 0 — DATA TYPE (AM, ECC, SEBCA) / FIELD CNT (-1)

7 6 5 4 3 2 1 0 — RG & WG CTL / STACK ENABL, INVALID NRZ, OUTPUT COMPARE EN, DATA XFER

7 6 5 4 3 2 1 0 — N.A. / 1F = STOP

00 = NO OP
01 = SET RG
10 = SET WG
11 = RESET WG

0 = ECC
1 = CRC

BRANCH CONTROL

RG • ECC = 1	RG • ECC = 0
000 = NO BRANCH	000 = NO BRANCH
001 = ECC ERROR STOP	001 = STOP ON INPUT
010 = NO COMPARE STOP	010 = STOP ON INDEX OR SECTOR
011 = ERROR OR NO COMPARE STOP	011 = STOP ON NO COMPARE
100 = GOOD ECC AND COMPARE	100 = BRANCH ON CARRY
101 = ECC ERROR	101 = BRANCH ON INPUT
110 = NO COMPARE	110 = BRANCH ON INDEX OR SECTOR
111 = ERROR OR NO COMPARE	111 = BRANCH ON NO COMPARE

NOTES:
1. ALL SEQUENCER RAM ADDRESSES AND DATA VALUES ARE IN HEX.
2. BLANKS = 00_H.
3. RD & WR DATA DON'T USE THE BRANCH REGISTER.

SOFT-SECTOR FORMAT: The given routine for soft-sector formatting is for two of the most common encoding schemes of MFM and 2,7 RLL. This format yields 32 sectors per track, and the sector size is 256 bytes. However, any sector size may be employed (please also refer to the extended data handling section).

Note: The sequencer map must be loaded for either MFM or RLL prior to executing the format track command sequence.

The format track command sequence is as follows:

1. Set Registers $E0_H$, $E1_H$, $E2_H$ and $E3_H$ with the first sector ID (cylinder, head, sector and flag respectively).

2. Set Register $C4_H$ to FF_H for 256 byte sector.

3. Load the following next address registers to structure the sequencer map for formatting:
   - Load Register $8C_H$ with 11_H.
   - Load Register $8D_H$ with 14_H.
   - Load Register 94_H with 16_H.

4. Set Register 78_H (Branch Register) with 16_H. This will format Gap 1 after index.

5. Set Bit 5 in Register $7A_H$ for a $6C_H$ data pattern. Otherwise, contents of the sector buffer will be used during a write to the data field.

6. Read operational control register and test if index bit (Register $7A_H$, Bit 0) is reset.

7. Set Register 79_H with 15_H. This will start the format operation. The AIC-610F waits for index after which a Gap 1 will be written.

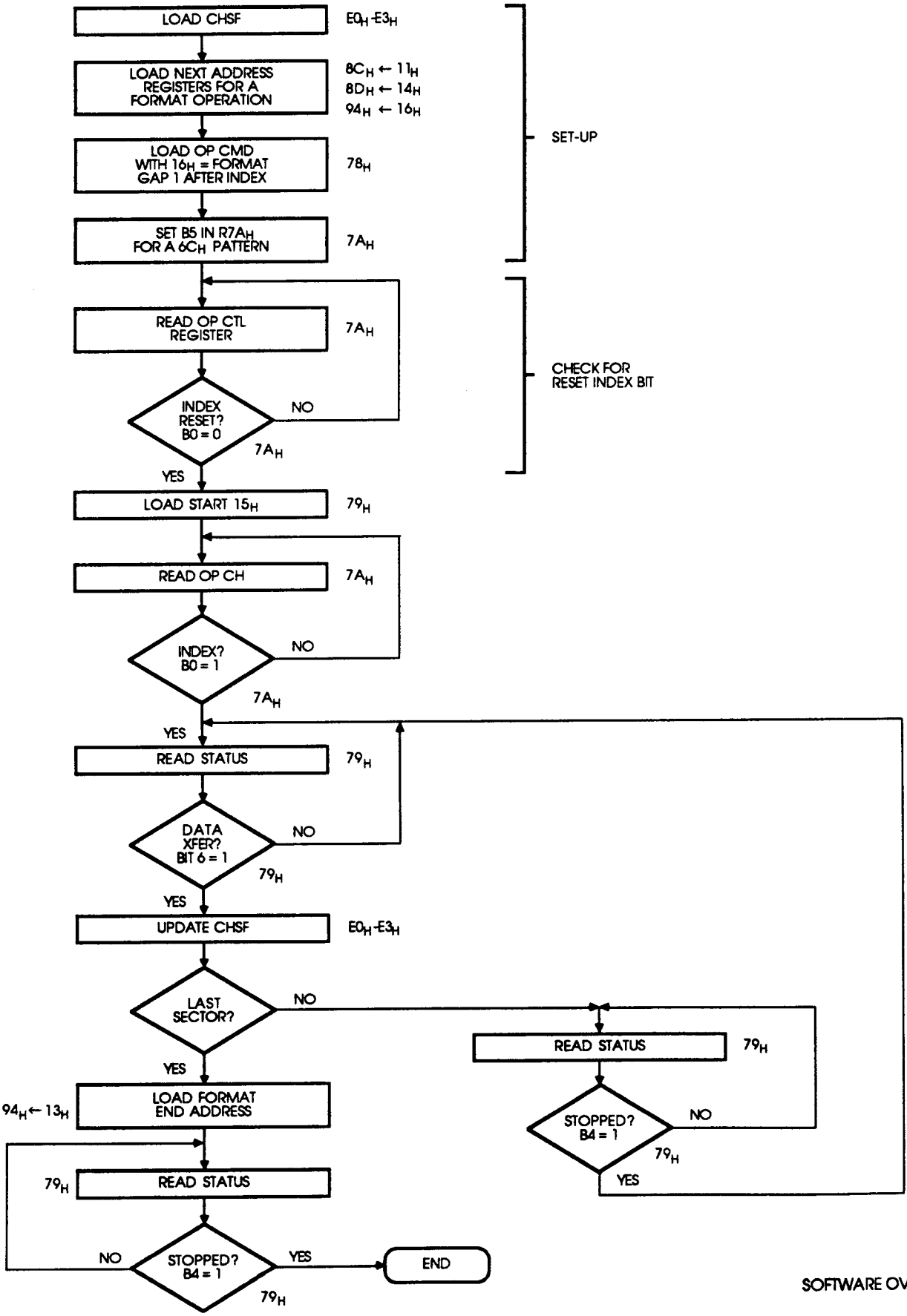
8. Read operational control register and wait till index past bit (Register $7A_H$, Bit 0) is set. The index bit being set means that the data field is being written.

9. Read status register (Register79_H) and wait until the data transfer bit is set (Register 79_H, Bit 6).

10. For a multiblock transfer, update Registers $E0_H$-$E3_H$ with the next ID field to be written.

11. Check if this is the last sector. If it isn't, read Register 79_H and wait until data transfer is complete and repeat Steps 9,10, and 11 until the number of sectors required per track are formatted. In this case repeat 31 more times (for 32 sectors/track).

12. Otherwise, load the next address at Register 94_H as 13_H. Read status register and test if stopped condition bit (Register 79_H, Bit 4) is set, indicating track format complete.


LOAD CHSF
E0H-E3H
LOAD NEXT ADDRESS REGISTERS FOR A FORMAT OPERATION
8CH ← 11H
8DH ← 14H
94H ← 16H
LOAD OP CMD WITH 16H = FORMAT GAP 1 AFTER INDEX
78H
SET B5 IN R7AH FOR A 6CH PATTERN
7AH
SET-UP
READ OP CTL REGISTER
7AH
INDEX RESET? B0 = 0
NO
7AH
YES
CHECK FOR RESET INDEX BIT
LOAD START 15H
79H
READ OP CH
7AH
INDEX? B0 = 1
NO
7AH
YES
READ STATUS
79H
DATA XFER? BIT 6 = 1
NO
79H
YES
UPDATE CHSF
E0H-E3H
LAST SECTOR?
NO
READ STATUS
79H
STOPPED? B4 = 1
NO
79H
YES
YES
LOAD FORMAT END ADDRESS
94H ← 13H
79H
READ STATUS
NO
STOPPED? B4 = 1
YES
79H
END

SOFT-SECTOR READ/WRITE: The following algorithm is for a soft-sector read or write for either MFM or RLL. In order to read or write data, the heads have to be positioned over the appropriate cylinder, and the relevant head must be selected. The following steps assume that the correct track has been reached.

1. Set Registers $E0_H$ through $E2_H$ with the required sector ID.

2. Set Register $C4_H$ to FF_H for a 256-byte sector.

3. Set the following next address registers to the appropriate addresses:

For Read ID and Read Data:
- Load Register $8C_H$ with 68_H.
- Load Register $8D_H$ with 74_H.
- Load Register 94_H with 05_H.

For Read ID and Write Data:
- Load Register $8C_H$ with 69_H.
- Load Register $8D_H$ with 14_H.
- Load Register 94_H with 05_H.

4. Set start address register (Register 79_H) with 05_H. This starts the read or write sequence by turning on Read Gate and enabling the VFO to look for an address mark.

5. Read the status register (Register 79_H). Test if Bit 4 or Bit 6 is set.

6. If no, check the condition of Register 78_H. If the value read is 07_H, start a short timeout as the AIC-610F is searching for a Data Address Mark. A timeout indicates no Data Address Mark was found and the sequencer can be stopped by storing $1F_H$ in Register 79_H. If the value is not 07_H, this indicates ID Address Mark not found. Start a timeout and return to Step 5. If yes, test Bit 4 of Register 79_H. If set, the sequencer is stopped on an ID error or miscompare.

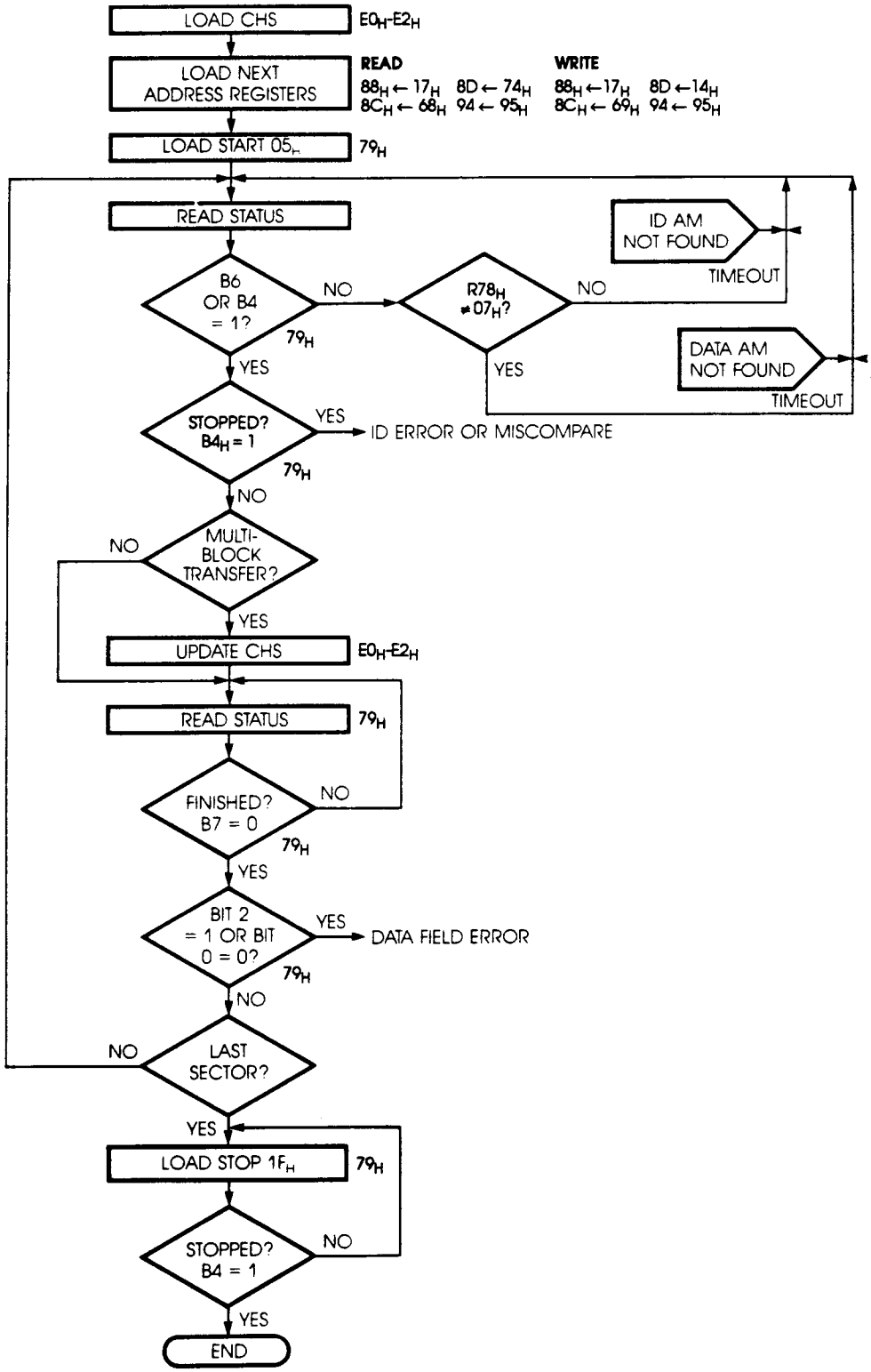
7. If Register 79_H, Bit 6, was set, it indicates read data is now being transferred to the sector buffer, or write data from the buffer. If this is a multiblock transfer, update Registers $E0_H$-$E2_H$ with the next sector ID while the data is being transferred.

