THE STANDARD OF
EXCELLENCE IN
MICROCOMPUTER
SYSTEMS

IMSAI

IMSAI MANUFACTURING CORP.
14860 Wicks Boulevard
San Leandro, CA 94577
(415) 483-2093
TWX: 910-366-7287

IMSAI EUROPE
Europa Terrassen
8 Rue Jean Engling
Dommeldange, Luxembourg
43-67-46 TWX: 1428

DISKETTE SYSTEM REFERENCE MANUAL

THE STANDARD OF
EXCELLENCE IN
MICROCOMPUTER
SYSTEMS

IMSAI

DISKETTE SYSTEM REFERENCE MANUAL

THE STANDARD OF
EXCELLENCE IN
MICROCOMPUTER
SYSTEMS

IMSAI

November, 1978

The information in this document is subject to change without notice and should not be construed as a commitment by IMSAI Manufacturing Corporation. IMSAI Manufacturing Corporation assumes no responsibility for any errors that may appear in this document.

DISKETTE SYSTEM REFERENCE MANUAL

The postage prepaid READER SUGGESTION form enclosed with this document requests the user's critical evaluation to assist us in preparing future documentation.

IMSAI Manufacturing Corporation
14860 Wicks Boulevard
San Leandro, California 94577

DISKETTE SYSTEMS REFERENCE MANUAL

TABLE OF CONTENTS

1 INTRODUCTION

This volume discusses IMSAI diskette systems hardware (for information on the diskette operating system software, refer to the IMDOS User's Manual). It starts off by reviewing the essential elements and operation of diskette storage systems and then proceeds to discuss IMSAI disk system configurations and the diskette I/O protocol that governs all IMDOS diskette system transactions. This comprises the System Section that is common to all system configurations. The System Section is followed by sections of detailed technical descriptions of the individual IMSAI diskette formatter/controllers available. Users who are primarily interested in learning about the diskette I/O protocol and how to use it may wish to skip section 2 and move on to section 3.

2 DISKETTE SYSTEMS, GENERAL

2-1 COMPONENTS AND INTERFACE

A diskette mass storage system typically consists of the three major elements shown in the block diagram of figure 1.

1) a controller
2) a drive
3) media

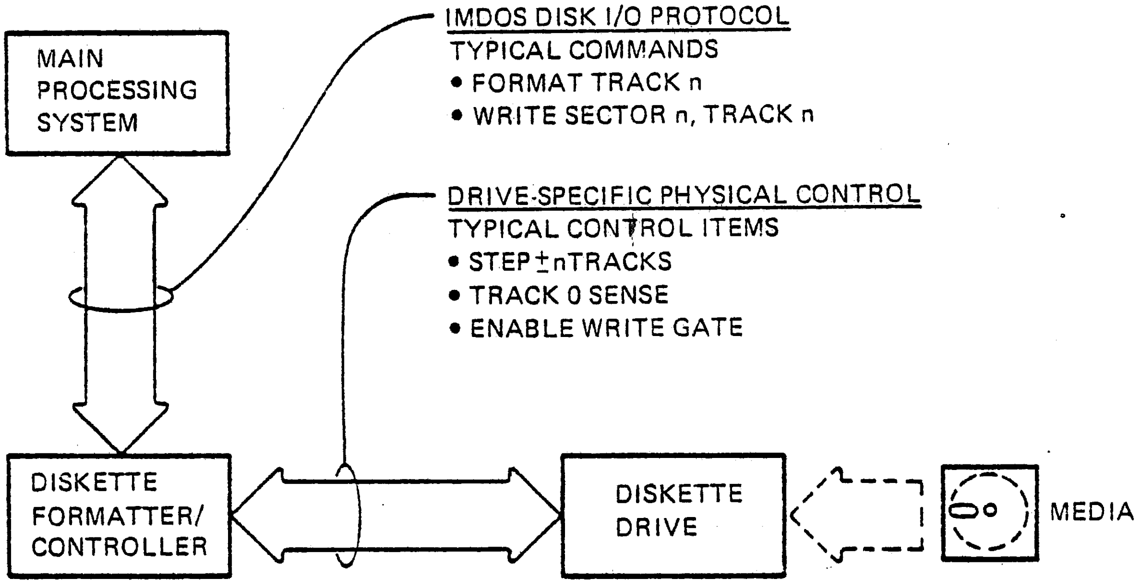


FIGURE 1 Disk System, Functional Block Diagram

2 - 1.1 CONTROLLER

The controller forms a bridge between the main processing system and the data storage mechanism. On the system end, it interfaces to the system bus structure and participates in the disk I/O protocol of the operating system. On the level of this system node, communications take place in the abstract world of data structures. The operating system (O/S) is oblivious to the physical details of the diskette drive mechanism and the recording media. It knows only of files that consist of multiple blocks of bytes which are identified by track and sector numbers (of which more, below), and of commands that facilitate the movement of data blocks to and from storage within the disk system. Central to the purpose of any controller is its ability to transform the high level O/S commands into the appropriate sequences of discrete steps that conform to the specific control requirements of the individual drive mechanism. On the peripheral end, the controller interfaces to the more elementary world of the drive mechanism.

2 - 1.2 DRIVE

The diskette drive contains the various electro-mechanical components and their associated control circuitry which a) rotate the diskette medium with a constant velocity, b) lower the read/write head onto and raise it off the medium, c) propel the head carriage assembly radially across the diskette surface, and d) detect a variety of special conditions in which some of the moving parts within the drive may find themselves. Among the special conditions that are detected in a typical drive, are a) read/write head positioning over Track-00, b) media hardware write protect interlock, c) diskette index hole passage past fixed point in drive, and the like.

Many of the activities within the drive mechanism are independent of one another and are controlled over dedicated lines. The on/off sequencing of these activities (such as drive motor on/off, head carriage stepping, head lowering and raising) is one of the three major tasks which the controller has to perform. The other two are a) FORMATing of the diskette and b) supervising the actual data transfer to and from the recording medium.

2 - 1.3 MEDIA

Data is stored on the surface of a circular sheet that consists of a mylar base coated with an oxide that has magnetic properties especially suited for saturation recording. This sheet is contained within a protective envelope called a cartridge. The inner surfaces of this cartridge are lined with low friction material that offer minimal resistance to the rotating sheet. The cartridge typically has three openings through which the disk may be accessed by a) the read/write head, b) the drive spindle, and c) the photosensor which detects the passage of the index hole on the disk. Optionally, a diskette cartridge may also have a 'hardware write protect' notch. This serves a purpose similar to the knock-out tab on a tape cassette: Prevention of unintentional erasure of recorded data.

Diskettes are sensitive to temperature and humidity variations and must be treated with great care. However, in routine utilization under normal ambient conditions, diskettes have proven to be quite rugged. Since the recording head is in contact with the medium during record or playback, it is important that the recording surface be kept free of dust particles and contaminants of any kind. A diskette that is not mounted in a drive should always be replaced in its dust jacket to prevent such contamination. For similar reasons, the diskette should only be handled by its jacket; and under no circumstances should the recording surface (exposed by the read/write head access slot) be touched.

2-1.4 TRANSFER MECHANISM

Data is transferred serially between the controller and the diskette drive. It is formatted and encoded in the controller prior to transcription to the recording medium, and it is decoded and decomposed to byte rendition prior to input to the main processing system.

Actual data transfer between the main processing system and the disk system (drive plus controller) may be designed to occur in three distinct forms:

1) Programmed Transfer
2) Interrupt Driven Transfer
3) Direct Memory Access (DMA) Transfer

PROGRAMMED DATA TRANSFER

In a system, in which data is moved to and from mass storage by a program that executes in the main CPU for the duration of an I/O transaction, data movement is said to occur via Programmed Transfer. The program that performs the data transfer may be considered part and parcel of the diskette controller and often takes the form of firmware which resides within the hardware bounds of the controller.

I/O transactions are commonly initiated by subroutine calls from the O/S to this controller firmware. The CPU of the main processing system will continue to execute various segments of this firmware until completion of the disk I/O transaction. Thus, the CPU is not available for other tasks during periods of disk I/O activity. It is for reasons of this constraint that this form of mass storage data transfer is only found in single user, single task oriented systems.

INTERRUPT DRIVEN DATA TRANSFER

A diskette controller may be designed to make use of an existing interrupt structure in the main processing system. Such a structure effectively segments central processing time into discrete apertures, each dedicated to the execution of a program segment that services the particular interrupt level that is currently 'on line'.

In a system of this sort, the O/S initiates a disk I/O transaction by setting up the data transfer parameters for the interrupt level which is occupied by the mass storage subsystem and subsequently issuing a particular disk I/O command to the controller. The controller then releases the system CPU and proceeds to implement the O/S command with the required sequence of control steps. During this time interval, the system CPU is free to perform other tasks (such as attend to other users or tasks in a multi-user/task environment). The controller will not interrupt the CPU again until it is either ready to transfer data to/from main memory or it has encountered an irrecoverable error condition. (Note: At present, no IMSAI controller uses this data transfer method)

DMA DATA TRANSFER

If the main processing system has the capability for direct memory access, then the diskette controller may be designed to make use of this capability. DMA structures frequently also make use of the interrupt structure for purposes of transaction initialization. But it is in the nature of the actual data transfer that a significant difference may be noted between a DMA-based system and either of the two discussed above.

In both the Programmed Transfer and the Interrupt Driven Transfer scheme, the system CPU is involved in the data transfer operation. Depending on serial data transfer rates between the controller and the drive, and whether the controller has its own intermediate data buffer, the system CPU may be required to dedicate its time to disk I/O data transfer operations for uninterruptible periods of considerable length. Such constraints can severely impact a system's capacity to service more than one user or to control real time processes.

A DMA structure alleviates this congestion by freeing the system CPU from direct involvement in the actual data transfer process. At the time of data transfer, the diskette controller requests and obtains control over the main memory bus long enough to transfer a byte (cycle stealing scheme), then immediately releases this bus to the main processing system. Though the system CPU will experience an overall slowdown in its program execution during times of disk I/O activity, it is never preempted from performing such program execution for extended periods of time.

2 - 2 FORMATS AND ENCODATION

2 - 2.1 TRACKS AND SECTORS

Unlike main memory RAM, in which each individual byte is uniquely identified by a single location within the system address space, a diskette mass storage system affords no such random access convenience on a byte level. Instead, data is stored bit-serially in segments of 128 bytes (or multiples of 128 bytes). These segments are known as sectors. A number of sectors are concatenated to form one continuous ring concentric about the center of the diskette. Such a ring is called a track. The diskette contains a number of such recorded tracks, each adjacent to another, radiating outward from the innermost track nearest the diskette center to the outermost track nearest the periphery of the diskette (see Figure 2).

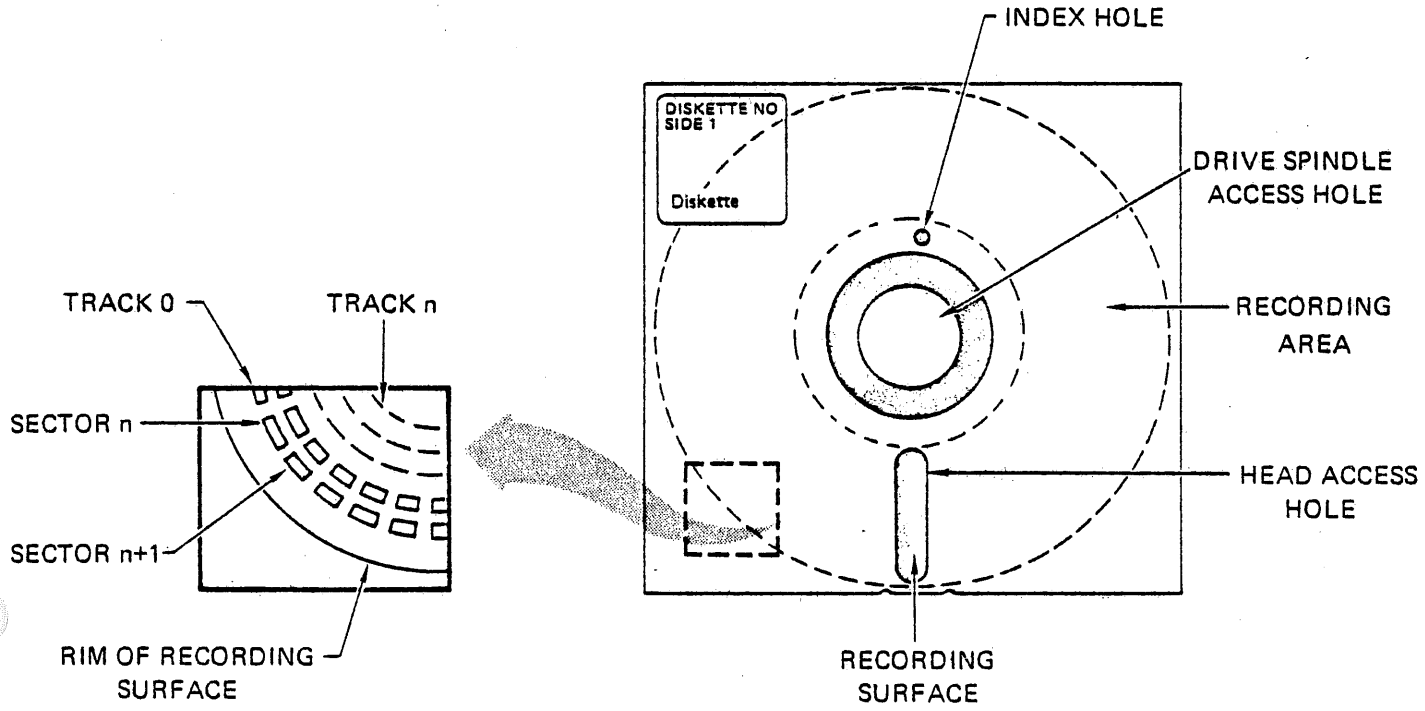


FIGURE 2 Diskette X-Ray View

Data is stored and retrieved one sector at a time. This is accomplished by positioning the read/write head over the appropriate track, locating the specific sector within that track, and updating the data field of the sector with new data (write operation) or reading the data field of the sector into main memory (read operation).

The read/write head passes over a particular track once per revolution of the diskette. An arbitrary track start/stop reference is established by the detection of the passage of an index hole past a sensing mechanism. Sectors are numbered in ascending order (starting with Sector 1) from this index reference all the way around the track.

There are two distinct means of identifying successive sectors within a track. One makes use of sector holes which are equidistantly placed around the hub of the diskette in addition to the index hole; this is known as 'hard sectoring'. In this scheme each sector is identified by the number of sector holes that separate the sector from the start of the track (index). Such a diskette is shown in Figure 3 (Note: IMSAI does not make use of the hard sector technique).

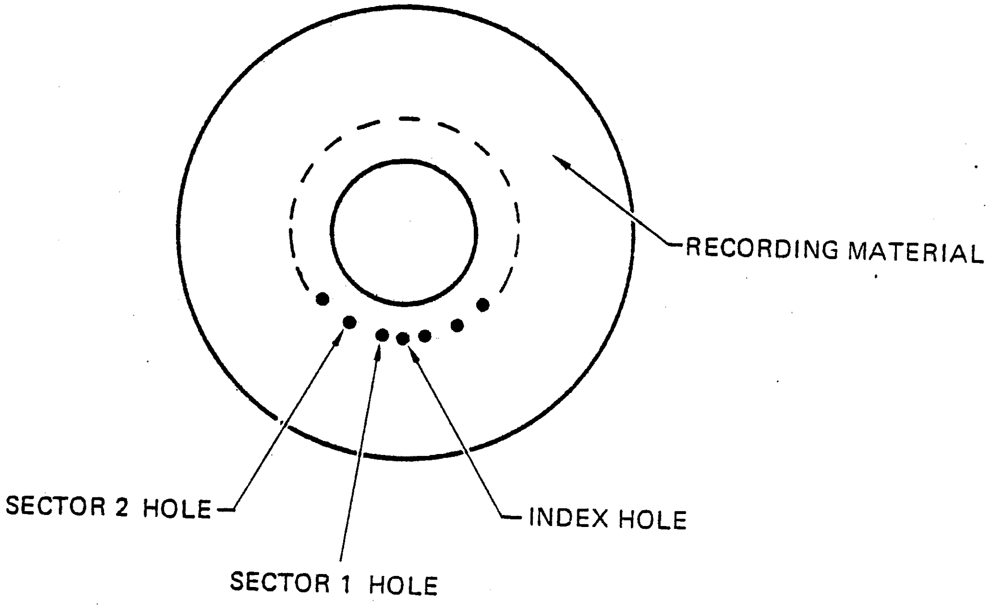


FIGURE 3 Hard-Sector Formatting

The other method does away with these sector holes and employs identification fields that preface the data fields of the sectors; this is known as 'soft sectoring'. The identification fields contain track number and sector number information as well as synchronization patterns. This is the method depicted in Figure 4. Before an operating system can access a diskette mass storage system and store data on a soft-sectored magnetic medium, the diskette must be prepared so that the aforementioned identification fields partition the tracks into individual sector areas and identify these as described; this is known as FORMATing the diskette.

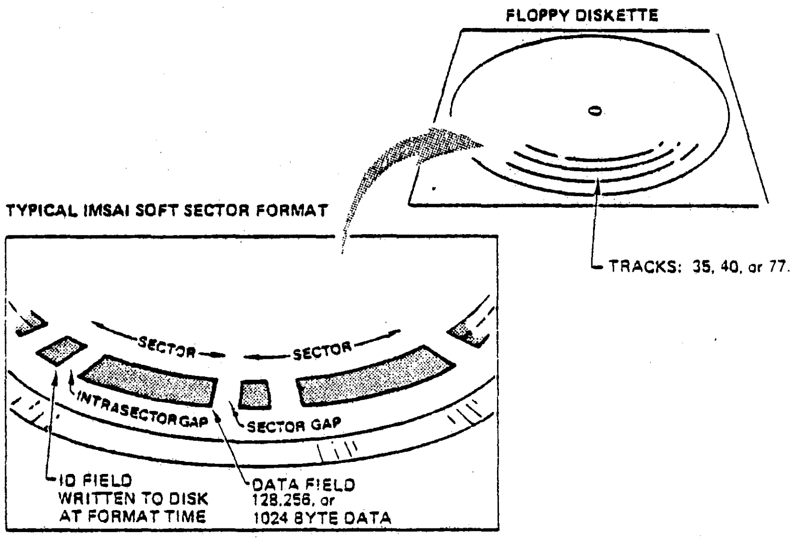


FIGURE 4 Soft Sector Formatting

The length of the sector affects the maximum number of such sectors that may be placed on one track. Standard sector lengths are 128, 256 and 1024 bytes. Another factor that affects total number of sectors per track is the recording density. The recording density is a function of the data encodation and recording method used. This subject is treated in detail in the following subsections.

2 - 2.2 DATA FORMATS

The particular organization of the data on a diskette is determined by the data formats that are used by the system. The format specifies the number of tracks on the diskette, the number of sectors on each track, the arrangement of synchronization and address mark patterns in the ID and data fields of each sector, the specific codes that constitute those unique patterns, and the arrangement of information regarding track number, sector number and sector length code.

Some of the parameters regarding the specific format that is to be used on a particular drive at any particular time are maintained in a table located in the RAM memory of the controller. This table is loaded from the current settings of some hardware configuration options at the time of system initialization (following a reset condition). Some entries of this table may further be updated by one of the O/S utilities under user control; paramount among these is the selection of recording density. In the IMDOS O/S, the command file STAT.COM serves this function. The user is referred to the technical reference sub-section on the particular controller for details regarding the function and usage of these parameters.

A typical sector format is shown in Figure 5. Each track contains one GAP1 immediately following the index reference. This index reference may be only the physical index pulse from the drive (as is the case in mini-diskette formats), or it may also include an encoded index field which is separated from the physical index pulse by a gap of type GAP4A (as is the case in standard diskette formats). Sector 1 follows GAP1 and is separated from the next following sector (Sector 2) by a GAP3. Except for the gap which separates the last sector from the beginning of the track (i.e. GAP4), all sectors are separated from adjacent sectors by a GAP3. Within each sector, the two major information fields (ID and DATA) are separated by a GAP2.

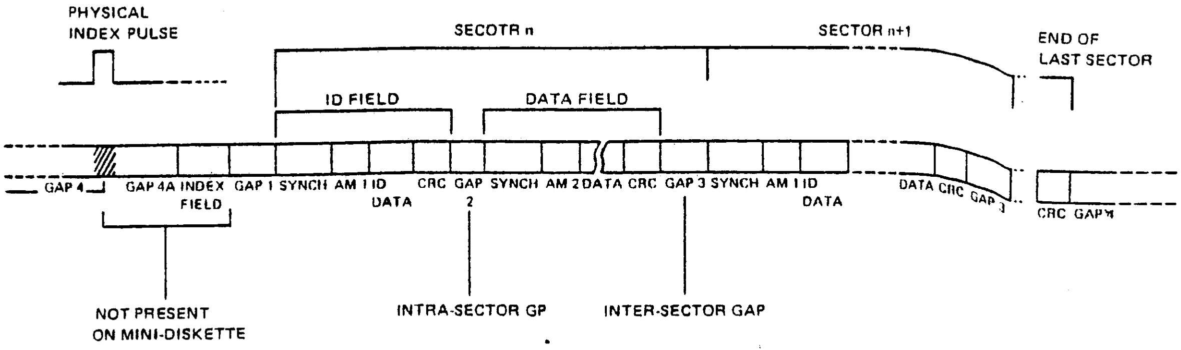


FIGURE 5 Typical Track Data Organization

The ID-field of a sector begins with a bit-synchronization sequence of null bytes. Next comes the ID-field address mark (AM). This is followed by several bytes of ID information regarding track #, sector# and a code which specifies sector length. It may also include information regarding diskette side selection (for future system expansion to incorporate double sided diskette drives). The two cyclical redundancy check (CRC) bytes complete the ID-field. These CRC bytes are the result of a mathematical procedure which introduces an evenly distributed redundancy over the information field. It is by means of this information redundancy that a majority of possible errors are intercepted during read operations. Conversely, if no error is encountered by this error detection scheme, then data integrity may be assumed with a high degree of certainty.

The data field also begins with a bit-synchronization sequence of null bytes, followed by the data field address mark. The main body of data comes next, with the two CRC bytes for the data field bringing up the rear.

2 - 2.3 DATA ENCODATION

Information is recorded on the magnetic surface of a diskette in the form of magnetic flux reversals that are induced by magnetic field reversals in the read/write head during a write operation. During a read operation, the recorded flux reversals induce electrical impulses in that same read/write head.

The significance of these flux reversals and their representative impulses is a function of the bit-serial data encodation process which the byte-parallel data undergoes in the disk controller prior to recording on the diskette.

During a read operation, the serial pulse stream undergoes a complementary decoding process in which the data is once again rendered in byte form. It is the unique physical characteristics of the magnetic recording process that dictate the preferred encodation of binary information.

2 - 2.4 BIT CELLS

The simplest representation of serial binary data is in the form known as Non-Return to Zero (NRZ). Figure 6 shows a series of ONEs and ZEROs in NRZ. In this waveform, only the boundary condition that separates a ONE from a ZERO or vice versa produces a logical level change. Contiguous ONEs or ZEROs result in the extension over longer periods of time of the same logical level. In such bit sequences, the boundaries that delimit each bit-cell become imaginary and can only be inferred from the duration of a particular logical level (dashed lines in Figure 6).

The magnetic field reversals are coincident with reversals in logical level of the waveform that drives the write circuitry of the diskette drive. If serial data is presented to this circuitry in NRZ from, then the only flux reversals that would be recorded are those that represent logic level changes in the NRZ waveform. Sequences of contiguous ONEs or ZEROs would, therefore, contain no information regarding bit-cell boundaries. This is a major weakness of NRZ data recording on magnetic media.

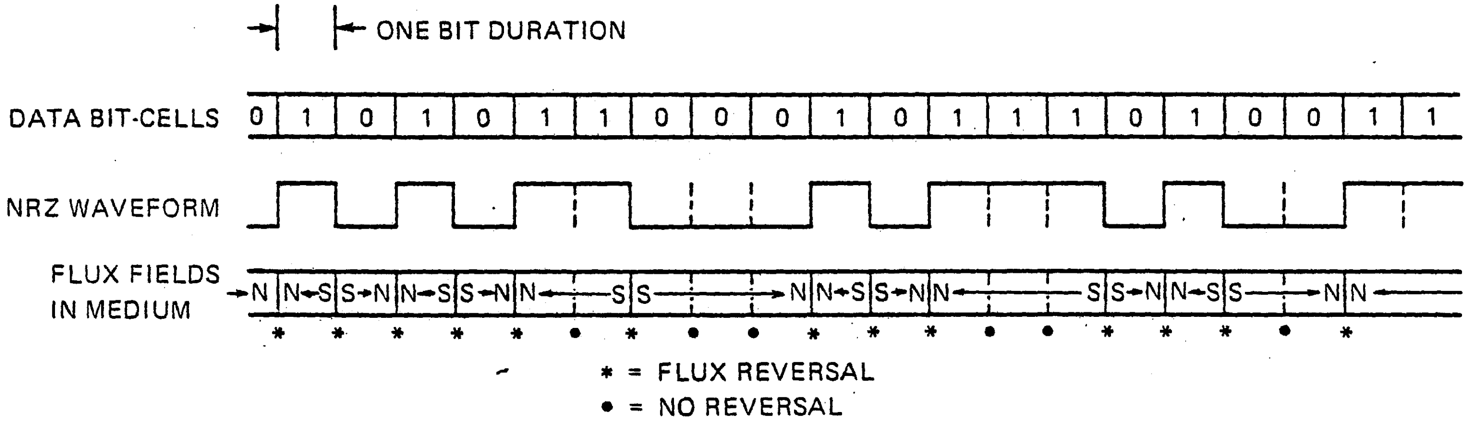


FIGURE 6 Bit-Cells and Flux Reversals

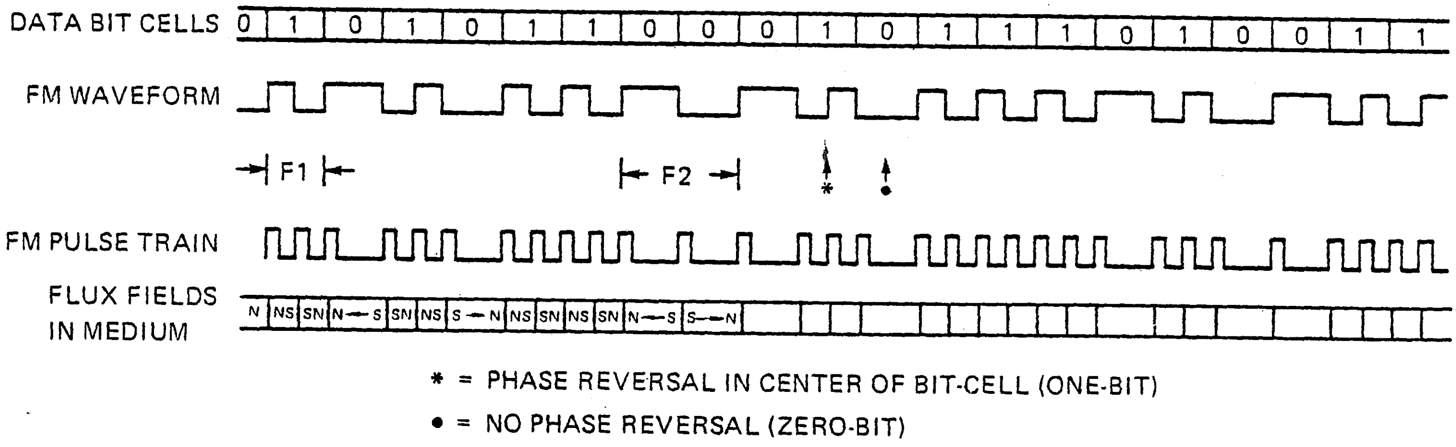

DATA BIT CELLS
0 1 0 1 0 1 1 0 0 0 1 0 1 1 1 0 1 0 0 1 1
FM WAVEFORM
F1
F2
FM PULSE TRAIN
FLUX FIELDS IN MEDIUM
N NS SN N—S SN NS S—N NS SN NS SN N—S S—N
* = PHASE REVERSAL IN CENTER OF BIT-CELL (ONE-BIT)
• = NO PHASE REVERSAL (ZERO-BIT)

FIGURE 7 FM Data Encodation

FM ENCODATION

The introduction of logic level transitions to signify bit-cell boundaries, and the adoption of the convention that the presence of a transition in the center of a bit-cell represents a ONE, while the absense of such a transition represents a ZERO, produces the data encodation method known as Frequency Modulation or FM. Figure 7 shows an equivalent FM representation of the bit sequence shown in NRZ form in Figure 6.

