ZENDEX ZX-200A SINGLE BOARD DISK CONTROLLER

HARDWARE REFERENCE MANUAL

by, Richard Brewster Main

Introduction

I founded Zendex Corporation (Dublin, CA) in 1979 and did all the engineering, manufacturing, marketing, and technical support for the original ZENDEX ZX-200A Single Board Disk Controller. I wrote the original Hardware Reference Manual for it, but I didn't retain any copies of it, nor can I seem to find any to buy or download in 2015. At present, I have four of these boards, and two of the Intel SBC-202 M2FM Double Density Disk Controllers they were intended by me to replace.

Design Goal

The ZENDEX ZX-200A Single Board Disk Controller hardware responds at its Multibus interface exactly like (1) the Intel SBC-202 M2FM Double Density Disk Controller with a base address of 078H, AND, (2) the Intel SBC-201 FM Single Density Disk Controller with a base address of 088H. It can DMA to shared Multibus Memory, and the I/O command/status ports are the same as in MDS-800 and MDS-225 Series-I and Series-II Intellec Development Systems.

The original MDS configuration was to use a port address base of 088H for their SBC-201 FM Single Density Disk Controller and to that system you could add the Intel SBC-202 M2FM Double Density Disk Controller with a base address of 078H. ISIS-II would happily work with both if the hardware was set up this way.

Using the ZX-200A

The ZX-200A expects up to four single-sided 8" diskette drives at its 50-pin edge connector like the Shugart Associates SA-801. That 50-pin edge connector was designed with the exact same pinout configuration as the SA-801, so a flat 5-wire ribbon cable with five 3M type edge connectors (one for the ZX-200A and four for the SA-801's) is all you need. Two SA-851 Double Sided Drives can be used that provide four recording surfaces by connecting the side-select signals in the drives to the drive-select pins. See Section 1.15.3 Spindle Drive System for a discussion of the disk drive interface and edge connector pinout.

All four physical drives (DS1-DS4), respectively as logical FO, F1, F2 or F3 under ISIS-II will read/write double density M2FM format. Only the first two physical drives (DS1-DS2) can be used to read/write the single density FM (IBM 3740) format. Read/write to logical drive F4 under ISIS-II will direct to physical drive DS1 and proceed as single density. Read/write to logical drive F5 under ISIS-II will direct to physical drive DS2 and proceed as single density. ISIS-II will regard this arrangement of logical drives as completely satisfactory.

The ZX-200A Controller simultaneously maps into I/O Ports 78H-7FH and 88H-8FH. The disk operation descriptor is placed in a 7-10 byte block of shared memory called the "Input Output Parameter Block" (IOPB). The host must then output the base address of the IOPB in memory in two port OUT of two-byte hex to a pair of ports on the ZX-200A. That second write to the second port is what triggers the ZX-200A to fetch in the IOPB constructed by the host to perform the operation described by the IOPB. A complete discussion of the Programming Interface is included here starting with Section 2-9 SWITCH AND JUMPER

CONFIGURATIONS, page 2-20, from the Intel SBC 202 Hardware Reference Manual (9800420A_iSBC_202_Hardware_Reference_Sep77).

When the task is complete, the ZX-200A will issue an interrupt signal for the host that result information is available. Performing a port input by the host will return result information over the data bus. The ZX-200A appears to a Multibus system as both an SBC-202 addressed at 078H (system) and an SBC-201 addressed at 88H (add-on).

The ZX-200A uses TTL random logic for the FDD Interface and format control and 8085A CPU for control, a 2716 EPROM (firmware BD-1) and an 8257 DMA Controller.

Controlling a 5.25" Floppy Disk Drive

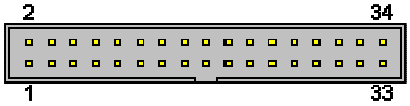
It is entirely doable and maybe advantageous in 2015 to make one of the first two drives, DS1 or DS2, a 5.25" Floppy Disk Drive. Best to use DS2, because CP/M-80 and ISIS-II need DS0 to boot. Unfortunately, the wiring of an adapter will be messy. (Another controller, a Zendex ZX-208 gives you both, a 50-pin for 8" and 34-pin for 5.25").

signal name	SA-800/850 8" drive	SA-400/450 5.25" drive	ZX-200A output/input
ground	all odd numbered pins	all odd numbered pins	n/a
disk change*	12		input
side select	14	32	output
in use*	16	16 (Drive Motor Enable)	output
head load*	18		output
index	20	8	input
ready	22		input
sector (SA851 only)	24		input
DS1	26	10 (NDS0)	output
DS2	28	12 (NDS1)	output
DS3	30	14 (NDS2)	output
DS4	32	6 (NDS3)	
stepper direction	34	18	output
step	36	20	output
write data	38	22	output
write gate	40	24	output
track 00	42	26	input
write protect	44	28	input

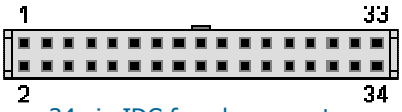
read data	46	30	input
sep-data (SA 851 only)	48		input
sep-clock (SA 851 only)	50		input

*These lines are alternate input/output lines and they are enabled by jumper plugs.

The following is an interesting discussion I found at http://pinouts.ru/Storage/InternalDisk_pinout.shtml about wiring in 3.5" floppies to 5.25" FDD connectors:



34 pin IDC male connector
at the motherboard & diskdrives



34 pin IDC female connector
at the cable

Controller pinout:

Pin	Name	Dir	Description
2	/REDWC	→	Density Select
4	n/c		Reserved
6	n/c		Reserved
8	/INDEX	←	Index
10	/MOTEA	→	Motor Enable A
12	/DRVSB	→	Drive Sel B
14	/DRVSA	→	Drive Sel A
16	/MOTEB	→	Motor Enable B
18	/DIR	→	Direction
20	/STEP	→	Step
22	/WDATE	→	Write Data
24	/WGATE	→	Floppy Write Enable
26	/TRK00	←	Track 0
28	/WPT	←	Write Protect
30	/RDATA	←	Read Data
32	/SIDE1	→	Head Select
34	/DSKCHG	←	Disk Change/Ready

Floppy Diskdrive pinout (Shugart interface):

Pin	Name	Dir	Description
2	/DCD	→	Disk Change Detect
3	Key		no pin in this position
6	/DS3		Device Select 3. Not sure but Amiga 500s schematics reveal that this signal might be used for motor control of internal DF1: on the Amiga 2000
4	/INUSE		A common open-collector LED driver signal? I have never seen this signal used anywhere.
8	/INDEX	←	Index
10	/DS0	→	Device Select 0
12	/DS1	→	Drive Sel B
14	/DS2	→	Device Select 2
16	/MTRON	→	Motor On
18	/DIR	→	Direction
20	/STEP	→	Step
22	/WDATE	→	Write Data
24	/WGATE	→	Floppy Write Enable
26	/TRK00	←	Track 0
28	/WPT	←	Write Protect
30	/RDATA	←	Read Data
32	/SIDE1	→	Head Select
34	/RDY	←	Drive Ready/Disk Changed

On a standard floppy drive there is absolutely no way to remap the motor on signal to another pin with jumpers. Therefore to have independent motor control for two drives the cable must provide this remapping with the traditional seven-conductor twist. If you jumper a floppy drive to work as drive A (unit 0) and connect it to an IBM PC controller with a direct cable, the drive will be selected when the controller tries to turn on the motor of drive A but nothing will happen and the drives motor will rotate when drive Bs motor is turned on.

The original Shugart interface (from which the IBM PC floppy interface is derived a long time ago) doesnt have separate motor on signals for the floppy drives but it does have a total of four device select lines. I have also seen floppy drives that wont turn on their motors unless the according device select signal is driven low. My guess is that this kind of drives strictly follow the original Shugart standard.

Also many synthesizers that have floppy drives use the standard Shugart interface pinout. This is why after replacing a faulty drive many people ask around the Internet why their new floppy drive doesnt work when connected to the synth but works fine on their PC.

The same thing affects the classic Amiga. It uses a very slightly modified Shugart interface pinout at the motherboard (the other /MTRON on pin 4) and a PC drive just doesnt work correctly unless the /DCD is remapped to its original pin. The correctly mapped /DCD is enough for AmigaOS but many trackloaders (X-Copy Pro for example) require the /RDY signal which the drive should set low when the motor rotation has stabilised. This signal does exist on most drives but at worst it requires relocating a soldered SMD jumper on the circuit board.

The floppy cable has 34 wires. There are normally five connectors on the floppy interface cable, although sometimes there are only three. These are grouped into three sets; a single connector plus two pairs of two each (for a standard, five-connector cable) or three single connectors. This how the connectors are used:

	Controller Connector: The single connector on one end of the cable is meant to connect to the floppy disk controller, eit or the motherboard.
	Drive A Connectors: The pair of connectors (or single connector in the case of a three-connector cable) at the opposite intended for the A: floppy drive. This is explained in more detail below.
	Drive B Connectors: The pair of connectors (or single connector in the case of a three-connector cable) in the middle of for the B: floppy drive.

The reason that the standard cable uses pairs of connectors for the drives is for compatibility with different types of drives. 3.5 drives generally use a pin header connector, while 5.25 drives use a card edge connector. Therefore, each position, A and B, has two connectors so that the correct one is available for whatever type of floppy drive being used. Only one of the two connectors in the pair should be used (they're too close together to use both in most cases anyway). The more common pin header (IDC) connector is shown below.