8. Read status Register 79_H and wait for Bit 7 to be reset, indicating that the ECC bytes have been written or read.

9. Test for ECC error or error from the compare bit. If either is set, a data field error requiring a user-specified correction process is implied.

10. Check if it was the last sector. If it isn't the last sector, repeat the sequence from Step 5.

11. If it is the last sector, load Register 79_H with $1F_H$ until the stop bit (Register 79_H, Bit 4) is set, indicating end of read/write sequence.

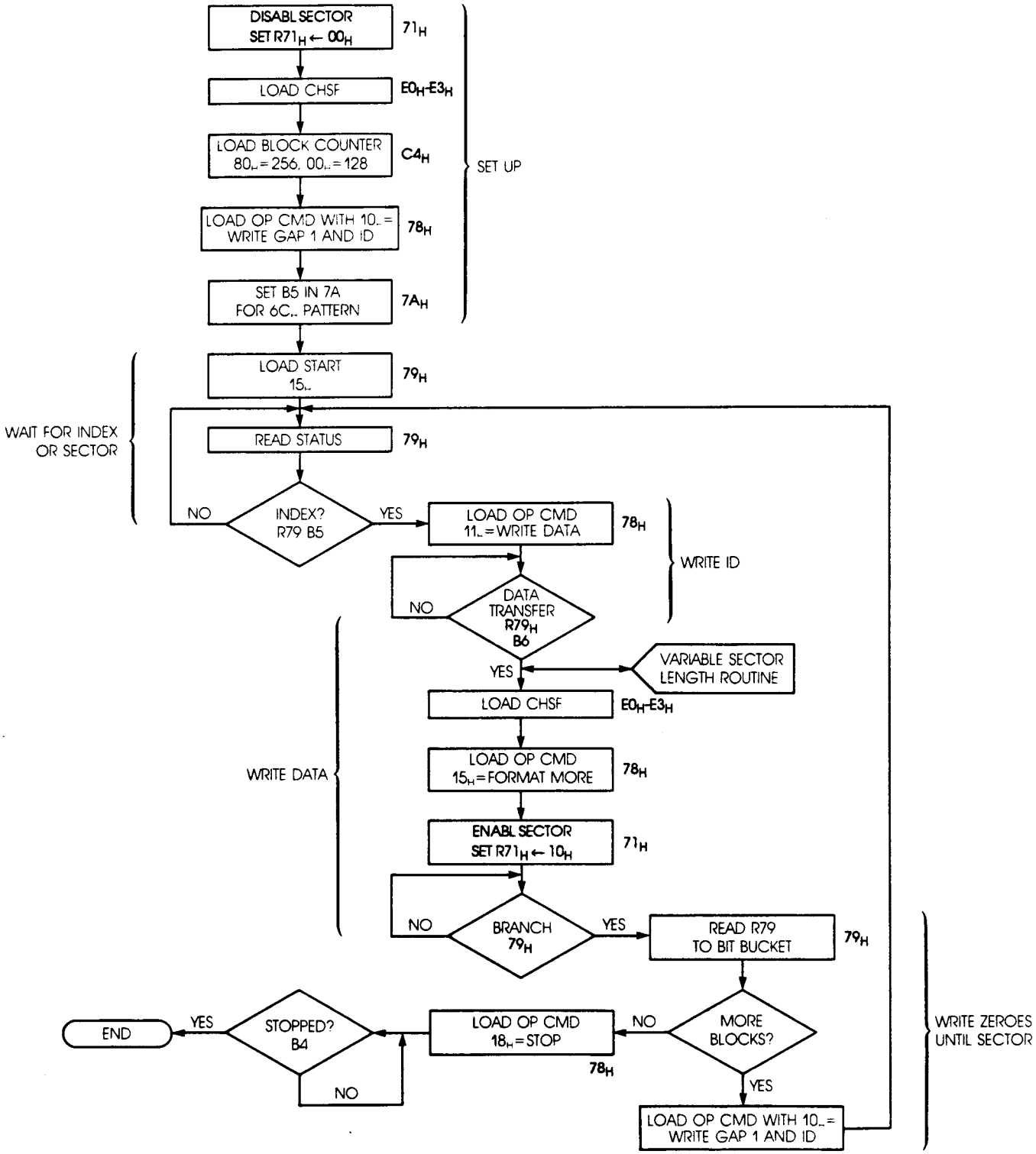

LOAD CHS
E0H-E2H
LOAD NEXT ADDRESS REGISTERS
READ
88H ← 17H 8D ← 74H
8CH ← 68H 94 ← 95H
WRITE
88H ← 17H 8D ← 14H
8CH ← 69H 94 ← 95H
LOAD START 05H
79H
READ STATUS
ID AM NOT FOUND
TIMEOUT
B6 OR B4 = 1?
79H
NO
R78H ≠ 07H?
NO
YES
YES
DATA AM NOT FOUND
TIMEOUT
STOPPED? B4H = 1
79H
YES
ID ERROR OR MISCOMPARE
NO
NO
MULTI-BLOCK TRANSFER?
YES
UPDATE CHS
E0H-E2H
READ STATUS
79H
FINISHED? B7 = 0
79H
NO
YES
BIT 2 = 1 OR BIT 0 = 0?
79H
YES
DATA FIELD ERROR
NO
NO
LAST SECTOR?
YES
LOAD STOP 1FH
79H
STOPPED? B4 = 1
NO
YES
END

HARD-SECTOR FORMAT: As mentioned earlier, the AIC-610F Programmable Storage Controller chip is capable of supporting hard-sector drives. Hard-sector drives differ from soft-sector drives in that, between every adjacent sector on a track, there is a sector mark, and this is used to identify the beginning of a sector.

A hard-sector format operation is performed as follows:

1. Disable sector mark by setting Register 71_H to 00_H. Thus, the controller chip will wait for the index mark before writing out Gap 1.
2. Load Registers $E0_H$-$E3_H$ with the sector ID, (cylinder, head, sector and flag).
3. Load Register $C4_H$ with the sector count.
4. Load Register 78_H with 10_H. This will cause Gap 1 to be written after index.
5. Set Bit 5 in Register $7A_H$ for $6C_H$ data pattern. Otherwise, sector buffer is used.
6. Load Register 79_H with 15_H to start formatting.
7. Read status from Register 79_H. If Bit 5 is set, then Gap 1 is being written. After this, the ID is written.
8. Load Register 78_H with 11_H. This tells the controller to write the data next.
9. Read status from Register 79_H. If Bit 6 is set, then data is being transferred.
10. Update Registers $E0_H$-$E3_H$ with the next sector ID. This has to be done before even checking if there are more blocks since, on a hard-sector drive, the timing is more critical.
11. Load Register 78_H with 15_H. This tells the controller to write zeros until the next sector mark is encountered.
12. Enable sector branch by setting Register 71_H to 10_H.
13. Read status from Register 79_H and branch when active (Bit 5 is set). This means that a sector mark was encountered.
14. Read status from Register 79_H and discard contents. This guarantees a reset.
15. Check to see if any more blocks have to be written. If there are no more to be done, then load Register 78_H with 18_H. This tells the controller to stop. Monitor Register 79_H. Bit 4 (stop bit) before leaving the routine.
16. If more blocks have to be written, then load Register 78_H with 10_H and repeat Steps 7 through 15.

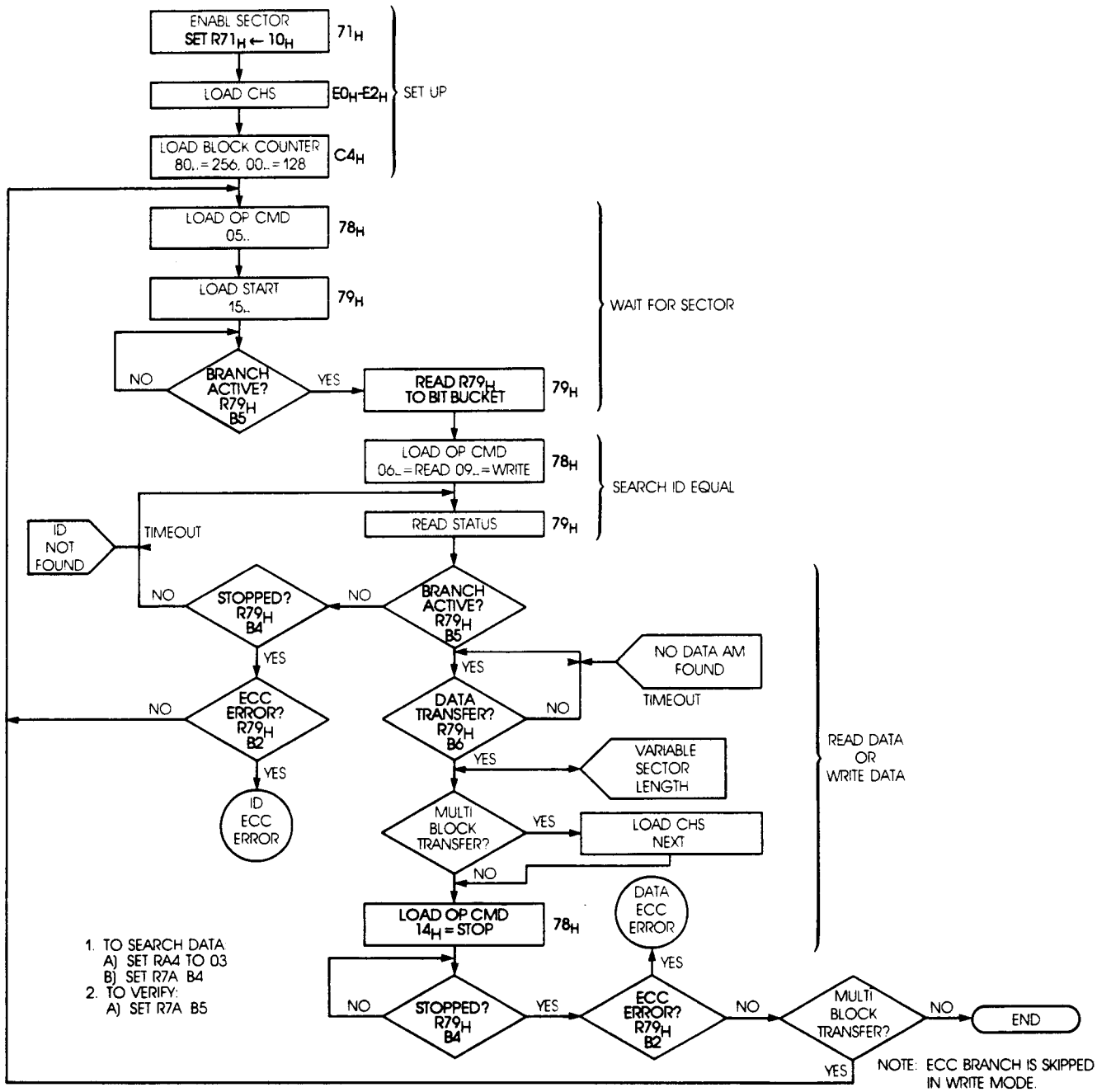
NOTE: It is suggested that Gap 1 length be kept to zero. Thus, during format, after the sector mark is encountered, the controller will write out the sync for ID field. Inter-record separation is provided by the controller writing 00_H from end of data field to next sector mark.