The recording mechanism will induce magnetic flux reversals for every logic level change in this FM waveform. As a result, each bit cell will be bounded by flux reversals, regardless of whether adjacent cells contain alternating ONEs and ZEROs or contiguous ONEs or ZEROs. This method is called FM because the resultant waveform contains two basic frequencies that are defined by the data rate and whose presence or absence is a function of the binary values of the bit stream that is encoded in this fashion. Each ONE bit cell is encoded by a complete cycle of F1. F1 is also the data rate. Each ZERO bit is encoded as one-half cycle of F2. The frequency of F1 is twice that of F2.

At the interface between the controller and the diskette drive, the FM encoded information is interchanged in the form of a pulse train rather than the FM waveform. This pulse train is also shown in Figure 7. Note that the frequency of flux reversals is 2 X F1.

The rules for FM encodation of serial binary data are, therefore:

1) Write data pulses in the center of ONE bit-cells
2) Write clock pulses at all bit-cell boundaries

FM data encodation has been known as Single Density recording since the advent of a Modified Frequency Modulation encodation technique which effectively doubles the amount of data that may be stored on any given stretch of magnetic medium at a specified velocity of linear travel. This technique is explained in the following section.

MFM ENCODATION

The FM recording technique is inefficient in that it requires two flux reversals for every ONE bit-cell. One of these is the actual reversal that specifies the logical value of the bit-cell (i.e. the flux reversal in the center of the bit-cell), the other forms the cell boundary with the next cell and provides bit synchronization clocking information.

If some of these clock pulses could be deleted at strategic locations in the bit stream where the characteristics of the surrounding data environment are sufficient to maintain bit synchronization, then the frequency of magnetic flux reversals would be lowered without undue degradation of recorded data integrity. Similarly, the recording frequency might then be increased until the frequency of magnetic flux reversals is, once again, that of the earlier FM process. This, in essence, is what is accomplished with the Modified FM (MFM) technique.

The rules for MFM encoding of serial binary data may be stated thus:

1) Write data pulses in the center of ONE bit-cells
2) Write clock pulses at the leading cell boundary if

a) the previous cell was a ZERO, and
b) the current cell is a ZERO

In MFM, the maximum frequency of flux reversal is equal to the data rate, while the F1 component is one-half of the data rate. Consequently, for any given rotational velocity of a diskette, twice as much data may be recorded in MFM on any given stretch of magnetic medium than is possible in FM. For this reason, the MFM encodation technique is also known as Double Density recording. Figure 8 shows the same serial data encoded in MFM. Note that this encoding scheme produces three basic frequency components, F1, F2 & F3, whose relationship may be expressed as follows: F1=2xF3; F2=1.5xF3.

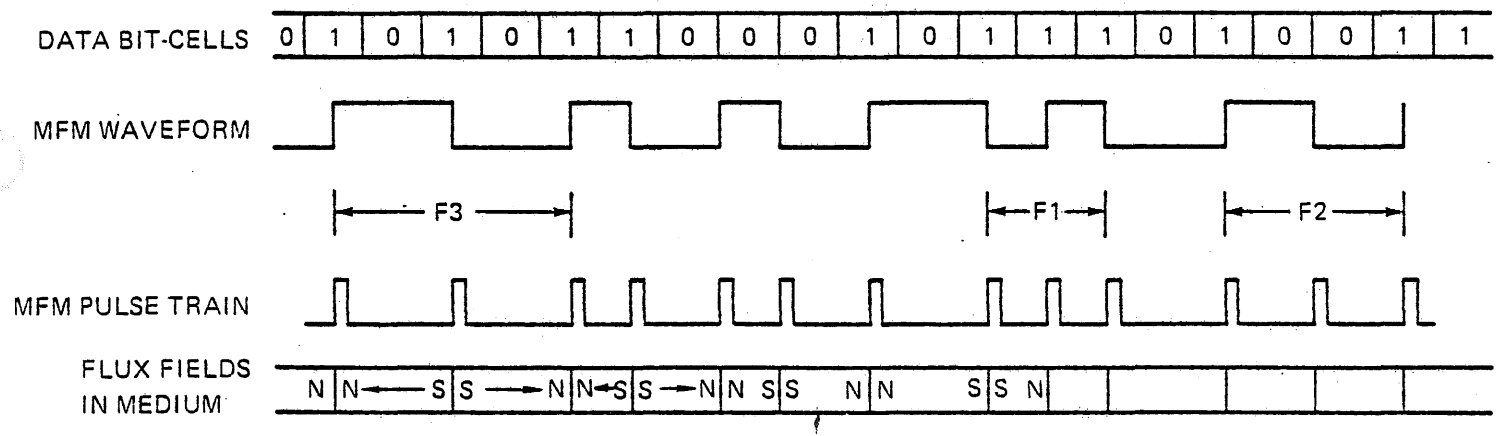


FIGURE 8 MFM Data Encodation

2-2.5 SYNCHRONIZATION

During a read operation, the read/write head is positioned over the track that contains the desired sector of data. The head may or may not already be in contact with the recording surface of the diskette by the time it arrives at the destination track; in any event, the head will begin to pick up magnetic impulses from this track starting at some arbitrary and random location within the track.

The controller receives these pulses in electrical form and must make desisions regarding their significance in terms of phase and frequency relationships, clock and data pulse separation, and, on a higher level, in terms of synchronization sequences that permit it to identify the start and stop of the ID and DATA fields within each sector over which the head passes.

FM SYNCHRONIZATION

In FM encoded tracks, the various intra- and inter-sector gaps contain contiguous ONE-bits (but note exceptions, below). This constitutes a pulse stream at the F1 frequency. When traversing such a sequence, the controller may or may not be in bit synchronization and thus may not be certain with regard to the identity of clock and data pulses in this stream. The bit synchronization circuitry of the controller searches for a sequence of ZERO-bits in order to orient itself with respect to clock and data pulses. Once it has encountered such a sequence, it searches for an ID address mark. In FM, this is a hex value of FE that differs from an ordinary data value of FE in that it is not encoded according to the standard FM encodation rules: some of its clock pulses have intentionally been deleted during the FORMATing of the diskette. Figure 9 shows the pattern for this address mark.

(It should be noted here that the MDIO uses ZERO-bits in gaps of the types GAP1,3 and 4, but uses ONE-bits in GAP2 and the trailing byte of the data field. The FIF employs ZERO-bits in all GAPs)

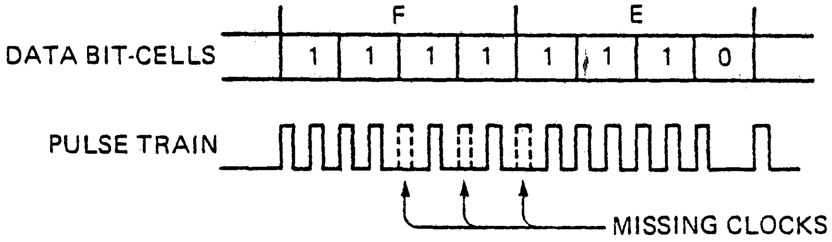


FIGURE 9 FM Address Mark

The recognition of this address mark signals to the controller that the succeeding bytes contain sector identification information. These bytes, in turn, are followed by the CRC bytes. A similar sequence of events takes place as the data field of the sector is traversed by the head.

During the data field update sequence in a sector write operation, the initial activation of the write current may glitch the magnetic medium in the region of GAP2. The exact location of this glitch varies from machine to machine, and may even vary from one write operation to the next. It is a function of a number of variables, primarily diskette drive speed variations. It may be stated that the purpose of GAP2 is to accommodate the aggregate dynamic range of these variables.

The spurious magnetic flux transition(s) which such a glitch may impart to the medium, can give rise to erroneous electrical pulses during playback. These, in turn, can derail the bit-synchronization which the controller had obtained while traversing the ID-field. It is for these reasons that the data field, too, starts with a bit-synchronization sequence of contiguous ZEROs.

At the conclusion of a write operation, the write current is turned off. As is the case with write current turn-on, turning it off may also induce spurious flux reversals in the recording medium. To ensure that this transient condition does not impact the trailing bits of the last data CRC byte, this turn-off is postponed until at least one byte after the last CRC byte.

Some controllers are designed to skip GAP2 during a read operation by counting off a fixed number of bytes and subsequently going into bit synchronization acquisition mode as the head enters the synchronization sequence of the data field. Others may rely on the fact that all Address Marks (AMs) are preceded by a certain number of ZERO-bytes and thus will not recognize an AM that is preceded by ONE-bits. In each case, however, the primary intent is the prevention of erroneous AM recognition. Similarly, the imposition of the criterion, that an AM must be preceded by a sequence of ZERO-bytes, also has the effect of reducing the frequency of false ID AM detection during sector search for both write and read operations. Whether or not a controller makes use of all the synchronization features of the recording format is a function of its implementation.

MFM SYNCHRONIZATION

MFM synchronization is not very different from that of FM. One of the adverse side effects incurred by going from FM to MFM is the considerable loss of clocking information on the recorded track. It will be remembered that the increase in data capacity per unit distance on a track was obtained at the cost of clock pulses. This imposes the requirement for tighter synchronization acquisition criteria in MFM.

The bit-synchronization sequence is still a sequence of ZEROs, only now there are more of them (typically twice as many as in FM). But it is in the Address Mark detection that the major difference is to be found. Whereas a single byte of FEH with modified clock pattern suffices in FM, the Address Mark for MFM encoded data consists of four bytes of which the first three are unique data/clock pulse patterns and the fourth byte is the actual Address Mark value with normal MFM encodation. Figure 10 shows the pulse train for the MFM ID Address Mark. The first three bytes are A1H.with a missing clock pulse as shown. This missing clock pulse is chosen such that the basic frequency contents of these patterns conforms with the F1, F2 & F3 components of conventionally MFM encoded data.

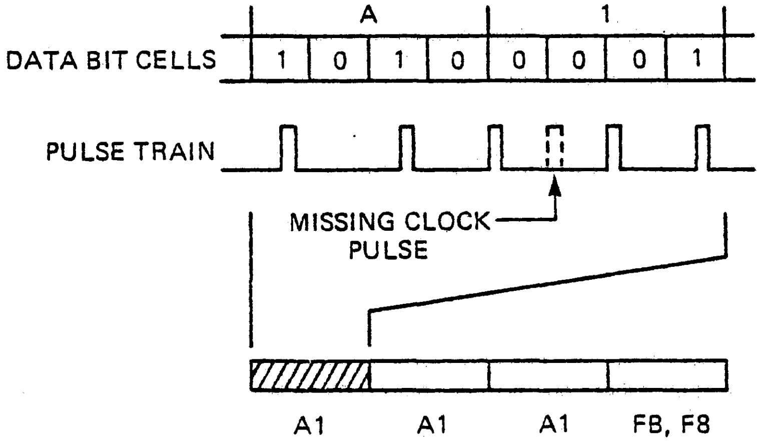


FIGURE 10 MFM Address Mark

Contiguous sequences of ONEs are indistinguishable from contiguous sequences of ZEROs encoded in MFM, except at boundary regions that separate two such sequences. It is for this reason that gaps are not filled with FFH sequences (as is the case in FM), but with sequences of another hex code which produces alternating groups of pulses at F1 and at F2, which, in turn, form an unbroken sequence of whole F1 and F2 lambdas (wavelengths). This code is 4EH, and the lambda sequence it generates is [..F1,F2,F2,F1,F2,F2...], shown in Figure 11. Such a sequence is characterized by symmetrical 'peak shifting' during playback, and it aids in frequency tracking of the playback signal. It also makes possible the discernment of ID-field and data field synch sequences from GAP sequences.

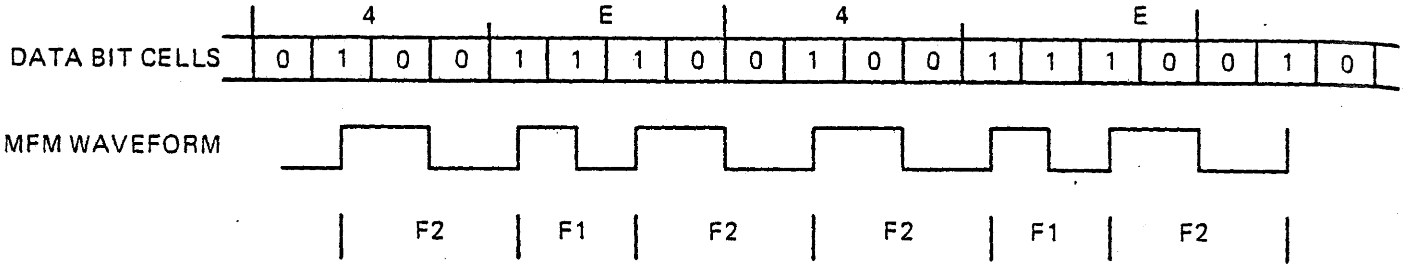


FIGURE 11 MFM Inter- and Intra-Sector Gap Pattern

As is the case with FM, the turn-on and turn-off of the write current during write operations may induce erroneous AMs in GAP2 and GAP3 (i.e. preceding and following the data field). To disallow erroneous AM detection on subsequent sector search operations, a controller may enforce the requirement that the AM be preceded by ZERO-bytes. Since glitch-induced AMs will be embedded in 4EH sequences, they are readily recognized (and therefore ignored) as imposters. A further constraint is imposed by the requirement that three consecutive A1H AM bytes be preceded by these ZERO-bytes and followed by a normally MFM encoded FEH byte.

During a read operation, controllers may follow either of the two procedures already pointed out in the discussion on FM , above, to circumvent fallacious recognition of AMs in the vicinity of data fields.

The combined effect of all these constraints imposed upon the synchronization acquisition and retention process is the prevention of false synch acquisition and, in consequence of this, optimized data integrity and system reliability.

2 - 3 WRITE AND READ PROCESSES

The discussion in the previous sections proceeded from the level of diskette system components down into the intricacies of patterns governing the magnetic encoding process. In the sections that follow, the higher level system interactions between a typical main processing system and the attached diskette system are discussed in terms of processes and the protocol that determines the particular form of their dynamic. Similarly, the sequences of operations that take place within the controller during typical read and write transactions are also presented in detail. Throughout this discussion, access conventions which characterize the diskette I/O protocol of IMSAI's IMDOS O/S serve as an example.

2 - 3.1 ACCESS CONVENTIONS

Regardless of the data transfer mechanism that is employed in a disk system (be it programmed, interrupt driven, or DMA data transfer), the disk I/O protocol usually consists of these major components: Command Parameters Set Up, Disk System Call-Up, Command Execution (1: Sector Search, 2: Data Transfer), and Status Reporting. The particular mechanics of command, data, and status transfer between main system and disk system is dictated by the data transfer mechanism, however. Thus, in a DMA organization, the command parameters must be set up somewhere in system RAM out of which it is executed by the disk controller subsequent to a command call from the main system. In a programmed data transfer environment, these same command parameters might be forwarded to the data transfer firmware module via the system stack and/or some of the CPU registers.

The primary items that must be specified in any disk I/O transaction are a) type of operation (read or write, format, whatever), b) site of origin of data block to be moved elsewhere (RAM area, in the case of a write operation; drive/track/sector, in the case of a read operation), c) site of destination of data block to be moved, and d) special considerations (FM or MFM, sector length).

The disk I/O protocol specifies the format of the data which conveys this information; it also specifies the sequence in which the entire disk I/O transaction transpires.

In the discussion that follows, the basic structure of a typical disk access protocol and its associated formats is presented first, and then the typical sequence for a write and a read operation is detailed.

BYTE COMMANDS AND COMMAND STRINGS

Disk access commands may be categorized into the two classes of Set Up Commands and I/O Commands. The former are executed to set up certain variables (write protect, write enable any drive; restore to track 00 any drive; modify Command String Address Array) and are called Byte Commands. The latter are executed to perform the actual data transfer to and from the disk system; these usually involve a number of parameters that are set up in system memory prior to the execution of what is known as a Command String.

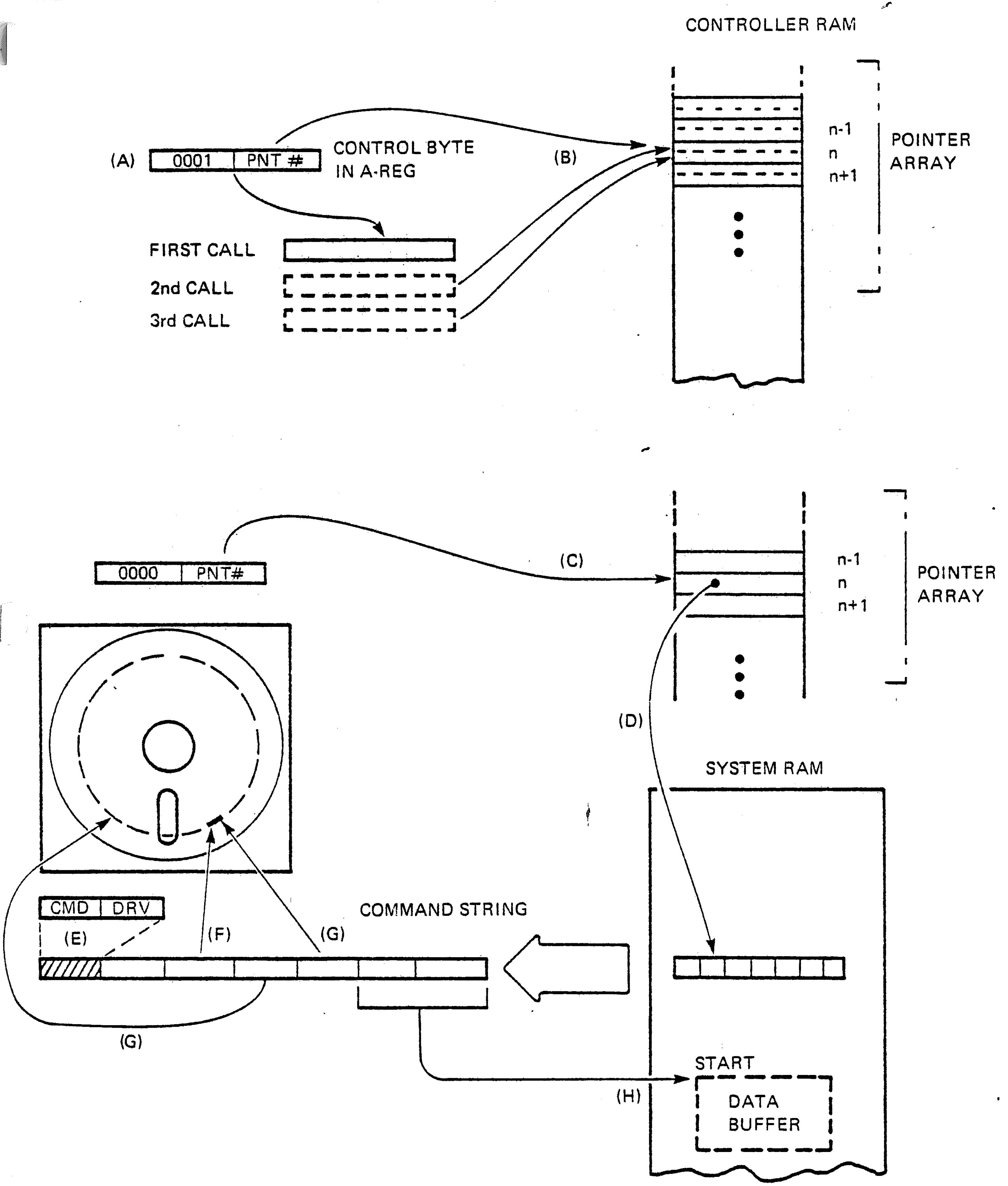
The basic control element of any disk access is the byte code which is forwarded to the disk controller in the A-Register of the CPU during the execution of an instruction that accesses the controller. In the case of the MDIO and the DIO controllers, the control byte is conveyed via the execution of a CALL instruction to the controller firmware. The FIF controller, on the other hand, is accessed via the execution of an OUT instruction that addresses the FIF controller command port. For ease of discussion, all subsequent references to the controller access mechanism presumes the use of a DIO or an MDIO controller.

The control byte consists of two 4-bit fields that specify a) the type of command (upper nibble) and b) either the drive or drives specified for the operation (in the case of Byte Commands 2, 3 and 4), or the Command String Pointer (in the case of an I/O Command). Distinction between Byte Command versus I/O Command is made by setting the upper nibble to ZERO for an I/O Command.

ACCESS SCHEMA

The interrelationship of the Control Byte, Byte Command, String Pointer, and Command String within the environment of system main memory and diskette may be visualized as shown in Figure 12. The following paragraphs describe this interrelationship. The letters in brackets refer to the appropriate region within Figure 12.

FIGURE 12 Disk I/O Access Schema



The Control Byte specifies either Byte Command or Command String execution [A]. Byte Command 1 is used to modify the contents of the Command String Address Array [B]. Command String execution is initiated by conveying the proper Control Byte in the A-Register via a CALL to the controller [C]. The controller uses the pointer to locate the start of the Command String residing somewhere in system RAM [D]. The first byte in this Command String identifies the type of command that is to be executed (Write Sector, Read Sector, Format Track, Verify Sector, Write Deleted Sector Mark) and the drive on which this action is to be performed [E]. The third byte specifies sector length, track number extension (0–255 or 256–511) and side selection (possible future expansion to double sided media) [F]. The fourth byte specifies the track, the fifth byte the sector that is involved in the transaction [G]. Bytes six and seven specify the start of the area in system main memory into or out of which data will be moved from or to the specified sector on the diskette [H]. Byte two is reserved for forwarding of the end-of-transaction status from controller to system.

All Command Strings require at least four bytes to specify the minimum set of parameters. The maximum number of parameter bytes that may compose a command string is seven. The complete Command String must be set up prior to the execution of the CALL to the controller. Variables that are controlled by Byte Commands, may be modified by these commands at any time between the execution of Command Strings.

2-3.2 WRITE/READ SEQUENCES

First, the Command String has to be set up properly, and the Command String Address Array must reflect the starting location of this new Command String. Any one of the sixteen possible pointers may be used. Usually, the array is set up once early in the system initialization phase immediately following power-on, and individual pointer values are modified only rarely thereafter. That segment of the O/S which is responsible for disk I/O, has to make sure that the Command String is set up in alignment with the current value of whichever pointer it will specify during its CALL to the disk controller.

The Command String is composed according to the format already indicated in the previous section (ACCESS SCHEMA). Whether or not Command String bytes five through seven need to be set up depends on the type of command that is to be executed. Next, the Control Code is placed in the A-Register and the system executes a CALL instruction to a specified location within the disk controller firmware. This firmware examines the contents of the A-Register. If it detects a hex-ZERO value in the upper nibble, it interprets the lower nibble as a pointer number for the Command String Address Array. It looks up the 16-bit address value stored in the array location(s) specified by this pointer and uses this address to locate the start of the current Command String.

The Command String contains all the necessary parameters needed by the disk controller to perform a disk I/O transaction. The first byte in this string identifies the Command String as either a write or a read operation, as well as the drive that is to be involved.

WRITE OPERATION

The controller issues a control sequence to the diskette drive which instructs this drive to

1) move the read/write head 'N' steps +/- from current location
2) lower the head onto the medium (if not already lowered)
3) perform a read operation

The controller then examines the pulse stream coming in from the drive and acquires bit- and sector-synchronization (as described in earlier section on SYNCHRONIZATION).

After it has verified that the read/write head has successfully traversed the ID field of the sought after sector (Address Mark check OK, CRC computation OK) and is entering the intra-sector gap (GAP2), it counts off a specified number of bytes and turns on the write current.

The controller then proceeds to write the bit-synchronization sequence for the data field, the Address Mark, the data-field itself, followed by the CRC bytes and the end-of-record byte. Once this is done, the controller orders the write current to be turned off.

If the transaction was accomplished without indication of any error, the controller places a 01 value in byte two of the Command String to indicate an 'all's well' status. This code is also placed in the A-Register before the controller firmware completes its business by returning to the CALLer.

If an error indication was noted by the controller, then it will either forward an appropriate error code to the system, or it will repeat the execution of the Command String a specified number of times before giving up and indicating the requisite status to the system. The controller's response to error conditions varies with the error type encountered and is treated in another section, below (ERROR HANDLING).

The controller will perform a somewhat different sequence of operations when it is informed to FORMAT a track. In this case, the controller instructs the drive to position the read/write head over the specified track and lower the head. The controller next awaits the index pulse from the drive electronics which signals start of track. It then lays sector ID-fields and dummy DATA-fields from start to end of track in the format specified by the third byte in the Command String that is currently being executed.