The three-connector cables are found either in very old systems or in ones where the manufacturer was trying to save a few pennies. They reduce the flexibility of the setup; fortunately these cables can be replaced directly by the five-connector type if necessary.

You will also notice that there is an odd twist in the floppy cable, located between the two pairs of connectors intended for the floppy drives. Despite the fact that this appears to be a hack (well, it really is a hack), this is in fact the correct construction of a standard floppy interface cable. There are some cables that do not have the twist, and it is these that are actually non-standard! What the twist does it to change the connection of the drive on the far end of the

twist so that it is different than the drive before the twist. This is done to cause the drive at the end of the cable to appear as A: to the system and the one in the middle to be as B:.

Heres how it works in detail. Traditionally, floppy drives used a drive select (DS) jumper to configure the drive as either A: or B: in the system. Then, special signals were used on the floppy interface to tell the two drives in the system which one the controller was trying to talk to at any given time. The wires that are cross-connected via the twist are signals 10 to 16 (seven wires). Of these, 11, 13, and 15 are grounds and carry no signal, so there are really four signals that are inverted by the twist. The four signals that are inverted are exactly the ones that control drive selection on the interface. Here is what happens when the twisted cable is used:

	Line 10	Line 12	Line 14	Line 16
Controller Signals	Motor Enable A	Drive Select B	Drive Select A	Motor Enable B
Drive Before the Twist Sees	Motor Enable A	Drive Select B	Drive Select A	Motor Enable B
Drive After the Twist Sees	Motor Enable B	Drive Select A	Drive Select B	Motor Enable A

Since the signals are inverted, the drive after the twist responds to commands backwards from the way it should; if it has its drive select jumpers set so that it is an A: device, it responds to B: commands, and vice-versa.

One might ask why the twist was needed. In short, because it was a big time-saver during setup back in the days when it was quite common to find two floppy drives in a machine. Without the twist, for two floppy drives to be used, one had to be jumpered as A: and the other as B:. With the twist, it was possible to leave them both jumpered as B:, and whichever was after the twist will appear to the system as A: because the control lines are inverted. Changing which drive is A: and which is B: is as easy as switching the cable. In systems with only one floppy drive, only the connector after the twist cable should be used. Large manufacturers, therefore, could arrange to have all of their floppy disks configured the same way without having to pull jumpers as the PC was assembled.

In order for this system to work, both drives must be jumpered as B: drives. Since the floppy cable with the twist is standard, this jumpering scheme has become the standard as well. Virtually all floppy disks that you purchase come pre-jumpered as B: drives so that they will work with this setup.

If this whole idea sounds similar to the seldom-used cable select protocol for IDE/ATA hard disks, thats because it is essentially the same thing. IDE/ATA hard disks require you to change

the master/slave jumpers in a similar manner, and cable select was invented to do away with this. The difference is, as usual, just one of inertia and history; the floppy drive system is the standard while cable select never caught on for hard disks.

Some newer BIOSes have taken things a step further. They include a BIOS parameter that will invert the A: and B: signals within the controller itself. When enabled, this lets you reverse whichever drive is A: with the one that is B:, without requiring you to even open the case. Note however that this is not compatible with all operating systems: in particular, both Windows NT and Linux can malfunction with this swap feature set, which can cause serious problems when trying to install the operating system. The reason this happens is that the swap setting only affects the way the BIOS handles the floppy drive, and confuses operating systems that go directly to the hardware.

Apparently, there is yet another floppy cable variant out there, that is used by some manufacturers. In this setup, there are actually two twists in the floppy cable. The drive placed after the first twist, in the middle of the cable, is A:, much as it is with the standard one-twist cable. The drive placed after the second twist is B:. The second twist reverses the effect of the first one and makes the connector at the end of the cable operate the same way a drive that appears before the twist in a regular cable does.

On some mainboards pin 3 is used as the key (missing pin) and on some pin 5 is used as the key pin, while a lot of mainboards dont have the key pin removed at all. This can all cause problems when using cables which have the key pin hole closed. As all odd pins are ground there are no technical implications in modifying such cables by removing the key pin closure by force.

Zendex®

ZX-200A SINGLE BOARD DISKETTE CONTROLLER

- MEDIA COMPATIBLE INTEL SINGLE/ DOUBLE DENSITY FORMATS
- UPGRADE INTELLEC TO DOUBLE DENSITY INTEL FORMAT
- DIRECT SHUGART SA801 INTERFACE
- PERFECT IN COMBINATION WITH ZX-85 FOR AN ECONOMY DEVELOPMENT SYSTEM THAT'S INTEL COMPATIBLE
- REPLACES BOTH SBC-201 (PORT 88H) AND SBC-202 (PORT 78H)
- ISIS-II, RMX COMPATIBLE
- CONTINUE TO USE INTEL SINGLE DENSITY LIBRARY ON HAND
- 5 VOLT ONLY POWER REQUIREMENT
- SECONDED SOURCED BOARD PRODUCT

The Zendex ZX-200 Diskette Controller is a one board solution to running Intel's single and double density media on an Intellec MDS or SBC system. The hardware interface to the computer meets the Multibus specification and the software protocol required of the host allows use of unmodified ISIS-II disk operating software in the MDS or RMX software in SBC systems.

The FDD Interface is pinned alike with Shugart's SA800 Series and this allows the use of inexpensive ribbon connector systems. Up to four diskette drives may be operated over the single FDD Interface.

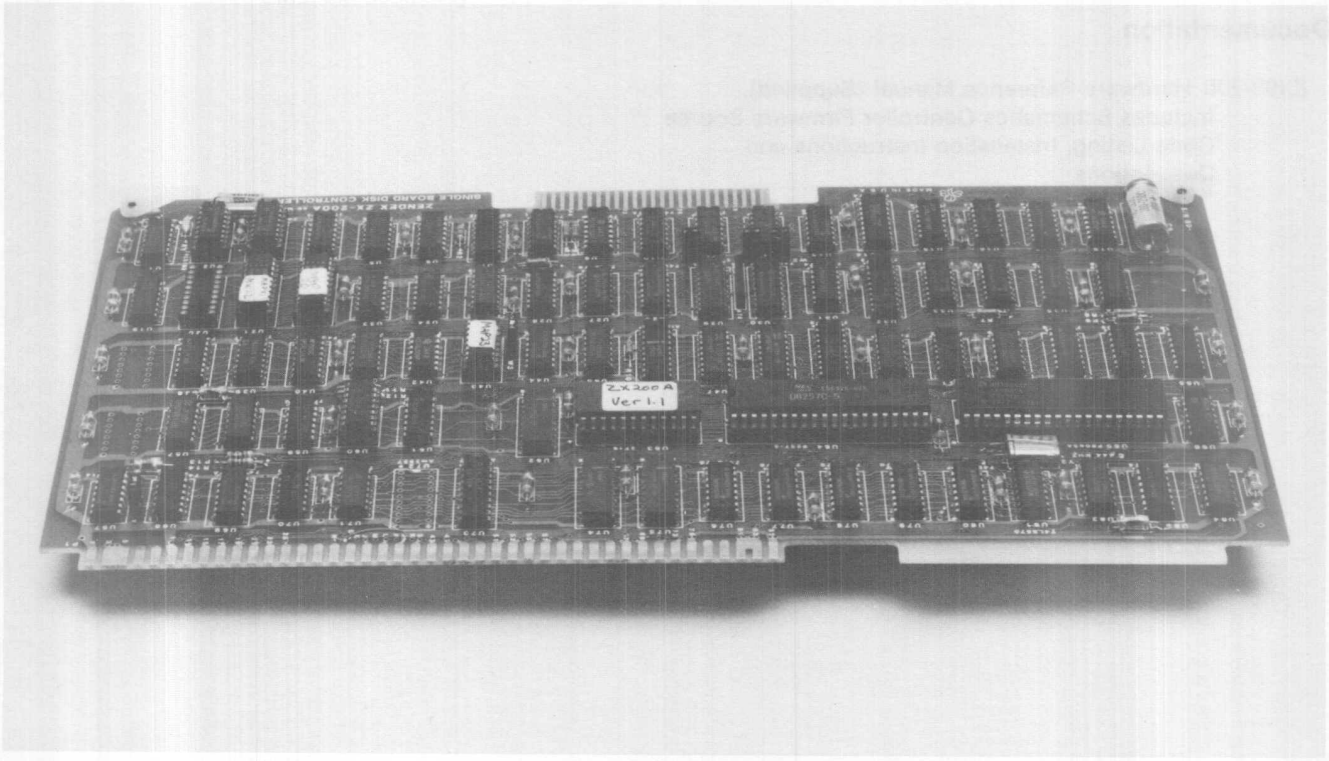


FIG. 1: ZX-200A CONTROLLER BOARD

September 1980

The ZX-200 supports up to four drives, single sided 8" only. All four drives will play double density format while the first two only will play the single density format. The selection, by the host CPU, of which drive is to respond in what density is controlled by the selection of logical device names. Use of F0, F1, F2 or F3 results in double density operation to physical drives 0, 1, 2 and 3, respectively. By using logical device names, F4 and F5, the operation will be single density to physical drives 0 and 1, respectively.