DISABL SECTOR
SET R71H ← 00H
71H
LOAD CHSF
E0H-E3H
LOAD BLOCK COUNTER
80H = 256, 00H = 128
C4H
LOAD OP CMD WITH 10H =
WRITE GAP 1 AND ID
78H
SET B5 IN 7A
FOR 6CH PATTERN
7AH
SET UP
LOAD START
15H
79H
WAIT FOR INDEX
OR SECTOR
READ STATUS
79H
NO
INDEX?
R79 B5
YES
LOAD OP CMD
11H = WRITE DATA
78H
WRITE ID
NO
DATA
TRANSFER
R79H
B6
YES
VARIABLE SECTOR
LENGTH ROUTINE
LOAD CHSF
E0H-E3H
WRITE DATA
LOAD OP CMD
15H = FORMAT MORE
78H
ENABL SECTOR
SET R71H ← 10H
71H
NO
BRANCH
79H
YES
READ R79
TO BIT BUCKET
79H
END
YES
STOPPED?
B4
NO
LOAD OP CMD
18H = STOP
78H
NO
MORE
BLOCKS?
YES
LOAD OP CMD WITH 10H =
WRITE GAP 1 AND ID
WRITE ZEROES
UNTIL SECTOR

HARD-SECTOR READ/WRITE: As mentioned earlier, a hard-sector drive has a sector mark between adjacent sectors. Thus, the controller starts reading the ID field after the next sector mark is encountered.

The read and write operations differ in that, after a write operation, no ECC test is necessary. The read or write data operation is performed as follows:

1. Enable branch on sector mark by setting Register 71_H to 10_H.
2. Set Registers $E0_H$, $E1_H$, and $E2_H$ with the desired sector ID.
3. Set $C4_H$ with the sector count.
4. Load Register 78_H with 05_H, the read ID command. The VFO will look for a sync of $A1_H$ after read gate is turned on.
5. Load Register 79_H with 15_H. The controller will wait for index or the next sector mark (since $R71_H$ was set to 10_H) and read the ID field.
6. Wait for Branch Active ($R79_H$, Bit 5). This means that the next sector mark or index has been encountered. The read gate will not be turned on.
7. Read and discard Register 79_H to ensure reset.
8. Load Register 78_H with 08, the read command, or a 09_H for a write command.
9. Wait for Branch Active ($R79_H$, Bit 5). If the correct ID field was read, the AIC-610F will continue on to read the data field. If an ID ECC error or incorrect sector was encountered, the stopped bit in Register 79_H will be set. If so, go back to Step 4.
10. Wait for Data Transfer ($R79_H$, Bit 6). Read data is now being transferred to the sector buffer; or from the buffer, in the case of a write.
11. If this is a multiblock transfer, update $E0_H$-$E2_H$ with next sector ID while data is being transferred.
12. Set Register 78_H with 14_H (Stop command). This will stop the controller chip at the end of the data field ECC.
13. Wait for Stopped ($R79_H$, Bit 4).
14. Test ECC Error ($R79_H$, Bit 2). If it is set, go to the error correction routine. If not, continue on to the next sector (Step 4) or end.


ENABL SECTOR
SET R71H ← 10H
71H
LOAD CHS
E0H-E2H
SET UP
LOAD BLOCK COUNTER
80.. = 256, 00.. = 128
C4H
LOAD OP CMD
05..
78H
LOAD START
15..
79H
WAIT FOR SECTOR
NO
BRANCH ACTIVE? R79H B5
YES
READ R79H TO BIT BUCKET
79H
LOAD OP CMD
06.. = READ 09.. = WRITE
78H
SEARCH ID EQUAL
READ STATUS
79H
ID NOT FOUND
TIMEOUT
NO
STOPPED? R79H B4
NO
BRANCH ACTIVE? R79H B5
YES
NO DATA AM FOUND
TIMEOUT
YES
NO
ECC ERROR? R79H B2
YES
DATA TRANSFER? R79H B6
NO
YES
VARIABLE SECTOR LENGTH
READ DATA OR WRITE DATA
ID ECC ERROR
MULTI BLOCK TRANSFER?
YES
LOAD CHS NEXT
NO
LOAD OP CMD
14H = STOP
78H
DATA ECC ERROR
1. TO SEARCH DATA
A) SET RA4 TO 03
B) SET R7A B4
2. TO VERIFY:
A) SET R7A B5
NO
STOPPED? R79H B4
YES
YES
ECC ERROR? R79H B2
NO
MULTI BLOCK TRANSFER?
NO
END
YES
NOTE: ECC BRANCH IS SKIPPED IN WRITE MODE.

Buffer Data Transfer

Buffer data transfers of the AIC-610F can be broken down into two fundamental operations:

1. Read
   - Single Block and Multiple Block
2. Write
   - Single Block and Multiple Block

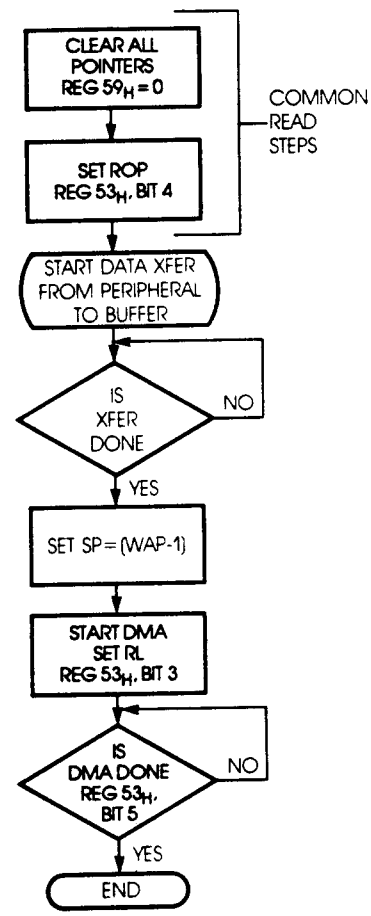
The following flowcharts illustrate the mechanism and set up required for a successful buffer data transfer:

- Single Block Read
- Multiple Block Read
- Single Block Write
- Multiple Block Write

SINGLE BLOCK READ:

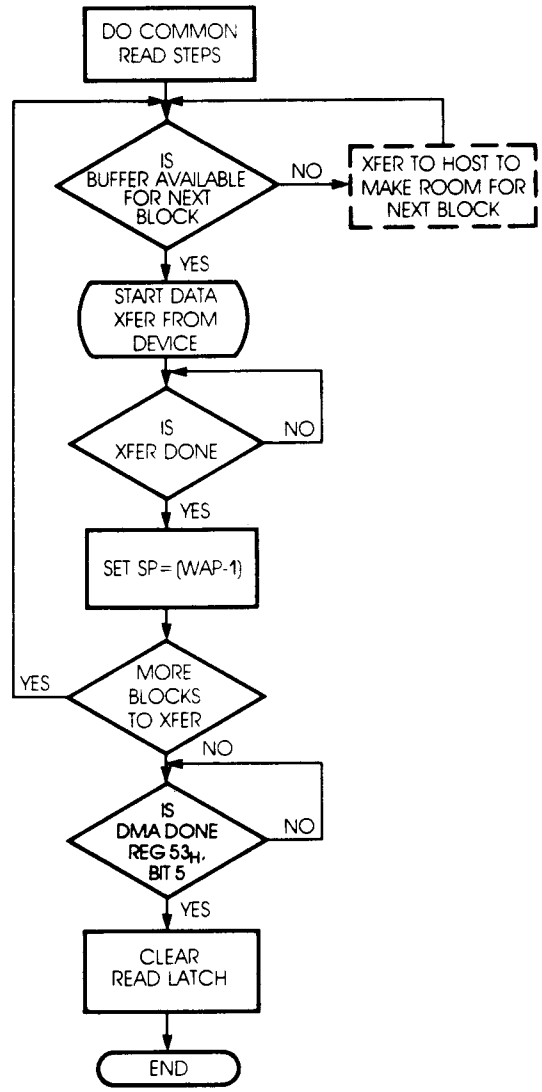
1. Clear all pointers (set Register 59_H to 0).*
2. Set up for read operation (set Register 53_H, Bit 4).
3. Transfer data to buffer (WAP will increment on each transfer).
4. At completion of transfer from device, set SP to (WAP-1) and set read latch for DMA read operation (Register 53_H, Bit 3). RAP will increment for each Request/Acknowledge cycle.
5. Monitor DMA Done bit (Register 53_H, Bit 5) to determine when the DMA transfer is complete (RAP equals SP).

*In the 15-bit or 16-bit addressing mode, after setting Register 59_H to 0, also set RAPH (Register $5B_H$) and WAPH (Register $5D_H$) to 0. This step clears external latches for the addresses.



MULTIPLE BLOCK READ:

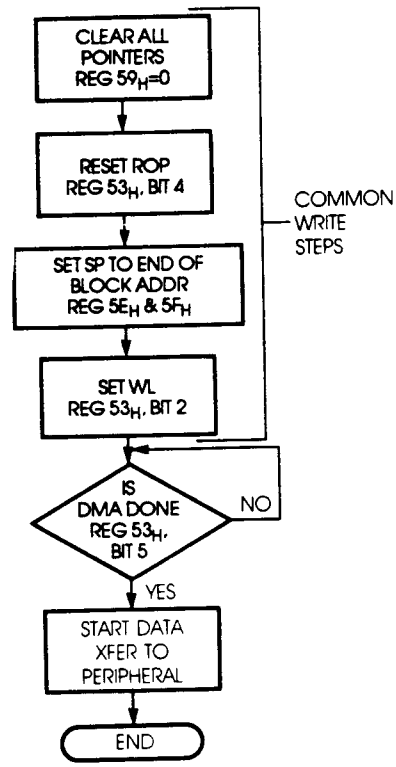
1. Do the same first four operations as single block read.
2. Read RAP to ensure that next block may be transferred from the device without overrunning the RAP.
3. Begin transfer of next block to buffer (WAP will increment on each transfer).
4. At end of transfer, set SP to new (WAP-1) address.
5. If DMA Done occurred, a restart of the DMA transfer will occur when the new SP address is set.
6. Return to Step 2 if more blocks are to be transferred.
7. Wait for DMA Done and clear Read Latch.



SINGLE BLOCK WRITE:

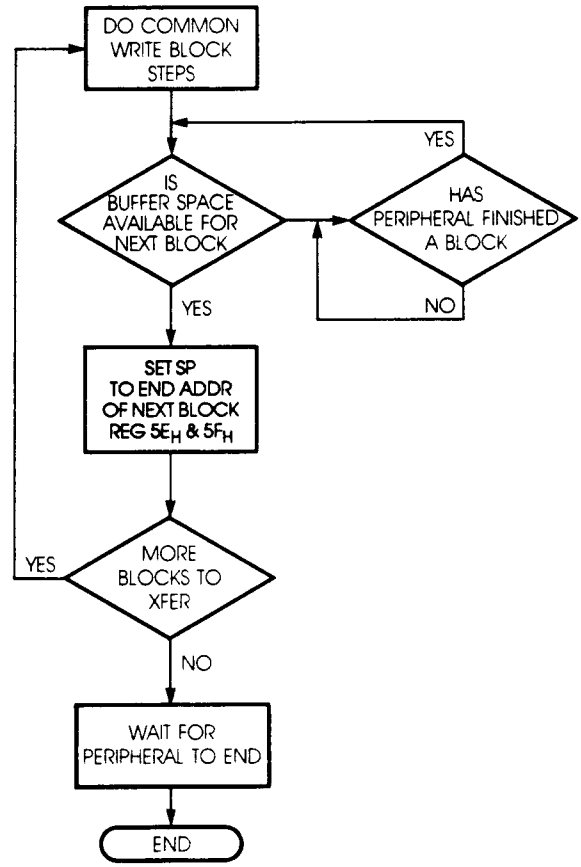
1. Clear all pointers (set Register 59_H to 0).*
2. Reset Read OP In DMA control (Register 53_H, Bit 4).
3. Set SP to the address at the end of the block to be transferred.
4. Set the Write Latch (Register 53_H, Bit 5). This causes the DMA cycles to begin.
5. Monitor DMA Done bit (Register 53_H, Bit 5) to determine when DMA transfer is complete (WAP equals SP).
6. Transfer to the peripheral may now begin.