READ OPERATION

The read operation differs only slightly from the write operation. The sequence that results in the positioning of the read/write head over the desired track and the identification of the proper sector within that track is the same as the one that is used in a write operation. After traversal of GAP2, the controller looks out for the bit-synchronization sequence and then the data field address mark (AM). Identification of AM leads to the subsequent readback of the data field into the system RAM region specified by bytes six and seven of the current Command String. Status reporting takes place as in a write operation.

2-3.3 ERROR HANDLING

Error handling may be either in the simple form of reporting each encountered malfunction immediately back to the system, or in the form of a sophisticated error recovery procedure within the disk controller itself. The latter implementation relieves the main processing system of much of the housekeeping task that is involved in the handling of errors.

The controller may be equipped with enough intelligence (i.e. necessary firmware routines) to make a determination regarding the nature of the anomaly encountered during a disk I/O operation and to respond accordingly. There are several error classes, and from some of these the controller may recover on its own initiative. Some of these conditions are outlined below.

In all of the cases where a transaction reaches a point where a CRC error is discovered by the controller, the latter may be permitted to perform a number of 'retries'. During a retry, the controller executes the current Command String all over again. This procedure is based on the statistical fact that a majority of such errors are 'soft' errors which have a low but finite probability of occuring every time data is recovered from a magnetic medium. Inability to come up with an error-free data recovery after a specified number of retries produces the inference that there is something seriously wrong with the magnetic medium in the region occupied by the sector in question. In such cases, the controller will discontinue retries and report a 'hard' error to the system.

Another type of error from which the controller may recover on its own, is the case where track information in the sector ID-field does not match the specified target track number. In this case, the controller may be allowed to reposition the read/write head several times in an attempt to achieve a track number match; the assumption being that the head carriage control mechanism fails, on occasion, to correctly position the head as instructed.

The remaining error types may be grouped into the three classes a) Command String Error, b) Recoverable System Error, and c) Hardware Error. The last class includes the previously described error conditions when these could not be corrected through retries by the controller. Command String Errors are usually the result of format errors in the Command String (track number too large, illegal command number, etc). Recoverable System Errors are those errors that may be corrected by altering a variable which currently disables an operation as specified by the Command String (hardware or software write protected diskette, drive not ready).

3 IMSAI DISKETTE SYSTEMS

3 - 1 GENERAL

The IMSAI family of disk systems offers the user a wide variety of system configurations. Both standard and mini drive systems are available in several hardware arrangements ranging from simple expansion drive units with or without controller, to fully integrated computing systems. Drives may be combined with different types of controllers to form the desired configuration. Among these controllers is available a broad range of features, including single and double density recording for both mini and standard drives, capability for attachment of additional drives, and the incorporation of several controllers within one computer mainframe. IMSAI's powerful IMDOS multi-disk operating system supports all of these system configurations.

3 - 2 CONFIGURATIONS

3 - 2.1 CONTROLLERS

The variety of disk system configurations is made possible by the flexibility of IMSAI disk controllers. These are the FIF, the MDIO, and the DIO. Of the latter, several models provide multiple single and double density formats. The major features of these controllers are outlined below. Detailed descriptions of their operational characteristics may be found in the reference sections that follow this common front-end text.

FIF CONTROLLER

The Floppy Interface consists of two S100 Bus printed circuit boards, the InterFace Master (IFM) and the Floppy Interface Board (FIB). It uses a data format that is fully compatible with the IBM 3740 format. The FIF attaches to a total of four standard diskette drives and supports single density recording (FM) only. The IFM contains a DMA-based data transfer mechanism that is organized around an internal 8080 microprocessor. The firmware program is contained in two (2) onboard 1K ROMs.

MDIO CONTROLLER

The MDIO is a single board diskette controller designed around an LSI floppy controller integrated circuit (1771). It attaches to a total of four mini-diskette drives and supports a single density format that is similar to the IBM 3740 format. The MDIO employs a Programmed Data Transfer mechanism and contains the diskette format/control firmware in an onboard 2K ROM/EPROM.

DIO CONTROLLER

The DIO is a two board controller consisting of a Disk I/O and a Programmable Data Separator printed circuit board. The DIO attaches to a total of four standard diskette drives and/or three (or four, depending on version) mini diskette drives, and supports both single and double density recording formats. The DIO comes in four versions, each with some unique features that are not available in the other versions.

The DIO–A is the original version and supports the two early IMSAI data formats: Format I, the IBM 3740 compatible single density format, and Format V, a 128 bytes/sector double density format. DIO–A uses Format I for both standard and mini drives, Format V for standard drives only.

DIO–B is a variant of DIO–A which differs from the latter only in that it incorporates a different step rate for standard drives to provide compatibility with the CalComp 142.

NOTE: Both the DIO–A and B models have been replaced by the DIO–C.

DIO–C supports the new single and double density formats and is designed to interface exclusively with standard drives. In addition to being backward compatible with the earlier Format I (though not V), it also communicates in Formats II, III and IV.

DIO–D also supports the new formats and is designed to interface exclusively with mini drives. Up to four mini diskette drives may be attached to a DIO–D.

3 – 2.2 HARDWARE COMPATIBILITIES

Though a variety of diskette drives, both mini and standard, may be attached to IMSAI diskette formatter/controllers, the range of permissible controller/ drive configurations is delimited by both hardware and format compatibility constraints that arise from the specific characteristics of these system components. The matrix shown in Figure 13 indicates hardware compatibilities among the IMSAI controllers and diskette drives of several manufacturers.

FIGURE 13 Controller/Drive Compatibilities

			STD		MIN		
CONTROLLER	DATA TRANSFER MODE	DRIVE TYPE	PERSCI 277	CALCOMP 142	SHUG 400	MPI 851	MICROP 1015
FIF ◉	DMA	STD		X			
MDIO ▨	PROG	MIN			X	X	
DIO-C ◹	PROG	STD	X ◥	X ◥			
DIO-D ◹	PROG	MIN				X ◤	X

LEGENDS:

◥ MUTUALLY EXCLUSIVE DRIVE TYPES

▨ STEP RATE JUMPER-SELECTABLE (12ms FOR MPI, 40ms FOR SHUG.) IF JUMPERED FOR SHUG 400, MDIO WILL SUPPORT MIX OF SHUG & MPI, BOTH 35 TRACKS

◹ DRIVE TYPE SWITCH SELECTABLE

◤ MPI STEP RATE IS 5ms

◉ USE WITH MPU-A ONLY

3 – 2.3 FORMAT COMPATIBILITIES

As was already indicated in the brief descriptions of IMSAI controllers, above, these controllers feature a range of data formats, not all of which are necessarily supported by each controller. Figure 14 shows a matrix in which format compatibilities are correlated with controller type and IMDOS versions. Note that IMDOS 2.05 (or later revision) supports all data formats used by DIO-C & DIO-D. Tables 1 through 4 show the field organization of the recording formats as they are utilized by the different controllers.

For details regarding media conversion techniques involving files created in systems utilizing DIO-A or DIO-B and to be transferred to systems utilizing DIO-C or DIO-D, the user is referred to the IMDOS User's Manual.

FIGURE 14 Format Compatibilities

CONTROLLER	FORMATS SUPPORTED BY IMDOS VERSION	
	2.02	2.05
FIF	I	I
MDIO	I	I II
DIO-C	I	I,II,III,IV
DIO-D	I	I,II,III,IV

FORMAT	DRIVE TYPE		RECORD DENSITY	NOTES
	STD	MIN		
I	128/26	128/18	SD	IBM 3740 SD FORMAT, IMSAI DEFAULT
II	256/15	256/9	SD	2nd IBM 3740 SD FORMAT
III	256/26	256/17	DD	IBM DD FORMAT 1, IMSAI DD DEFAULT
IV	1024/8	1024/5	DD	IBM DD FORMAT 2

TABLE 1

FIF RECORDING FORMAT

						SECTOR											
				INDEX FLD		ID FIELD					DATA FIELD						
MODE	SECTORS PER TRACK	DATA BYTES PER SECTOR	GAP 4A	SYNC	INDEX ADR MARK	GAP 1	SYNCH	ADR. MRK	ID DATA	CRC	GAP 2	SYNC	ADR. MRK.	DATA	CRC	GAP 3	GAP 4
FM	26	128	45 0		1 FC	31 0		1 FE*	4 NOTE 1	2 NOTE 2	17 0		1 NOTE 3	128	2 NOTE 2	32 0	NOTE 4

* FM ID ADDRESS MARK CLOCK PATTERN = C7

NOTE 1: ID DATA CONTAINS FOLLOWING 4 BYTES

1) TRACK NUMBER (0-N)
2) ZERO
3) SECTOR NUMBER (0-N)
4) ZERO

NOTE 2: CRC BYTES ARE GENERATED BY CCITT STD V41, STARTING WITH FIRST AM BYTE

NOTE 3: FM DATA AM: FBH FOR 'DATA', F8H FOR 'DELETED DATA', CLOCK PATTERN = C7

NOTE 4: TRAILING ZERO BYTES BETWEEN END OF LAST SECTOR AND INDEX

TABLE 2

MDIO RECORDING FORMAT

				SECTOR											
				ID FIELD					DATA FIELD						
MODE	SECTORS PER TRACK	DATA BYTES PER SECTOR	GAP 1	SYNCH	ADR. MARK	ID DATA	CRC	GAP 2	SYNCH	ADR. MARK	DATA	CRC		GAP 3	GAP 4
FM	18	128	12 00	2 00	1 FE*	4 NOTE 1	2 NOTE 2	11 FF	6 00	1 NOTE 3	128	2 NOTE 2	1 FF	11 00	NOTE 5
FM	9	256	12 00	2 00	1 FE*	4 NOTE 1	2 NOTE 2	11 FF	6 00	1 NOTE 3	256	2 NOTE 2	1 FF	11 00	NOTE 5

* FM ID ADDRESS MARK CLOCK PATTERN = C7

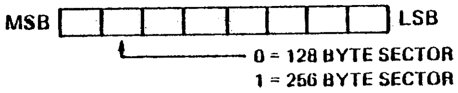
NOTE 1: ID DATA CONTAINS FOLLOWING 4 BYTES

1) TRACK NUMBER (0-N)
2) SIDE NUMBER (0 or 1)
3) SECTOR NUMBER (see also NOTE 4, below)
4) SECTOR LENGTH: 0 = 128, 1 = 256

NOTE 2: CRC BYTES ARE GENERATED BY CCITT STD V41, STARTING WITH FIRST AM BYTE

NOTE 3: FM DATA AM: FB FOR 'DATA', F8 FOR 'DELETED DATA', CLOCK PATTERN = C7

NOTE 4: FOR PURPOSES OF SECTOR FORMAT RECOGNITION DURING SECTOR SEARCH BY 1771 LSI, THIS BYTE ALSO INCORPORATES SECTOR LENGTH SPECIFICATION:



NOTE 5: TRAILING ZERO BYTES BETWEEN END OF LAST SECTOR AND INDEX

TABLE 3

DIO-C RECORDING FORMATS STANDARD DRIVE

							SECTOR										
				INDEX FIELD			I.D. FIELD					DATA FIELD					
MODE	SECTORS PER TRACK	DATA BYTES PER SECTOR	GAP 4A	SYNC	INDEX ADR. MARK	GAP 1	SYNC	ADR. MARK	I.D. DATA	CRC	GAP 2	SYNC	ADR. MARK	DATA	CRC	GAP 3	GAP 4
FM	26	128	40 FF	6 00	1 FC	26 FF	6 00	1 FE*	4 NOTE 2	2 NOTE 3	11 FF	6 00	1 NOTE 4	128 NOTE 6	2 NOTE 3	27 FF	247 FF
FM	15	256	40 FF	6 00	1 FC	26 FF	6 00	1 FE*	4 NOTE 2	2 NOTE 3	11 FF	6 00	1 NOTE 4	256 NOTE 6	2 NOTE 3	42 FF	170 FF
MFM	26	256	80 4E	12 00	4 NOTE 7	50 4E	12 00	4 NOTE 1	4 NOTE 2	2 NOTE 3	22 4E	12 00	4 NOTE 5	256 NOTE 6	2 NOTE 3	54 4E	598 4E
MFM	8	1024	80 4E	12 00	4 NOTE 7	50 4E	12 00	4 NOTE 1	4 NOTE 2	2 NOTE 3	22 4E	12 00	4 NOTE 5	1024 NOTE 6	2 NOTE 3	116 4E	654 4E

* FM ID ADDRESS MARK, CLOCK PATTERN = C7

NOTE 1: MFM ID ADDRESS MARK 3 BYTES OF A, CLOCK PATTERN = 0A

FOLLOWED BY 1 BYTE OF FE, NO MISSING CLOCK BITS

NOTE 2: ID DATA CONTAINS FOLLOWING 4 BYTES

1) TRACK NUMBER (O - N)
2) SIDE NUMBER (O OR 1)
3) SECTOR NUMBER (1 - N)
4) SECTOR LENGTH, 0 = 128 BYTES/SECTOR; 1 = 256 BYTES/SECTOR, 3 = 1024 BYTES/SECTOR

NOTE 3: CRC BYTES ARE GENERATED BY CCITT STANDARD V41, STARTING AT FIRST ADR. MARK BYTE

NOTE 4: FM DATA ADR. MARK: FB FOR DATA, F8 FOR DELETED DATA CLOCK = C7

NOTE 5: MFM DATA ADR. MARK: 3 BYTES OF A1 CLOCK PATTERN = 0A, FOLLOWED BY FB (DATA) OR F9 (DELETED DATA), NO MISSING CLOCK BITS

NOTE 6: USER SPECIFIED DATA INITIALIZED TO 4E

NOTE 7: MFM INDEX ADR. MARK: 3 BYTES OF C2 (CLOCK PATTERN = 14), FOLLOWED BY 1 BYTE OF FC, NO MISSING CLOCK BITS

TABLE 4

DIO D RECORDING FORMATS MINI DRIVE

				SECTOR										
				I. D. FIELD					DATA FIELD					
MODE	SECTORS PER TRACK	DATA BYTES PER SECTOR	GAP 1	SYNC	ADR. MARK	I.D. DATA	CRC	GAP 2	SYNC	ADR. MARK	DATA	CRC	GAP 3	GAP 4
FM	18	128	12 FF	6 00	1 FE*	4 NOTE2	2 NOTE3	8 FF	6 00	1 NOTE4	128 NOTE6	2 NOTE3	10 FF	79 FF
FM	9	256	40 FE	6 00	1 FF*	4 NOTE2	2 NOTE3	11 FF	6 00	1 NOTE4	256 NOTE6	2 NOTE3	27 FF	241 FF
MFM	17	256	50 4E	12 00	4 NOTE1	4 NOTE2	2 NOTE3	22 4E	12 00	4 NOTE5	256 NOTE6	2 NOTE3	34 4E	216 4E
MFM	5	1024	50 4E	12 00	4 NOTE1	4 NOTE2	2 NOTE3	22 4E	12 00	4 NOTE5	1024 NOTE6	2 NOTE3	110 4E	220 4E

* FM ID ADDRESS MARK, CLOCK PATTERN = C7

NOTE1: MFM ID ADDRESS MARK 3 BYTES OF A1, CLOCK PATTERN = 0A
FOLLOWED BY 1 BYTE OF FE, NO MISSING CLOCK BITS

NOTE2: ID DATA CONTAINS FOLLOWING 4 BYTES

1) TRACK NUMBER (0 - N)

2) SIDE NUMBER (0 OR 1)

3) SECTOR NUMBER (1 - N)

4) SECTOR LENGTH, 0 = 128 BYTES/SECTOR; 1 = 256 BYTES/SECTOR; 3 = 1024 BYTES/SECTOR

NOTE3: CRC BYTES ARE GENERATED BY CCITT STANDARD V41, STARTING AT FIRST ADR. MARK BYTE

NOTE4: FM DATA ADR MARK, FB FOR DATA, F8 FOR DELECTED DATA CLOCK = C7

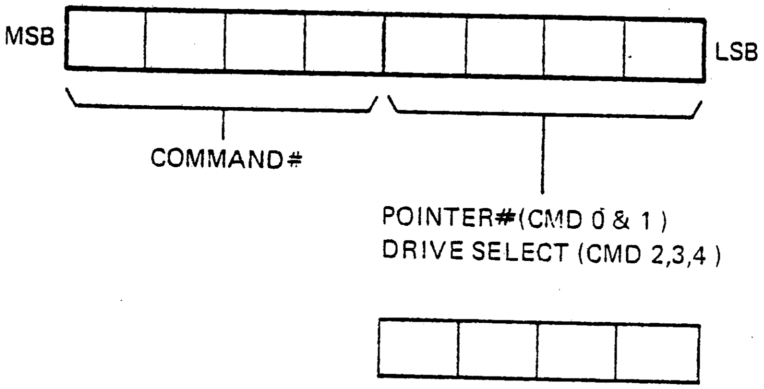
NOTE5: MFM DATA ADR MARK: 3 BYTES OF A1 CLOCK PATTERN =0A, FOLLOWED BY
FB (DATA) OR F9 (DELETED DATA), NO MISSING CLOCK BITS

3 - 3 TRANSACTIONS

In this section, the IMDOS disk I/O protocol is discussed in detail. Emphasis is placed on the expositon of command formats and error codes. The reader has already been introduced to these formats and the general disk access mechanism in sections, titled BYTE COMMANDS AND COMMAND STRINGS and ACCESS SCHEMA, above. Here the specific byte and string commands are explained and their formats defined.

3 - 3.1 BYTE COMMANDS

The Byte Command is an eight bit code structured so that the upper four bits contain the Byte Command Number and the lower four bits contain either a pointer number or a drive select number, depending on the Byte Command Number involved (see Figure 15).



DRIVE 0	X	X	X	1
DRIVE 1	X	X	1	X
DRIVE 2	X	1	X	X
DRIVE 3	1	X	X	X

FIGURE 15 Byte Command Format

The Byte Commands are listed below according to the Byte Command Number (as defined above).

COMMAND 0:

Execute Command String identified by pointer. The controller uses the lower four bits of the command code as a pointer into an array that contains sixteen 16-bit addresses. One of these 16-bit addresses (the one identified by the lower four bits in the command code) specifies the start of a string of bytes that comprises a Command String. This is the Command String that is to be executed. Note that prior to using this command type, the 16-bit address must have been initialized as required. This may be accomplished by the Byte Command described next.

COMMAND 1:

This command instructs the controller firmware to expect two additional bytes from the calling program (by way of two additional calls). These two bytes constitute a 16-bit address that is to be placed into the pointer array at the location(s) specified by the lower four bits of the Command-1 byte.

COMMAND 2:

Restore Drive(s) instructs the controller to cause the read/ write head to be positioned over track-0 on any and all drive(s) specified by the lower four bits of this command code. The controller will take due notice of the drive(s) specified, and will perform the restore operation as first order of business the next time such drive(s) is (are) referenced by a Command String.

COMMAND 3:

This command sets software write protect flag(s) for drive(s) specified in the lower four bits of the command code. Note that the controller firmware presets all drives to Write Enabled during the initialization phase that follows system power-on; more of this later.

COMMAND 4:

This command resets software write protect flag(s) for drive(s) specified in the lower four bits of the command code.

COMMAND 5: (FIF only)

Software reset causes the IFM to jump to location 0000 and re-initialize. If drive 0 is ready, track 0 sector 1 will be read into location 0000 of system memory. This command can be used to effectively "cold start" the system.

NOTE: This command is implemented in REV. 4 and later firmware.

COMMANDS 6 - 15: (FIF only)

These commands perform no operation, except to reset interrupt if interrupt mode is being used. IMSAI standard RESET INTERRUPT command is COMMAND 15.

COMMANDS 5 – 15: (MDIO and DIO)

Illegal commands. Controller returns with error code to system.

POINTERS

The pointer is a number from 0 to 15 which points into an array that contains sixteen 16-bit addresses. Each of these addresses, in turn, points to the leading byte of a Command String that resides somewhere in system RAM. Byte Commands 0 and 1 interpret the lower four bits of the Byte Command Code as such a pointer into the Command String Address Array. Byte Command 1 is used to modify addresses within this array, while Byte Command 0 signals the order to execute the Command String whose starting address is (indirectly) specified by the lower four bits of the Byte Command 0 code.

During system initialization, the initialization entry point of the controller firmware is called by the operating system (or by BOOT, in case of a cold start). Among the many functions that are performed during this phase, the Command String Address Array is initialized to the following default values (hex):

0: 0080	4: 4000	8: 8000	C: C000
1: 1000	5: 5000	9: 9000	D: D000
2: 2000	6: 6000	A: A000	E: E000
3: 3000	7: 7000	B: B000	F: F000

DRIVE SELECT NUMBERS

Byte Commands 2, 3 and 4 specify one or more drives with the lower four bits of the command code. The drives are bit-encoded, a ONE in any bit position signifying the selection of that particular drive. A command that does not specify at least one drive will be rejected with an error indication to the system.

3 – 3.2 COMMAND STRINGS

Command Strings are indirectly executable in that the exact operation to be performed is not specified in the code conveyed to the controller during the CALL. Instead, the Command String resides somewhere in system RAM in variable length format. The control code imparted to the controller during the CALL specifies the start address of this string by vectoring through the Command String Address Array. Controller firmware interprets the string and performs the specified operation.

All Command Strings consist of at least four bytes of information and may include as many as seven. The definition of each of these bytes is given below. (See also Figure 16)

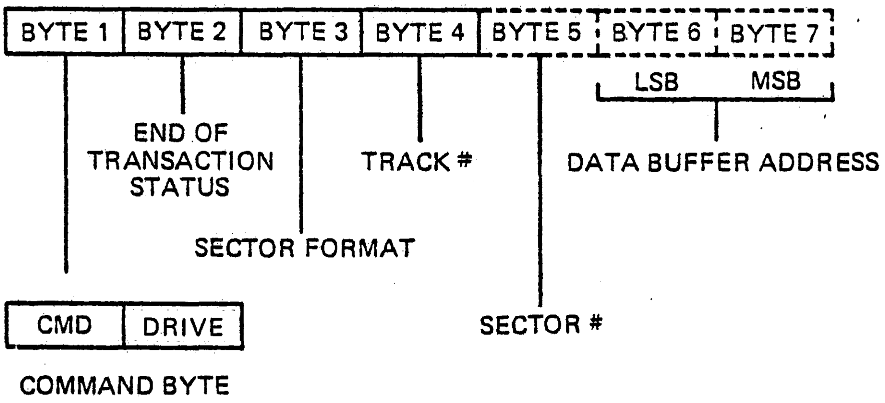


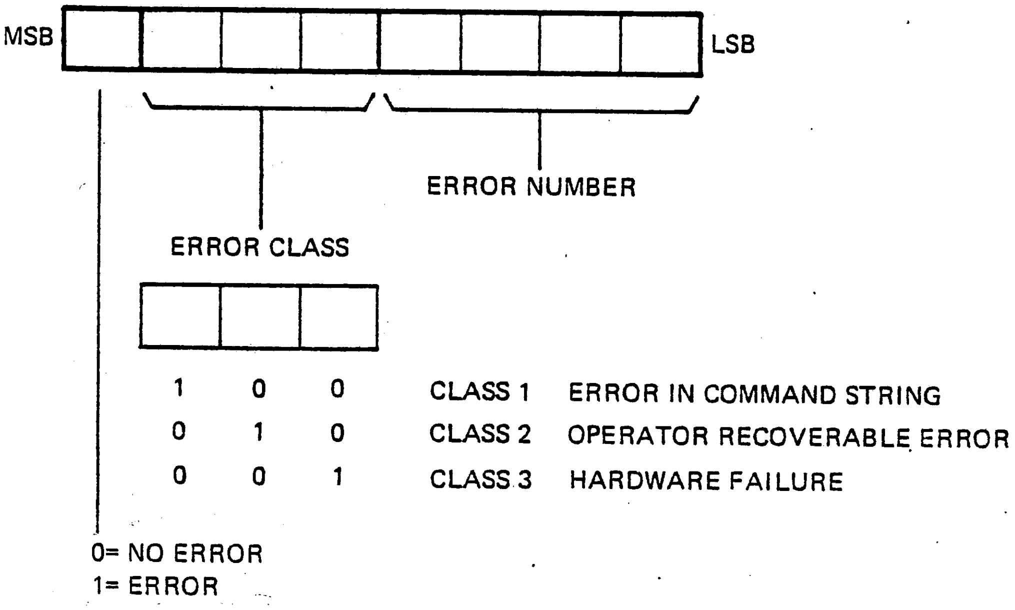
FIGURE 16 Command String Organization

BYTE 1 - Command Byte

This byte contains a command number in the upper hexadecimal digit and a drive select number in the lower digit. The operation of each Command Number is explained in the next section. A drive is selected if its corresponding bit is set. Commands that specify more than one drive are rejected with an error indication to the system.

BYTE 2 - Status Byte

This byte indicates to the CALLer the status of the controller after completion of the last disk transaction. A hex value of 01H signifies successful completion. The MSB is set if an error condition had been encountered prior to completion of the operation. The same code that is placed into this byte location within the Command String is also forwarded to the system in the A-Register. The significance of the various codes is given in the subsection, titled ERROR CODES, below (see also Figure 17).



1	0	0	CLASS 1	ERROR IN COMMAND STRING
0	1	0	CLASS 2	OPERATOR RECOVERABLE ERROR
0	0	1	CLASS 3	HARDWARE FAILURE

FIGURE 17 Status Byte Format

BYTE 3 – Format & Extended Track

This byte consists of several single and multiple bit fields that specify a) MSB for extended track numbers (256–511), b) sector length, c) side select (for future expansion to double sided diskettes) and d) platter number (for future expansion to multi-platter fixed disk drives). Sector lengths that may be specified, are 128, 256 and 1024 bytes (see Figure 18).

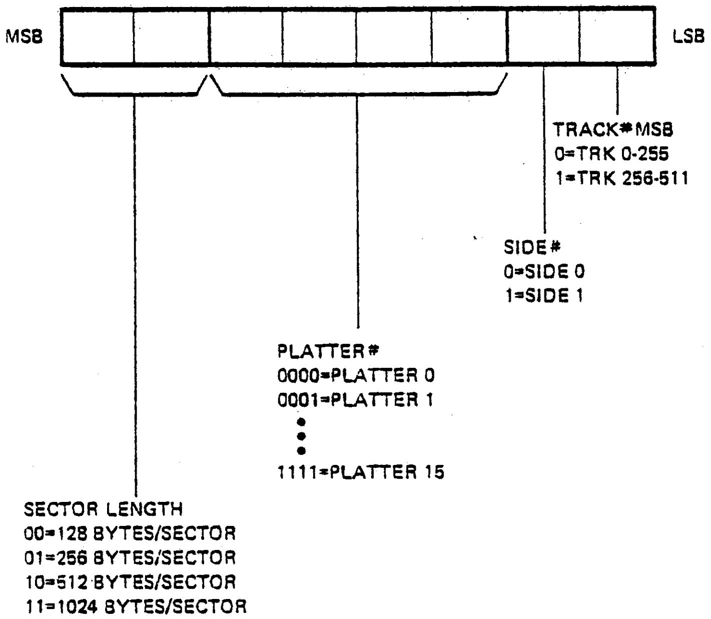


FIGURE 18 Command String Byte 3 Format

BYTE 4 – Track Number

This byte specifies the track number which may be 0 through 76 for standard diskettes and 0 through 34, 39 or 76 for mini diskettes. MPI type B51 drives can accommodate 40, Micropolis-1015 77 tracks. Identification of what constitutes a legal track number specification is a function of controller type, drive type and format. Illegal track number specification in this byte is flagged and results in an error indication to the system.