The ZX-200 Controller maps into IO Ports 78H-7FH and 88H-8FH. The disk operation descriptor is placed in a 7-10 byte block of host memory called the IOPB. The host then outputs the address of the IOPB to a pair of ports on the ZX-200. The ZX-200 then fetches the IOPB constructed by the host and performs the operation described in the IOPB. When the task is complete the ZX-200 can issue an interrupt to signal the host that the result information is available. Performing a port input by the host will return result information over the data bus. The ZX-200 appears to a Multibus system as an SBC-202 addressed at 78H (system) and an SBC-201 addressed at 88H (add-on).

The ZX-200 uses TTL random logic for FDD Interface and format control and 8085A CPU for control, a 2716 EPROM and an 8257 DMA Controller.

SPECIFICATIONS

Electrical

Power — 5 Volts @ 1.0 Amps (Typical)
Transfer Rate — 250K, 500K
Data Bus — 8 Bit Parallel
IO Address — 8 Bit Parallel
MEM Addressing — 16 Bit Parallel
Master Modes — Multibus Slave or Master

Connectors

Bus: 86 PIN @ 0.156" Centers (Multibus)
Disk: 50 PIN @ 0.1" Centers (SA800)

Documentation

ZX98-200 Hardware Reference Manual (Supplied), Includes Schematics Controller Firmware Source Code Listing, Installation Instructions and Descriptions

Physical

Height — 6.75 Inches
Width — 12.00 Inches
Thickness — 0.50 Inches (Max)
Weight — 10 Ounces Net, 2 Pounds Shipping

Operating Temp — 0° to 50° C, 5 to 95% R.H.

Ordering Information

Number	Description
ZX-200A	Single Board Flexible Diskette Controller. (Includes Manual.)

1.15.3 Spindle Drive System

The spindle drive system consists of a spindle assembly driven through a drive belt by a brushless D.C. motor/tachometer.

The servo electronics required for speed control are located on the printed circuit board.

The control circuitry contains an interface control line. When the drive motor control interface line is false (high), the drive motor is allowed to come up to speed.

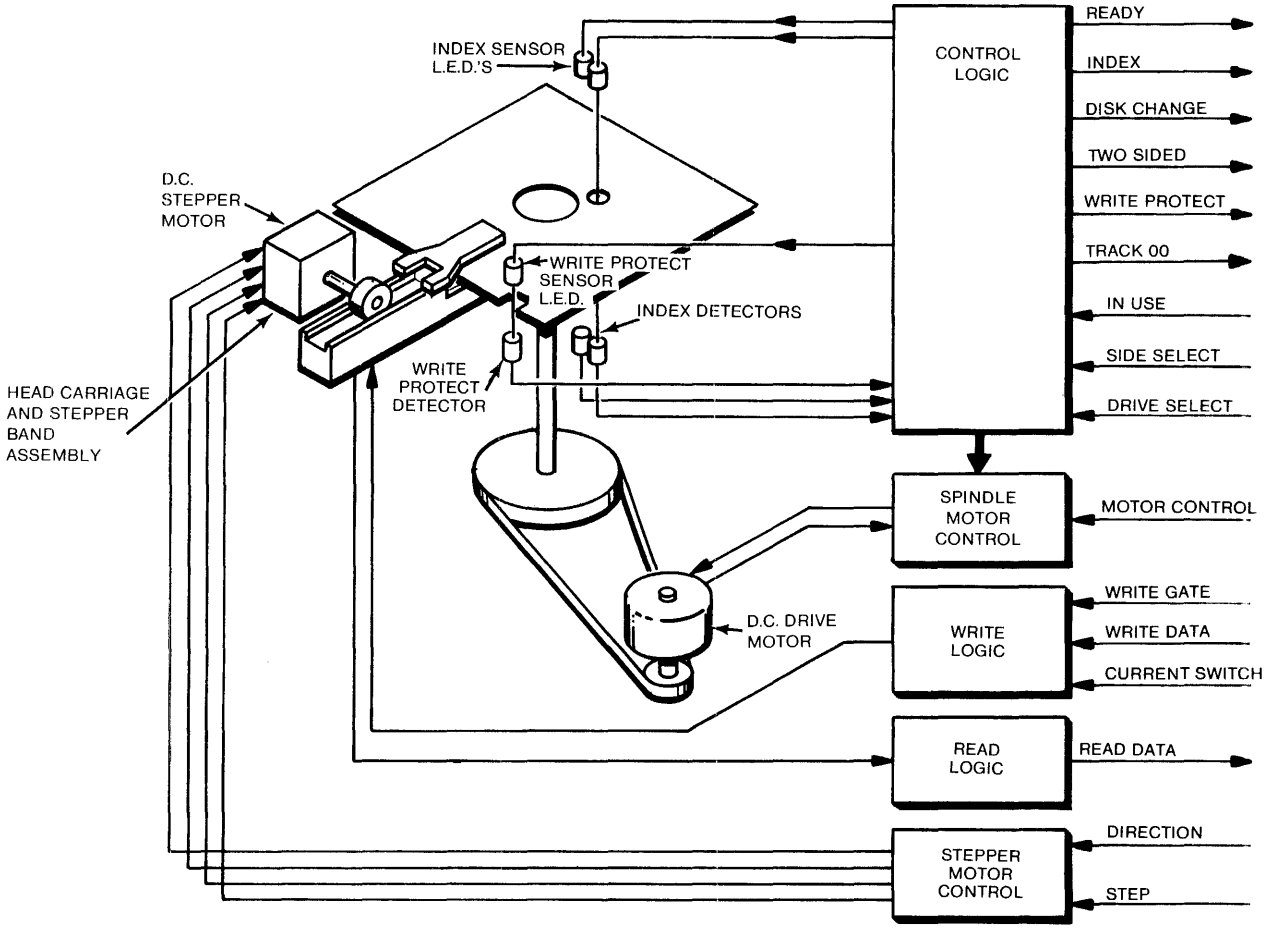


Figure 1-2
TM848 Disk Drive Functional Block Diagram

1.15.4 Positioner Control

The head positioning system uses a bipolar-driven motor drive, which changes one phase for each track advancement of the read/write carriage. In addition to the logic necessary for motor control, a gate is provided that inhibits positioner motion during a write operation.

1.15.5 Data Electronics

Information can be recorded on the diskette by using a double-frequency code. Figure 1-3 illustrates the magnetization profiles in each bit cell for the number sequence shown for FM recording.

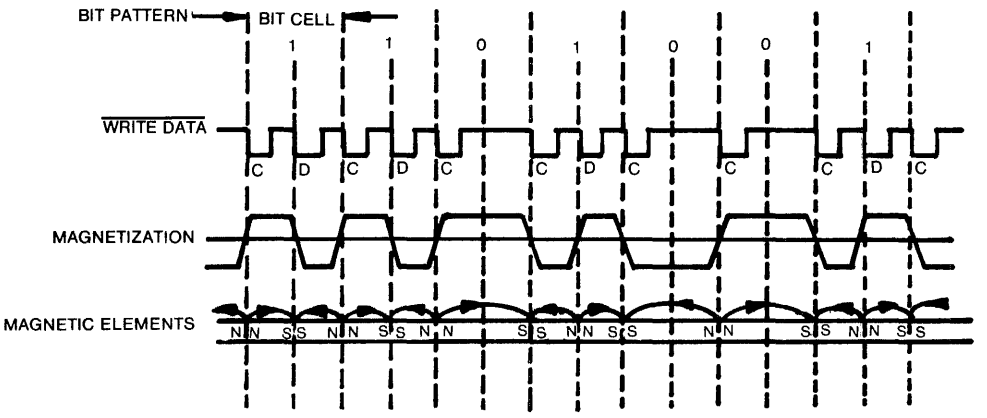


Figure 1-3
FM Recording Magnetization Profiles

The erase gaps provide a guard band on either side of the recorded track.

All signals required to control the data electronics are provided by the user system and are shown in the TM848 drive functional block diagram (see Figure 1-2). These control signals are:

1. Select
2. Write Gate
3. Write Data
4. Side Select
5. Write Current Switch

Composite read data is sent to the user system via the Read Data interface line.

A. Data Recording

The write electronics consist of a switchable write current source, a write waveform generator, an erase current source, the trim erase control logic, and the head selection logic (see Appendix I).

The read/write winding on the head is center-tapped. During a write operation, current from the write current source flows in alternate halves of the winding, under control of the write waveform generator.

The conditions required for recording, i.e. drive ready must be established by the user's system, as follows:

1. Drive speed stabilization occurs 700 milliseconds after the drive motor is started.
2. Subsequent to any read/write operation, the positioner must be allowed to settle. This requires 18 milliseconds maximum after the last step pulse is initiated, i.e., 3 milliseconds for the step motion and 15 milliseconds for settling.
3. The foregoing operations can be overlapped, if required.