*In the 15-bit or 16-bit addressing mode, after setting Register 59_H to 0, also set RAPH (Register $5B_H$) and WAPH (Register $5D_H$) to 0. This step clears external latches for the addresses.



MULTIPLE BLOCK WRITE:

1. Do the same first five operations as single block write.
2. Begin block transfer to peripheral.
3. Check that there is enough buffer for the next block without overrunning the RAP. If buffer space is available, set SP to end of next block.
4. If DMA Done was on, setting the new SP address will clear it and renew DMA transfers.



ABSOLUTE MAXIMUM RATINGS AND D.C. CHARACTERISTICS

Absolute Maximum Ratings

Storage Temperature . . . . . –65°C to 150°C
Voltage on Any Pin with Respect to Ground . . . . . –0.5 to 7 Volt
Power Dissipation . . . . . 0.6 W

NOTE: Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. Characteristics (Conditions: V_{CC} = 5.0V ±5%, 0°C <T <70°C)

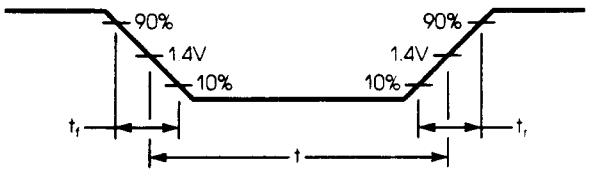
SYMBOL	PARAMETER	MIN	MAX	UNITS	CONDITIONS
V_{IL}	Input Low Voltage	–0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.5	V	I_{IH} = 10 µA
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2 mA*
V_{OH}	Output High Voltage	2.4			I_{OH} = 400 µA
I_{CC}	Supply Current		100	mA	V_{CC} = 5.5V
I_{IL}	Input Leakage	–2	2	µA	0 <V_{IN} <V_{CC}
I_{OL}	Output Leakage Off State	100	100	µA	0.45 <V_{OUT} <V_{CC}
C_{IN}	Input Capacitance		10	pF	
C_{OUT}	Output Capacitance		30	pF	

*NOTE: For RG and WG, I_{OL} = 4 mA.

TIMING DIAGRAMS

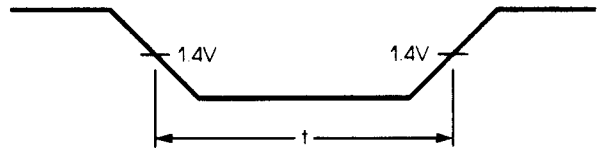
A.C. Characteristics (Conditions: V_{CC} = 5.0V ±5%, 0°C <T <70°C)

A.C. INPUT TIMING CONDITIONS



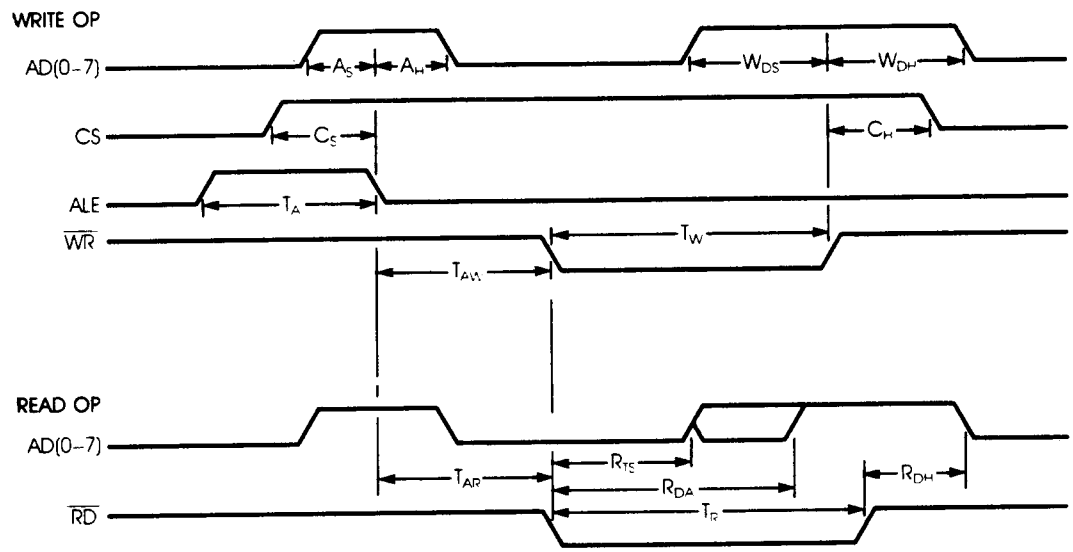
t_f < 5 ns t_r < 5 ns

A.C. OUTPUT TIMING CONDITIONS



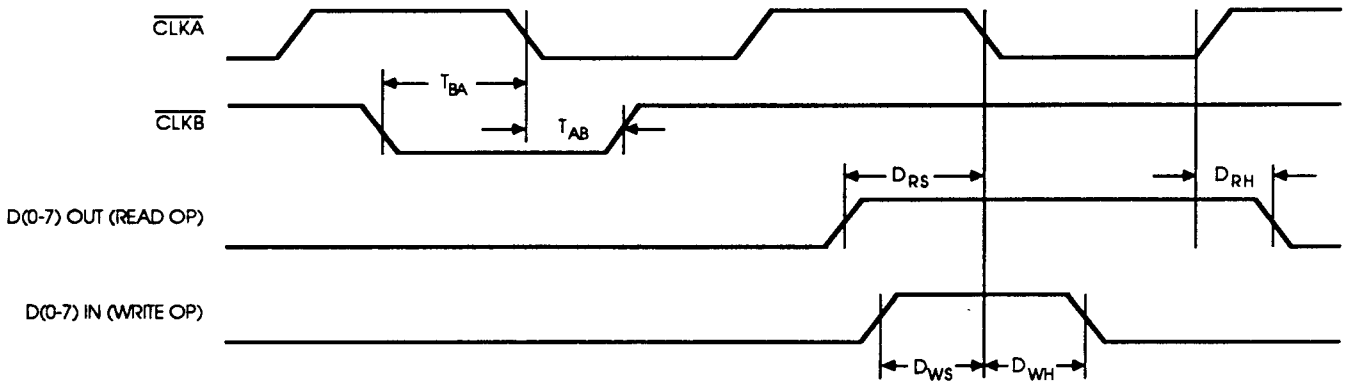
C_L = 50 pF

AIC-610F Microprocessor Interface Timing



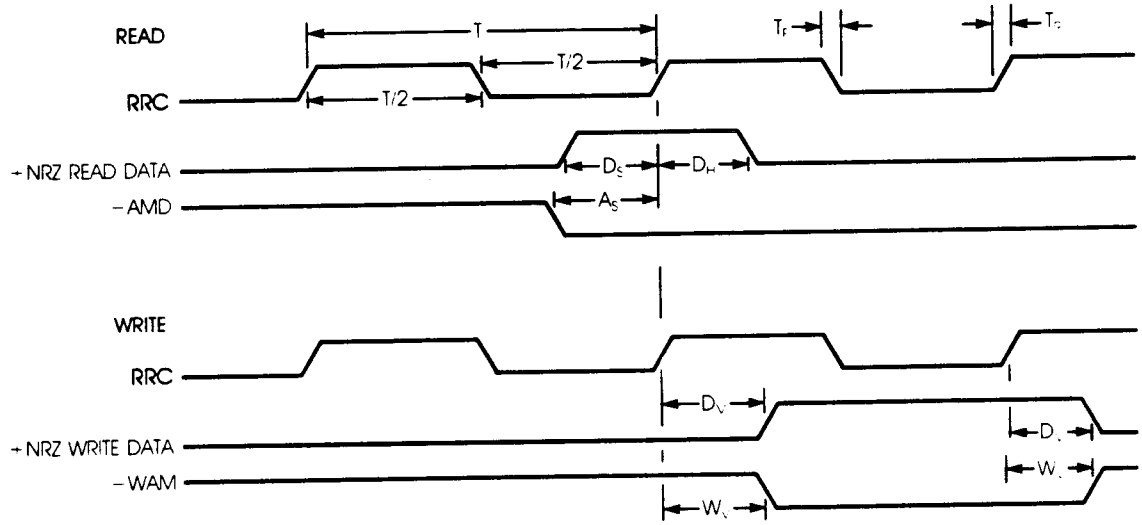
SYMBOL	PARAMETER	MIN	MAX	UNITS
T_A	ALE Width	15		ns
T_{AW}	ALE ↓ to $\overline{WR}$ ↓	15		ns
T_{AR}	ALE ↓ to $\overline{RD}$ ↓	15		ns
T_W	$\overline{WR}$ Width	40		ns
T_R	$\overline{RD}$ Width	40		ns
A_S	ADRS Valid to ALE ↓	5		ns
A_H	ALE ↓ to ADRS $\overline{Invalid}$	5		ns
C_S	CS Valid to ALE ↓	5		ns
C_H	$\overline{RD}$ or $\overline{WR}$ ↑ to CS ↓	0		ns
W_{DS}	Write Data Valid to $\overline{WR}$ ↑	70		ns
W_{DH}	$\overline{WR}$ ↑ to Write Data $\overline{Invalid}$	5		ns
R_{TS}	$\overline{RD}$ ↓ to AD(0-7) Active	15		ns
R_{DA1}	$\overline{RD}$ ↓ to Data Valid (Registers 71-7F)		55	ns
R_{DS2}	$\overline{RD}$ ↓ to Data Valid (All Other Registers)		55	ns
R_{DH}	$\overline{RD}$ ↑ to Data $\overline{Invalid}$	10	30	ns

Read/Write Data Bus Timing



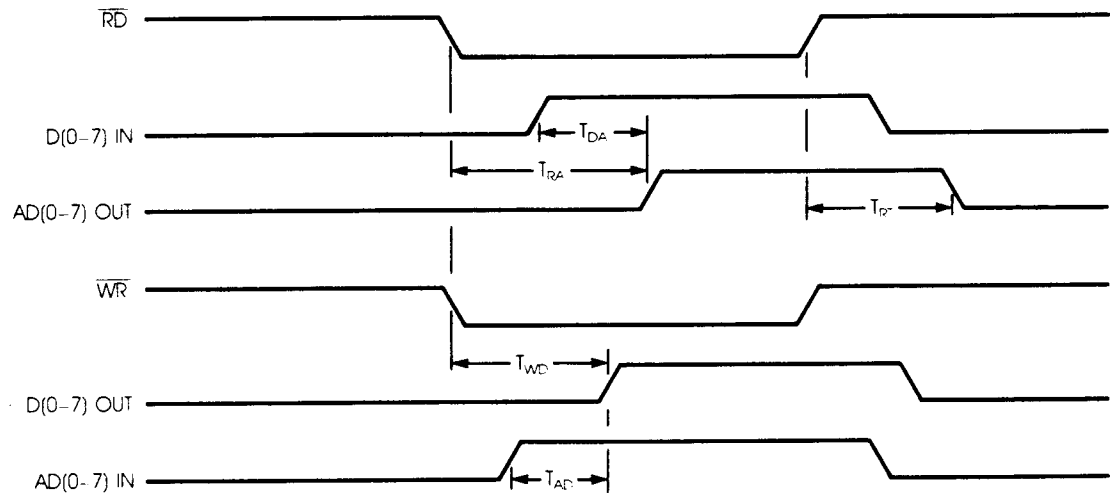
SYMBOL	PARAMETER	MIN	MAX	UNITS
T_{BA}	$\overline{CLKB}$ ↓ to $\overline{CLKA}$ ↓	90		ns
T_{AB}	$\overline{CLKA}$ ↓ to $\overline{CLKB}$ ↑	55		ns
D_{RS}	D(0-7) Out Valid to $\overline{CLKA}$ ↓	30		ns
D_{RH}	$\overline{CLKA}$ ↑ to D(0-7) Out Valid	-30	30	ns
D_{WS}	D(0-7) In Valid to $\overline{CLKA}$ ↓	45		ns
D_{WH}	$\overline{CLKA}$ ↓ to D(0-7) In $\overline{Invalid}$	15		ns
$T_R = T_F$	$\overline{CLKA}$, $\overline{CLKB}$ Rise/Fall Time		15	ns