BYTE 5 – Sector Number (when required)

This byte specifies the sector that will be involved in the disk I/O transaction. The legal sector numbers are:

1 to 18 for single density (SD) 128 byte sectors;

1 to 9 for SD 256 byte sectors;

1 to 17 for double density (DD) 256 byte sectors; and

1 to 5 for DD 1024 byte sectors when accessing mini diskettes.

1 to 26 for SD 128 byte sectors;

1 to 15 for SD 256 byte sectors;

1 to 26 for DD 256 byte sectors;

1 to 8 for DD 1024 byte sectors; and

1 to 58 for DD 128 byte sectors when accessing standard diskettes.

Applicability of these ranges of sector numbers and associated formats is governed by compatibility considerations involving the several IMSAI controller types and versions (see sections on hardware and format compatibility, above).

BYTE 6 & 7 – Memory Address (when required)

These two bytes specify the starting address of that contiguous block of memory locations within system RAM to or from which the specified sector is to be transferred. Byte 6 contains the least significant, byte 7 the most significant address byte. Specifying a value of E0H for byte 7 results in an error indication (start of address space occupied by the controller).

COMMAND TYPES

The individual Command String commands are identified by the value of the most significant hex digit of Byte 1 of a Command String. The definitions of these commands are given below by Command Number.

COMMAND 0 – Not used

COMMAND 1 – Write Sector

This command instructs the controller to write the data block whose starting address is specified by bytes 6 and 7 of the Command String, into the sector specified by byte 5, on the track specified by byte 4, according to the format called out in byte 3.

COMMAND 2 – Read Sector

The sector specified by byte 5, of length specified by byte 3, is read from the track called out in byte 4 of the Command String, and is written into the data buffer in system RAM whose starting address is specified by bytes 6 and 7 of the Command String.

COMMAND 3 – Format Track

This command formats the track specified in byte 4, according to the format called out in byte 3 of the Command String. It encodes the appropriate number of sector ID–fields, constructs dummy data fields and fills all intra– and inter–sector gaps with the required bit patterns. All data information previously contained in the specified track is eliminated.

COMMAND 4 – Verify Sector

This command informs the controller to read the specified sector and to check the validity of the CRC bytes. Byte 5 of the Command String specifies the sector of interest.

COMMAND 5 – Write Deleted Data Address Mark

Execution of this command results in the placement of a Deleted Data Address Mark in the data field of the sector whose number is specified in byte 5 of the Command String. The sector is located on the track specified by byte 4, and is of a length specified by byte 3 of the Command String.

COMMAND 6 – Configuration Check (FIF only)

This command causes the floppy controller to test for the presence of drives and return a bit map of the drives found. Only the drives indicated in the low order nibble of the command will be tested. Within 1 ms of issuing the command, the controller will return 40H in the status byte to indicate that the tests are in progress. When the operation is complete, the controller returns 2XH in the status byte where "X" is the map of drives tested and found.

NOTE: This command is implemented in REV. 4.0 and later firmware.

NOTE: Command 60H can be used to test for the presence of REV 4.0 or later firmware. No drives are tested but 40H and 20H are returned in the status byte as described above.

COMMANDS 7 – 11 (FIF only)

Commands 7 through 11 are identical with commands 1 through 5 and are used to read diskettes which have the logical and physical track address physically different. The logical track address to be used is always contained in the two bytes directly following the other command data called for in the COMMAND STRING (0 – 5). For example, if the basic command used just 4 bytes, then the logical track number would be in bytes 5 and 6. If the basic command used 7 bytes, then the logical track address number would be in bytes 8 and 9.

COMMANDS 5 – 15 (MDIO and DIO)

Illegal commands. Controller returns with error code.

3-3.3 ACCESSING THE CONTROLLER

This section describes the procedure that is observed by the BDOS segment of the IMDOS operating system when it accesses the disk system. A user who intends to utilize custom (non-BDOS) disk accessing software driver(s) must ensure that such driver(s) follow(s) the procedure described below. In most cases, there is no need for special disk driver module(s), as the file management facilities provided through BDOS are extensive and very flexible. For a description of these IMDOS facilities, the user is referred to the IMDOS PROGRAMMING GUIDE chapter of the IMDOS User's Guide.

PARAMETER SET UP

Prior to any disk access, a variety of parameters must be specified. Some of these must be specified with each disk I/O transaction (drive, track, sector, sector length, data buffer address pointer); others are variables that are changed less frequently (media software write protect, data buffer address array et al) but are still under the control of the disk I/O driver module. Still other variables, such as recording density, are commonly specified by the operating system (although the driver module may have access to some of these variables as well).

VARIABLES CONTROLLED BY BYTE COMMANDS

The Byte Commands (excluding Byte Command 0) serve the purpose of allowing alteration of some of the infrequently changed variables. Their descriptions were given above. Though their execution also transpires via a CALL to the controller firmware, accesses of this variety are kept separate from command executions that result in actual data transfers that involve the media itself.

Byte Commands 3 and 4 are used to modify the software write protect status associated with any, some or all drives attached to a particular controller. The proper byte command code is placed in the high nibble of the byte, the drive selection specification in the lower nibble. The drives are bit-encoded (see format above) and a logical ONE selects, while a ZERO de-selects the drive encoded by a particular bit position in the lower nibble of the byte command code. This code is placed in the A-Register.

The variable update (write protect a certain drive that was previously enabled, for example) is effected by the issuance of a CALL to the proper controller firmware entry point. In the case of a standard drive (STD), this entry point is at hex address E006; for a mini drive (MIN) it is at hex address E009.

Byte Command 1 is used to change the Command String Address Array contents associated with a particular Pointer into this array. It will be remembered that it is through this array that the starting addresses of Command Strings are specified during Command String executions (of which more below).

The entire process of changing one 16-bit address within the array involves three CALLs to the controller firmware. In the first CALL, the Byte Command 1 code is conveyed to the controller. This code consists of a 1-value in the upper nibble, and the array pointer (a hex value in the range of 0 - F) in the lower nibble.

The second CALL sends the low 8-bit byte , the third CALL the high 8-bit byte of the address that shall henceforth be associated with the array pointer in question. The controller places these address bytes in the appropriate cells within the array which is maintained in the RAM on-board the controller printed circuit board (PCB).

Byte Command 2 is used to obtain track 'tracking' within a disk system. Execution of this command 'restores' the read/write head carriage assembly to the periphery of the diskette and positions the head over track-0. Typically, this command is used following a system reset condition to provide a known reference for the succeeding head repositioning activities during disk I/O. The controller firmware keeps track of subsequent head position changes for each drive that is attached.

If the controller encounters an error condition while a disk I/O transaction is taking place, its firmware may find it necessary to issue a restore order to the drive in an attempt to recover from the error. This is treated in greater detail in a section on error handling, below.

SETTING UP COMMAND STRINGS

Command Strings are executed 'indirectly' and must be set up in system RAM prior to the CALL that initiates the execution of a command string. The composition of a Command was discussed in the section, titled COMMAND STRINGS, above, and setting up such a string is a simple matter of loading the byte locations with the appropriate values. At least the first four bytes must be specified for each Command String. Depending on command type, one or three additional bytes must be appended to these four bytes to complete the Command String. Byte 2 of the string must be cleared (00H).

Perhaps the most critical aspect to be kept in mind when constructing a Command String is address alignment of a) start of Command String, b) Value of intended Command String Pointer, and c) 16-bit address associated with the pointer in the Command String Address Array.

If the Command String is set up regularly in the same contiguous locations in system RAM, then one of the sixteen available array entries need only be set up once at the outset, and subsequently referenced by the appropriate pointer.

If, on the other hand, location of the Command String within system RAM is not fixed, but varies from one disk I/O transaction to another, then it is necessary to update the Cmd. String Address Array prior to each Command String CALL. An exception to this is the case where the Command String is located in one of a relatively fixed set of Command String locations. In this case it is expeditious to assign separate Pointers for each member of this set of locations and to vary the Pointer value of the control byte according to the location of the next to be executed Command String.

COMMAND CALL

To invoke the execution of a Command String, the Byte Command 0 code with appropriate array pointer is loaded into the A-Register and a CALL is executed to one of two controller firmware entry points:

E006 - - for disk I/O with a standard diskette drive
E009 - - for disk I/O with a mini diskette drive

The CALL instruction is the means whereby the MDIO and the DIO are accessed. Since it makes use of the DMA data transfer method (in place of the Programmed Data Transfer method used by both MDIO and DIO), the FIF requires a different means of access than the CALL instruction. The FIF is provided with a command port through which it receives the control byte. This port is accessed by an OUT instruction that outputs the contents of the A-Register to the port. The standard address for this FIF command port is FDH.

The controller firmware examines the control code, detects the ZERO value in the high nibble, and proceeds to execute the Command String specified by the pointer value of the low nibble. As it implements the Command String, this firmware examines the parameters for compliance with format rules, and rejects any Command String that deviates from these rules by returning to the CALLer with a unique error code.

If the Command String parameters check out OK, then actual execution of the string commences. The sequence of activities that takes place as the controller directs Command String execution has already been described in detail elsewhere (section 2 - 3, WRITE AND READ PROCESSES). Suffice it to repeat here that the controller will instruct the diskette drive to go through the required steps to access the specified track, that it locates the specified sector (where necessary) by examining the pulse stream which it receives from the drive, and that it then either writes into (or reads from) the data field of that sector from (or to) a data buffer located somewhere in system RAM.

Either upon successful completion of the transaction or as a result of having encountered an irrecoverable error, the controller will execute a RETURN from subroutine CALL and informs the calling program of its post transaction status.

STATUS

A status code is returned by the controller in both Byte 2 of the Command String responsible for the last disk I/O transaction and the A-Register. Byte 2 had been reset to ZERO prior to the last command CALL. To indicate successful completion of the transaction, the controller will set the LSB of the status byte.

To indicate an error condition, the controller will set the MSB of the status byte and encode the specific error condition in the remaining bits of the status byte. Furthermore, the controller will also set the sign-bit of the CPU status byte. The calling program may make a determination regarding the outcome of the transaction by testing this sign-bit.

ERROR CODES

Error codes returned by the controller are grouped into three classes which are bit-encoded by bits 6, 5 and 4 of the status byte. The specific error type is encoded in the low nibble of the byte. A description of these error codes is presented by class and type (see also Figure 17).

Class 1 - (Bit-6 is set, status code has the form CXH)

This class of errors deals with illegal specifications in the Command String.

C2	No drive selected
C3	More than one drive selected
C4	Illegal command number
C5	Illegal track address
C6	Illegal sector address
C7	Illegal buffer address
C8	Illegal byte-3 format

Class 2 - (Bit-5 is set, status code has the form AXH)

This class of errors deals with system errors from which the operator may recover through appropriate intervention.

A1 Selected drive not ready.

A2 Selected drive is hardware write protected and attempt had been made to write onto its diskette.

A3 Selected drive is software write protected and attempt had been made to write onto its diskette.

A4 Sector length specified by byte 3 of command string does not correspond to actual sector lengths found on diskette.

Class 3 - (Bit-4 is set, status code has the form 9XH)

This class of errors deals with hardware malfunctions that inhibit completion of disk I/O transaction.

91 Selected drive not operable; controller unable to position head over track-0 or drive became 'Not Ready' during transaction.

92 Track address error while attempting to read/write onto diskette; (controller makes 3 attempts to reposition head over desired track before this error is indicated to caller).

93 Data synchronization error; controller unable to find desired sector on track specified within permissible time; (controller makes 3 attempts to reposition head before this error is indicated to caller).

94 CRC error in the ID-field of desired sector.

95 Failure to recognize DATA AM after recognizing sector ID-field

96 CRC error in the DATA-field of desired sector; (data is transferred independent of error discovery).

97 Deleted Data Address Mark in DATA-field;
(data is transferred independent of error discovery, no CRC error involved).

98 FORMAT operation unsuccessful;
(diskette revolution time too short to permit recording of entire track, will occur when diskette revolution rate exceeds nominal rate by more than 2%).

As has already been pointed out above (see section titled, ERROR HANDLING, under 2 - 3), IMSAI controllers have enough intelligence to attempt recovery from certain error types. Typically, the recovery sequence involves ten attempts to locate the desired sector before issuing a restore command to the drive and then again trying to locate the sector. This is repeated a number of times before a hard error is suspected by the controller and an error code is forwarded to the CALLer.

3 - 4 SYSTEM INITIALIZATION

The great variety of system configurations and their differing system initialization procedures are treated extensively elsewhere (IMDOS User's Manual). In this section, only a controller feature summary with regard to system initialization is presented. A general discussion introduces some of the concepts, and the characteristics of the FIF, MDIO and DIO controllers are then explained.

GENERAL

For any system to function in an orderly and predictable fashion, it must go through a known (and intended) sequence of states upon initial application of power. It must, furthermore, be possible for an operator to direct such a system to go through an initialization sequence that will set up specific initial system conditions and then arrive at an intended end-state; this 'state' is typically the interrogation loop of the command processor. Every system monitor (such as the MPU-B firmware monitor) and every O/S (such as IMDOS) has such a command processor.

In an IMDOS-based system, the initialization process takes place in three phases. Phase 1 is hardware initialization. This occurs on the register level within the CPU and peripheral circuits, and entails the clearing and/or presetting of registers, address pointers and flags. Hardware initialization always results in the execution of program segments starting with the OP-code located in absolute address 0000H.

Phase 2 is firmware initialization. All IMSAI systems utilizing the MPU-B multi-function processor PCB go through this phase following hardware initialization. The MPU-B firmware monitor is imaged into low system RAM (starting at location 0000H) and is executed as soon as the power-on reset condition ends. One of the first things the monitor does, is to 'relocate' itself into high system memory space (D800H-DFFFH) by switching the firmware ROM out of the 0000H-07FFH addressing range. This prepares the way for bringing the actual BOOT program off the diskette and into low system memory space 0000H-00FFH (of which more, below).

One of the functions of the monitor is to establish on which I/O port the system console (input/output device) is attached, and, in the case where this console is a serial data terminal, determine the serial baud rate. Another function is the 'booting' of the O/S from diskette into system RAM. The monitor interrogates the diskette controller(s) (if any) regarding their readiness to be accessed for a disk I/O transaction. Normally, there is no diskette in any drive at the time of power-on, and the monitor will go into its command processor loop.

Once the system is powered up, the system diskette (containing the IMDOS O/S) is placed in drive A and the O/S may then be booted in one of two ways. This is Phase 3 of the initialization process.

In this phase, the O/S is read from diskette into system RAM with subsequent activation of the command processor of the O/S. This may be invoked by inputing the Boot (B) command to the monitor, or by actuation of the RST switch on the operator's panel of systems that are equipped with such a panel. In either case, the 'bootstrap' of the monitor reads sector 1 of track 0 into low system RAM (starting at address 0000H), executes an OUT instruction to a control port, then executes a JMP to 0000H. The significance of the OUT instruction is that the MPU-B hardware will switch the monitor ROM out of the system address space as soon as the subsequent JMP instruction has been executed. This frees the high end of system address space for the subsequent loading of IMDOS proper.

The 128 bytes of program that had been brought in from sector 1 of track 0 contain the actual BOOT whose execution results in the transfer of the O/S into system RAM.

It must be noted in this connection that all IMDOS system diskettes feature single density, 18 sectors/track and 128 bytes/sector recording format on track 0. This makes it possible for any IMSAI system to boot the O/S off any system diskette, regardless of the recording density and format used elsewhere on the diskette.

Each of the disk controllers also has a bootstrap contained in its firmware. In systems that use the MPU-B, this bootstrap is used by the O/S in the event of a warm re-start request to the command processor (control-C). In systems that do not use the MPU-B (an I-8080 with an MPU-A, for example), the bootstrap contained in the controller is also used during a cold-start. The variations in utility of these bootstraps is explained in detail in the IMDOS User's Manual.

FIF CONTROLLER

The FIF controller firmware contains a bootstrap program that must be DMA'ed into system RAM before it can be executed. In a system that has an operator's panel (I-8080), the DMA operation that transfers the bootstrap program from FIF firmware into system RAM may be initiated by performing several (two to three) EXAM operations with the appropriate control switch on the panel. This action releases the CPU from the STOPed condition and permits DMA to take place. Once the bootstrap resides in system RAM, the CPU is allowed to commence execution of the code, starting with address 0000H. This is effected by pushing the RUN/STOP switch into the RUN position.

MDIO, DIO-C AND DIO-D

These controllers contain bootstraps with common firmware entry points for standard drives and for mini drives. They also have an additional entry point for system initialization. The entry points are located at these addresses:

E000H	Boot for standard diskette drive
E003H	Boot for mini diskette drive
E00CH	Controller initialization

Accessing either E000H or E003H of the MDIO firmware results in the execution of the bootstrap. This bootstrap first calls the initialization entry point, and then attempts to read sector 1 of track 0 of physical drive 9: (lowest number mini drive attached to the MDIO) into 0000H-00FFH of system RAM. The bootstrap ignores all error conditions and will attempt the load operation indefinitely. Upon successful loading of the sector, the MDIO firmware executes a JMP to 0000H.

Similarly, accessing E000H of the DIO-C firmware results in a boot operation involving physical drive 5: (lowest number standard drive attached to DIO). Access to E003H of a DIO-C returns an error condition code.

The DIO-D, being designed to handle mini drives exclusively, will return an error code for any access to E000H. A JMP to E003H, on the other hand, will result in a boot operation involving physical drive 9: (lowest number mini drive attached to DIO-D).

An access to E00CH in the MDIO firmware results in the presetting of several controller variables. One of these is a function of the drive type selection jumper. This jumper specifies either a 35-track drive with a 40ms track to track step rate, or a 40-track drive with a 12ms track to track step rate.

An access to E00CH in the DIO firmware also results in the presetting of several controller variables. These include initial recording density selection as well as specification of drive type.

The actual codes used to specify these variables, and the RAM array in which they are kept, are listed and explained in the user guides to be found in the technical reference subsections on the individual controller.

4 FIF CONTROLLER

4 - 1 INTRODUCTION

The functional and operational characteristics of the FIF Controller are presented in detail in this section. A functional description is given first, followed by a discussion of data transaction processes and the system protocol that governs them. Next comes the user guide in which configuration options are explained. The last section is the theory of operation; it is divided into two parts, each dedicated to one of the two boards that, together, comprise the FIF Controller.

4 - 2 FUNCTIONAL DESCRIPTION

The IMSAI FIF Controller provides for control of up to 4 standard diskette drives.

Data formats are fully compatible with the IBM 3740 format, providing a total storage capacity of 1.94 M bits per flexible disk. This is organized as 77 tracks with 26 sectors per track. Each sector contains 128 bytes of data.

To allow for non 3740 compatible formats, provision is made for reading all clock and data bits in an unformatted mode. The firmware may also be changed by reprogramming to support varying densities and formats.

The FIF is an intelligent dedicated controller with a DMA capability to free the main processor from the overhead associated with floppy disk control processing.

Commands to the Floppy Disk System are initiated from the main processor by the execution of an OUT instruction that references the FIF Port Command (Byte Command). The actual command is executed from a command string located in the main system memory. Up to 16 different command string pointers may co-exist at any one time, with the values of these command string pointers being User definable.

The Controller Set is composed of two boards, the IFM (Interface Master) and the FIB (Floppy Interface Board).

The IFM is a separate 8080 based processor used to control floppy disk functions. It contains an Intel 8080A, 512 bytes of RAM, 2K bytes of EPROM, and all support logic for the 8080A chip. Communications with the main system processor takes place through the DMA channel or the single output port.

The FIB contains all the control logic necessary to drive the floppy disk from the IFM processor.

The FIF Controller relies on external data separation and expects data and clock pulses on two separate input lines. This data separation is performed by a Phase Locked Oscillator (PLO) board that typically resides in the same cabinet that contains the diskette drive(s).

The FIF contains all the logic necessary to interface up to four floppy disk drives from the 8080A microprocessor. The firmware, located in the Controller's 2K bytes of EPROM, contains the driving program for the IFM processor and supports the IBM 3740 Data Format.

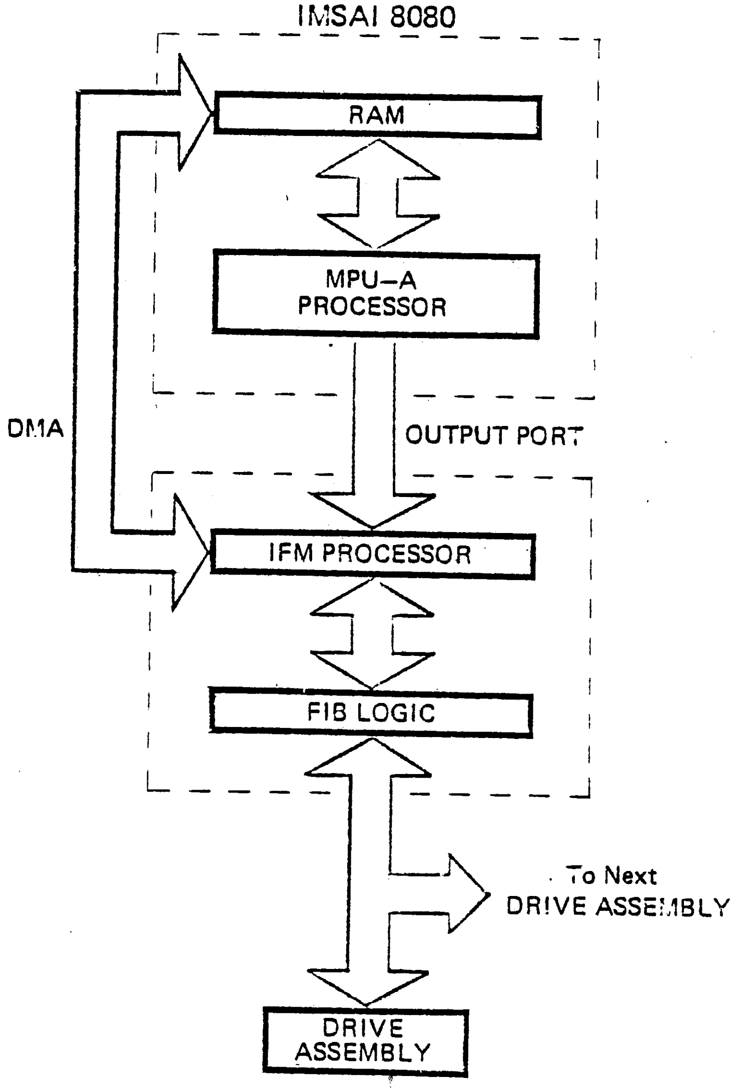


FIGURE 1 FIF - based System

The IMSAI Interface Master Board (IFM) is a complete 8080A based single board microcomputer configured as an intelligent interface controller. The overhead normally associated with peripheral processing is eliminated from the system microprocessor (SMPU) since: 1. the IFM moves data between the SMPU and the peripheral via a DMA Channel; and 2. the IFM exercises routine control over the peripheral.

IFM I/O

The SMPU communicates with the IFM via a bi-directional DMA Channel and an isolated output port. The electronics for both are implemented in the IFM and require no additional support circuitry. The DMA Channel normally gives the IFM access to the lower 32K bytes of the SMPU's memory. The isolated output port is used for issuing single byte commands to the IFM from the SMPU. Its address is jumper selectable and can range from 0 to 255 (decimal).

DMA PRIORITY

When used with other DMA devices, e.g. multiple IFM Boards, the DMA priority is resolved using a daisy chain scheme.

IMPLEMENTATION

The IFM is implemented using an 8080A microprocessor, its support chips, 2K bytes of EPROM, 512 bytes of RAM, and the additional electronics required to interface to the SMPU busses. The IFM is powered from the Mother Board and uses one slot. External interface connections to peripherals or other interface boards are made via a 50 pin edge connector at the top of the Board.

The Floppy Interface Board (FIB) is used in conjunction with an IFM board to form a complete floppy controller. The FIB uses the Mother Board Connector only to connect to an interrupt line and derive its voltage and ground from the proper pins.

The interface board works on a memory mapped basis and an 8205 three to eight line decoder is used to generate the proper command signals for the interface logic. All connections to the data bus are driven with tri-state drivers and all signals to the disk drives are driven with open collector drivers capable of sinking 40 milliamps. All lines from the disk drives are terminated with a resistor divider which is designed to match the impedance of the cable so there is no ringing effect on the signals. Two hex latches (74174s) are used to control the status information for the disk drive. These latches are loaded on output commands from the IFM board.

There are also two 24 pin, 8 bit shift registers used to accumulate the serial clock and data bits from the disk drives and then transmit them in a parallel mode to the microprocessor, or to receive a parallel load from the microprocessor and to transmit them serially to the disk. The write clock for the disk is derived by dividing down the 2 megahertz, phase 2 signal which comes from the MPU-A board using a 74LS193 counting chip. Finally the determination of when a byte has been assembled is done using a 74LS193 counting chip to count the 8 bits.

4 – 3 DATA TRANSACTION PROCESSES

The FIF observes the IMDOS disk I/O protocol as specified in section 3 – 3, and it supports single density Format I. Table 1 shows a detailed breakdown of the various fields in this sector format. It may be found in subsection 3 – 2.3, titled Format Compatibilities.

Because of its use of DMA in data transfers, some of the processes that take place during an FIF I/O transaction differ from those that are characteristic of Programmed Data Transfer Transactions. For this reason, the write and read processes already described in section 2 – 3 are reiterated here with appropriate adjustments to reflect operations that are unique to the FIF.

For additional clarification of the IMDOS disk I/O protocol, refer also to the Programming Guide of the IMDOS User's Manual.

FIF WRITE PROCESSES

MAIN SYSTEM PROCESSES

Assume there exists a block of data located in the main system RAM to be transferred to floppy disk. The main processor needs to first set-up the COMMAND STRING in main memory with

1) the command number for a sector write
2) zero in the Status Byte
3) the destination track and sector number
4) the address in memory of the data block to be transferred

The main processor then executes an OUT instruction referencing the IFM Command Port to initiate the execution of the write. The acknowledgement of a completed operation will be indicated by a non-zero value being stored in the status word of the COMMAND STRING, Thus, once the processor issues the BYTE COMMAND 0, it only need wait for the status word to go non-zero before proceeding with another disk operation.