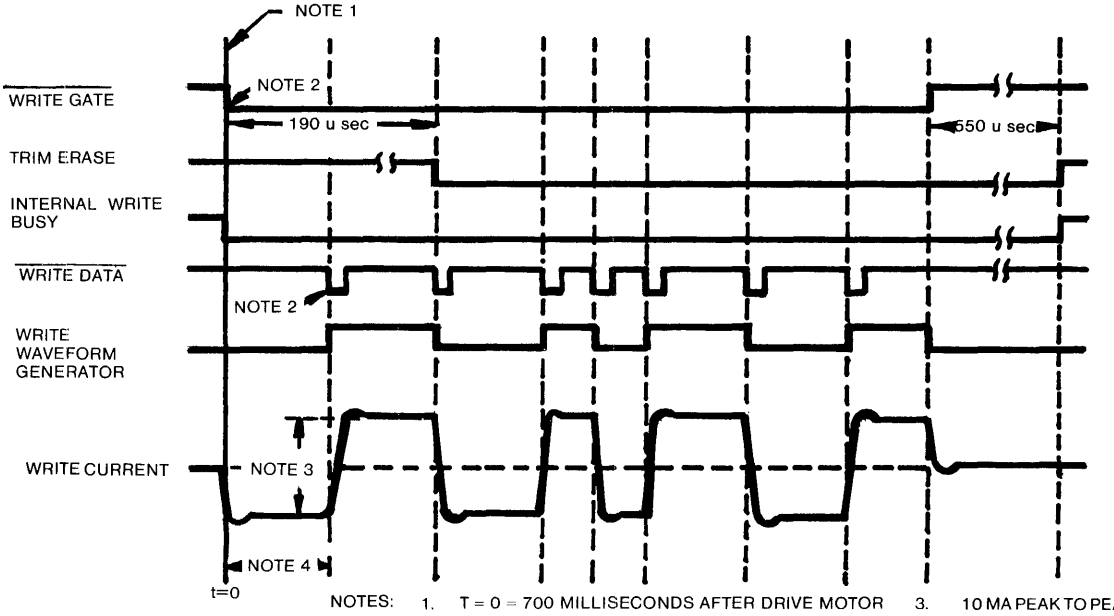
Figure 1-4 illustrates the timing diagram for a write operation. At t = 0, when the unit is ready, the write gate interface line goes true. This enables the write current source. Write current is switched via the write current switch interface line to a lower value by the user's controller at Track 43.

The Trim Erase control goes true 190 microseconds after the Write Enable interface line since the trim erase gaps are behind the read/write gap. It should be noted that this value is optimized between the requirements at Track 00 and at Track 76, so that the effect of the trim erase gaps on previous information is minimized.

Figure 1-4 shows the information on the write data interface line and the output of the write waveform generator, which toggles on the leading edge of every write data pulse.

A maximum of 4 microseconds between write gate going true and the first write data pulse is only required if faithful reproduction of the first write data transition is signifcant.

At the end of recording, at least one additional pulse on the write data line must be inserted after the last significant write data pulse to avoid excessive peak shift effects.



NOTES:
1. T = 0 = 700 MILLISECONDS AFTER DRIVE MOTOR STARTS OR 20 MILLISECONDS AFTER LAST STEP PULSE, WHICHEVER IS THE LATEST TIME.
2. UNSYCHRONIZED
3. 10 MA PEAK TO PEAK, TRACK 0 TO TRACK 42; 7.5 MA PEAK TO PEAK, TRACK 43 TO TRACK 77.
4. 2 USEC MINIMUM
   4 USEC MAXIMUM

Figure 1-4
Write Operation Timing Diagram

The duration of a write operation is from the true going edge of write gate to the false going edge of erase. This is indicated by the internal write busy waveform shown (see Figure 1-4).
The Read electronics consist of:

1. Read Switch/Side Select
2. Read Preamplifier
3. Filter
4. Differentiatior
5. Time Domain Filter and Digitizer

The read switch is used to isolate the read amplifier from the voltage excursion across the head during a write operation. The side select is used to enable one of the read/write/erase heads.

The drive must be in a ready condition before reading can begin. As with the data recording operation, this ready condition must be established by the user system. In addition to the requirements established in this section, a period of 100 microseconds is necessary after a trim erase operation occurs to allow the read amplifier to settle after the transient caused by the read switch returning to the read mode.

The output signal from the read/write head is amplified by a read preamplifier and filtered by a low-pass linear phase filter to remove noise (see Figure 1-5). The linear output from the filter is passed to the differentiator, which generates a wave form whose zero crossovers correspond to the peaks of the Read signal. This signal is then fed to the zero crossing detector and digitizer.

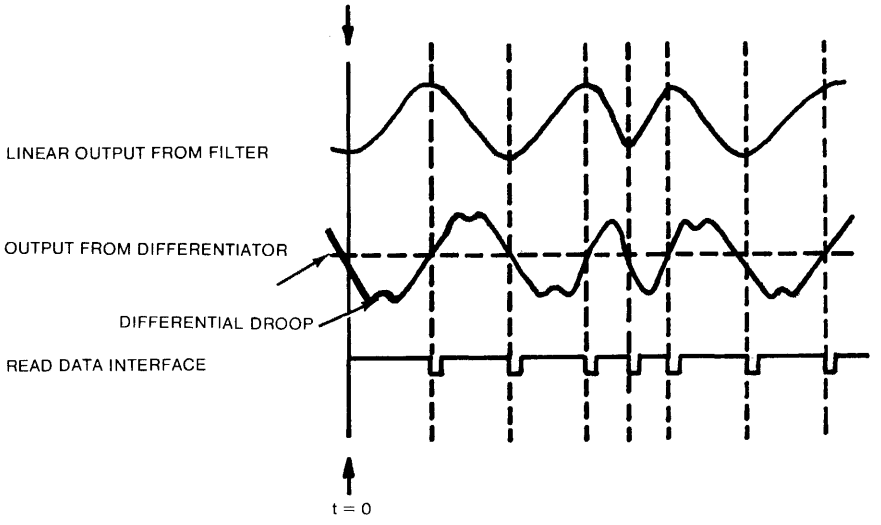


Figure 1-5
Read Timing Diagram

Note

T = 0 is defined as 250 milliseconds after drive motor starts, or 20 milliseconds after a step command, or 100 microseconds after termination of write busy, whichever is the latest time.

The zero crossover detector and digitizer circuitry generate a 200 nanosecond read data pulse, corresponding to each peak of the read signal. The composite read data signal is sent to the user system via the read data interface line.

1.16 INTERFACE ELECTRONICS

All interface signals are TTL compatible. Logic true (low) is +0.4 volt maximum, logic false (high) is +2.4 volts minimum. The maximum interface cable length is ten feet. It is recommended that the interface cable be flat ribbon cable that has a characteristic impedence of 100 ohms.

1.16.1 Interface Connector Pin Assignments, P13

The interface connector pin assignments, P13, are listed in Table 1-3.

1.16.2 Power Connector Pin Assignments

The power connector pin assignments are listed in Table 1-4.

TABLE 1-3

INTERFACE CONNECTOR PIN ASSIGNMENTS

Ground	Pin Number	Signal
1	2	Write Current Switch
3	4	Motor Off Control 1
5	6	Motor Off Control 2
7	8	Motor Off Control 3
9	10	Two Sided (option) (Model TM848-2 only)
11	12	Disk Change (option)
13	14	Side Select (Model TM848-2 only)
15	16	Activity Indicator (option)
17	18	Head Load
19	20	Index
21	22	Ready
23	24	Motor Off Control 4
25	26	Drive Select 1 (Side Select Option, TM848-2 only)
27	28	Drive Select 2 (Side Select Option, TM848-2 only)
29	30	Drive Select 3 (Side Select Option, TM848-2 only)
31	32	Drive Select 4 (Side Select Option, TM848-2 only)
33	34	Direction Select (Side Select Option, TM848-2 only)
35	36	Step
37	38	Write Data
39	40	Write Gate
41	42	Track 00
43	44	Write Protect
45	46	Read Data
47	48	Alternate I/O
49	50	Alternate I/O

TABLE 1-4

POWER CONNECTOR PIN ASSIGNMENTS

Pin	Supply Voltage
1	24V D. C.
6	Return
3	Return
2	Return
5	5V D. C.

1.17 TERMINATED LINES

1.17.1 Input Line Terminations From Removable Resistor Pack

The drive has the capability of terminating the following input lines:

1. Write Current Switch
2. Write Data
3. Write Gate
4. Side Select (TM848-2 only)
5. Direction
6. Step
7. Head Load

These input lines are individually terminated through a 150 ohm resistor pack that is installed in the dip socket located at integrated circuit location RP1. In a single-drive system, this resistor pack should be installed to provide the proper terminations. In a multiple-drive system, only the last drive on the interface is to be terminated. All other drives on the interface must have the resistor pack removed (see Figure 1-6).

1.17.2 Drive Select

The Select lines provide a means of selecting and deselecting a drive. These four lines -- DS1 through DS4--allow independent selection of up to four drives attached to the controller.

When the signal logic level is true (low), the drive electronics are activated and the drive is conditioned to respond to Step or to Read/Write commands. When the signal logic level is false (high), the input control lines and the output status lines are disabled.

The drive select address is determined by a movable shorting plug installed on the circuit board. Select lines one through four provide a means of daisy chaining a maximum of four drives to a controller. Only one line can be true (low) at a time. An undefined operation might result if two or more units are assigned the same address or if two or more select lines are in the true (low) state simultaneously (see Figure 1-7). A select line must remain stable in the true (low) state until the execution if a Step or Read/Write command is completed.

1.17.3 Program Shunt

The program shunt is AMP Part Number 435704-8. The program shunt positions are programmed by cutting the particular shunt. The program shunt is installed in a dip socket. At the user's option, the program shunt may be removed and replaced by a dip switch. Pins 8 and 9 of the program shunt are not used. See Table 1-5 for a listing of the program shunts.