Disk Read/Write Timing



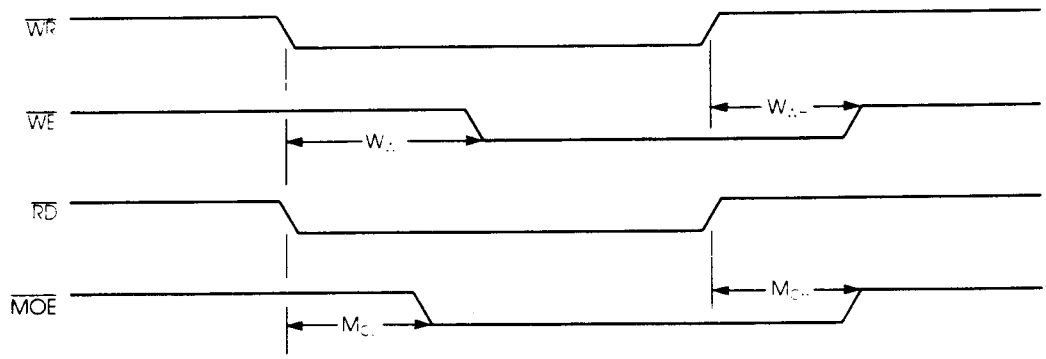
SYMBOL	PARAMETER	MIN	MAX	UNITS
T	RRC Period	65	5000	ns
T/2	RRC Period ÷ 2	25		ns
T_R	RRC Rise Time		10	ns
T_F	RRC Fall Time		10	ns
D_S	Data In Valid to RRC ↑	10		ns
D_H	RRC ↑ to Data In $\overline{\text{Invalid}}$	20		ns
A_S	AMD Valid to RRC ↑	10		ns
D_V	RRC ↑ to Data Out	5	40	ns
W_V	RRC ↑ to W_{AM} Out	5	40	ns

Registers 50, 51, and 70 Timing



SYMBOL	PARAMETER	MIN	MAX	UNITS
T_{DA}	D(0-7) In Valid to AD(0-7) Out Valid		45	ns
T_{RA}	$\overline{RD}$ ↓ to AD(0-7) Out Valid		80	ns
T_{RT}	$\overline{RD}$ ↑ to AD(0-7) Out Tri State	20	80	ns
T_{WD}	$\overline{WR}$ ↓ to D(0-7) Out Valid		40	ns
T_{AD}	AD(0-7) In Valid to D(0-7) Out Valid		45	ns
T_{DH70}	$\overline{WR}$ to D(0-7) Out Valid	10		ns

Read/Write Register 70

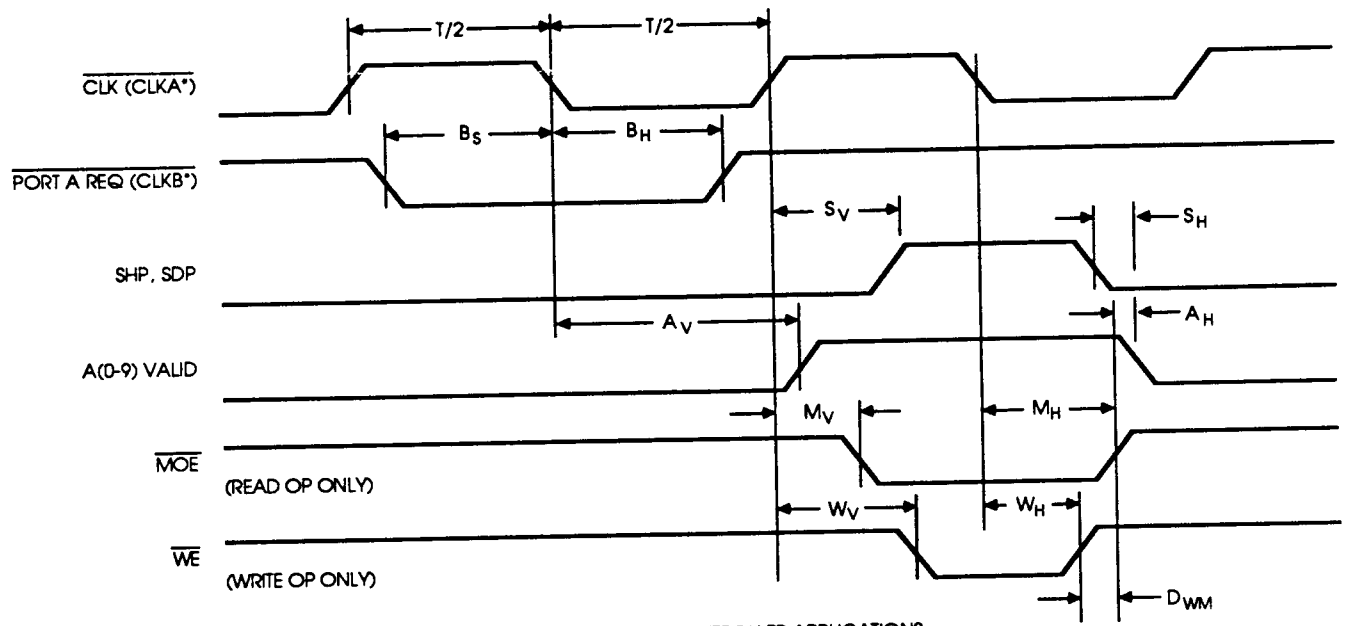


SYMBOL	PARAMETER	MIN	MAX	UNITS
M_{CL}	$\overline{RD}\downarrow$ to $\overline{MOE}\downarrow$		30	ns
M_{CH}	$\overline{RD}\uparrow$ to $\overline{MOE}\uparrow$		30	ns
W_{WL}	$\overline{WR}\downarrow$ to $\overline{WE}\downarrow$		30	ns
W_{WH}	$\overline{WR}\uparrow$ to $\overline{WE}\uparrow$		30	ns

Registers 6E and 6F Timing

SYMBOL	PARAMETER	MIN	MAX	UNITS
F	$\overline{RD}$ or $\overline{WR}$ ↓ to 6E or 6F		30	ns
R	$\overline{RD}$ or $\overline{WR}$ ↑ to 6E or 6F		30	ns

Buffer Ram Interface

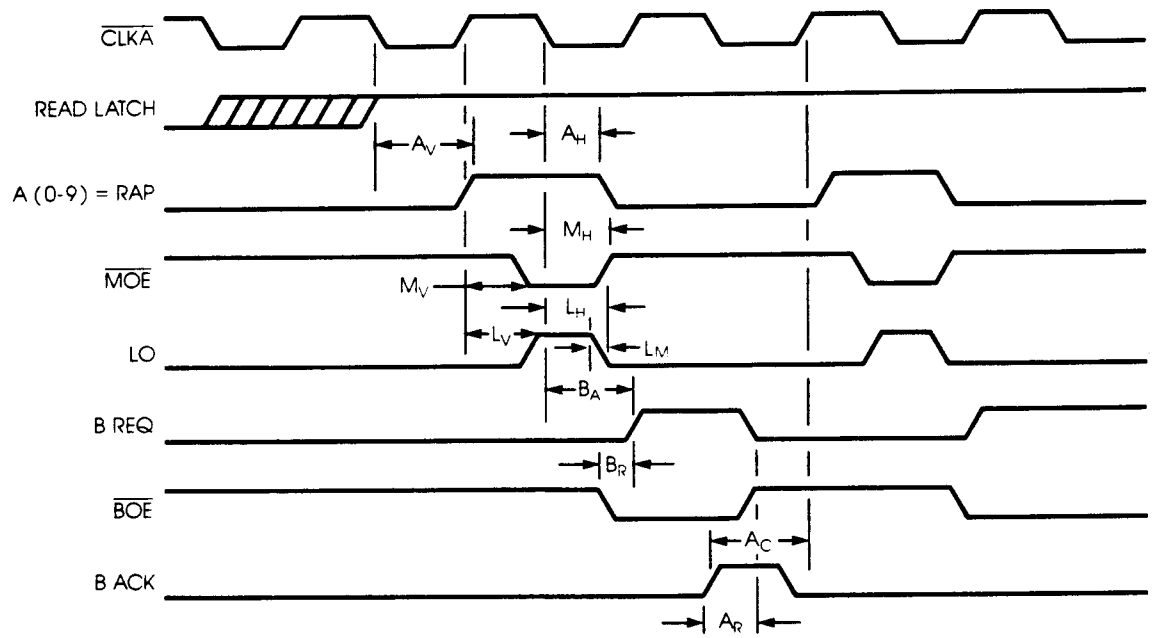


*CONNECTS TO THESE SIGNALS ON THE AIC-010 OR THE AIC-100 IN DISK CONTROLLER APPLICATIONS.

SYMBOL	PARAMETER	MIN	MAX	UNITS
T/2	Clock Half Cycle	145		ns
B_S	CLKB ↓ to CLKA ↓ (Set-Up)	90		ns
B_H	CLKA ↓ to CLKB ↑ (Hold)	10		ns
A_V	CLKA ↓ to Address Valid		55	ns
A_H	MOE ↑ to Address Invalid (Hold)	5		ns
M_V	CLKA ↑ to $\overline{MOE}$ ↓		20	ns
M_H	CLKA ↓ to $\overline{MOE}$ ↑		30	ns
W_V	CLKA ↑ to WE ↓		40	ns
W_H	CLKA ↓ to WE ↑	5	25	ns
D_{WM}	WE ↑ to MOE ↑			ns
S_V	CLK ↑ or SHP ↑	10	50	ns
S_H	SHP ↓ or SDP ↓ to Address Invalid	10		ns

Buffer to Host Interface

READ OPERATION

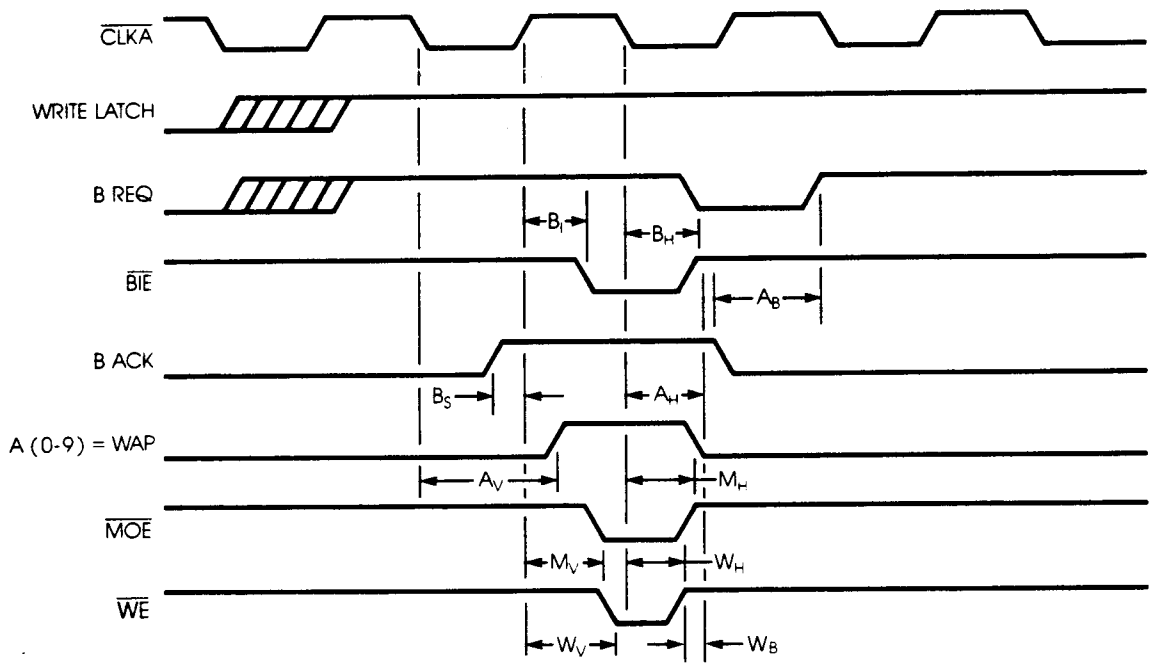


SYMBOL	PARAMETER	MIN	MAX	UNITS
A_V	$\overline{CLKA}$ ↓ to Address Invalid		55	ns
A_H	$\overline{CLKA}$ ↓ to Address Valid	5		ns
M_V	$\overline{CLKA}$ ↑ to $\overline{MOE}$ ↓		20	ns
M_H	$\overline{CLKA}$ ↓ to $\overline{MOE}$ ↑		30	ns
L_V	$\overline{CLKA}$ ↑ to LO ↑		50	ns
L_H	$\overline{CLKA}$ ↓ to LO ↓	0		ns
L_M	LO ↓ to $\overline{MOE}$ ↑	5		ns
B_A	$\overline{CLKA}$ ↓ to BREQ ↑		140	ns
B_R	$\overline{BOE}$ ↓ to BREQ ↑	45		ns
A_C	BACK ↑ to $\overline{CLKA}$ ↑ Set-Up		30	ns
A_R	BACK ↑ to BREQ ↓		20	ns

NOTE: A Port A Req cycle has priority over a DMA cycle; i.e., if Port A Req is not active during the falling edge of CLK, the following cycle is available for DMA. If Port A Req is active, $\overline{BIE}$ will be disabled for one CLK cycle.