CONTROLLER PROCESSES

When the IFM board receives the output byte (BYTE COMMAND 0), it lowers the Main Processor's READY line. The Main Processor goes into a WAIT state and the IFM Processor reads the output word from the system data bus into its own accumulator. Once the BYTE COMMAND is read, the IFM Processor raises the System READY line to allow the SYSTEM to continue. The output byte is decoded by the IFM firmware as being a command to execute from the COMMAND STRING located in the System RAM.

DMA FUNCTIONS

The IFM processor will present a HOLD REQUEST to the System, and the System will respond with HOLD ACKNOWLEDGED. At this point the System is in a HOLD STATE and the IFM processor will disable all of the Main Processor's address, data, and status lines (with the exception of PHLDA). The

IFM processor then gates its own address, data and status lines onto the System Bus.

The COMMAND STRING is now transferred to the IFM RAM from the System RAM, and the HOLD REQUEST is released, allowing the System to continue with its own processing activity.

The Command Number is decoded as a WRITE operation. The IFM processor transfers the Data block to the IFM RAM. This transfer is accomplished by DMA'ing a byte at a time in the HOLD MODE during state T3 (or the state following T3) of the main processor's machine cycle.

To proceed with the WRITE operation, the IFM processor computes and stores the CRC characters in its own RAM.

TRACK POSITIONING

A request is issued to the FIB to load the head, sync the PLO, and then to synchronize on the ID Address Mark. The FIB then places the IFM Processor in a WAIT State until it has found the desired missing clock pattern.

Once the FIB has recognized the Address Mark, it raises the IFM READY line, allowing the IFM processor to read and check the track address. A compare is made to see if the head is positioned over the desired track. If not, the direction and Step lines are used to reposition the head over the destination track.

SECTOR POSITIONING

Once again the IFM Processor issues a request to the FIB to synchronize on the ID Address Mark. The IFM is again placed in a WAIT State until the ID Address Mark is recognized. Once the IFM is allowed to continue, it reads and checks track and sector number, this time looking for the destination sector. If the head is verified to be positioned over the desired sector, the IFM processor waits 12 bytes before writing the remaining five 0 bytes, the Data Address Mark, 128 bytes of data, and the 2 CRC characters according to the IBM 3740 Data Format.

COMPLETION OF THE OPERATION

At this point, the Write operation is complete and the IFM board will indicate the results of the operation to the Main System by storing a non-zero value in the Status Word of the COMMAND STRING, via the DMA channel.

READ PROCESSES IN THE IMSAI FLOPPY DISK SYSTEM

MAIN SYSTEM PROCESSES

To prepare for a Read operation the main processor sets up the COMMAND STRING in its RAM with:

1) The Command Number for a sector Read;
2) zero in the Status Byte;
3) the track and sector number for the data block to be read from the diskette;
4) the Address of the destination in Main memory.

The main processor then issues BYTE COMMAND 0 to initiate the READ operation. The main processor waits until the Status Byte of the COMMAND STRING goes non-zero before proceeding with another disk operation.

CONTROLLER PROCESSES

As before, the IFM processor will receive the output byte from the main processor and decode it as an execute from COMMAND STRING.

The COMMAND STRING will be transferred from the System RAM to the IFM memory via the DMA access channel. The Command Number is decoded as a READ operation and the IFM processor positions the read/write head over the desired track and sector as before. Once the head is correctly positioned, the IFM processor waits for the Data Address Mark.

When the Data Address Mark is recognized, 128 bytes of data are read into the IFM RAM. The two CRC characters are then read and checked to verify the data block. If the data block is verified, it is written into the main processor's RAM via the DMA channel.

To acknowledge completion of the READ operation, the IFM processor will store a non-zero value in the Status Byte. This value is then passed to the COMMAND STRING located in the main system RAM via the DMA channel.

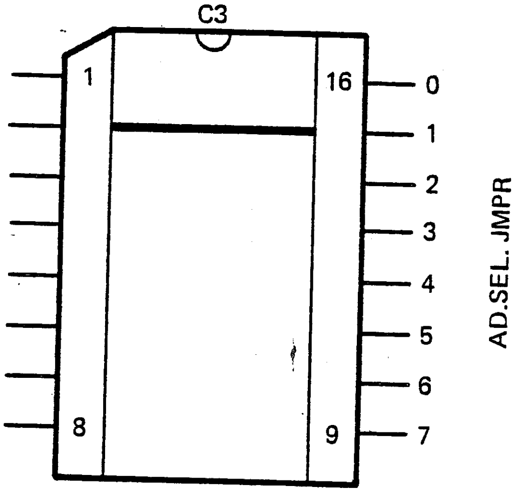
4 – 4 USER GUIDE

PORT ADDRESSING

The address of the Command Output Port must be selected by jumpers (or a DIP switch) in position C3. Socket pins 1 through 8 correspond to Address Bits A0 through A7 respectively. Pins 9 – 16 are used for ground connections.

The address is selected as follows. For each Address bit which is to be a "0", the pin corresponding to that Address bit must be jumpered to ground. If a "1" is desired in any bit position, the corresponding jumper position is left open.

For example, to select address FD Hex, pin 2 is connected to ground pin 15. Pins 1, 3, 4, 5, 6, 7 and 8 are left open. To select address F6 Hex, pins 1 and 4 are connected to ground pins 16 and 13. Pins 2, 3, 5, 6, 7, and 8 are left open. FD is in IMSAI standard port address for floppy disks. Figure 2 depicts these connections.



JUMPERS SHOWN SET FOR FD

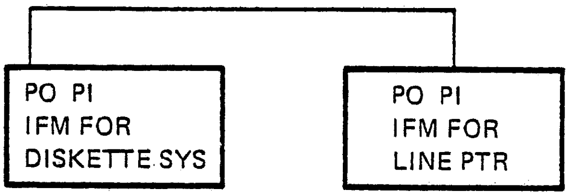
FIGURE 2

DMA PRIORITY

When multiple DMA channels are used in an IMSAI System, they are usually serviced on a first-come, first-served basis. When coincident DMA requests occur, priority is resolved using a daisy chain technique. To implement this, the Priority Output (PO) of one IFM Board is connected to the Priority Input (PI) of the IFM Board with the next lower priority. When using multiple IFM controllers, care must be taken when putting the SMPU into a Hold State for the duration of a block transfer of data, as all other controllers, regardless of priority, will be locked out for the duration of the transfer. Figure 3 depicts the DMA Priority connections for a Floppy Disk controller.

EXTERNAL INTERFACE CONNECTIONS

All External Interface Connections are made via the 50 pin edge connector, J2, at the top of the IFM Board. Refer to Table 1 for a list of signals available and the corresponding pin assignments.



DMA PRIORITIZATION: 1. DISKETTE
2. LINE PTR

FIGURE 3

J2 SIGNAL DEFINITIONS

PIN	SIGNAL NAME	PIN	SIGNAL NAME
1	RESET	2	/E7
3	/RESET	4	/E6
5	T0(2)	5	/E5
7	RDYIN	8	/E4
9	MA10	10	/E8
11	MA11	12	MA14
13	MA13	14	MA12
15	MA15	16	MA9
17	MA8	18	INTE
19	WAIT	20	SYNC
21	MEMRW	22	INT
23	NC	24	HOLD
25	-5V	26	/INTA
27	DMA15	28	/MEMR
29	NC	30	/MEMRW
31	DISBUS	32	PADDR
33	DBO	34	MA7
35	DB1	36	MA6
37	DB2	38	MA5
39	DB3	40	MA4
41	DB4	42	MA3
43	DB5	44	MA2
45	DB6	46	MA1
47	DB7	48	MA0
49	+5V	50	+5V

TABLE 1

IFM I/O

The IFM communicates with the SMPU via a DMA Channel and a Single Byte Command Port. As viewed from the IFM's CPU, all IFM Input/Output is Memory Mapped (Table 3). Input and output instructions DB Hex and D3 Hex must not be used in the IFM's Firmware, or System malfunction will result.

COMMAND PORT

As viewed from the SMPU, the Command Port is an isolated output port whose address can be selected on the IFM Board. The address may range from 0 to 255 (decimal). When the SMPU outputs a byte to the Command Port, the IFM places the SMPU into a Hold State until it is ready to accept another command byte. This simplifies SMPU software requirements, as no additional data transfer synchronization (e.g. status bit) is required. A sequence of commands may be programmed as an inline string of output instructions.

SYSTEM BOOTSTRAP

A "bootstrap" is a short program which reads another program from some storage medium into system RAM and executes it. This simple, yet general procedure gives the user freedom to load in any kind of operating system s/he desires. The IMSAI Floppy Disk System bootstrap reads sector 1 of track 0 from drive 0 into system RAM at 0-7F and then jumps to location 0.

The following procedure should be used when bootstrapping from an IMSAI IMDOS System Diskette.

1. Insure that the diskettes are removed from the drives.
2. Power up the 8080.
3. Power up the floppy disk drive.
4. Insert a system diskette in drive 0.
5. When the drive becomes READY, press RESET.
6. Set the ADDRESS switches for 0000 and press EXAMINE.
   A "C3" should appear in the DATA lights.
7. Press RUN.

At this point, the operating system should be loaded and run.

If a hardware error occurs, the error code (see FIB Software section) will be displayed in the PROGRAMMED OUTPUT lights. The bootstrap will be retried until it is successful, or until it is stopped. If unable to bootstrap, check the points described at the end of the SYSTEM INITIALIZATION section.

If REV 3 Firmware is not available, a Bootstrap Simulator is described in the IMSAI IMDOS Documentation.

4-5 THEORY OF OPERATION

4-5.1 INTERFACE MASTER (IFM)

The operation of the IFM Board is easily understood if it is broken down into its major functional blocks. The IFM consists of:

1. CPU Circuits
2. On-Board Address Decoding
3. Memory Circuits
4. I/O Circuits
5. DMA Circuits
6. DMA Priority Circuits

CPU

The CPU employed by the IFM consists of an 8080A microprocessor, an 8224 System Clock, and an 8228 System Controller. This circuitry is described in detail in the Intel 8080 Microcomputer System User's Manual.

IFM ON-BOARD ADDRESS DECODING

IFM Address Lines MA10 - MA15 are decoded by the 8205, located in position A5, to originate signals /E1 through /E8.

The 8205 is enabled when MA15, MA14, and MA13 are present on the IFM Address Bus in a low state. At this point, MA10 - MA12 are decoded to select one of the eight outputs /E1 - /E8, which are active low. Note that /INTA going low will disable the 8205.

The outputs /E1 - /E8 are used for gating and selecting on-board functions as listed in Table 4. They may also be used for external device and peripheral control functions, as /E4 - /E8 are available at the connector J2.

MEMORY

The IFM employs 2K Bytes of Eraseable Programmable Read Only Memory (EPROM) and 512 Bytes of Random Access Read Write Memory (RAM). The EPROM consists of two 8708 chips and the RAM consists of four 8111 chips.

On-board RAM is selected when the /E3 output of the 8205 (A5) goes low. This will occur when the address 08XX - 09XX appears on the IFM Address Bus.

The IFM RAM is organized as two blocks of 256x8 bit words. Each block of 256 is composed of two 4 bit half blocks since the organization of the 8111 is 256x4.

A RAM chip is selected when its inputs /CE1 and /CE2 are both low. Since the input of /CE2 is /E3 and the input to /CE1 is driven from MA8, an address of 08XX will select chips B9 and B12, while an address of 09XX will select chips B10 and B11.

Selection of an 8 bit word within a block of 256 is achieved through the use of MA0 – MA7. Output Disable and R/W inputs to the RAM chips are driven by /MEMR and /MEMW respectively.

The IFM PROM consists of two 8708 chips, organized as 1Kx8. PROM 0000, location A10, is enabled when /E1 goes active low. This will occur when an address in the range of 0000 – 03FF appears on the IFM Address Bus. Similarly, PROM 0400, location A11, is enabled when /E2 goes active low. This will occur when an address in the range of 0400 – 07FF appears on the IFM Address Bus. Selection of a word within each 1K block of PROM is achieved through the use of MA0 – MA9.

All of the IFM's memory (both EPROM and RAM) has less than 500 nanosecond access time so no WAIT states are used for memory references.

Table 4 On–Board Address Decoding

MA0 – MA15 ADDRESS	SIGNAL	FUNCTION
0000 – 03FF	/E1	Enable PROM 0
0400 – 07FF	/E2	Enable PROM 1
0800 – 09FF	/E3	Enable RAM
0C00 – 0FFF	/E4	not assigned
1000 – 13FF	/E5	not assigned
1400	/E6	Command Data Port
1800	/E7	Command Status Port
1C00 – 1FFF	/E8	not assigned

INPUT/OUTPUT

All I/O on the IFM Board is implemented using the Memory Mapped I/O technique. The I/O facilities consist of a single byte command port from the SMPU and any control ports implemented in the peripheral or external interface circuitry (e.g. FIB or LIB boards). The DMA channel can be viewed as 32K I/O ports but because of the extensive circuitry involved, it will be considered separately.

As viewed from the SMPU, the single byte command port is an isolated output port. When the SMPU outputs to this port, the address is decoded by the 7485 Comparators (C4 and C5) causing PADR to go high. This is gated with PWR and SOUT by a 74LS11 and gate (B3), whose output CIN is high if and only if the SMPU is outputting to this port. CIN is connected to the STB input of the 8212 8–bit Latch (C8) and causes the data on the SMPU Data Bus to be loaded into the latch. This, in turn, sets the 8212 Internal Service Request Flip/ Flop, causing /IORQ to go low and a Hold Request to be initiated. (Hold Request and DMA logic is discussed below.)

As viewed from the IFM CPU, the single byte command port consists of a Memory Mapped Status Input Port and a Memory Mapped Data Input Port. The Status Port address is fixed at 1800H. When 1800H is decoded by the 8205 (A5), it causes /E7 to go low. This in turn, places IORQ on bit 0 of the Data Bus.

The Memory Mapped Data Input Poart address is fixed at 1400H. When 1400H is decoded by the 8205 (A5), it causes the signal /E6 to go low. This enables the Tri–State outputs of the 8212, allowing the CPU to read the contents. /E6 also resets the 8212 Service Request Flip/Flop, causing /IORQ to go high and the associated Hold Request is removed.

Control I/O ports may be implemented in the peripheral equipment or external interface circuitry. This is possible since all of the IFM's busses are available at the J2 connector.

DIRECT MEMORY ACCESS

As viewed from the IFM CPU, the DMA Channel can be considered to be 32K Memory Mapped I/O Ports. Alternately, it can be considered as a virtual extension of the IFM memory. A read or write in the upper 32K bytes of the IFM address space causes a DMA read or write in the lower 32K of the SMPU address space. This can be changed to the upper 32K bytes of the SMPU address space under IFM program control, if the requisite latch is implemented on an additional board (e.g. LIB or FIB).

A DMA transaction is initiated when MA15 is high and either /MEMR or /MEMW is low. The IFM CPU is put into a Wait State, (RDYIN pulled low), and a Hold Request in initiated (A2-11 goes high).

If /IORQ and /PRIIN are high, the CLR input to the shift register at A3 goes high. At this point, the 74195 shift register, A3, is in the LOAD mode. If /PHOLD is not currently being pulled low by another DMA device, the output Qa of A3 is latched high on the trailing edge of the SMPU 02 clock, (S02). The output Qa of A3 drives /PHOLD active low.

The SMPU acknowledges its entry into the HOLD State by driving PHLDA high. This, in turn, latches the Qb output of A3 high. The SMPU Drivers are disabled at this time and the IFM drives the SMPU Address, Control, and Status Busses. Note: When a byte is output to the single byte command port, a sequence of actions occurs similar to that which put the SMPU into a Hold.

Since the Qb output of A3 is high, the signal, DMA, (A2-6) goes high, placing the 74195, A3, into the SHIFT Mode. The 8216 Bi-Directional Bus Drivers, (C6 and C9), are also enabled at this time, with the direction of the data transfer determined by /MEMR.

On the next trailing edge of S02, the Qc output of A3 goes high, driving PWAIT high. /PWR is also driven low at this time if a write operation is to be executed.

One S02 period later, the Qd output of A3 goes high. At this time PRDY is gated to drive RDYIN. Thus the IFM CPU resumes operation when PRDY is high. The DMA transaction terminates when both /MEMR and /MEMW are high.

If, at the end of the DMA transaction, /IORQ is high, the CLR input to A3 goes low and control of the Busses is returned to the SMPU. /PHOLD goes high. The SMPU ceases Holding and resumes normal operation until the next DMA operation.

If, at the end of a DMA transaction, /IORQ is low, a single byte command is pending and the CLR input to A3 remains high. Since the DMA REQ, A2-11, is now low, DMA, A2 - 6, goes low and the 74195 shift register, A3, returns to the LOAD Mode. The first two bits of A3 remain high, but the second two bits go low. The SMPU continues to Hold. The 74195 remains in this state until /IORQ goes high or another DMA transaction is initiated. Thus, the SMPU is held after outputting a byte to the IFM until the IFM CPU reads the byte. The IFM is able to continue DMA operation during this time and will DMA slightly faster since it no longer is delayed by waiting for PHLDA.

DMA PRIORITY

Conflicts for DMA access by multiple DMA channels is resolved using a daisy chain scheme. The /PRIOUT output of a DMA Controller is connected to the /PRIIN input of the controller with the next lowest priority. The /PRIIN input of the highest priority controller is left open and is pulled up by an on-board 1K resistor.

A controller pulls its /PRIOUT low if its /PRIIN input is low or if that controller is executing a DMA transaction. /PRIIN low indicates that a higher priority controller is executing a DMA transfer. Any pending DMA request must be delayed until /PRIIN returns high and /PHOLD is high.

If /PRINN and /PHOLD are high, the IFM may initiate a DMA transaction by pulling /PHOLD low synchronized to the trailing edge of the SMPU 02 Clock. Simultaneously, /PRIOUT is pulled low. If /PRIIN remains high until the next trailing edge of S02, no higher priority DMA device requires service, and the DMA transaction may continue. However, if /PRIIN is pulled low, a higher priority channel requires service and the transaction must be aborted. DMA priority applies only to DMA transactions initiated on the same trailing edge of S02. All other DMA transactions are serviced on a first-come first-served basis. Once a DMA transaction is in progress, higher priority controllers must not initiate a DMA transaction or system malfunction will occur.

4 - 5.2 FLOPPY INTERFACE BOARD (FIB)

INTERNAL ADDRESS DECODING

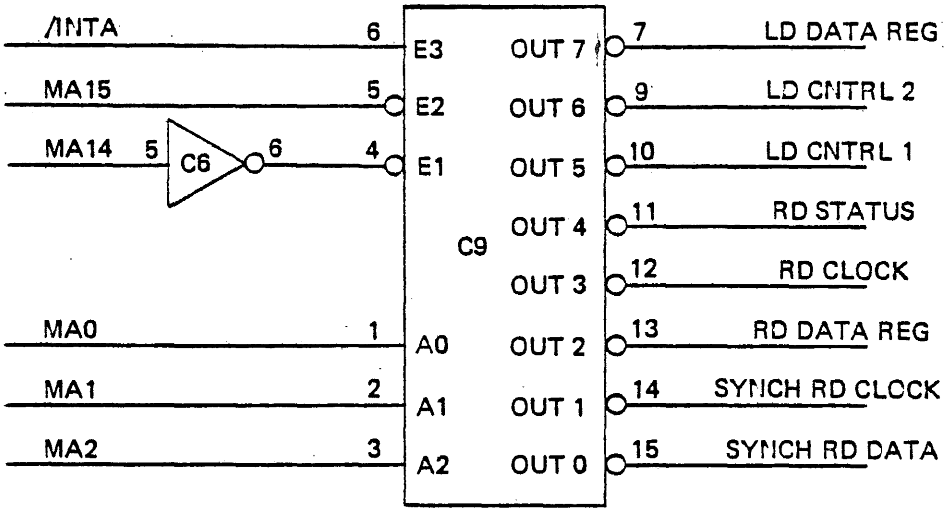


FIGURE 4 Internal Address Decoding

All major functions on the FIB board are controlled from the IFM processor via a memory-mapped I/O technique. Address decoding is achieved through the use of the 8205 in position C9.

The 8205 is enabled when MA15 is low and MA14 is high. It may be disabled when INTA goes low.

Once C9 is enabled, MA0 - MA2 are decodes to produce one of eight select function. The function of each select line is given below.

MA0 - MA15 = 4000: SYNCH ON READ DATA

The command is used to tell the floppy interface to go into a synchronization mode. The interface will then issue a HOLD command to the microprocessor until it has shifted a clock pattern into the shift register which has bit 5 missing. Note that the input data to the shift register is complemented, so this actually looks like bit 5 of the 74198 is a one.

When this bit becomes a 1, it says that the clock byte is missing a bit in this position. All missing clock patterns of significance in the 3740 have this bit as the first missing clock bit. Once this bit is detected, the microprocessor is allowed to read first the data and then the clock shift registers.

The firmware then compares them in order to determine whether or not this is the proper address and clock byte.

The read for this must be done with a 2 byte memory reference instruction to ensure the rapid timing between reading of the data byte and the clock byte (LHLD). This will result in the clock byte being read 2.5 microseconds after the data byte.

Since in a double density disk drive this would be longer than the next bit time, the clock register is inhibited from shifting during the time that we are waiting to read this clock byte. This is accomplished by pulling the mode control to 0 on the 74198.

MA0 - MA15 = 4001: SYNCH READ CLOCK

This is the Synch Read Clock Command. The command, as mentioned above, should be generated by referencing a Sync Read Data address (4000) with a two byte instruction (LHLD). When this command is operational, it gates the clock shift register onto the data bus through the 8T98 tri-state drivers. Note that inverting drivers are used because that clock and data registers both shift in the complement of the data.

MA0 - MA15 = 4002: READ DATA REGISTER

MA0 - MA15 = 4003: READ CLOCK REGISTER

These two commands operate in the same mode as the 4000 and 4001 commands with the following exception. When an input from the data register is to be accomplished, then the microprocessor must be held in a WAIT state until 8 bits have been assembled so that there is a complete byte.

This differs from the SYNCH READ CLOCK and SYNCH READ DATA commands where we are waiting to synch on a clock pattern. Once we have received the 8 bit byte, the data from the shift register is gated onto the data bus using the inverting tri-state drivers, and the microprocessor is taken out of the HOLD state.

Note that the single reference to the read data register will cause the data register to be transmitted. Only the read data operation may be used when attempting to read data from a floppy disk which has a bit time faster than 2.5 microseconds since, otherwise, the clock register will become out of synch.

MA0 - MA15 = 4004: READ STATUS

This command causes the contents of the disk drive status to be gated into the microprocessor. The sector input pulse is input to the microprocessor even though it is not used elsewhere in the design. It could be used at some later time to initiate a firmware system which works with a hard sectored disk as opposed to the IBM electronically sectored disk.

MA0 - MA15 = 4005: LOAD CONTROL 1

This command causes the content of the data bus to be transferred to the lower half of the floppy disk control register. The gating of these signals can then be controlled by the microprocessor to select a disk drive, to load the head on that drive, and to indicate to that drive when it is above track 43.

MA0 - MA15 = 4006: LOAD CONTROL 2

This command causes the contents of the data bus to be transferred to the upper half of the control register. It is used to initiate or generate seeks on the disk drive, to put the drive in the WRITE ENABLED mode, and to synch the PLO on the drive.

The command also sets a flip-flop which forces clock bits of all ones and data bits of all zeros to be written for one byte. This feature is used to initialize the WRITE logic when the WRITE ENABLE is first turned on. After this time, the controlling of logic is by the load data register command. The command is also used to clear the index pulse holding flip-flop whenever it is issued with DB07 a one.

The top two bits of this register are used to enhance the communication between the IFM and MPU memory. The firmware gets the value for this bit from the memory locations (bit 15) called out by the command strings and then forces bit 15 to be a one for internal operations so the DMA will function properly.

The second bit is used as an interrupt request bit and can be jumpered any of the eight lines used as interrupt request inputs for the FIB. This bit is set whenever a command string execution has been completed (i.e., when the status byte value is modified from zero). It is cleared whenever a command is accepted from th MPU. Note that any command not defined is treated as a NOOP command and may be used to clear this bit.

MA0 - MA15 = 4007: LOAD DATA REGISTER

Load Data Register commands will transfer the contents of the data bus to the parallel shift register. The clock register is also loaded on the command with a value of FF, EF, D6 or C7 for memory address bits 3 and 4 equal to 00, 01, 10 or 11 respectively.

CONTROL OF IFM READY LINE

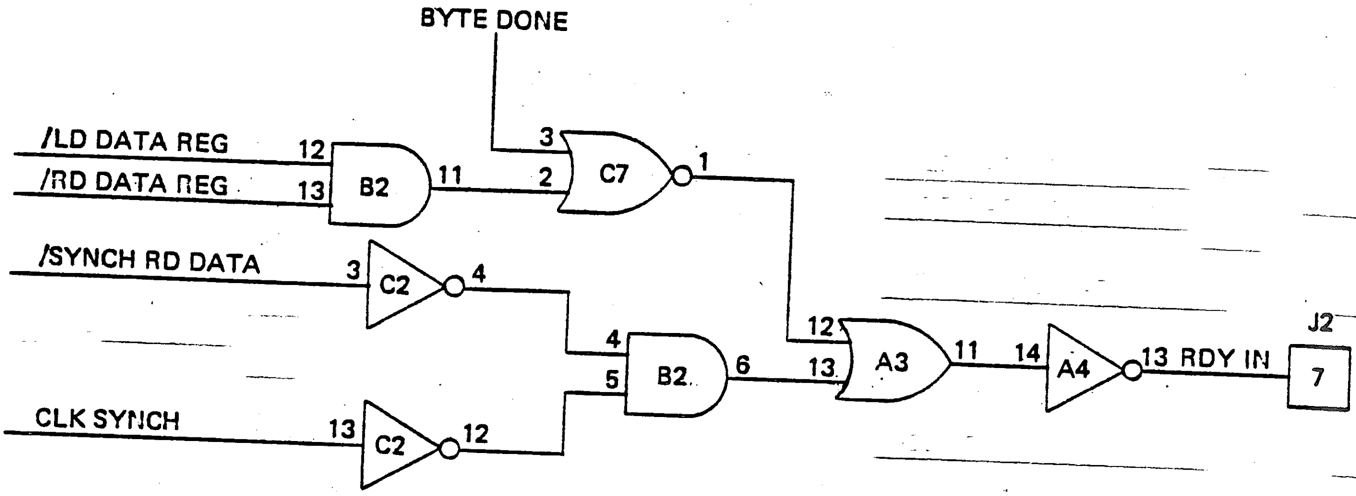


FIGURE 5 Control of IFM Ready Line

If a SYNCH READ DATA COMMAND (4000) is issued to the FIB board, the output of the AND gate a B2 – 6 will go high. The high level signal is inverted at A4 – 13, lowering the IFM READY line.