2-9. SWITCH AND JUMPER CONFIGURATIONS

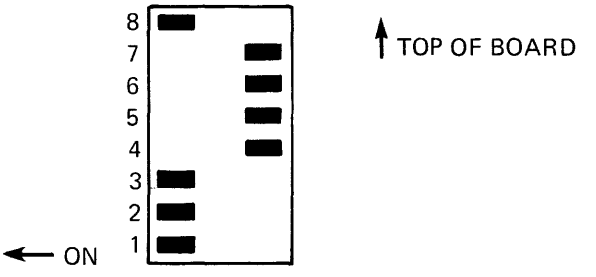
The following paragraphs provide instructions for configuring the I/O Base Address switch and the Interrupt Level Select switch. The memory base address, which is under program control, is described in paragraph 4-4.

2-10. I/O BASE ADDRESS SELECTION

The user must assign a base address to the Diskette Channel. The base address is defined by the five most significant bits of the eight-bit I/O port address. The three least significant bits, then, can be used to differentiate between eight input or eight output channel commands. When the CPU accesses the Diskette Channel by executing an I/O instruction the base address (BASE) is used to select the Diskette Channel, while the three low-order address bits select one of the channel commands, as described in Chapter 3.

A base address is assigned by opening or closing the five most significant switch positions of the S1 switch (S1 -4, 5, 6, 7, 8) on the Channel Board (see sheet 1 of the Channel Board schematic in Chapter 5). When a switch position is closed (on) (tied to ground) it represents the assignment of a logical 0 address bit. When a position is open (off) (+5V), it represents a logical 1 selection.

The following sketch represents a base address selection of 78_{16}.

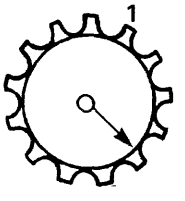


2-11. INTERRUPT LEVEL SELECTION

The user can assign the Diskette Channel's interrupt request line to any one of eight interrupt priority levels (INT0/-INT7/) by moving the interrupt level select switch (S1) on the Interface Board to the desired position. This eight position rotary switch is shown on sheet 3 of the Interface Board schematic in Chapter 5. The eight switch positions are associated with the following priority levels:

SWITCH POSITION	INTERRUPT PRIORITY LINE	RELATIVE PRIORITY (INTELLEC MDS SYSTEM)
1	INT0/	HIGHEST
2	INT1/	
3	INT2/	
4	INT3/	
5	INT4/	
6	INT5/	
7	INT6/	
8	INT7/	LOWEST

The following sketch shows the switch settting 3 corresponding to priority line INT2/.



CHAPTER 3
PROGRAMMING INFORMATION

3-1. INTRODUCTION

All operations must be initiated by the Central Processor Unit (CPU). Once initiated the controller completes the specified operation without further intervention on the part of the CPU. From the CPU's point of view, there are only three general steps required to complete any diskette operation.

- The CPU must prepare and store in system memory an I/O Parameter Block (IOPB) for each operation to be performed. An IOPB (seven bytes) specifies a particular diskette operation and provides all of the parameters required for execution of that operation.
- The CPU must then pass the memory address of the IOPB to the Controller Channel.
- The CPU must process the result information from the Controller Channel upon completion of the operation(s).

The following paragraphs define the system operation.

The 7-byte parameter block (IDPB) must adapt the following format:

Byte 1 Channel Command
2 Diskette Instruction
3 Number of Records
4 Track Address
5 Sector Address
6 Buffer Address (Lower)
7 Buffer Address (Upper)

The preparation of the IOPB by the CPU, in itself, requires no interaction with the Controller Diskette Channel. The passing of the memory address for the IOPB and the result processing, however, do require interaction. Six channel commands have been defined to allow the CPU to perform these interactive steps. Three of the channel commands are the result of the CPU executing an output instruction to a dedicated I/O port address, while the other three commands are the result of input instructions to dedicated ports. The six channel commands are:

(1) Write memory address lower (output)
(2) Write memory address upper and start the diskette operation (output)
(3) Reset the channel (output)
(4) Read subsystem status (input)
(5) Read result type (input)
(6) Read result byte (input)

The channel command provides the Controller with information which:

(1) Determines the method of assigning logical sector addresses.
(2) Enables or disables a series of possible diskette interrupts.
(3) Determines the length of the data word to be transferred.

The CPU outputs the memory address of the IOPB by executing channel commands 1 and 2. Upon execution of channel command 2, the Controller Channel will request master control of the Multibus, fetch the diskette instruction and associated parameters from the IOPB, and proceed to perform the specified diskette operation. The diskette instruction byte in the IOPB can specify any one of seven diskette operations:

(1) Recalibrate (seek track 00)
(2) Seek
(3) Format a track
(4) Write data (with data address marks)
(5) Write data (with deleted address marks)
(6) Read data
(7) Verify CRC

The Controller Channel can interrupt the CPU when the operation is completed or when the diskette ready status changes. The host system software can implement its CPU interrupt mechanism via this direct interrupt feature or it can "poll" the Controller Channel by executing channel command 4 (read subsystem status). When the CPU determines that the operation sequence has been completed (either by receiving an interrupt request or by reading the interrupt status), the CPU should execute channel commands 5 and 6 (read result type and read result byte) to determine whether the diskette operations were successfully completed, and if not which type of error occurred.

Thus, in summary, we see that certain channel commands are executed by the CPU to point the Controller Channel to an IOPB in system memory, and initiate the operation sequence. The Controller Channel, then, accesses the IOPB to perform the diskette operation specified by the instruction byte of the IOPB. The Controller Channel will, if enabled by the IOPB, generate an I/O complete interrupt request upon completion of each diskette operation or detection of an error. The CPU, then, executes other channel commands to determine the result of the diskette operation.

3-2. CHANNEL COMMANDS

There are six channel commands to which the Controller Channel will respond. Three of the channel commands are issued when a CPU in the system executes output (I/O write) instructions with the appropriate eight-bit I/O addresses. The other three commands are issued when the CPU executes input (I/O read) instructions with the appropriate I/O addresses.

When the CPU executes one of the output channel commands, it activates the I/O write (IOWC/) line and duplicates the appropriate 8-bit I/O address on address lines ADR0/-ADR7/ and ADR8/ - ADRF/ of the System bus. Depending on the particular channel command, the CPU may also place relevant data on data lines DAT0/-DAT7/ of the System bus. The CPU maintains the data lines until the Controller Channel returns the transfer acknowledge (XACK/) signal.

When the CPU executes one of the input channel commands, it activates the I/O read (IORC/) line and duplicates the appropriate I/O address on both halves of the System bus. The CPU expects the Controller Channel to activate the transfer knowledge (XACK/) line when it has placed the requested data on data lines DAT0/-DAT7/.

The Controller Channel differentiates between the different channel commands by interrogating the I/O read (IOCR/) and I/O write (IOWC/) lines and the three least significant address lines (ADR0/ - ADR2/). The five most significant I/O address lines (ADR3/ - ADR7/) define the switch-selectable BASE address for the Controller Channel.

If the Controller Channel is not busy, it will respond to an output channel command within 3 microseconds. It is is busy, the "write MA lower" and "write MA upper" commands are ignored; no acknowledge is returned. (Note: Because no acknowledge is returned in this case, it could be possible to "hang up" the host system if the system does not include a Fail Safe time-out provision, as is provided on the Front Panel Control Module in the system). The "reset" command, however, is acknowledged even if the Controller Channel is busy. "Reset" is executed immediately (if issued during a data write operation, garbled data will be written).

The Diskette Controller responds to "read subsystem status" and "read result type" input channel commands within 1 microsecond. The information returned in response to a "read subsystem status" command is always valid. The eight bits of data returned in response to a "read result type" command, however, are only valid if the Controller Channel had previously issued an interrupt request to the CPU. The Controller Channel will, if not busy, respond to a "read result byte" input command within 3 microseconds. If the Controller Channel is busy, however, it ignores the "read result byte" command (i.e., no acknowledge is returned). The "read result type" and "read result byte" commands must be executed sequentially ("read result type" first), and should be executed only in response to an interrupt request from the Controller Channel; execution at other times could produce erroneous result data.

The use and format of each of the six channel commands is described below:

WRITE MEMORY ADDRESS LOWER (OUTPUT)

This channel command outputs the low order byte of the 16-bit memory address that points to byte 1 ("channel word") of the IOPB.

System address bus: BASE + 1

System data bus: Eight least significant bits of the 16-bit memory address that points to the first IOPB.

WRITE MEMORY ADDRESS UPPER AND START THE DISKETTE OPERATION (OUTPUT)

This channel command outputs the high order byte of the 16-bit memory address that points to byte 1 of the IOPB. This command also causes the Controller Channel to begin executing the diskette operation specified in byte 2 (instruction byte) of the addressed IOPB.

System address bus BASE + 2

System data bus: Eight most significant bits of the 16-bit memory address

RESET DISKETTE SYSTEM (OUTPUT)

This output channel command causes all control logic in the Controller Channel to be reset in an initialized state. If this command is issued while a "write data" diskette operation is in progress, the data in the sector currently being written will be garbled. This command is intended to clear a "hang up" in the Controller Channel.