Host to Buffer Interface

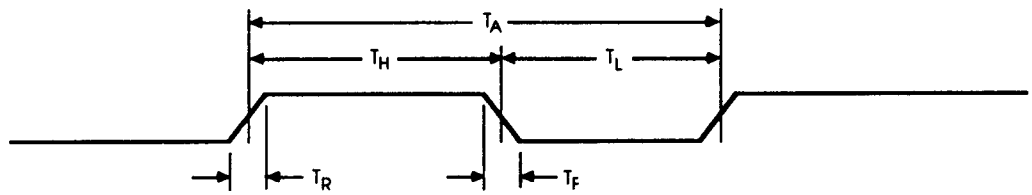
WRITE OPERATION



SYMBOL	PARAMETER	MIN	MAX	UNITS
B_I	CLKA ↑ to BIE ↓		40	ns
B_H	CLKA ↓ to BREQ ↓		40	ns
B_S	BACK ↑ to CLKA ↑	45		ns
A_V	CLKA ↓ to Address Valid		35	ns
A_H	CLKA ↓ to Address Invalid	5		ns
M_V	CLKA ↑ to $\overline{MOE}$ ↓		20	ns
M_H	CLKA ↓ to $\overline{MOE}$ ↑		30	ns
W_B	$\overline{WE}$ ↑ to $\overline{BIE}$ ↑	5		ns
W_V	CLKA ↑ to WE ↓		40	ns
W_H	CLKA ↓to WE ↑	5	25	ns
A_B	BACK ↓ to BREQ ↑		25	ns

NOTE: The assertion of $\overline{BIE}$ and $\overline{WE}$ will have identical timing relationships with respect to CLK.

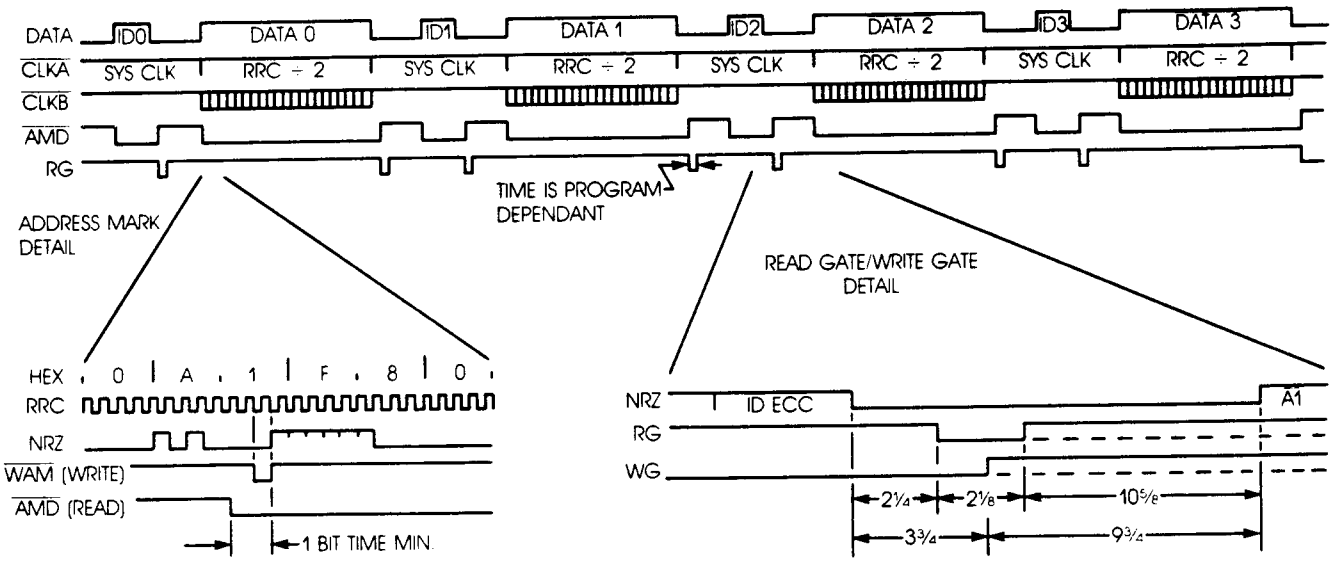
System Clock Timing



SYMBOL	PARAMETER	MIN	MAX	UNITS
$T_R = T_F$	SYS CLK Rise/Fall Time		10	ns
T_A	SYS CLK Period	200		ns
T_L	SYS CLK Low	45		ns
T_H	SYS CLK High	45		ns

REFERENCE TIMING DIAGRAM

(For ST412/506 Sequencer Map In Table 8, Located on Page 51.)



NOTE: NUMBERS ARE GIVEN IN BYTE TIMES. NOTICE THAT THE SUM OF RG TIMES IS 1.5 BYTES LONGER THAN THE SUM OF WG TIMES DUE TO 1 BYTE DATA DELAY IN THE AIC-610F AND 1/2 BYTE DELAY IN THE ENCODER.

APPENDICES

A1 NOTES ON ECC

Differences Between the 32-Bit and 48-Bit ECC

Both 32-bit and 48-bit ECC codes are computer generated. The 32-bit ECC code is included in the data sheet. To obtain rights to the 48-bit ECC polynomial, a license agreement with Adaptec must be signed.

The way the 32-bit ECC is set-up is as follows:

REG 72		REG 73		REG 74	REG 75	REG 76	REG 77
7	1	7	1	7 POLY 8	7 POLY 16	7 POLY 24	7 NOT USED
6	1	6	1	6 POLY 7	6 POLY 15	6 POLY 23	6 POLY 31
5	1	5	1	5 POLY 6	5 POLY 14	5 POLY 22	5 POLY 30
4	1	4	1	4 POLY 5	4 POLY 13	4 POLY 21	4 POLY 29
3	1	3	1	3 POLY 4	3 POLY 12	3 POLY 20	3 POLY 28
2	1	2	1	2 POLY 3	2 POLY 11	2 POLY 19	2 POLY 27
1	1	1	1	1 POLY 2	1 POLY 10	1 POLY 18	1 POLY 26
0	1	0	1	0 POLY 1	0 POLY 9	0 POLY 17	0 POLY 25

NOTE: For a 32-bit ECC, Register 72_H and Register 73_H must be set to FF_H, ensuring correct operation.

For the 48-bit ECC, the polynomial is set-up as follows:

REG 72	REG 73	REG 74	REG 75	REG 76	REG 77
7 POLY 8	7 POLY 16	7 POLY 24	7 POLY 32	7 POLY 40	7 NOT USED
6 POLY 7	6 POLY 15	6 POLY 23	6 POLY 31	6 POLY 39	6 POLY 47
5 POLY 6	5 POLY 14	5 POLY 22	5 POLY 30	5 POLY 38	5 POLY 46
4 POLY 5	4 POLY 13	4 POLY 21	4 POLY 29	4 POLY 37	4 POLY 45
3 POLY 4	3 POLY 12	3 POLY 20	3 POLY 28	3 POLY 36	3 POLY 44
2 POLY 3	2 POLY 11	2 POLY 19	2 POLY 27	2 POLY 35	2 POLY 43
1 POLY 2	1 POLY 10	1 POLY 18	1 POLY 26	1 POLY 34	1 POLY 42
0 POLY 1	0 POLY 9	0 POLY 17	0 POLY 25	0 POLY 33	0 POLY 41

Programming 8-Bit ECC Correction With a 32-Bit ECC Polynomial

After each read data operation, a read error may have occurred. This may be determined by reading Register 79. If Bit 2 is set, an error did occur and the following procedure is employed to determine if the error is correctable. Note that the majority of read errors are soft (i.e., caused by noise) and that the correction algorithm is time consuming. It is recommended that the record be reread before attempting correction.

The general flow of the algorithm for 8-bit correction is as follows:

1. Off-load the 32-bit syndrome into local RAM.
2. Shift the syndrome back into the ECC register in reverse order; swapping the syndrome end for end.
3. Change the ECC polynomial from forward to reverse.
4. Shift the ECC until all bits, except the high order (24-31) bits, are zero (correctable) or the number of shifts is greater than the number of bits in the record (uncorrectable).
5. If correctable, the number of shifts represents the displacement of the error from the end of the record (the last bit of the ECC). The error pattern is located in Bits 24-31 of the ECC register. This pattern is exclusive ORed with the appropriate bits in memory to correct the error.

DETAILED PROGRAMMING STEPS:

1. After a read error is detected, disable feedback by setting $R71_H = C4_H$.
2. Store 00_H in RAM (x).
3. Store 00_H in RAM (x + 1).
4. Store contents of $R73_H$ in RAM (x + 2).
5. Shift ECC eight times by setting $R71_H = C6_H$ eight times.
6. Store contents of R73 in RAM (x + 3).
7. Shift ECC eight times by setting $R71_H = C6_H$ eight times.
8. Store contents of $R73_H$ in RAM (x + 4).
9. Shift ECC eight times by setting $R71_H = C6_H$ eight times.
10. Store contents of $R73_H$ in RAM (x + 5).
11. Clear ECC and disable feedback by setting $R71_H$ to $C8_H$ and then $C4_H$.
12. Right rotate location RAM (x + 5) and test if any carry is set (i.e., test Bit 0). If set, then load $R71_H = 07_H$. If not set, then load $R71_H = 06_H$. Repeat the operation seven more times to load the entire byte.
13. Repeat Step 12 for RAM locations x + 4, x + 3, x + 2, x + 1, and x until all 32 bits of the syndrome are loaded into the ECC in reverse order along with the 16 zero bits.
14. Load $R74_H = 82_H$ and $R77_H = 21_H$ to enable the reverse polynomial and disable the forward polynomial.
15. Compute record length in bits: Number of bits per data field = ECC + Data + AM and SYNC for a 256-byte record. Length in bits = 4 • 8 + 256 • 8 + 2 • 8 = 2096.
16. Enable feedback by setting $R71_H = C0_H$.
17. Shift ECC once by setting $R71_H = C2_H$ and increment a software counter.
18. Test to see if the software counter is greater than the record length. If yes, the error is uncorrectable. Re-enable the forward polynomial and end operation.
19. Test to see if $R72_H = 00_H$. If yes, go to Step 20. If no, go to Step 17.
20. Subtract hardware offset of 7 from the shift count. If a correctable error is located within the ECC or the SYNC and AM bytes (shift count < = 32), the data field is good and no further action is required. Otherwise, subtract 32 from the shift count.
21. The bit displacement (shift count) must now be converted to a byte offset by right shifting the count three times. The value of the shift count equals the bit displacement from end of the record.
22. $R73_H$ is the mirror image of the error pattern. Form the error mask data (two bytes) by concatenating $R73_H$ with a zero byte.
23. Get the shift count (E) for error mask data by extracting the lower three bits from the shift count obtained in Step 20.
24. Right shift the error mask data with MSB (Bit 15) set to zero. Repeat E-1 times more.
25. Mirror the error mask data byte-by-byte. For example, if the original error mask data is 5C 9A, after mirroring, the data is 3A 59.
26. The two-byte error mask data may now be XORed with the data in memory (obtained with a Register 70_H access) to correct the error. Byte offset obtained from Step 21.