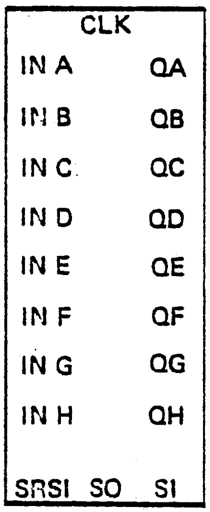
The IFM processor remains in a WAIT state until CLK SYNC appears high at C2 – 13. CLK SYNC goes high when a missing clock pulse is detected in bit position 5 of the clock shift register. This causes the output of the AND gate B2 – 6 to fall and the READY line to rise.

If either a READ DATA REF (4002) or LD DATA REG (4007) command is issued to the FIB board, the output of the OR gate at A3 – 11 will go high. This gets inverted at A4 – 13, and the IFM READY line is lowered.

The IFM processor will remain in the WAIT state until the BYTE DONE signal appears high at C7 – 3. The BYTE DONE signal becomes active when a 8 bit byte has been assembled in the shift registers (for a read) or 8 bits have output (written) to the disk.

The appearance of an active BYTE DONE signal causes the output of the OR gate A3 – 11 to fall. After inversion at A4 – 13, the READY line is raised, allowing the IFM processor to enter the RUN state.

THE 74198



SO	SI	FUNCTION
1	1	PARALLEL LOAD OUTPUTS = PARALLEL INPUTS ON RISING EDGE OF CLOCK
1	0	SHIFT RIGHT ON RISING EDGE OF CLOCK – $Q_{n+1} = Q_n$ / Q_A = SRSI
0	0	DATA IS STABLE AT OUTPUTS QA – QH

FIGURE 6 The 74198

The 74198 is a serial/parallel load shift register. This chip is used to assemble and transmit both the data and clock pulses used by the floppy disk. The functions used are shown in figure 7.

CLOCK SHIFT REGISTER

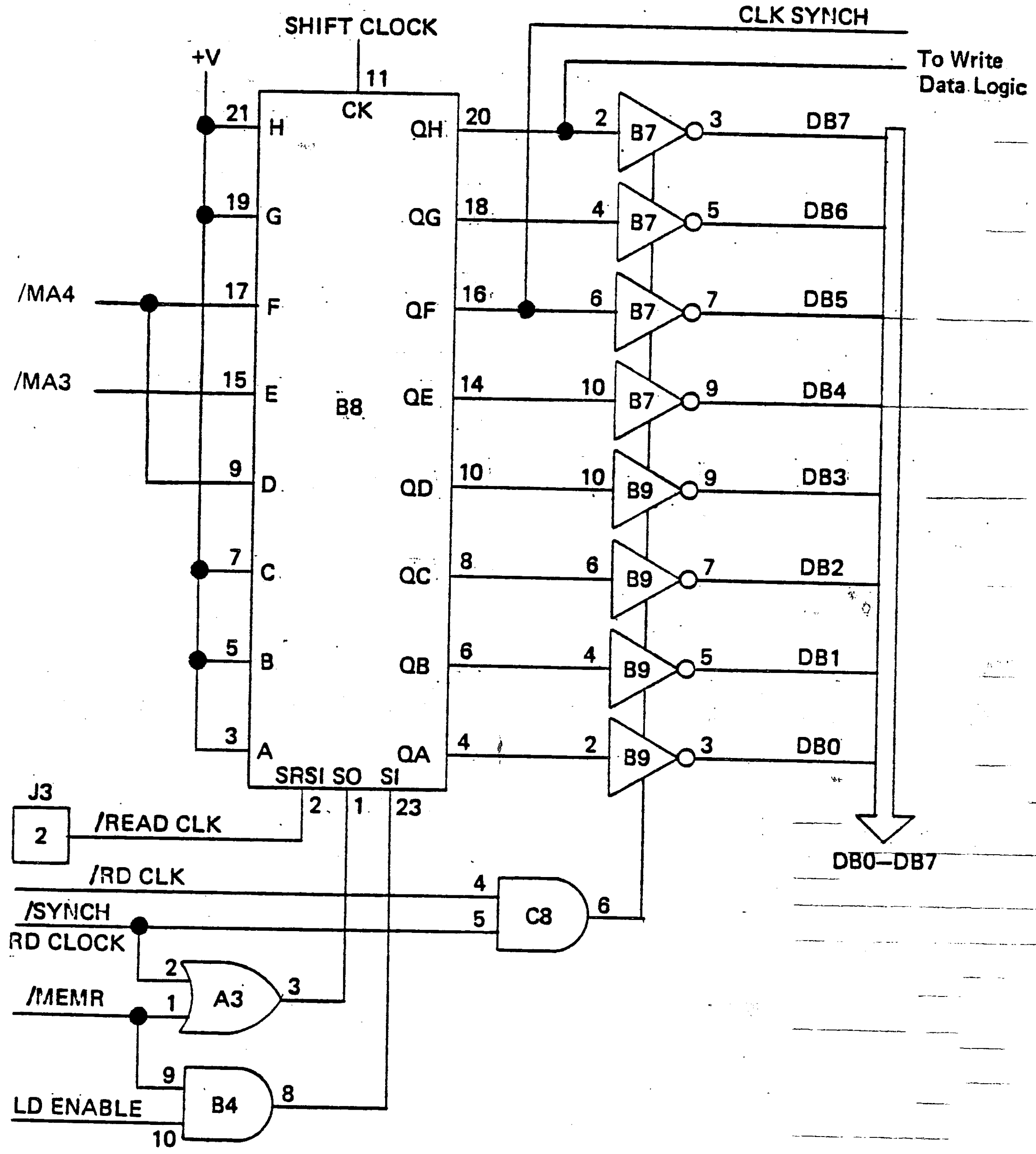


FIGURE 7 Clock Shift Register

The 74198 in position B8 is used to output and input clock signals to or from the floppy disk.

READING THE CLOCK

A SYNC RD CLD (4001) command or a RD CLK (4003) command will output the 8 parallel bits of the 74198 at B8 onto the IFM Data Bus by driving the output of the AND gate at C8 - 6 low to enable the inverting drivers at B7 and B9.

Note: 8T98 inverting drivers are used since clock and data is input from the disk in inverting form.

Use of the SYNC RD CLK (4001) and RE CLK (4003) command-assumes that an 8 bit word has been shifted into the clock shift register. These commands should only be used following a SYNC RD DATA command (4000) or RD DATA command (4002). To insure the proper timing of the read, only the LHLD instruction should be used to access the clock shift register.

SERIAL LOADING

The READ CLOCK line from the disk provides the serial input to the 74198 shift register. Clock data is input at the SRSI input to the 74198 and is shifted right on the positive edge of the shift clock.

WRITING CLOCKS

The shift register at B8 is also used in writing missing clock patterns to the disk. If a write to 4001, 4009, 4011 or 4019 occurs, one of four missing clock patterns is parallel loaded into the shift register.

A write to any one of these four addresses causes the S0 input of B8 to go high. When LOAD ENABLE goes high, the S1 input to B8 also goes high, and the shift register is loaded from the parallel inputs. MA3 and MA4 are used in writing missing clock pulses in positions D3, D4 and D5.

Once the parallel clock data is loaded, it is written to the disk. The serial output appears at the output Qh of B8 and shifts right on the rising edge of the Shift Clock.

DATA SHIFT REGISTER

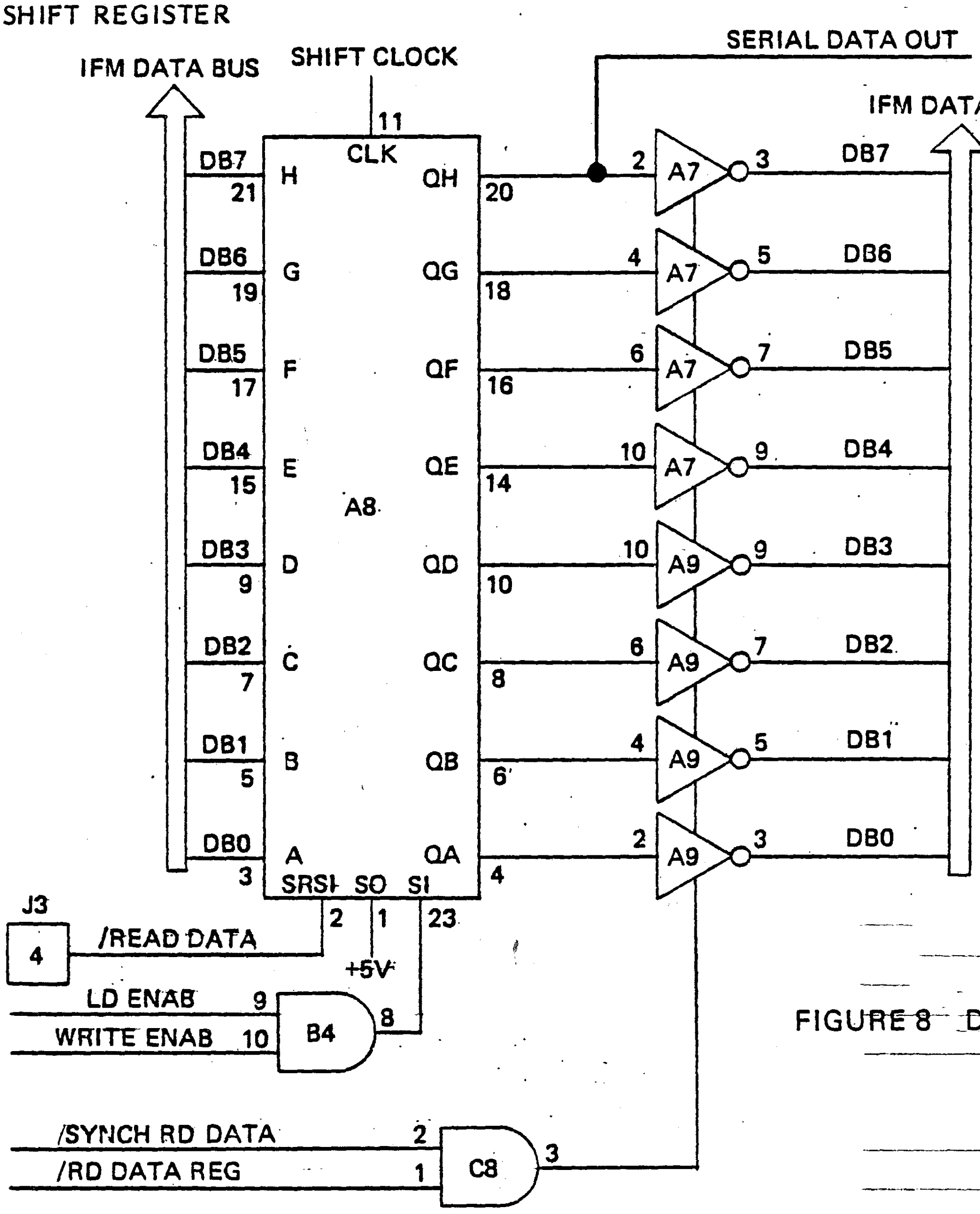


FIGURE 8 Data Shift Register

PARALLEL LOADING

If the WRITE ENABLE is set, the LD ENABLE signal will gate parallel data from the IFM Data Bus into the data shift register at A8.

Once the parallel data is loaded, it is written to the disk on the rising edge of the Shift Clock. Serial data output appears at Qh of A8.

SERIAL LOADING

When the input S1 equals 0, serial data from the floppy disk is input to the shift register in inverted form at the SRSI input of A8 on the rising edge of the Shift Clock.

READING THE SHIFT REGISTER

If a SYNC RD DATA or READ DATA REG appears, the tri-state drivers at A7 and A9 are enabled. This allows the IFM processor to read the parallel outputs of the data shift register.

SYNCHRONIZATION

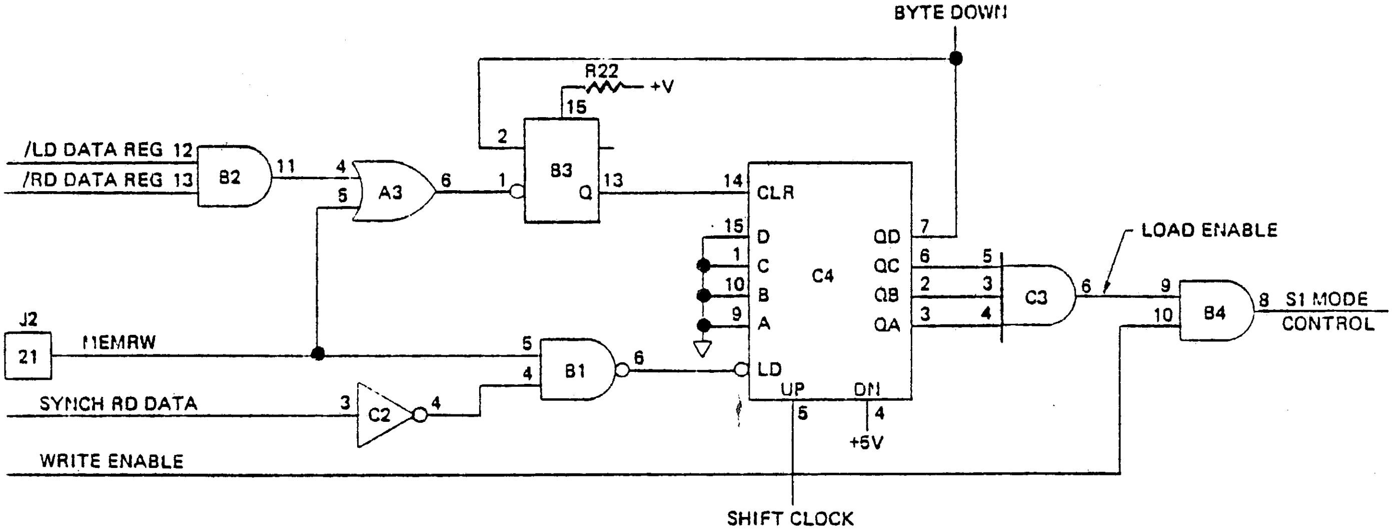


FIGURE 9 Synchronization

WRITE SYNCHRONIZATION

The LD DATA REG signal is used to load the data shift registers on a WRITE operation. The trailing edge of the LD DATA REG signal is used to fire the one-shot (B3) to clear the 4 bit counter (C4).

Count pulses are provided by the shift clock concurrent with clock and data pulses to/from the disk. At a count of seven, the LOAD ENABLE signal appears at the output of the AND gate C3 - 6. If the WRITE ENABLE is set, the S1 Mode Control signal goes high to allow parallel loading of the data and clock shift registers at B8 and A8 once the BYTE CONE signal appears.

At the count of 8, the Qd output is used to generate the BYTE DONE signal indicating that an 8 bit byte has been output from the clock and data shift registers to the floppy disk.

READ SYNCHRONIZATION

When the SYNCH RD DATA line goes low, the counter at C4 is held clear. Once the IFM processor recognizes the missing clock pattern SYNCH RD DATA will go high to enable the counter to start the count. Data is synchronized at this time.

The RD DATA REG line goes low, the IFM processor waits for the BYTE DONE before reading the 8 bits of data from the shift registers. Once RD DATA REG goes high, the one-shot at B3 is fired to clear the counter C4, and the count repeats itself for the next 8 bits read from the disk.

CLOCK W CLOCK/ DATA W CLOCK GENERATION

Both Write clocks are generated using the 4 bit counter at C5 to divide down the 02 clock. These two clocks are on opposite 2 microsecond clock intervals, and both clocks have a pulse width equal to the IFM 02 clock pulse width (275ns).

If the WRITE ENABLE is not set, both clocks are disabled since the LD input to the counter goes to zero and all outputs Qa - Qd go low.

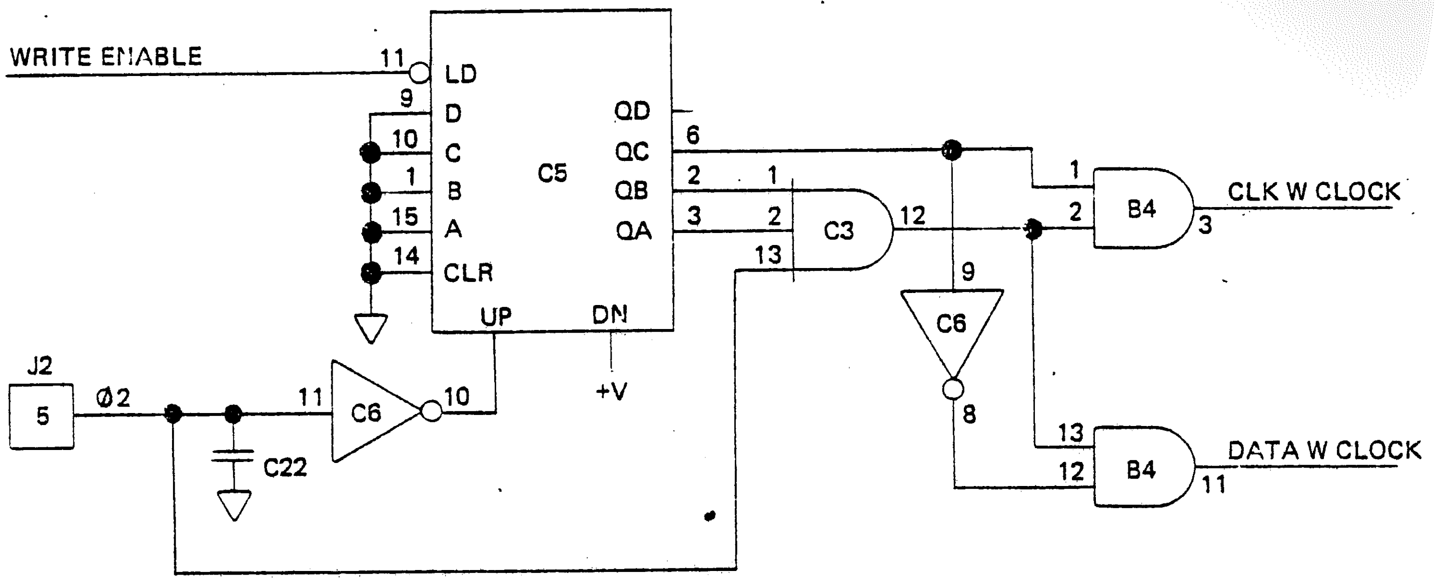


FIGURE 10 Clock W Clock/Data W Clock Generation

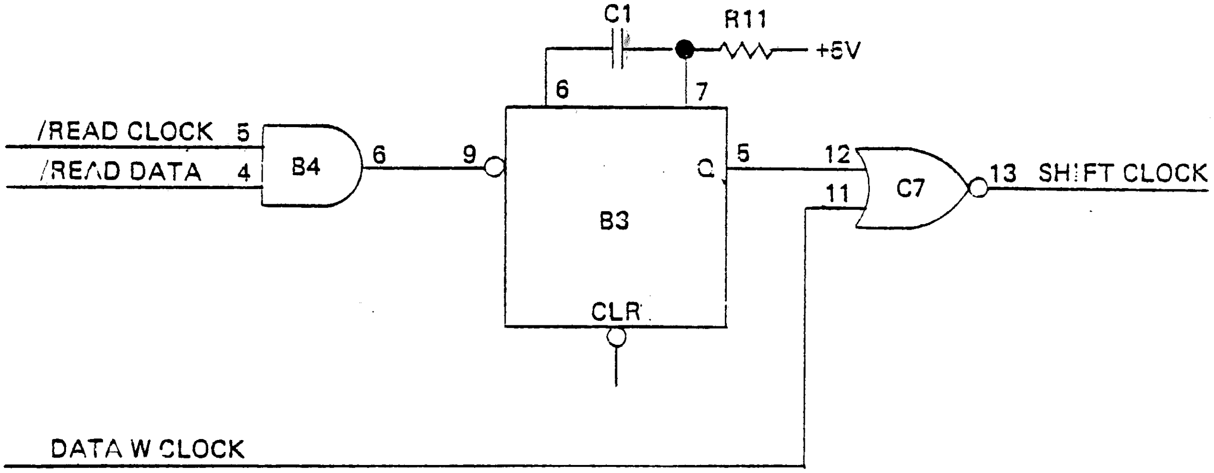


FIGURE 11 Shift Clock

4 – 6 SCHEMATICS AND I/O INTERCONNECT

FIB TO DRIVE INTERCONNECTION

J 3 PIN #	SIGNAL NAME	SIGNAL DIRECTIONAL
1	GND	
2	READ CLOCK	<——
3	GND	
4	READ DATA	<——
5	GND	
6	WT DATA	——>
7	GND	
8	READY 2	<——
9	READY 3	<——
10	INDEX	<——
11	READY 1	<——
12	TRACK 0	<——
13	READY 0	<——
14	SECTOR	<——
15	HEAD LD	——>
16	WT ENABLED	<——
17	SELECT 0	——>
18	PLO SYNC	<——
19	SELECT 1	——>
20	WT ENABLE	——>
21	SELECT 2	——>
22	DIRECTION	——>
23	SELECT 3	——>
24	STEP	——>
25	ABOVE TRK 43	——>
26	GND	

NOTE: All signals are low true, except DIRECTION which is bi-polar.

——> means: control signal to drive
<—— means: sense line from drive

5 MDIO CONTROLLER

5 - 1 INTRODUCTION

This section presents a detailed discussion of the functional and operational characteristics of the MDIO diskette formatter/controller.

The first sub-section (Functional Description) describes the major characteristics of the MDIO on a block diagram level. The next sub-section contains little more than a vector to another section in this manual in which the IMDOS disk I/O protocol is explained in detail. A subsection, called User Guide comes next; here, various user options are explained. This is followed by a Theory of Operation and support documentation.

5 - 2 FUNCTIONAL DESCRIPTION

GENERAL

The MDIO is a single board, S100 Bus compatible mini-diskette formatter/ controller. It supports up to 3 mini-diskette drives and an IBM 3740 compatible (single density, FM) recording format. Through the use of the FORMAT utility available with IMSAI's IMDOS operating system, the MDIO may be directed to FORMAT mini-diskettes in one of four possible formats:

1) 18 sectors of 128 bytes/sector, 35 tracks/disk
2) 9 sectors at 256 bytes/sector, 35 tracks/disk
3) 18 sectors at 128 bytes/sector, 40 tracks/disk
4) 9 sector at 256 bytes/sector, 40 tracks/disk

The number of tracks recorded on each diskette is a function of the drive type. MPI B51 drives support 40, Shugart 400 drives support 35 tracks.

The MDIO is accessed by the execution of CALL instructions that reference one of several entry points of the MDIO firmware. Such a CALL initiates a disk I/0 with the control byte that is conveyed in the A-Register. The nature of the requested transaction is specified by a Command String which is located in system main memory.

The MDIO firmware interprets this Command String and performs the desired data transfer. Upon completion of the transaction, the MDIO returns to the CALLer with a status describing the outcome.

IMPLEMENTATION

The MDIO is designed around an LSI-chip, the 1771, which performs a majority of the mini drive control and all of the data and status transfer operations that implement MDIO firmware command sequences. These sequences, in turn, are activated by system calls requesting diskette access. The remainder of the drive control requirements are handled by a memory-mapped port.

Functional Description

MDIO occupies 4K bytes of the system address space (E000 - EFFF) which may be released to RAM-space under system control. Within this 4K space, MDIO contains the MDIO firmware and 256 bytes of on-board RAM. The firmware is located in a 2Kx8 ROM/PROM module.

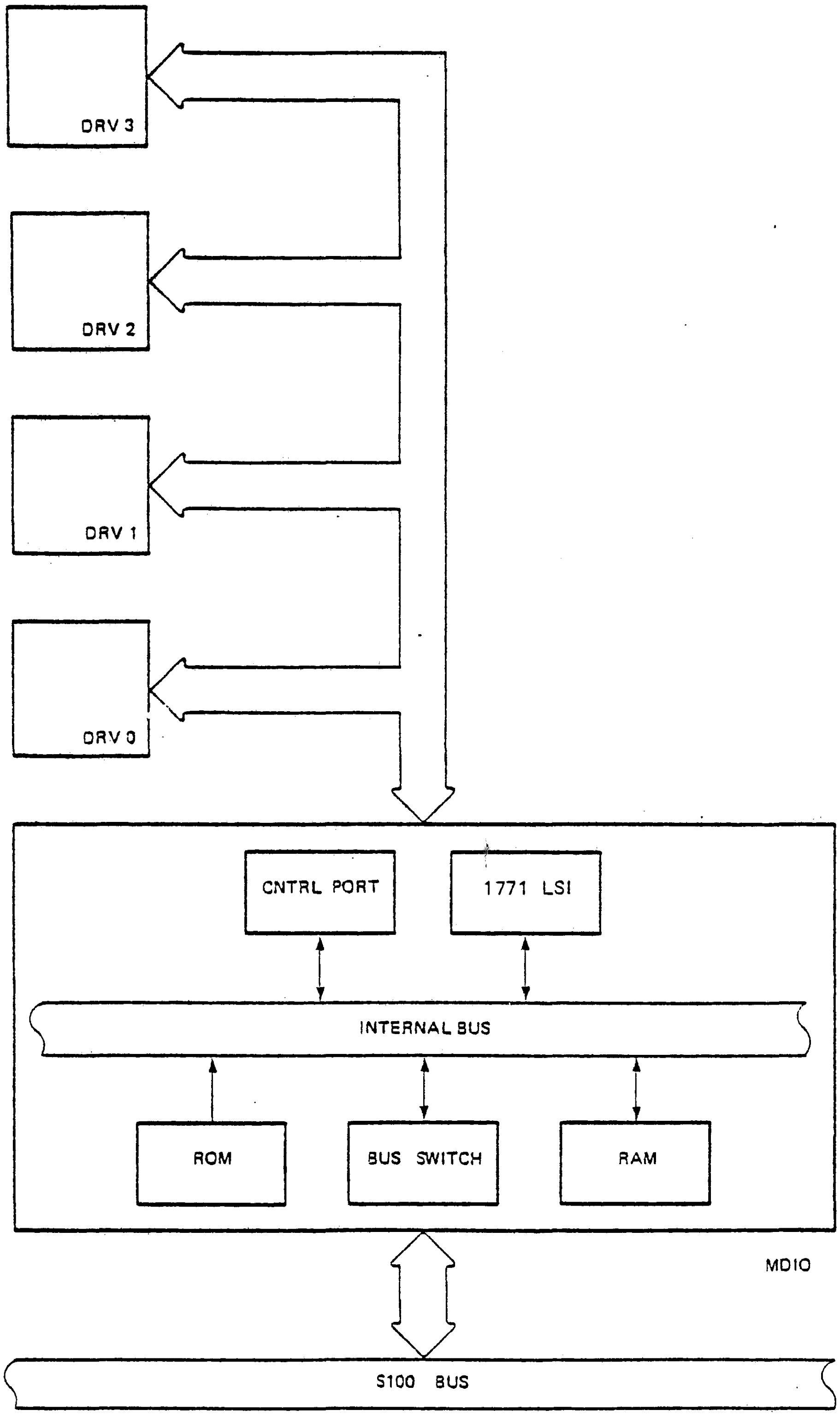
The MDIO is organized around an on-board, bi-directional data bus that links the major components of the controller. This bus is connected to the DO and DI lines of the S-100 bus through a bi-lateral bus-switch.