System address bus: BASE + 7

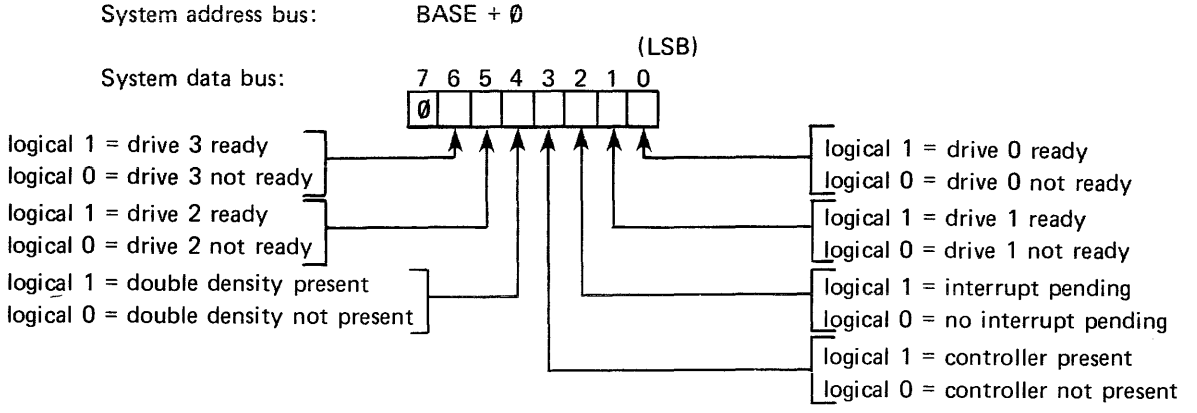
System data bus: Not used.

READ SUBSYSTEM STATUS (INPUT)

This input channel command causes the Controller Channel to return.

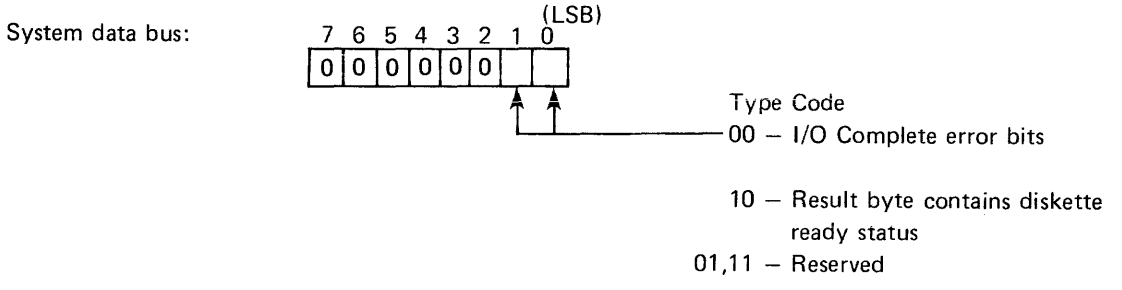
bit 0 — ready status of drive 0
bit 1 — ready status of drive 1
bit 2 — state of the channel's interrupt flip-flop
bit 3 — controller presence indicator
bit 4 — double density controller presence indicator
bit 5 — ready status of drive 2
bit 6 — ready status of drive 3

These indications allow the operating system to monitor the operation of the Controller Channel.



READ RESULT TYPE (INPUT)

This input channel command causes the Controller Channel to return eight bits of information to the CPU. The two least significant bits specify one of four different types of result byte (see next paragraph) associated with diskette operations.

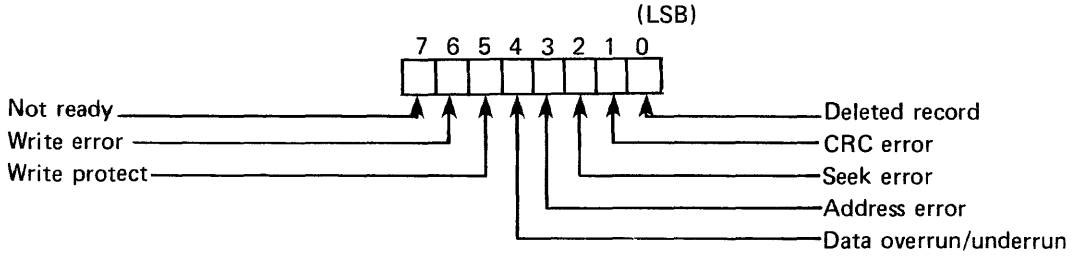


READ RESULT BYTE (INPUT)

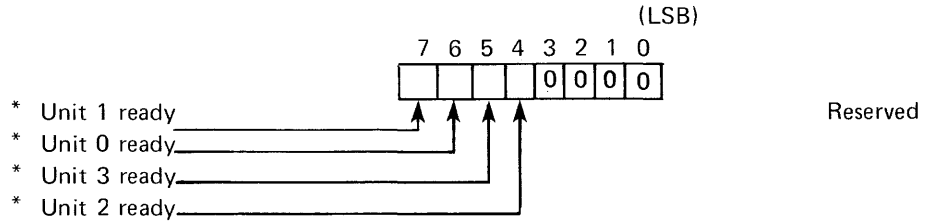
This input channel command causes the Controller Channel to return eight bits of information to the CPU. The interpretation of these bits is dependent upon the type code returned in the result type word (see previous paragraph). The "read result byte" channel command should only be executed after a "read result type" command has been executed.

System address bus: BASE + 3

System data bus: If the type code in the result type word = 00, the result byte, input on the data bus, will contain error bits (see Paragraph 3-5 for error explanations) and will be formatted as follows:



If the type code = 10, the controller has detected a change in the ready status of a drive and the contents of the result byte will indicate the current ready status of the diskette drives:



*NOTE: A logical 1 means that the drive is currently ready; a logical 0 means the drive is not ready. It is the responsibility of the host system software to maintain appropriate tables to track these status changes. There is one instance in which a drive can appear "not ready" to the host system, when in fact it is ready. For example, assume that while drive 0 is selected, drive 1 just goes not ready then returns to the ready state (perhaps the diskette platter was changed). When the drive 0 operation is completed, the diskette controller will return two consecutive status change interrupts, the first showing drive 1 not ready, the second showing drive 1 ready. The first interrupt, indicating drive 1 to be not ready, is returned even though the drive is now actually ready because it is important that the operator know that the ready status of the drive changed while the other drive was selected. For instance, this would protect against inadvertently accessing an "unknown" disk, if the drive went not ready then ready again because someone changed disk platters.

3-3. DISKETTE OPERATIONS

The Diskette System is capable of performing seven different operations: recalibrate, seek, format track, write data (with data marks), write data (with deleted data marks), read data, and verify CRC. To initiate any diskette operation, the CPU will output both bytes of the 16-bit memory address that points to the first byte of an I/O Parameter Block (IOPB). The second byte in the IOPB specifies one of the seven diskette operations (see Paragraph 3-4 for IOPB format). After the Diskette Controller receives the upper byte of the 16-bit memory address, it accesses the IOPB to determine the operation to be performed and to acquire the various parameters that are necessary for execution of the diskette instruction. The Diskette System will perform the specified operation, then set its interrupt flip-flop.

NOTE: The Diskette Channel automatically unloads the read/write head after a fixed length of time following a diskette operation. This feature is meant to reduce head wear. The feature is implemented by counting index pulses after a "read result byte" channel command is executed. When the specified count is achieved, the head is unloaded, and the count is reinitialized. At present, the count is set for 6; that is, the head will remain loaded for at least five complete revolutions following each diskette operation or group of linked diskette operations.

The seven diskette operations are defined in the following paragraphs:

RECALIBRATE

This operation causes the head of the selected diskette unit to be moved over track 00. The diskette drive's track 0 sensor is sampled to determine successful completion of this operation. This is often the first instruction executed after a diskette is loaded, or when a seek error occurs (see Paragraph 3-5).

SEEK

This operation causes the head of the selected diskette unit to be moved over the track specified in byte 4 of the IOPB. The Diskette Channel will verify the head position by reading the track address from the diskette platter before completing the operation. If at the completion of the head movement, the head is not over the expected track, a "seek error" will be indicated (see Paragraph 3-5).

FORMAT TRACK

This operation initializes the track specified in byte 4 of the IOPB, by writing all address marks, gaps, address fields and data fields, as shown in Figure 3-1.

The method of assigning logical sector addresses, which are written into the sector address fields, is specified by bit 6 of the first IOPB byte (the channel word). If this bit is equal to logical 0 the sequence of logical sector addresses will match the physical sequence on the diskette (i.e., sector address "01" is written into the first physical sector after the index mark, sector address "02" is written into the second physical sector, and so on). In addition, the data byte stored in the memory location specified by the 16-bit buffer address contained in bytes 6 and 7 of the IOPB will be written into the 128-byte locations of each sector's data field. No other data bytes need to be stored in this buffer.

If, on the other hand, the sequence of logical addresses being assigned to the sectors is "random" (that is, do not match the physical sequence of sectors), bit 6 of the channel word will be equal to logical 1, and 104 bytes (52 pairs) of data will be stored in memory beginning at the 16-bit buffer address contained in bytes 6 and 7 of the IOPB. Each of the 52 pairs of

data bytes will specify the logical sector address to be written into the sector address field of the corresponding physical sector, and the data character which will be written (128 times) into the data field portion of that sector. For example, if the first four bytes of the buffer are:

Byte	Contents (hex)
1	01
2	FF
3	0E
4	00
.	.
.	.