NOTES:

1) For proper 32-bit ECC polynomial operation, Register 72_H must be loaded with a FF_H and Register 73_H must be loaded with a FF_H.
2) For five-bit ECC correction, the following modification is necessary.
   - Step 17: Test to see if $R72_H = 00_H$ and $R73_H$, Bits 0, 1, and 2, are zero. If yes, go to Step 13.
   - Step 18. $R73_H$, Bits 3-7, are the mirror image of the error pattern (0-7 for eight-bit ECC).

Programming 8-Bit ECC Correction With a 48-Bit ECC Polynomial

After each read data operation, a read error may have occurred. This may be determined by reading Register 79_H. If Bit 2 is set, an error did occur and the following procedure is employed to determine if the error is correctable. Note that the majority of read errors are soft (i.e., caused by noise) and that the correction algorithm is time consuming. It is recommended that the record be reread before attempting correction.

The general flow of the algorithm for 8-bit correction is as follows:

1. Off-load the 48-bit syndrome into local RAM.
2. Shift the syndrome back into the ECC register in reverse order; swapping the syndrome end for end.
3. Change the ECC polynomial from forward to reverse.
4. Shift the ECC until all bits, except the high order (40-47) bits, are zero (correctable) or the number of shifts is greater than the number of bits in the record (uncorrectable).
5. If correctable, the number of shifts represents the displacement of the error from the end of the record (the last bit of the ECC). The error pattern is located in Bits 40-47 of the ECC register. This pattern is exclusive ORed with the appropriate bits in memory to correct the error.

DETAILED PROGRAMMING STEPS:

1. After a read error is detected, disable feedback by setting $R71_H = 44_H$.
2. Store contents of $R73_H$ in RAM (x).
3. Shift ECC eight times by setting $R71_H = 46_H$ eight times.
4. Store contents of $R73_H$ in RAM (x + 1).
5. Shift ECC eight times by setting $R71_H = 46_H$ eight times.
6. Store contents of $R73_H$ in RAM (x + 2).
7. Shift ECC eight times by setting $R71_H = 46_H$ eight times.
8. Store contents of $R73_H$ in RAM (x + 3).
9. Shift ECC eight times by setting $R71_H = 46_H$ eight times.
10. Store contents of $R73_H$ in RAM (x + 4).
11. Shift ECC eight times by setting $R71_H = 46_H$ eight times.
12. Store contents of $R73_H$ in RAM (x + 5).
13. Clear ECC and disable feedback by setting $R71_H$ to 48_H and then 44_H.
14. Right rotate location RAM (x + 5) and test if any carry is set (i.e., test Bit 0). If set, then load $R71_H = 07_H$. If not set, then load $R71_H = 06_H$. Repeat the operation seven more times to load the entire byte.
15. Repeat Step 14 for RAM locations x + 4, x + 3, x + 2, x + 1, and x until all 48 bits of the syndrome are loaded into the ECC in reverse order.
16. Load Registers 72_H-77_H to enable the reverse polynomial and disable the forward polynomial.
17. Compute record length in bits: Number of bits per data field = ECC + Data + AM and SYNC for a 256-byte record. Length in bits = 4 • 8 + 256 • 8 + 2 • 8 = 2096.
18. Enable feedback by setting $R71_H = 40_H$.
19. Shift ECC once by setting $R71_H = 42_H$ and increment a software counter.
20. Test to see if the software counter is greater than the record length. If yes, the error is uncorrectable. Re-enable the forward polynomial and end operation.
21. Test to see if $R72_H = 00_H$. If yes, go to Step 22. If no, go to Step 19.
22. Subtract hardware offset of 7 from the shift count. If a correctable error is located within the ECC or the SYNC and AM bytes (shift count < = 48), the data field is good and no further action is required. Otherwise, subtract 48 from the shift count.
23. The bit displacement (shift count) must now be converted to a byte offset by right shifting the count three times. The value of the shift count equals the bit displacement from end of the record.
24. $R73_H$ is the mirror image of the error pattern. Form the error mask data (two bytes) by concatenating $R73_H$ with a zero byte.
25. Get the shift count (E) for error mask data by extracting the lower three bits from the shift count obtained in Step 22.
26. Right shift the error mask data with MSB (Bit 15) set to zero. Repeat E-1 times more.
27. Mirror the error mask data byte-by-byte. For example, if the original error mask data is 5C 9A, after mirroring, the data is 3A 59.
28. The two-byte error mask data may now be XORed with the data in memory (obtained with a Register 70_H access) to correct the error. Byte offset obtained from Step 23.

NOTES:

1) For five-bit ECC correction, the following modification is necessary.
   - Step 17: Test to see if $R72_H = 00_H$ and $R73_H$, Bits 0, 1, and 2, are zero. If yes, go to Step 15.
   - Step 18. $R73_H$, Bits 3-7, are the mirror image of the error pattern (0-7 for eight-bit ECC).

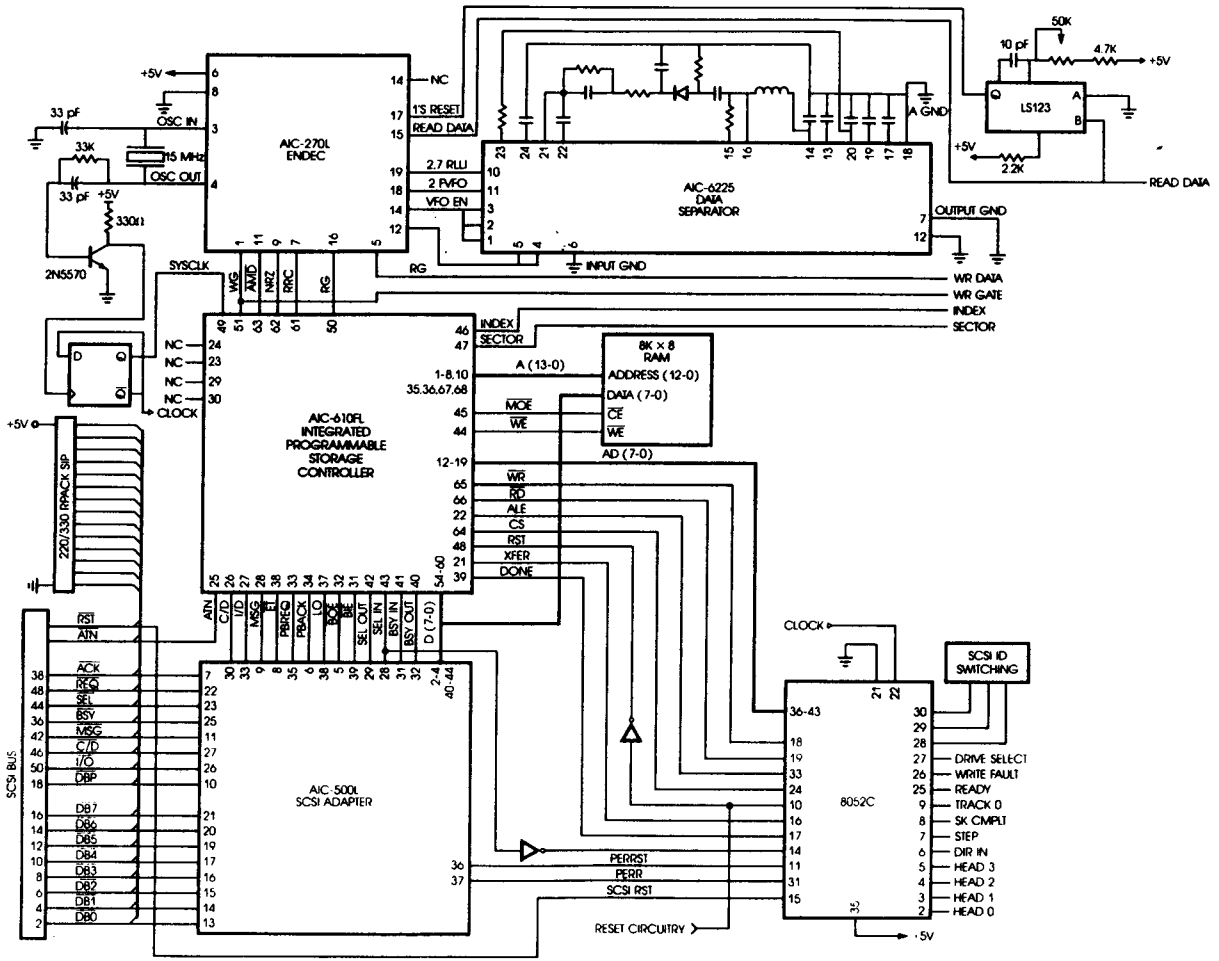
ECC Application Issues

1. ECC calculation begins:
   Write Operation: The sequencer word that sets AM data type (Bit 7 of the Count Field) while WG is on.
   Read Operation: While RG is on, AM data type is active and the sync byte is found.

2. Due to internal timing relationships, different code implementations of the Error Correction Code Algorithm (code specific applications) may need to use slightly different values (± a few bits) for the total number of bits in the record to find the exact displacement location of the error. Once this value is established (for a specific application), it will not change.

3. Due to internal timing delays, it is recommended for proper operation of the device, that the first PAD byte following the ECC/CRC data field in the sector layout have a value of 00_H only. If a value of 00_H is not used, an ECC/CRC may occur.

A2 SCHEMATIC OF AN EMBEDDED CONTROLLER DESIGN WITH THE AIC-610F



A2 SCHEMATIC OF AN EMBEDDED CONTROLLER DESIGN WITH THE AIC-610F

A3 SOFT-SECTOR FORMAT SEQUENCING EXAMPLE FOR THE AIC-610F

The following example shows the operation of the AIC-610F (programmed for MFM) during a soft-sector format operation. The examples is presented using "Snapshots" of the AIC-610F and system activity descriptions. The drive is formatted for 256-byte sectors.

SYSTEM ACTIVITY: Host initializes AIC-610F registers.

Register $E0_H$ = ID Cylinder Value
Register $E1_H$ = ID Head Value
Register $E2_H$ = ID Sector Value
Register $E3_H$ = ID Flag Value
Register $8C_H = 11_H$
Register $8D_H = 14_H$
Register $94_H = 16_H$
Branch register $78_H = 16_H$
Host writes 15_H to the AIC-610F sequencer start register (Register 79_H)

CURRENT ADDRESS	DATA	DATA TYPE			COUNT	CONTROL		BRANCH CONDITION	NEXT ADDRESS	BRANCH ADDRESS REGISTER
15_H	00_H	0	0	0	00000	0	0	110	10101	16_H

Data: 00_H
Data Type: Normal
Count: 1
Control: No-Op
Branch Condition: Branch On Index Or Sector
Next Address: 15_H (Loop Until Branch)

SYSTEM ACTIVITY: AIC-610F waits for the leading edge of Index to begin writing Gap 1. When index is detected by the AIC-610F, the Branch Active Flag is set.

CUURENT ADDRESS	DATA	DATA TYPE			COUNT	CONTROL		BRANCH CONDITION	NEXT ADDRESS	BRANCH ADDRESS REGISTER
16_H	$4E_H$	0	0	0	01010	8	0	000	10000	16_H

Data: Gap 1 Character
Data Type: Normal
Count: 11
Control: Set Write Gate
Branch Condition: No Branch
Next Address: 10_H

SYSTEM ACTIVITY: AIC-610F has asserted Write gate and the NRZ output is the Gap 1 character. The host has detected the Index pulse.

CURRENT ADDRESS	DATA	DATA TYPE			COUNT	CONTROL		BRANCH CONDITION	NEXT ADDRESS	BRANCH ADDRESS REGISTER
10_H	00_H	0	0	0	00000	8	0	000	01110	16_H

Data: Sync Character
Data Type: Normal
Count: 1
Control: Keep Write Gate Set
Branch Condition: No Branch
Next Address: OE_H

SYSTEM ACTIVITY: AIC-610F NRZ output is 00_H Sync characters.