Several MSI-TTL functions and a few gates form an external data separator which tracks the FM-encoded pulse stream coming from the drive during a read operation. It samples this stream a 2MHz, decodes the pulses and steers separated clock and data pulses to the 1771.

A section of low-complexity combinational logic takes care of address decoding and timing requirements.

NOTE: MDIO's with firmware revision 2.2 or later will accept up to four (4) drives.
Earlier revisions accept up to three (3) drives.

FIGURE 1 MDIO Functional Block Diagram



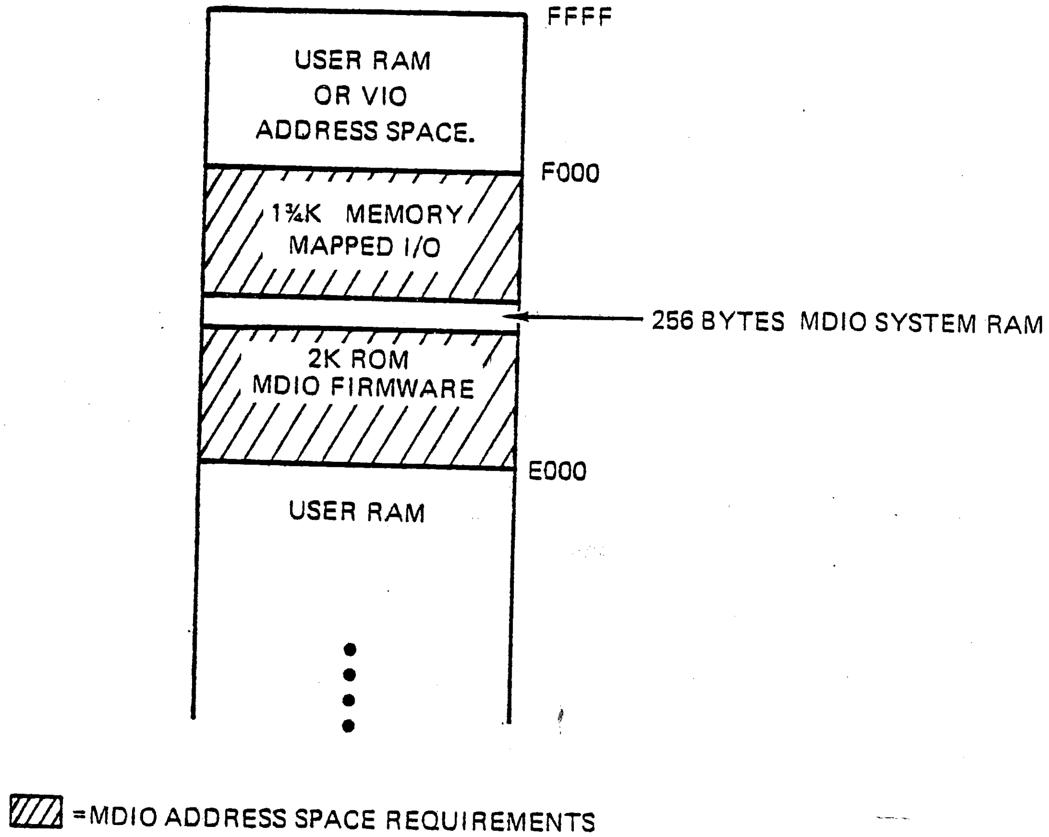


FIGURE 2 MDIO Address Space Requirements

5 – 3 DATA TRANSACTION PROCESSES

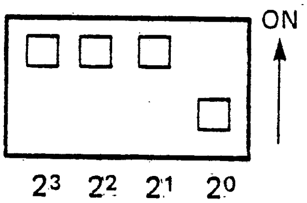
Data transaction processes involving IMSAI disk systems are governed by the IMDOS disk I/O protocol. They are, therefore, nearly identical for all IMSAI diskette controllers. A detailed description of this protocol, its diskette access conventions, and the data and command formats may be found in section 3 – 3, titled TRANSACTIONS. Additional clarification of the protocol may be found in the Programming Guide of the IMDOS User's Manual.

The MDIO supports two single density (FM) recording formats (Formats I and II). A breakdown of these sector formats into the various fields is shown in table 2, subsection 3 – 2.3, titled Format Compatibilities.

5 – 4 USER GUIDE

MDIO ADDRESS ASSIGNMENT

The MDIO may be switched into (selected) or out of (deselected) the system addressing space. This is accomplished by the execution of OUT instructions that reference the OUTPUT port address assigned to the MDIO. This port address assignment is made with the appropriate selection of SWI switch settings. The four (4) switches of SWI decode the upper nibble of the port address (N). Execution of an OUT instruction to port NFH selects, to port NEH deselects the MDIO.



MDIO CONTROL PORT SELECTION
(SHOWING EXH SETTING)

MDIO JUMPER CONFIGURATIONS

The following is a complete summary of MDIO configuration jumpers and their use.

ADDRESS DECODING:

The board is pc'ed for 65K maximum system configuration and the FANTOM line is enabled:

	A14	O – P
	A15	R – S
FANTOM	A16	U – V

For systems with memory configured for more than 65K, these changes in the jumper configuration must be made:

Delete	O – P	Add	P – Q
	R – S		S – T
	U – V		X – Y

BOARD SELECTION

The board is pc'ed to power-up in the MDIOIN-state. To have it come up /MDION, i.e., deselected, these jumpers are changed:

Delete	L – M	Add	M – N

BUS BUFFER TYPE SELECTION

This is primarily a manufacturing option to permit installation of either buffer type.

INBOARD is generated as /INBOARD for 74LS244 use. To change this for 74LS241 use:

Delete	AE – AD	Add	AD – AF

ROM/EPROM SIZE SELECTION

The board is pc'ed for 2K (2716) ROM/EPROM. To change to 1K capacity, change thusly:

Delete	I – K	Add	I – J
	F – H		F – G

MPU-A COMPATIBILITY

Although pc'ed for MPU-B (8085-based microprocessor) operation, the MDIO board may be modified to work with an 8080-based microprocessor. The following jumpers should be changed.

Delete	AA – AB	Add	AA – AC

DRIVE-TYPE SELECTION

The MDIO is pc'ed to attach to MPI B51 mini-diskette drives and may be modified to accept other drive types. Note that the MDIO supports 40 tracks on MPI drives and 35 tracks on other drives. Parameters that specify either the MPI drive or a non-MPI drive are loaded into MDIO RAM by the initialization routine of MDIO firmware upon power-on (see also the section titled The 1771 Ready Status Bit). If micropolis 1015 drives (up to four) are attached to the MDIO, then the user must change the drive type parameter code from 6 to an AH, if IMDOS is to be informed of the availability of 77 tracks. This may be effected by a cold start command that loads AH values into the appropriate MDIO RAM locations.

In addition, this cold start command must also load the value 4DH in MDIO RAM location E850H so as to inform MDIO firmware that track numbers 0 through 76 are to be legal. For information regarding the use of the cold start command feature, the user is referred to IMDOS User's Manual.

To use Shugart SA400 or Wangco 82:

Delete	AJ - AH	Add	AJ - AG

To use a Micropolis 1015:

Delete	AJ - AH A - C	Add	AJ - AG A - B

The parameters which specify the drive type are kept in MDIO locations:

E804	mini-drive	0
E805		1
E806		2

The codes used to identify these parameters are:

6	Shugart 450 (35 tracks)
8	MPI B51 (40 tracks)

5 – 5 THEORY OF OPERATION

5 – 5.1 HARDWARE STRUCTURE

BUS ORGANIZATION

The DAL–bus (see Figure 3) is a bi–directional bus which interconnects the several MDIO components with the S–100 bus where data pathing occurs on two uni–directional buses, namely DO0 – DO7 for data transfers from CPU to peripheral and DI0 – DI7 for data transfers from peripheral to CPU. Interfacing between the DAL–bus and the DI, DO lines is accomplished by two 74LS244 bus drivers (U30 and U31).

MAJOR LOGIC BLOCKS

In addition to the bus–switch, MDIO is composed of these major functional components:

- 1771 controller, a 40–pin LSI chip that supervises and directly controls communications with the selected mini drive.

- Data separator, an MSI–implemented FM pulse stream decoder interposed between the drive and the 1771.

- Memory mapped port, a six–bit strobed latch which exercises control over drive motor as well as, drive selection in system configurations containing more than one mini drive.

- ROM/EPROM, a 24 pin device which contains the 2K bytes (optionally 1K) of MDIO firmware.

- RAM, 256 bytes of random access memory used by MDIO firmware for variable data and status storage.

- Timing and control logic, which performs address decode, bus control and device strobing functions.

Operation with regard to device selection and access timing of the ROM/EPROM, RAM, memory–mapped port, and bus switch is straight forward and is discussed below. Operation of the 1771 and the data separator is more complex and is treated separately.

MDIO ADDRESSING

The MDIO resides at E000 - EFFF within a 65K address space, and within the highest 65K block in any configuration space that exceeds 65K. Several gates (chips U2, U15, U25 and U20 decode address lines A12 - A19 for a partial decode of MDIO board select. This decode is EXXX for an address space of 65K, FEXXX for greater than a 65K space.

The quad-XOR (U6), together with the 4-position DIP switch (SWI), a 7422 (U18), 1/4 of U9 and one D-FF (1/2 of U5) forms the control logic whereby the MDIO may be switched into and out of the address space. Execution of an OUT instruction to port NE deselects, to port NF selects the MDIO. When the MDIO is deselected, any RAM existing "behind" the MDIO becomes available to the system. The signal generated by this logis is MDIOIN.

Together with EXXX (or FEXXX), MDIOIN forms BDSELX. The X in this signal name signifies that MDIO board selection has not yet been qualified with negation of S-100 bus status signals SINTA, SINP, and SOUT. (For reasons elaborated upon below, these status signals are kept out of MDIO access decoding until the next logic layer.) The product of /SINTA, /SINP, and /SOUT forms the term MEM. Subsequent ANDing of BDSELX and MEM produces BDSEL. This term appears as qualifier at several nodes within the remainder of the address decode and timing logic.

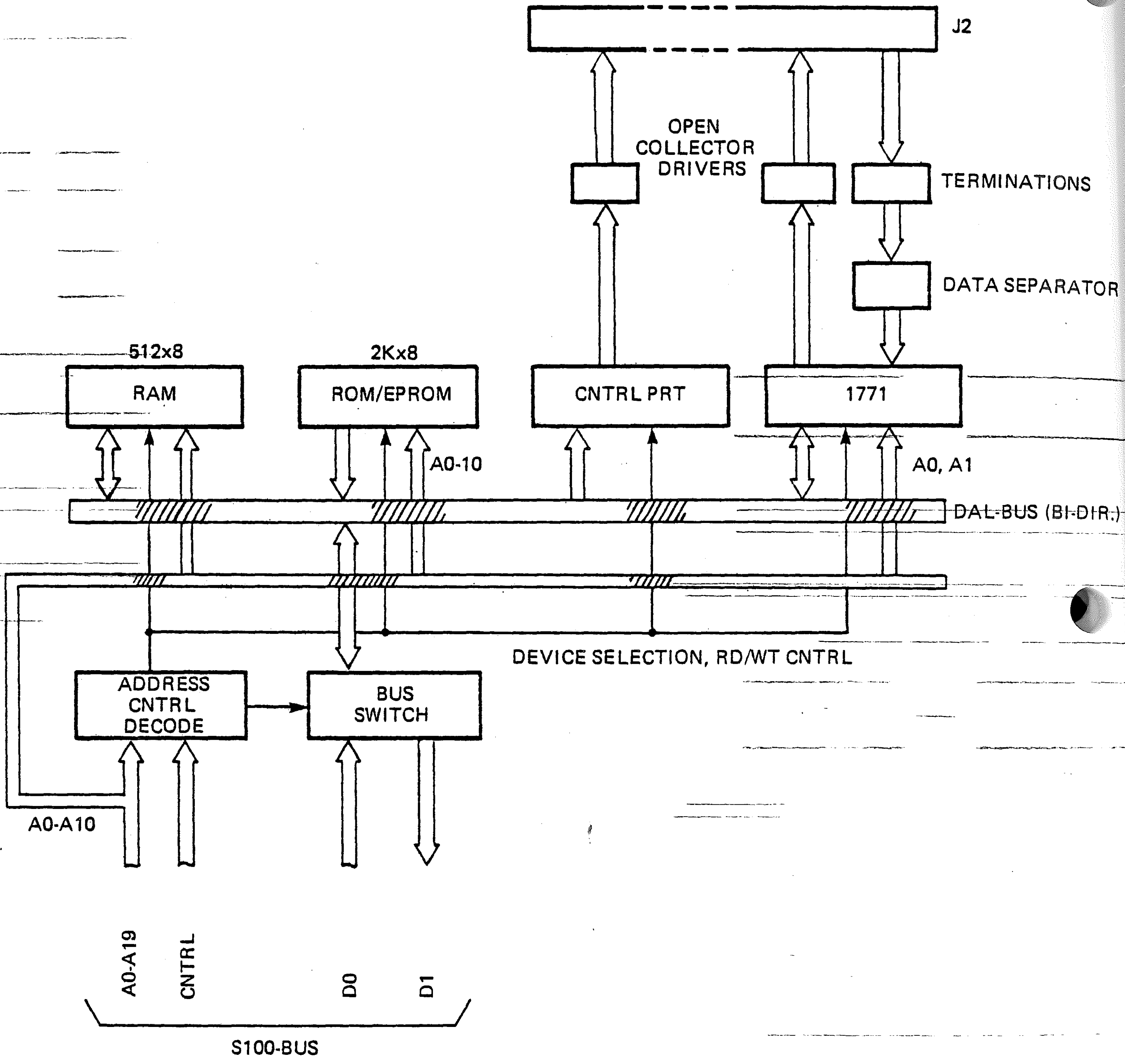


FIGURE 3 MDIO Organization

MDIO RAM is located at E800 through E8FF. Access to this space is decoded by BDSEL X8XX which produces RAMSEL (U18, U26 and U17). Similarly, ROM/ EPROM resides an E000 through E7FF. Any 1K firmware will image in two regions, E000 – E3FF and E400 – E7FF, because A10 is not decoded. The standard 2K MDIO firmware will, of course, occupy the entire MDIO ROM/EPROM space. This space is distinguished from RAM space by the inversion of A11 and access is decoded by BDSEL•X(0XXX)XX which produces ROMSEL.

Located at E9X (X1XX), the memory–mapped port is enable by BDSEL•X9XX•A2 which forms the signed PORTSEL. SEL1771, on the other hand, is decoded by ANDing BDSELX•X9XX•/A2. The results in the following address assignments:

memory–mapped port	E9X7
1771 data register	E9X8
1771 sector register	E9X9
1771 track register	E9XA
1771 status register	E9XB (read)
1771 command register	E9XB (write)

The DAL/S–100 bus switch is controlled by BDSEL PDBIN for data going off–board and BDSEL•/SMEMR for data heading on–board. The control signals are OFFBRD and INBRD, respectively.

5 – 5.2 CONTROL LOGIC TIMING

WRITE OPERATIONS

The S–100 bus signal /PWR is the write strobe which trips destination latches with its trailing edge during CPU to peripheral device write operations.

/PWR controls the R/W control line of RAM directly while the negation of SMEMR controls the output–disable (OD) line. During write operations, SMEMR will be low, and RAM output thus disabled (OD = 1).

/INBRD, meanwhile, will be active (/(BDSEL•/SMEMR) = 0), thus enabling data from the DO lines onto the DAL bus. At the end of /PWR, this data is latched into the RAM cell addressed by address lines A0 – A7.

Writing into the memory–mapped port occurs at the end of /PORTWT, the product of /(PORTSEL• PWR). In this case the write strobe also contains the device selection term.

The leading (i.e., falling) edge of /PORTWT triggers the DRIVE SELECT ONESHOT (1/2 of U1) which has a period of around 4 – 5 seconds. When set, the Q–output of this oneshot removes the reset on U23, thus allowing updating of the latches at the time of the trailing (i.e., rising) edge of /PORTWT.

Head loading takes place within the mini drive concurrent with drive selection. Conversely, head de–selection is equivalent to drive deselection. It follows that the head of a selected drive will remain loaded as long as the DRIVE SELECT ONESHOT is retriggered prior to time–out.

If the oneshot is allowed to time out, the Q–output (U1 - 5) will revert to a low state which, in turn, resets U23. This results in drive de–selection and hence head unloading.

Since the term MOTORON is stored in an independent memory cell (FF U5), the motor(s) of the drive(s) will remain on even after DRIVE SELECT ONESHOT time–out has occured. This ensures that the drive(s) continue at full rotational speed.

Upon drive re–selection, only the normal head load settling time will have to be reckoned with by the controller.

READ OPERATION

During read operations, BSDEL PDBIN generates OFFBRD which enables data flow off the DAL bus and onto the DI lines. SMEMR will be high, while PWR will remain inactive.

SMEMR will thus remove the data out disable during RAM access (OD = 0) allowing the accessed RAM cell to output its contents onto the DAL bus. ROM/EPROM is enabled by ROMSEL going true during a ROM/EPROM access.

Accessing the 1771 is a little more involved. Due to its long access time, the 1771 must be selected well in advance of strobing time at the destination (i.e., at the CPU). Similarly, the 1771 read strobe must arrive as early as possible.

This is accomplished by a) deferring the MEM qualification and deriving SEL1771 solely from address bus decoding plus MDIOIN, and b) generating an advanced "PDBIN" and qualifying that with MEM. This latter signal becomes ADVREAD, the read window for the 1771.

ADVREAD is the latched result of sampling the status line SMEMR as soon as it becomes stable on the bus. This happens to be just prior to the rising edge of PSYNC 01 in an 8080–based system and /PSYNC 01 in an 8085–based system. ADVREAD is reset by the trailing edge of PDBIN at the conclusion of the data transfer.

5 - 5.3 THE 1771 LSI CONTROLLER

Next to the firmware, the 1771 LSI chip is the most important single item in the MDIO design. It performs all the diskette drive control operations such as head positioning, head loading, and read or write operation sequencing. If contains the requisite logic to sense TRK0, DRIVE READY, INDEX and WTPROT and use these inputs in conjunction with the various operation it may be instructed to perform. These operations are very complex and are initiated by commands which the MDIO firmware issues to the 1771 command register. A summary of those commands which are utilized within the MDIO design, is given below:

1771 COMMAND SUMMARY

RESTORE

Forces head carriage assembly to reposition over track-0.

SEEK

Computes difference between current track and destination track, then issues appropriate number of steps in desired direction.

WRITE TRACK

Writes an entire track from index to next index, including all sector formatting information. All information to be written is supplied by the firmware.

WRITE

Searches for desired sector in track over which the head is located, then overlays data field with new block of data.

READ

Searches for desired sector in track over which the head is located, then reads that data field into system main memory.

READ ADDRESS

Inputs the six ID-field bytes of the next encountered sector. This is used to examine validity of Track and Format information when an ID-field CRC error is encountered during a sector search.

FORCE INTERRUPT

Used solely for the purpose of placing the 1771 into a state that allows continuous monitoring of the index pulse.

These commands are used in conjunction with three additional 1771 registers through which track, sector and data information is interchanged between the 1771 and the firmware. To inform the external world of its current status, the 1771 updates the contents of its status register as required. A summary of the status byte codes is given below:

MSB	S7	S6	S5	S4	S3	S2	S1	S0	LSB

STATUS REGISTER SUMMARY

BIT	ALL TYPE I COMMANDS	READ ADDRESS	READ	READ TRACK	WRITE	WRITE TRACK
S7	NOT READY	NOT READY	NOT READY	NOT READY	NOT READY	NOT READY
S6	WRITE PROTECT	0	RECORD TYPE	0	WRITE PROTECT	WRITE PROTECT
S5	HEAD ENGAGED	0	RECORD TYPE	0	WRITE FAULT	WRITE FAULT
S4	SEEK ERROR	ID NOT FOUND	RECORD NOT FOUND	0	RECORD NOT FOUND	0
S3	CRC ERROR	CRC ERROR	CRC ERROR	0	CRC ERROR	0
S2	TRACK 0	LOST DATA	LOST DATA	LOST DATA	LOST DATA	LOST DATA
S1	INDEX	DRQ	DRQ	DRQ	DRQ	DRQ
S0	BUSY	BUSY	BUSY	BUSY	BUSY	BUSY

The MDIO employs the Programmed Data Transfer method and thus does not make use of the 1771's interrupt and data request lines. Instead, the MDIO firmware supervises data transfer activity by constantly monitoring the 1771 status register, and in particular, the DRQ and BUSY bits of the status byte.

For a detailed description of the 1771 the reader is referred to the 1771 Data Sheet at the end of this section.

THE 1771 READY STATUS BIT

During initialization, the READY input of the 1771 is used to determine which of the two permissible jumpers, AG - AJ and AJ - AH, is installed on the board.

MDIO firmware strobes the control port with a dummy drive select byte that selects no drive (but does turn on drive motor(s)). This strobe fires the DRIVE SELECT ONESHOT, while also latching a "0" into 4Q at U23, thus forcing jumper point AG high.

The firmware then examines the 1771 status byte for a READY. If it finds a READY, it infers that the board is strapped for MPI B51 use and readies the appropriate parameters for step rate and number of tracks per diskette (12 ms, 40 tracks). If it finds a NOT READY, it infers that the board is not strapped for MPI B51 use and readies the default parameters (40 ms, 35 tracks). All future control bytes strobed into

U23 during drive selection activities will have bit 6 set to force a low at jumper point AG.

Before issuing a READ or a WRITE command to the 1771, MDIO firmware looks at the status of the chip to determine whether the READY flag has gone false. This flag serves the combined purpose of sensing both the readiness of a drive (where the drive does output such a status line) and the condition of the DRIVE SELECT ONESHOT.

If the firmware detects a NOT READY condition, it assumes that the oneshot has timed out. Re-selecting the specified drive will, therefore, result in the loading of the head. To allow for the required head load settling time, the firmware issues an alternate version of the READ or WRITE command which raises the HLD signal (U11 - 28). This fires the HEAD LOAD ONESHOT. The Q-output of this oneshot (U1 - 4) drives the HLT input at U11 - 23 low for approximately 75 ms. Sensing a low at HLT, the 1771 will defer commencement of the actual data transfer operaton until HLT once again has gone high.

If, on the other hand, the firmware detects a READY condition, it will merely re-select the specified drive (thus re-firing the DRIVE SELECT ONESHOT) and issue a normal READ or WRITE command. In response to this, the 1771 assumes that the head is already loaded and begins the data transfer operation immediately.

DATA SEPARATOR

The separator consists of two stages: a sychronization front end, followed by a quadrature circuit. The front end sychronizes the incoming FM pulse stream to the on-board 2 MHz clock. It does this by first converting the pulse stream to an NRZ waveform (U19) then reconstructing the now sychronized pulse stream (U20, U7 and U9). This is traced in the Data Separator timing diagrams.

Each FM pulse broadsides the 4-bit binary counter (U21) with the value N101, where N is the current binary value of Q (D), the output of the fourth stage of the counter.

In the timing diagram, Q (D) is assumed to be low at the beginning of the first bit-cell shown. It could just as easily have been high, due to asynchronism in bit-timing between the drive output and the separator, but a low is chosen at this time for ease of discussion.

LOADing of the counter at time "a" will jam the value 1101 into its stages (Q (D) will remain the same) and the next pulse coming from pin 10 of U20 will be routed to the FDCLOCK line. Three clock cycles later, the counter rolls over and Q (D) toggles, ready now to steer the (possible) next pulse to the FDDATA line.

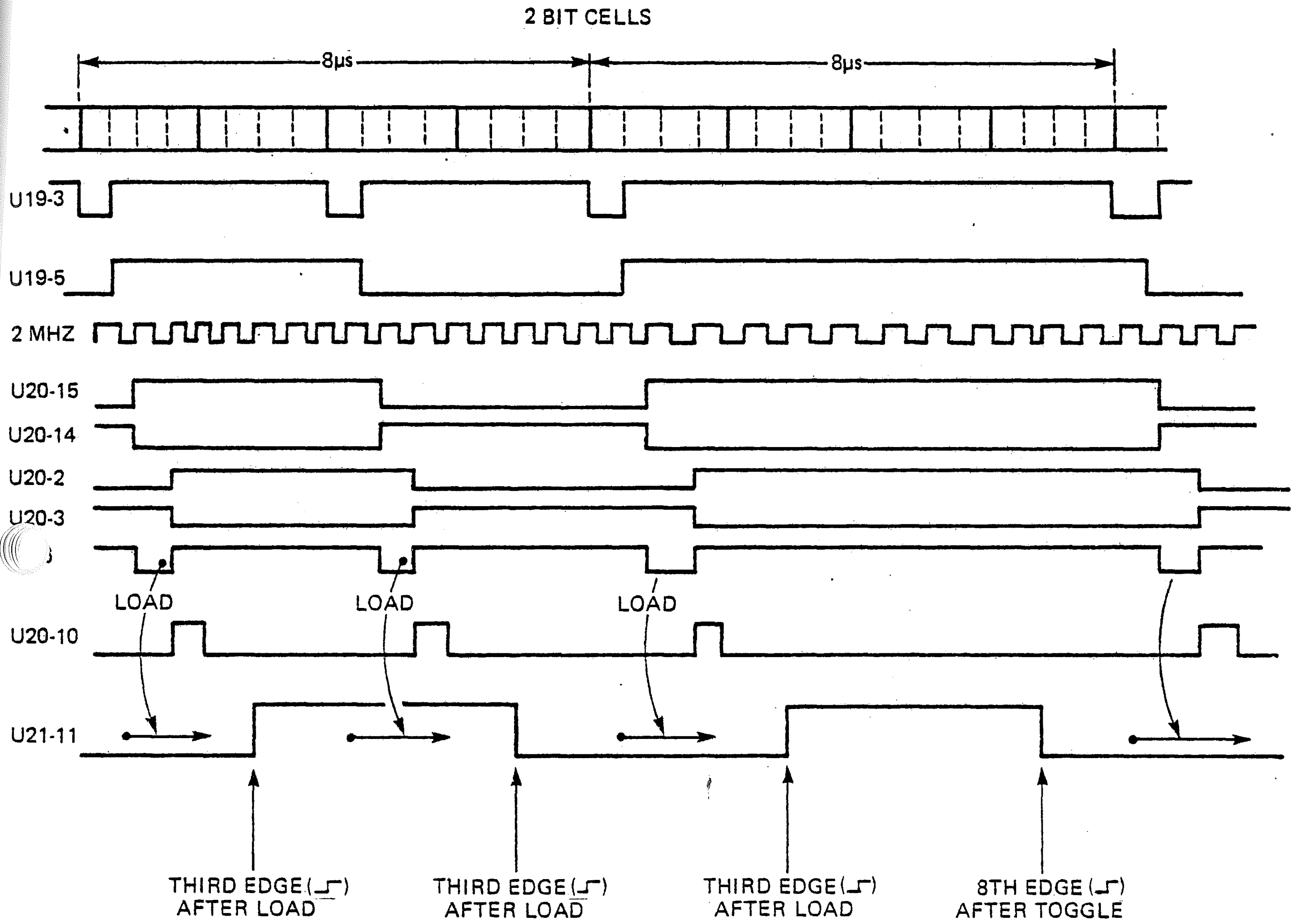
In the absence of a data pulse, the counter does not get LOADed and, hence, Q (D) will toggle after eight more clock cycles, ready, again, to steer the next pulse to the FDCLOCK line.