Then, sector address "01" will be written into the sector address field of the first physical sector after the index mark, and "FF_{16}" (all ones) will be written into each of the 128 byte locations in the data field portion of this sector. The sector address "$0E_{16}$" (14_{10}) will be written into the sector address field of the second physical sector (i.e., the sector which is physically next to the first sector), and "00_{16}" (all zeros) will be written into each of the 128 byte locations in the data field portion of this sector. And so on, until a logical sector address has been written into the sector address field of each of the 52 physical sectors on the track, and a data byte is written into each of the 128 byte locations in the data field portion of each of the 52 sectors.

The firmware implementation of the format command is such that in order to format track n ($n \neq 0$), track n-1 must already be formatted (i.e., already have readable address information written into it). Track 0 can always be formatted even if no valid address information is written on the disk.

During formatting, a "data mark" (i.e., a character which has a clock pattern equal to 70_{16} and a data pattern equal to $0B_{16}$; see Figure 3-2) is written into the "data/deleted data address mark" character position of each sector (i.e., the character position immediately preceding the 128 byte data field.

If, when the format track operation is initiated, the head is not already positioned over the track specified in byte 4 of the IOPB, the format track instruction will cause the head to move (seek) to the proper track before the actual formatting begins.

WRITE DATA

This operation transfers N x 128 bytes of contiguous data from memory to the diskette. N represents the number of sectors to be written. N is specified by the contents of byte 3 of the IOPB. The 16-bit buffer address stored in bytes 6 and 7 of the IOPB specifies the memory location containing the first data byte to be transferred. The contents of bytes 4 and 5 of the IOPB (track and sector addresses, respectively) specify the logical address of the first sector to be written into.

Each 128 byte data field will be preceded by a "data" address mark (see Figure 3-2) that is used for synchronization. Two bytes (16 bits) of CRC check bits will be generated and written after each data field; the CRC bytes are generated from the address mark, as well as the 128 data bytes.

A multi-sector operation (i.e., $N \geq 2$) may begin at any sector, but must not go beyond the last logical sector on a track (sector 52).

If the head is not already positioned over the track specified in byte 4 of the IOPB, the write data instruction will cause the head to move (seek) to the proper track before the actual writing begins.

WRITE "DELETED" DATA

This operation is identical to the WRITE DATA operation, described above, except that each 128 byte data field is preceded by a "deleted data" address mark, shown in Figure 3-3.

READ DATA

This operation transfers N sectors of data (128 bytes per sector) from diskette to memory. N is specified by the contents of byte 3 of the IOPB. The contents of bytes 4 and 5 of the IOPB (track and sector addresses, respectively) specify the logical address of the first sector to be read. The 16-bit buffer address stored in bytes 6 and 7 of the IOPB specifies the memory location into which the first data byte will be written.

Two bytes of CRC check bits will be generated as each sector is being read. When the "data" address marks and all 128 data bytes of a sector have been read, the generated CRC bits are compared with the 16 CRC bits previously written. If there is a mismatch, a CRC error is indicated (see Paragraph 3-5).

A multi-sector operation (i.e., $N \geq 2$) may begin at any sector, but must not go beyond the last logical sector on a track (sector 52).

If the head is not already positioned over the track specified in byte 4 of the IOPB, the read data instruction will cause the head to move (seek) to the proper track before the actual data reading begins.

VERIFY CRC

This operation is identical to the READ DATA operation, described above, except that no data is transferred to memory.

3-4. I/O PARAMETER BLOCK

The CPU in the system initiates a diskette operation by outputting a 16-bit address that points to the beginning (the channel word) of the I/O Parameter Block (IOPB) in system memory. The Diskette Channel then accesses the IOPB. An IOPB specifies one of the diskette operations (see Paragraph 3-3) and provides all of the parameters required for the completion of that operation. An IOPB consists of seven bytes, as shown in Table 3-1.