CURRENT ADDRESS	DATA	DATA TYPE			COUNT	CONTROL		BRANCH CONDITION	NEXT ADDRESS	BRANCH ADDRESS REGISTER
$0E_H$	00	0	0	1	01010	8	0	000	00110	16_H

Data: Sync Character
Dta Type: Normal
Count: 11
Control: Keep Write Gate Set
Branch Condition: No Branch
Next Address: 06_H

SYSTEM ACTIVITY: AIC-610F NRZ output is 00_H Sync characters. The host is waiting for data transfer.

CURRENT ADDRESS	DATA	DATA TYPE			COUNT	CONTROL		BRANCH CONDITION	NEXT ADDRESS	BRANCH ADDRESS REGISTER
06_H	$A1_H$	1	0	0	00000	0	2	000	00111	16_H

Data: Address Mark Character
Data Type: Address Mark
Count: 1
Control: Set Compare Enable Bit
(Not Valid For Compare With RG=0)
Branch Condition: No Branch
Next Address: 07_H

SYSTEM ACTIVITY: AIC-610F NRZ output is the $A1_H$ Address Mark character. The AM output will be strobed at the bit time defined by the WAM control register.

CURRENT ADDRESS	DATA	DATA TYPE			COUNT	CONTROL		BRANCH CONDITION	NEXT ADDRESS	BRANCH ADDRESS REGISTER
07_H	FE_H	0	0	0	00000	0	2	000	00000	16_H

Data: ID Field Sync Character
Data Type: Normal
Count: 1
Control: Keep The Compare Enable Bit Set
Branch Condition: No Branch
Next Address: 00_H

SYSTEM ACTIVITY: AIC-610F NRZ output is FE_H ID field sync character.

CURRENT ADDRESS	DATA	DATA TYPE			COUNT	CONTROL		BRANCH CONDITION	NEXT ADDRESS	BRANCH ADDRESS REGISTER
00_H	CYL	0	0	0	00000	1	2	000	00001	16_H

Data: Cylinder Value Established By The Host
Data Type: Normal
Count: Set Stack Enable So That Read Data Is Pushed Onto 8-Bit Stack
Keep Compare Enable Bit On
Branch Condition: No Branch
Next Address: 01_H

SYSTEM ACTIVITY: AIC-610F NRZ output is ID cylinder, head, sector, flag bytes as sequencer executes locations 00_H, 01_H, 02_H, and 03_H. The next location after 03_H (Flag byte write) is $0C_H$ ID ECC write.

CURRENT ADDRESS	DATA	DATA TYPE			COUNT	CONTROL		BRANCH CONDITION	NEXT ADDRESS	BRANCH ADDRESS REGISTER
$0C_H$	00_H	0	1	0	00011	0	0	000	10001	16_H

Data: 00
Data Type: ECC
Count: 4
Control: Reset All Control Bits
Branch Condition: No Branch
Next Address: 11_H

SYSTEM ACTIVITY: AIC-610F NRZ output is ID ECC characters (4 bytes).

CURRENT ADDRESS	DATA	DATA TYPE			COUNT	CONTROL		BRANCH CONDITION	NEXT ADDRESS	BRANCH ADDRESS REGISTER
11_H	00_H	0	0	0	00101	8	0	000	01001	16_H

Data: Data Field Sync Bytes
Data Type: Normal
Count: 6
Control: Set Write Gate
Branch Condition: No Branch
Next Address: 09_H

CURRENT ADDRESS	DATA	DATA TYPE			COUNT	CONTROL		BRANCH CONDITION	NEXT ADDRESS	BRANCH ADDRESS REGISTER
09_H	00_H	0	0	1	01011	8	0	000	01010	16_H

Data: Data Field Sync Bytes
Data Type: Set SEBCA
Count: 12
Control: Keep Write Gate Set
Branch Condition: No Branch
Next Address: $0A_H$

SYSTEM ACTIVITY: The AIC-610F NRZ output is 18 bytes of 00_H (Data Field Sync bytes).

CURRENT ADDRESS	DATA	DATA TYPE			COUNT	CONTROL		BRANCH CONDITION	NEXT ADDRESS	BRANCH ADDRESS REGISTER
$0A_H$	$A1_H$	1	0	1	00000	0	2	000	01011	16_H

Data: Data Field Address Mark
Data Type: Address Mark
(Keep SEBCA Set)
Count: 1
Control: Set Compare Enable Bit
Branch Condition: No Branch
Next Address: 08_H

SYSTEM ACTIVITY: Write 1 byte Data Field AM. AM will be strobed as defined by the WAM control register. $\overline{CLKA}$ and Bit Ring will synchronize.

CURRENT ADDRESS	DATA	DATA TYPE			COUNT	CONTROL		BRANCH CONDITION	NEXT ADDRESS	BRANCH ADDRESS REGISTER
$0B_H$	$F8_H$	0	0	0	00000	0	2	000	00100	16_H

Data: Sync Byte
Data Type: Normal
Count: 1
Control: Keep Compare Enable Bit Set
Branch Condition: No Branch
Next Address: 04_H

SYSTEM ACTIVITY: The AM output strobe is followed by a $F8_H$ sync byte.

CURRENT ADDRESS	DATA	DATA TYPE			COUNT	CONTROL		BRANCH CONDITION	NEXT ADDRESS	BRANCH ADDRESS REGISTER
04_H	$6C_H$	1	1	1	11111	0	5	000	01101	16_H

Data: $6C_H$ Format Character
Data Type: Not Valid
Count: 256 (All Eight Bits Of This Register Represent Count As The Data Transfer Bit 6, Register 79_H, Is Set)
Control: Set Data Transfer And Output Bits
Branch Condition: No Branch
Next Address: $0D_H$

SYSTEM ACTIVITY: The AIC-610F NRZ output is the sector data from the host buffer for a fill value of $6C_H$. Host buffer data will be transferred if the Suppress Transfer bit (Register $7A_H$, Bit 5) is off. If this bit is set, then the $6C_H$ fill byte will be output. At this point, the Data Transfer flag (Register 79_H, Bit 6) will be set. Setting Inhibit Data Field Carry will allow another 256-byte transfer (for 512 byte sectors).

CRRENT ADDRESS	DATA	DATA TYPE			COUNT	CONTROL		BRANCH CONDITION	NEXT ADDRESS	BRANCH ADDRESS REGISTER
$0D_H$	00_H	0	1	0	00011	0	0	000	10100	16_H

Data: 00
Data Type: ECC
Count: 4
Control: Reset All Control Bits
Branch Condition: No Branch
Next Address: 14_H

SYSTEM ACTIVITY: AIC-610F NRZ output is Data Field ECC (4 bytes indicate 32-bit ECC). At the end of the ECC, branch is active.

CURRENT ADDRESS	DATA	DATA TYPE			COUNT	CONTROL		BRANCH CONDITION	NEXT ADDRESS	BRANCH ADDRESS REGISTER
14_H	00_H	0	0	0	00001	0	0	000	16_H or 13_H	16_H

Data: 00_H
Data Type: Normal
Count: 2
Control: All Control Bits Are Reset
Branch Condition: No Branch
Next Address: 16_H or 13_H

SYSTEM ACTIVITY: AIC-610F NRZ output is two bytes of 00_H as post data field. If it is not the last sector, the next address is 16_H. If it is the last sector, the address is 13_H.

CURRENT ADDRESS	DATA	DATA TYPE			COUNT	CONTROL		BRANCH CONDITION	NEXT ADDRESS	BRANCH ADDRESS REGISTER
13_H	$4E_H$	0	0	0	00000	0	0	010	10011	16_H

Data: 4E
Data Type: Normal
Count: 1
Control: All Control Bits Reset
Branch Condition: Stop On Index Or Sector
Next Address: 13_H

SYSTEM ACTIVITY: The AIC-610F continues to write $4E_H$ bytes until index is encountered. At index, the AIC-610F stops and track format is complete.

A4 EXTENDED DATA HANDLING OPERATIONS

Variable Sector Size

The AIC-610F has an eight-bit data field length counter. This field is programmable and, by setting the field to any value from 00_H to FF_H, any sector length up to 256 bytes can be written to the drive. Note: The value written in the count field is always one less than the actual sector length.

There are two techniques that may be used for sector sizes from 256 to a full track. The first approach is to use the INHIBIT CARRY bit (Register $7A_H$, Bit 7) in the operation control register. By setting this bit during data transfer before the first count has expired, the AIC-610F will be inhibited from going on to the next sequencer word and another 256 bytes of data will be transferred. The inhibit carry bit will automatically be reset when the counter overflows. By testing this bit, a count of 256 bytes can be transferred. For example, a 532-byte sector can be transferred by setting up the counter with 14_H and enabling the inhibit carry bit twice.

The second method is to use as many sequencer words as necessary to implement the count for the data field. The next address field for the first sequencer word points to a subsequent field and so on. Thus, a 532 byte sector would require three sequencer words: two with a field count of 256 bytes and a third with a field count of 20 bytes (14_H).

Multisector Read Or Write

Multisector reads or writes are accomplished by loading the next address to be found while DATA TRANSFER is active (Register 79_H, Bit 6). In case of an error, the AIC-610F will have to be restarted after restoring the sector address that failed to transfer successfully.

Verify Sector

A verify sector is accomplished by setting the SUPPRESS TRANSFER bit (Register $7A_H$, Bit 5) and then performing the read data command sequence. This will verify that the ECC is good for the data field without generating a $\overline{CLKB}$ (no transfers to or from buffer).

Search Sector Data

A search of the data field is performed by setting the SEARCH OPERATION bit (Register $7A_H$, Bit 4) in the Operation Control register and COMPARE ENABLE in the appropriate Control register field (Registers $A0_H$ through $B7_H$, Bit 1) for the data transfer sequencer word and then entering the read data sequence. The contents of the sector buffer will be compared byte-for-byte with the data read from the disk. The result of this comparison is latched into the Status register (Register 79_H, Bits 0 and 1). Be sure to reset both the Operation Control register (Register $7A_H$, Bit 4) and the appropriate byte in the control field (Bit 1) of the Sequencer RAM after the Search Operation is completed.

A5 AIC-610F INTERRUPTS

Data Transfer Interrupt

The Data Transfer Interrupt indicates the start of the data transfer to or from the buffer memory, or stopped condition. The processor then executes the interrupt routine for the sequence of events resolving the interrupt request.

The Data Transfer Interrupt goes active (low) when the actual data transfer process begins: at the same time Register 79_H, Bit 6, is set. The Data Transfer Interrupt also goes active (low) during stopped conditions, such as the end of a transfer process, or an ID error or miscompare. In other words, at any time Register 79_H, Bit 4, is set.

The Data Transfer Interrupt is deasserted upon the completion of a data transfer or when the sequencer map is started (from a stopped condition).

DMA Done Interrupt

The DMA Done Interrupt occurs when the Read Address Pointer (RAP) = Stop Pointer (SP) during a read cycle or when the Write Address Pointer (WAP) = SP during a write cycle. This interrupt can be used to indicate the completion of DMA transfers to the host and allows the microprocessor to perform the necessary function of updating the SP only if it is required. The interrupt will occur after the REQ and ACK lines are inactive.

The DMA Done Interrupt is deasserted after the pointers are updated in a read or write cycle.

This interrupt can be used to simplify the buffer operations. For example, for a single block read operation, a DMA Done Interrupt indicates the end of the operation meaning the software routine does not have to monitor Bit 5, in Register 53_H, to test the end of the operation.

A6 SPECIAL APPLICATION NOTES

Sequencer RAM

Since the Sequencer RAM is essentially a dual port RAM, when the appropriate control signals (CS, ALE, and Address 7) are present expecting to be accessed on the next cycle. If it is not accessed on the next cycle, the Sequencer RAM contents may be destroyed.

Where the possibility exists for these control signals to be inadvertantly generated, care must be taken to ensure Sequencer RAM data integrity.

For example, in an I8031 microprocessor application with the device residing in external data memory: using CS = 4000_H decode (using address 14 = on) which can be set in Port 2 or in DPTR (DPH) the following issues must be taken into consideration: ALE is activated twice every machine cycle. Address 7 is tristated when not being driven (HIGH).

SOLUTION 1

If using the 16-bit address Mov X @ DPTR format, then set P2 = 00_H and no external hardware modifications are necessary.

SOLUTION 2

If using Port 2.6 for the CS and the 8-bit address Mov X @ Ri instruction format, then it is recommended that a 1K pull-down resistor be used on address 7 and only CMOS devices be used on the A/D Bus.