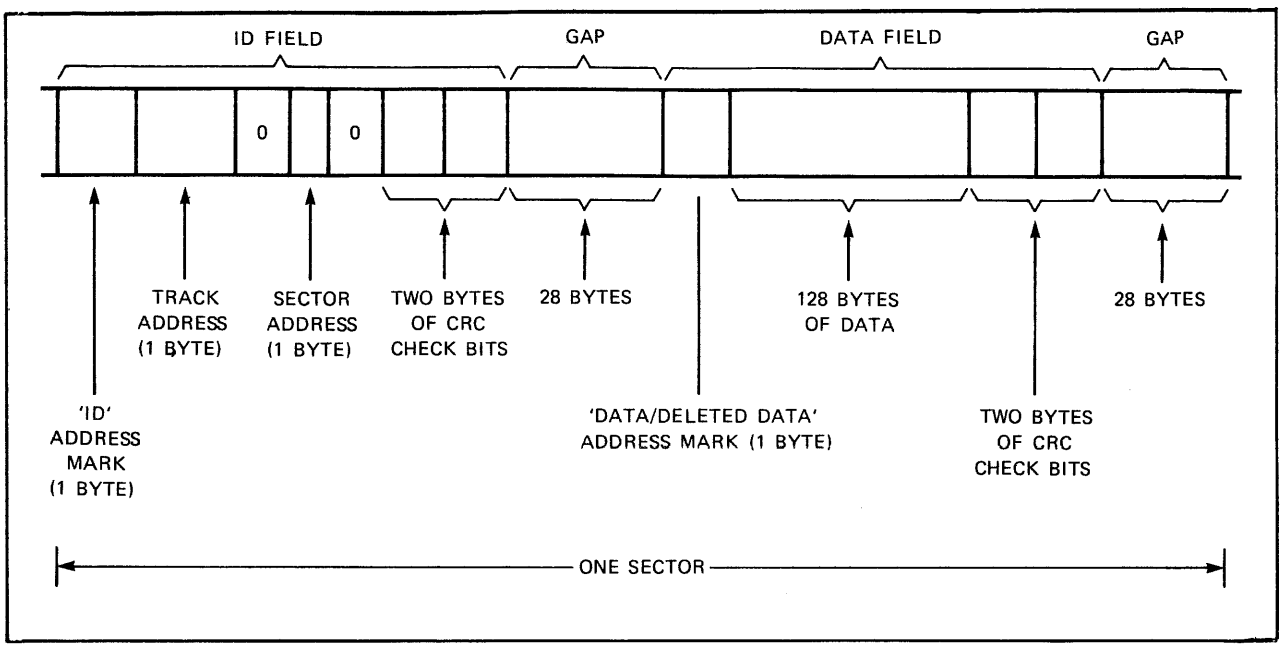


Figure 3-1. Sector Format

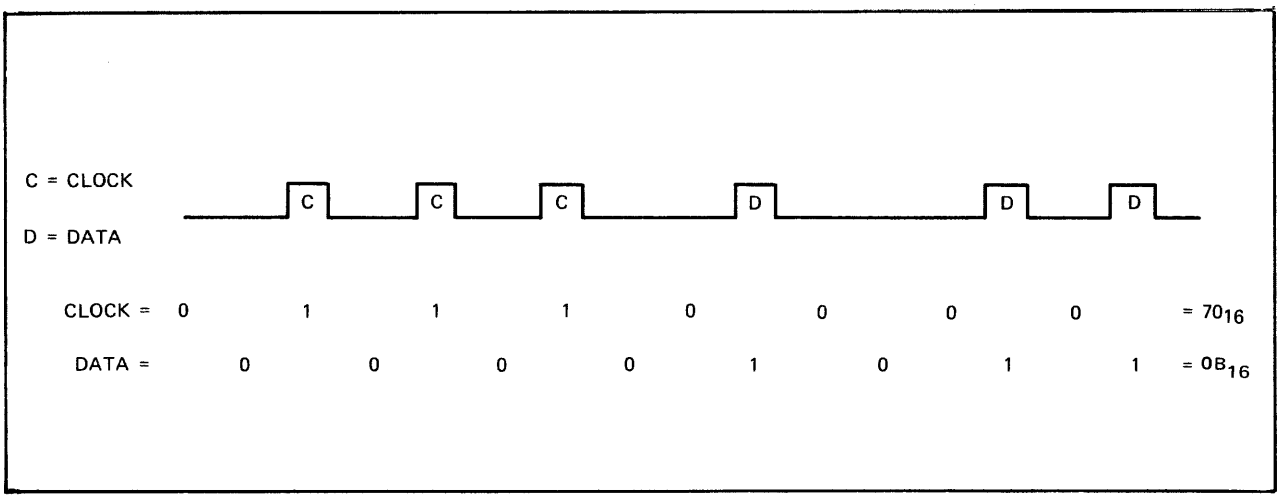


Figure 3-2. 'DATA' Address Mark

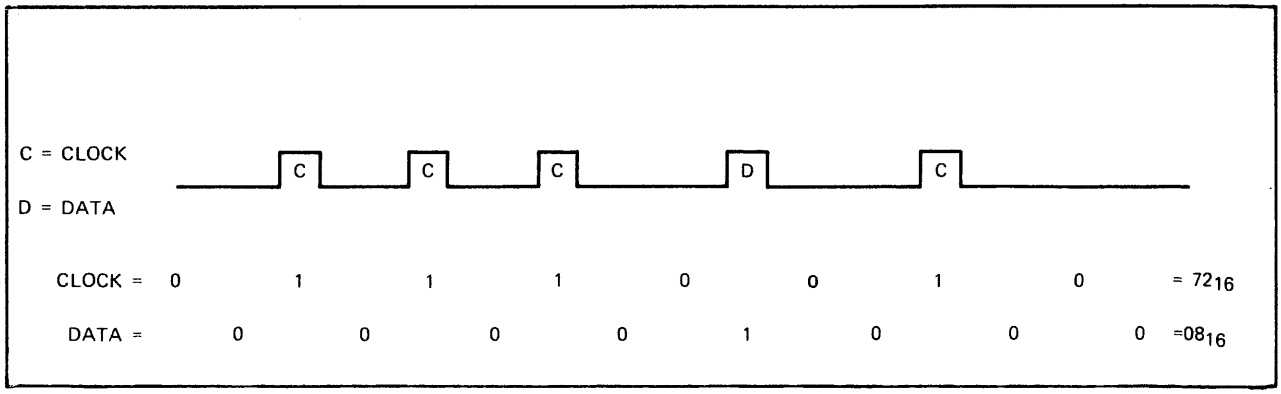
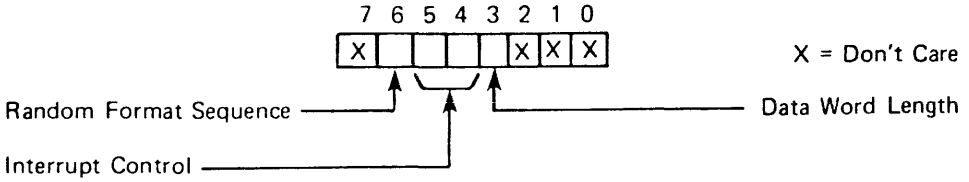


Figure 3-3. 'DELETED DATA' Address Mark

Byte 1. Channel Word

This byte contains channel control information to be used by the Diskette System. Bit assignments in this byte are as follows:



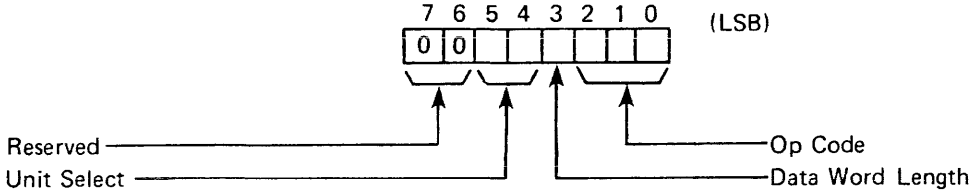
The "random format sequence" bit (6) specifies the method of assigning logical sector addresses when formatting a track. If this bit is reset (logical 0), sector addresses are assigned in sequential order. If this bit is set (logical 1), sector addresses are assigned in random order according to the pattern listed in the 52 byte memory buffer, which begins at the location addressed by the contents of IOPB bytes 6 and 7. (Refer to the description of the FORMAT TRACK operation in Paragraph 3-3.)

The "interrupt control" bits (4 and 5) enable or disable Controller Channel interrupts according to the scheme shown in Table 3-2.

The "data word length" bit (3) must be reset (logical 0) when the Diskette Controller is being used with 8-bit systems, or set (logical 1) when being used with 16-bit systems. This bit must be logical 0 when being used with the SBC 80 system (an 8-bit system).

Byte 2. Diskette Instruction

This byte specifies the diskette operation to be performed and identifies the diskette unit to be used:



The "unit select" bits (4-5) specify the drive address as follows:

00 = drive 0
01 = drive 1
10 = drive 2
11 = drive 3

The "data word length" must contain the same value as the corresponding bit in the channel word (byte 1).

The "op code" bits (0-2) specify one of the seven diskette operations (refer to Section 3-3):

BIT: 3	2	1	OPERATION
0	0	0	No operation
0	0	1	Seek
0	1	0	Format Track
0	1	1	Recalibrate
1	0	0	Read data
1	0	1	Verify CRC
1	1	0	Write data
1	1	1	Write 'Deleted' Data

Table 3-1. I/O Parameter Block (IOPB) Format

BYTE	IOPB FORMAT
*1	Channel Word
2	Diskette Instruction
3	Number of Records
4	Track Address
5	Sector Address
6	Buffer Address (Lower)
7	Buffer Address (Upper)

*The 16-bit address output to the Controller Channel by the two 'Write MA' channel Commands points to the first byte of an IOPB.

Table 3-2. Interrupt Control Bits

BIT: 5	4	FUNCTION
0	0	I/O complete interrupt request to be issued (a) upon completion of diskette operation, (b) upon detection of an error in any operation.
0	1	All I/O complete interrupts are disabled.
1	1	Illegal code
1	0	

NOTE: The interrupt control bits do not affect interrupt requests which are issued as the result of a change in diskette ready status.

Byte 3. Number of Records

This binary number specifies the number of sectors to be transferred. Multi-sector operations are allowed, but they must not go beyond the last sector on a track (sector 52); that is, an address error (see Paragraph 3-5) will be indicated if (starting sector address) + (number of records) × 52_{10}. Therefore, the maximum block transfer is 52 sectors (from sector 1 to sector 52).

Byte 4. Track Address

This binary number identifies the track. Acceptable values are 0 to $4C_{16}$ (76_{10}), inclusive.

Byte 5. Sector Address

Bits 5 through 0 of this byte contain a binary number which specifies the first sector to be accessed during transfer operations. Acceptable values are 1 to 34_{16} (52_{10}), inclusive. Bits 6 and 7 are not used.

Byte 6. Buffer Address (Lower)

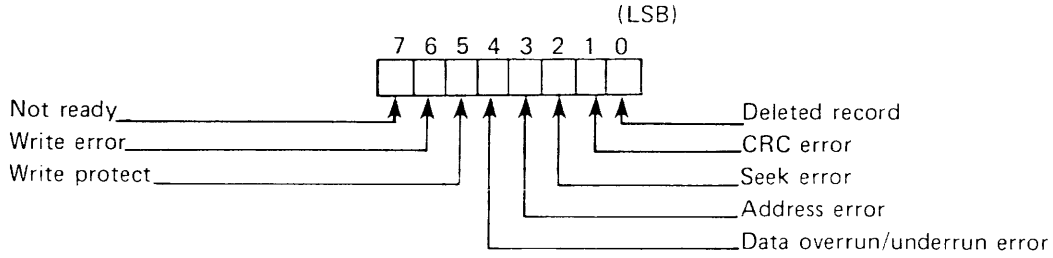
This byte contains the eight least significant bits of the 16-bit buffer memory address.

Byte 7. Buffer Address (Upper)

This byte contains the eight most significant bits of the 16-bit buffer memory address. Bytes 6 and 7 together contain the 16-bit address of the first word of the buffer in system memory. During read data operations, the data from the diskette is transferred to the buffer. During write operations, data from the buffer is written to diskette. During format track operations, the address assignment pattern and/or the data field "format characters" are stored in the buffer.

3-5. ERROR INDICATIONS

If the CPU executes a "read result byte" channel command (in response to a "read result type" channel command which returned a code of 00), the Diskette Channel will return the following result word on the system data bus:



The bits are defined as follows:

NOT READY. This bit (7) indicates that the selected unit was not ready or that the selected unit changed to a not ready state during an operation.

WRITE ERROR. This bit (6) indicates that, during a write operation, a condition existed which precluded data integrity. This error is detected by the drive and monitored by the Controller Channel. An example of a condition that could cause this error is an attempt to write through an unloaded head.

WRITE PROTECT. This bit (5) indicates that the selected drive contains a diskette platter which is in the "read only" mode. This condition is checked on format track, write data (with data address marks) and write data (with deleted data address marks) operations.

DATA OVERRUN/UNDERRUN ERROR. This bit (4) indicates that the Diskette Controller was not able to service a byte transfer request from the drive before the next request occurred. The data byte is "lost".

ADDRESS ERROR. This bit (3) indicates that the disk address received from the CPU is invalid; that is:

- track address > 76_{10},
- sector address = 00,
- sector address > 52_{10}, or
- sector address + number of records > 52_{10}

SEEK ERROR. This bit (2) indicates that, at the completion of a head movement sequence, the head is not positioned over the expected track. This bit indicates the Diskette System Controller and/or drive are malfunctioning, and a recalibrate diskette operation (see Paragraph 3-3) should be performed. Because all of the diskette operations may implicitly cause the head to move, a seek error can occur during any diskette operation.

CRC ERROR. This bit (1) indicates that the two CRC characters generated during a read data or verify CRC operation were not the same as the two CRC characters appended to the data field when it was written on diskette.

DELETED RECORD. This bit (0) indicates that a sector addressed during a read data or verify CRC operation was preceded by a deleted data address mark.

Three other error conditions are indicated when more than one error bit is true:

ID CRC ERROR. If the address error (3) and CRC error (1) bits are true, it indicates that the CRC characters generated during the reading of an ID field were not the same as the CRC characters appended to the field when it was written by a format track operation.

NO ADDRESS MARK. If the address error (3), seek error (2) and CRC error (1) bits are true, it indicates that no address mark was encountered for a full revolution of the diskette. This usually indicates that the track has not been formatted.

DATA MARK ERROR. If the address error (3), seek error (2), CRC error (1), and deleted record (0) bits are true, it indicates that the data field of a particular sector was not preceded by either a data mark or a deleted data mark.