By loading the counter on each FM pulse, the quadrature waveform (Q (D)) is updated at least once per bit-frame (and twice during every "1" bit). This ensure tight tracking of the raw FM pulse stream by the data separator.

Note that the 1771 - 1 is interested in the leading edges of FDCLOCK and FDDATA. While in frequency synchronization, these edges will fall on or near the center of the quadrature waveform which, of course, extends 25% of bit-time back- and forward in time. Considerable pulse jitter can thus be tolerated and changes in pulse separation, between any two consecutive pulses, of nearly 25% of bit-time (i.e., 2 us) will not impair data recovery.

Depending on initial conditions, Q (D) may or may not be in phase synchronization with the FM pulse stream. The separator may, therefore, output data pulses on the FDCLOCK line. This is of no consequence, as the 1771 - 1 will perform phase synchronization upon detection of synch sequences consisting of multiple NUL bytes.

FM DATA SEPARATOR
(AFTER NATIONAL)


2 BIT CELLS
8µs
8µs
U19-3
U19-5
2 MHZ
U20-15
U20-14
U20-2
U20-3
LOAD
LOAD
LOAD
U20-10
U21-11
THIRD EDGE (_Г) AFTER LOAD
THIRD EDGE (_Г) AFTER LOAD
THIRD EDGE (_Г) AFTER LOAD
8TH EDGE (_Г) AFTER TOGGLE

- LOAD BROADSIDES N 101 INTO 74161 (WHERE N IS THE CURRENT VALUE OF QD)
- 3 CLOCK PULSES AFTER LOAD, QD TOGGLES.
- IF NO LOAD OCCURS FOR ANOTHER 8 CLOCK PERIODS,
- QD WILL TOGGLE AGAIN

5 - 6 1771 DATA SHEET

A comprehensive data sheet detailing the function and operation of the 1771 LSI Controller chip may be obtained from National Semiconductor Corporation.

Data Sheet Title: INS1771-1 Floppy Disk Formatter/Controller
Pub. No.: 426305468-001 (Apr. 1977)

5 – 7 SCHEMATICS AND I/O INTERCONNECT

MDIO TO DRIVE INTERCONNECTION

J2 PIN #	SIGNAL NAME	SIGNAL DIRECTION
2	——	
4	——	
6	——	
8	INDEX	<——
10	DS1	——>
12	DS2	——>
14	DS3	——>
16	MTRON	——>
18	DIRSEL	——>
20	STEP	——>
22	WT DATA	——>
24	WT GATE	——>
26	TRK 00	<——
28	WRT PROT	<——
30	RAW DATA	<——
32	——	
34	DS 4	——>

NOTE: All signals are low true, except DIRSEL which is bi-polar. All odd-numbered J2 pins are signals ground.

——> means: control signal to drive
<—— means: sense line from drive

6 DIO CONTROLLER

6 - 1 INTRODUCTION

This section presents a detailed discussion of the functional and operational characteristics of the DIO diskette formatter/controller. A common functional description precedes two user guide subsections, one for the DIO-C, the other for the DIO-D configuration. These are followed by a common theory of operation section. In this section, differences between the two DIO versions are pointed out within the text. The subsection titled DATA TRANSACTION PROCESSES, refers the user to a discussion of the IMDOS disk I/O protocol.

6 - 2 FUNCTIONAL DESCRIPTION

The DIO is a two-board, S-100 Bus compatible diskette formatter/controller. It is available in two versions. The DIO-C supports up to four (4) standard diskette drives; the DIO-D supports up to four (4) mini-diskette drives. Each version supports four (4) data formats, two in single density (FM) and two in double density (MFM) recording technique. These formats are discussed in greater detail in subsection 3 - 2.3, title FORMAT COMPATIBILITIES.

Each DIO controller consists of a DIO board (either DIO-C or -D) and a Programmable Data Separator (PDS) board. The PDS decodes the FM or MFM pulse sequences sourced by the diskette drive during a disk read transaction. It inputs separated clock and data pulses to the DIO board. The PDS is not involved in a write transaction.

The DIO is accessed by the execution of CALL instructions that reference one of several points of the DIO firmware. Such a CALL initiates a disk I/O with the control byte that is conveyed in the A-register. The nature of the requested transaction is specified by a command string which is located in system main memory.

The DIO firmware interprets this command string and performs the desired data transfer. Upon completion of the transaction, the DIO returns to the CALLer with a status describing the outcome.

DIO BOARD

The DIO Board operates on a memory mapped basis and occupies 1000H locations (4096 bytes). The addresses which are not used for the DIO ROM and RAM in the 4K address space are used for memory mapped I/O. The DIO has the capability of residing in the same location as RAM memory and can be switched in and out using I/O instructions.

Data is transferred between the floppy disk drives and the main system memory under program control. The CPU Ready line is used to introduce Wait States and thereby provide synchronization with the floppy disk drive data transfer rates.

The DIO is designed to make the operation as easy as possible for the operating system. All initialization sequences and error recovery procedures are contained within the firmware in the floppy disk controller. Hence, if a hardware error is indicated by the floppy disk firmware, it is an unrecoverable error and the user need not have error recovery procedures in the software. Similarly, the floppy firmware is designed to do the necessary head positioning and to remember the existing head positions, so the user need only execute read and/or write functions.

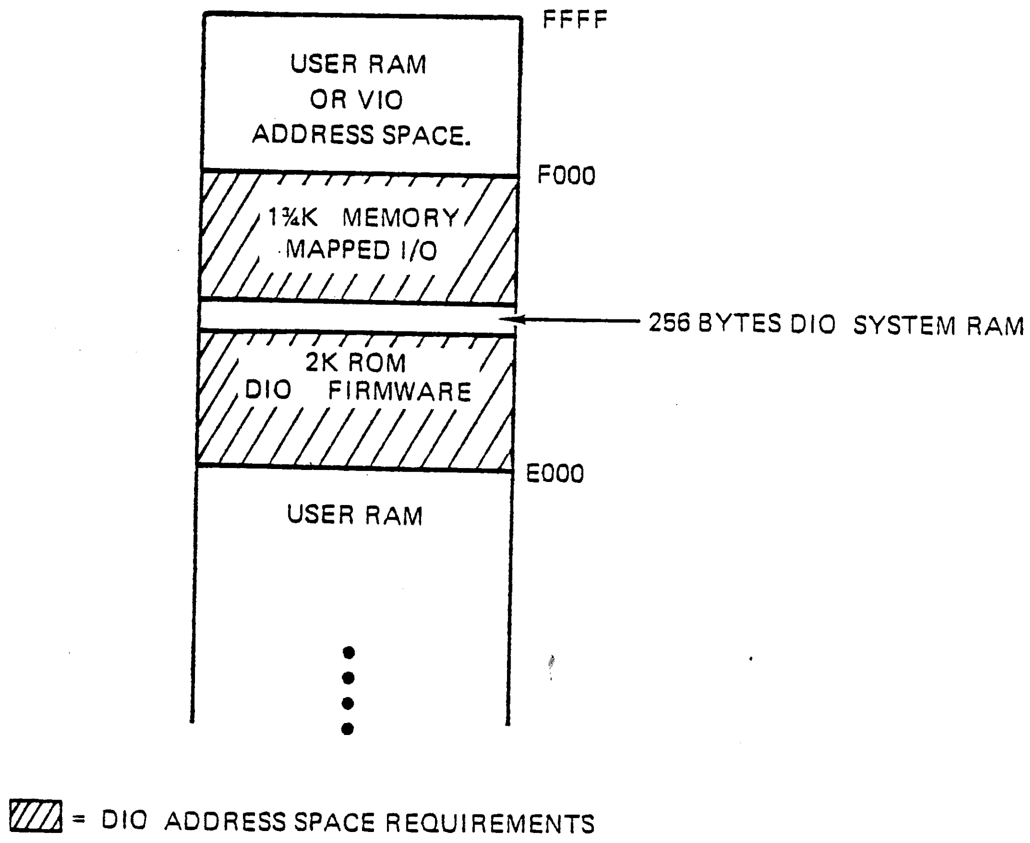


FIGURE 1 DIO Address Space Requirements

The DIO firmware is designed to pick up from its internal RAM memory the type of drive and recording technique being used each time a read or write operation is requested. Therefore, under program control the operating system can modify RAM locations and change the recording technique used on the same physical drive. Thus, a single-drive system is capable, under program control, of reading or writing a IBM 3740 format diskette and then switching to a high or double-density format to achieve economy in storage using the same physical drive. These settings are initialized by the initialization or bootstrap call to the value which is defined by the hardware switch settings on the DIO board and is initialized during the GENESYS phase of IMDOS.

PDS BOARD

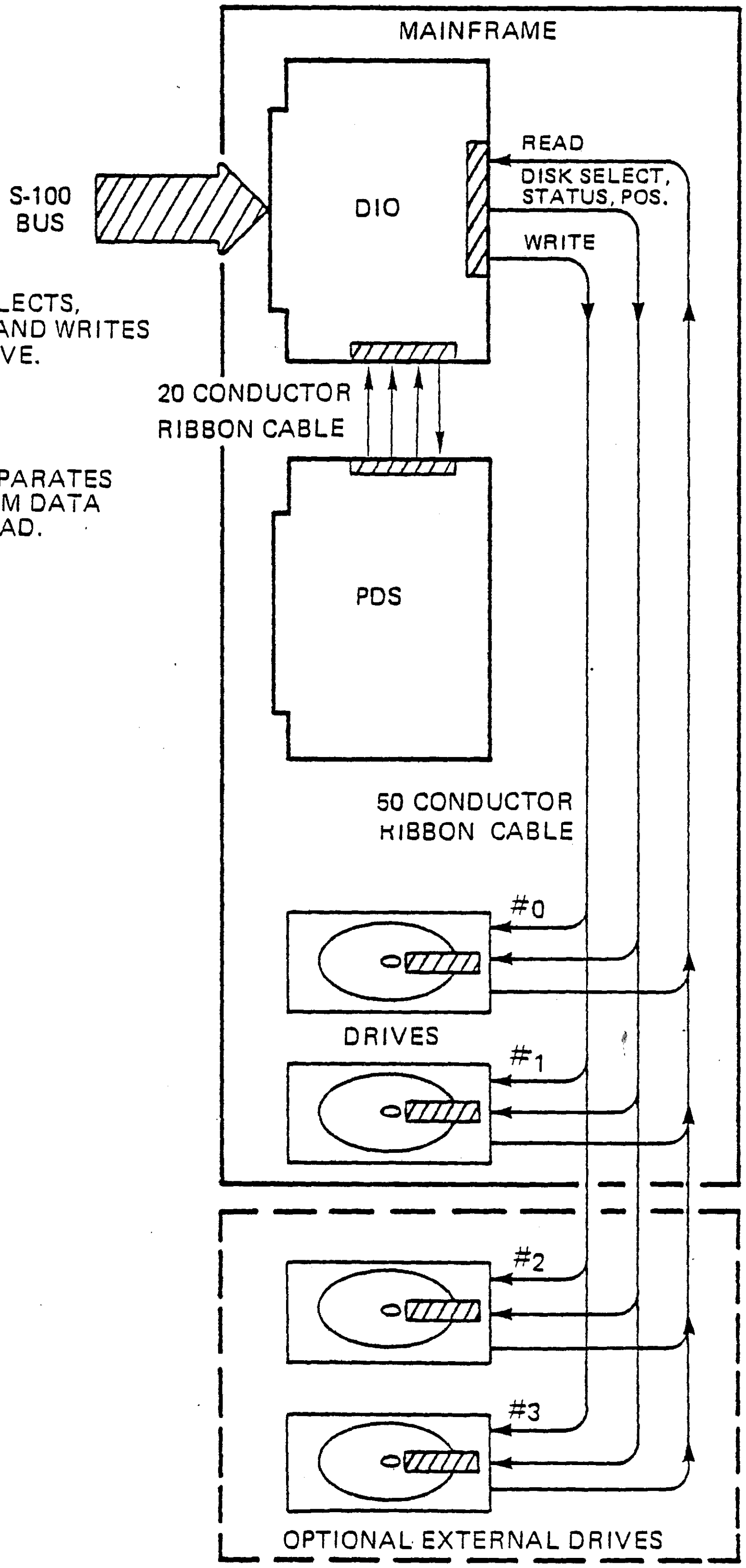
The PDS is an S-100 board used to separate the clock and data pulses which are received as a continuous pulse stream from the diskette drives electronics. The PDS is designed around a programmable Phase Lock Loop. The PLL phase locks on incoming signals from the drive and maintains lock even under missing clock or data bits.

The PDS can be programmed via the proper software operations to read and write with both the 250 KHz FM for single density or 500 KHz MFM (Modified FM) for double density formats, or 125 KHz FM for single density or 250 HHz MFM for double density format when used with DIO-D.

In terms of overall operation and interaction between the DIO-C and the PDS boards, the board set should be viewed as a single unit with two basic functions spread across circuitry in two boards.

The overall block diagram for the DIO and PDS in an S-100 system is shown below. Note that the DIO is connected to both the PDS and the drives. The PDS is used when read operations are required. For writing on the diskettes, the DIO sends the digital data directly to the drive.

As shown in the diagram, up to four (4) standard floppy disk drives may be operated from one DIO board. Selection of drives is done under software control. The DIO contains an on-board DIP switch that allows selecting one of 16 unique addresses for the DIO-C.


MAINFRAME
S-100 BUS
DIO
READ
DISK SELECT, STATUS, POS.
WRITE
20 CONDUCTOR RIBBON CABLE
PDS
50 CONDUCTOR RIBBON CABLE
#0
DRIVES
#1
#2
#3
OPTIONAL EXTERNAL DRIVES

THE DIO SELECTS, POSITIONS AND WRITES TO THE DRIVE.

THE PDS SEPARATES CLOCK FROM DATA DURING READ.

••• UP TO 15* ADDITIONAL DIO AND PDS BOARDS ON COMMON S-100 BUS.

Add Standard drive on 50 conductor cable

FIGURE 2 DIO/PDS Block Diagram

6 – 3 DATA TRANSACTION PROCESSES

Data transaction processes involving IMSAI disk systems are governed by the IMDOS disk I/O protocol. They are, therefore, nearly indentical for all IMSAI diskette controllers. A detailed description of this protocol, its diskette access conventions, and the data and command formats may be found in section 3 – 3, titled TRANSACTIONS. Additional clarification of the protocol may be found in the Programming Guide of the IMDOS User's Manual.

The DIO supports two single density (FM) and two double density (MFM) recording formats (Formats I and II, and III and IV, respectively). A breakdown of these sector formats into the various fields is shown in Tables 3 and 4; the former for DIO-C, the latter for DIO-D. The tables may be found in subsection 3 – 2.3, titled Format Compatibilities.

6-4 DIO-C USER GUIDE

All IMSAI DIO-C disk controllers are designed to occupy 4K of address space in an IMSAI S-100 mainframe. The specific addresses are at the top of memory at locations E000 to EFFF hex. Each DIO-C uses two I/O Ports in the 8080/8085. The ports are used by the operating system to switch a specific DIO-C in and out of operation. Each DIO-C contains a DIP switch which allows the user to select several default options for the controller. Four of these switch positions are used to give the DIO-C board a port address of XE or XF where X is a value between 0 and F. The XE port is used to disable a DIO-C with address switch set to X and the XF port is used to enable a DIO-C with the address switch set to X. (the value of X is set in binary on the switch; this is explained later).

Thus up to 16 DIO-C controllers can occupy the same DIO-C address space (E000 to EFFF) and any one can be enabled at a specific time. During this time all remaining DIO-Cs must be disabled by the operating system.

The DIO-C is designed so it can occupy its memory space along with real RAM memory, providing that the RAMs use A16 (backplane pin 16) to disable their selection logic. Alternately the DIO-Cs may exist without any RAM in their address space. Both these conditions are described below.

SINGLE DIO-C WITH NO OVERLAPPING RAM

In the situation where the DIO-C is to exist without overlapping RAM, when the DIO-C receives a RESET command (RESET on the front panel), the DIO-C is enabled and no further I/O instructions need be executed for the DIO-C. The I/O ports for the DIO-C should be switch selected to be I/O ports which are NOT used in the rest of the system.

SINGLE DIO-C WITH OVERLAPPING RAM

In the situation where the DIO-C is to exist with overlapping RAM when the DIO-C receives a front panel RESET command, the DIO-C is enabled. To disable the DIO-C (and thus enable the RAM locations at the same addresses) an OUT XE instruction should be executed. For the I/O instructions, the contents of the A register (i.e. the data value) is not used and can be any value. The value of X should be switch selected to be unique to that DIO-C and the appropriate output instruction executed to enable or disable the DIO-C for DIO-C and RAM references respectively.

CAUTION: Note that this definition precludes the transfer of data directly to or from the RAM locations E000-EFFF using the DIO-C Floppy Disk System.

MULTIPLE DIO-C INTERFACES

Multiple DIO-C Interfaces may be used in a single system by selecting a different set of I/O ports (e.g. a different value of X) for each DIO-C. Then the derived DIO-C can be enabled, or all of them disabled for referencing overlapping RAM memory. In addition, reset command jumpers (described later) are inserted so that the primary DIO-C board is enabled and the others are disabled by the Reset Command.

JUMPER AND DRIVE SELECTIONS

The DIO–C board requires switch settings to delineate the type of standard floppy drive and the recording format for that drive for use by the Firmware. These switch values are read by the Firmware when bootstrapping and each time the system is initialized to determine the type of drive, default density, type of processor (8080 or 8085),and so on. The results are stored in specific DIO–C address locations. The value in the RAM memory is then used by the Firmware when transferring data to or from the disks. In this manner, the same physical drive can be used, under program control, to read and record in different formats. For more complete information on how to do this, the reader is referred to the Programming Options section of this guide.

SWITCH AND JUMPER SETTINGS

This section gives the physical configuration requirements to accomplish the above alternatives. The switches at location U3 are used for the I/O Port selection and the Drive selection. The discrete jumper locations are called out alphabetically as shown on the assemble diagram.

Switches:

Address Assignment

The standard address assignments are selected by the top half of the DIP switch (switches 1 – 4). The assignment for the first DIO–C is E, D for the second, C for the third, and so forth in descending order. Note that a 1 corresponds to OFF on the switch. Switch physical position does not correspond to binary weighting.

SW#	1	2	3	4
Binary Weight	2^2	2^3	2^1	2^0

Recording Density

The recording density used by the drives is selected by switch 6. It should be ON for single density FM and OFF for double density MFM. Number of sides Switch 8 selects 1 or 2 sided operation and is currently not implemented. Place in OFF position. Drive types switch 5 selects the drive type. It must be set on for Persci drives (used in the VDP–80).

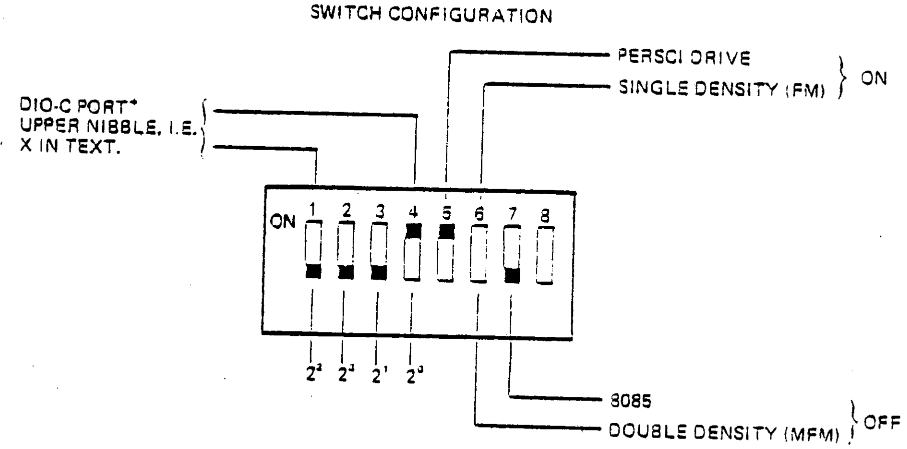
MPU Selection

Due to differences in system clock frequency the type of MPU being used must be set in switch 7. It should be OFF for 3 mHz 8085 (MPU–B) systems such as the VDP–80.

Jumpers:

Reset Command Selection (L, M, N)

The DIO–C has a trace from L to M which causes the DIO–C to be enabled by a Reset Command (RESET is usually pressed to make a cold-boot). To cause the particular DIO–C to be disabled, cut the trace from L to M and insert a jumper from L to N. Note only one DIO–C in a group may be the default DIO–C that's enabled by RESET.



*NOTE: Binary values of this switch do not follow normal 3, 2, 1, 0 weighting. Weighting is actually 2, 3, 1, 0 as shown in the above example. Also the ON position is a binary 0 and the OFF position is a binary 1. Thus for port set to D in the above example, switch 3 on the DIP switch would be set on and switch 4 would be off. In the example above the switch is set to E.

FIGURE 3 DIO-C Switch Organization

SYSTEM BOOTSTRAP

A "bootstrap" is a short program which reads another program from some storage medium into system RAM and executes it. This simple, yet general procedure gives the user freedom to load in any kind of operating system desired. The IMSAI Floppy Disk System bootstrap reads sector 1 of track 0 from drive 0 into system RAM addresses at 0–127 (0–7F) and then jumps to location 0. Note: On all IMSAI standard diskettes, regardless of density and format, track 0 is always single density 18 sectors/track and 128 bytes/sector. This allows any IMSAI computer with IMDOS to always be able to read in a system. Drive 0 of the mini drive is used by the bootstrap starting at E000.

PROGRAM OPTIONS

Under program control, the main program can alter the recording format used on a standard disk drive, modify the timing used by the DIO–C Firmware and program input/output pins on the 50 pin flat cable to use options available on the disk drives which are not supported by the firmware.

ALTERING RECORDING FORMATS

The first four locations of the DIO-C RAM Memory are used to define the type of drive present and the recording format used for that drive. The locations are:

E800 - Specify drive 0
E801 - Specify drive 1
E802 - Specify drive 2
E803 - Specify drive 3

These four locations are loaded with the switch selected drive type and recording density each time Initialization Command or Bootstrap is executed.

The codes used to identify those parameters are given below:

SD	DD	
0	1	Persci 277 Drive-side 0 (77 tracks)
2	3	Other STD Drive (Calcomp 142) (77 tracks)
4	5	Persci 277 Drive-side 1 (77 tracks)

Bit-4 is reserved for future expansion to include double-sided media specifications; thus, 15H would specify a media organization at: STD diskette, double-sided, 77 tracks, double density, mounted on Persci 277 drive-side 1.

DISK DRIVE OPTIONS

The DIO-C uses memory mapped transfer for transferring its I/O data. There are four output registers and two input registers used for the control functions. For a complete description of all bits in these registers the reader is referred to the Firmware Reference Information below. When using the control ports, care must be taken to ensure that all bits of that port are set appropriately.

Register Name	Memory Address	RAM Address
Output Control 1 (OC1)	E900	E80F
Output Control 2 (OC2)	E902	E810
Output Control 3 (OC3)	EA01	E811
Output Control 4 (OC4)	EA02	E812
Input Sense 1 (IS1)	E901	None
Input Sense 2 (IS2)	EA00	None

FIRMWARE REFERENCE INFORMATION

The DIO Floppy Disk Interface is designed to function with a PDS Programmable Data Separator to form a Floppy Disk Controller. The DIO uses 1000 Hex address locations beginning at E000 Hex for its self-contained ROM, RAM and Memory Mapped I/O. The following paragraphs provide a detailed description of the address use within the 1000 Hex locations. For jumper options available on this board, the reader is referred to the User Guide.

The format for the following discussion will be to give the address locations in hex followed by a description of the use of those locations. Locations labeled as undefined will cause indeterminate results if referenced with a read or write operation.

E000 to E7FF - Used to address the 2048 bytes of ROM contained on the board. The ROM contains all of the firmware required to operate all supported combinations of drives.

E800 to E8FF - Used to address the 256 bytes of RAM contained on the board.

E900 - Output Control 1 register, write only. The individual bits of this register are used as follows:

Bit 0 - Enable the CRC calculation on the CRC chip.

Bit 1 - Bit 0 (LSB) of the four bit clock pattern which is used for identifying the soft-sectored Address Marks.

Bit 2 - Sync Signal to PDS

Bit 3 - Bit 3 of the clock recognition pattern.

Bit 4 - Enable a write on the selected drive. Controls the Write Gate line for all drives.

Bit 5 - LSB of the Write Precompensation ROM group select. These two bits are used as follows:

00	FM Data
01	FM Address Mark
10	MFM Data
11	MFM Address Mark

Bit 6 - MSB of the Write Precompensation ROM group select.

Bit 7 - Enable the CRC bytes to be shifted out onto the data line (for recording CRC)

E901 - Input Sense 1, read only. The individual bits contain the following input values:

Bit 1 - Contains the value of Switch 6.

Bit 2 - Contains the value of Switch 5.

Bit 3 - Seek complete signal from selected Persci drive, 0 when complete.

Bit 4 - Contains the value of Switch 7.

Bit 5 - Side 1 ready from selected Percsi drive, 0 when ready.

E902 - Output Control 2 register, write only. The individual bits of this register are used as follows:

Bit 6 - Bit 1 of the clock recognition pattern.

Bit 7 - Preset the CRC value in the CRC chip to all ones.

E903 - Write only, used to configure the 8255 chip containing the above three locations. Must be loaded with 82 Hex after any RESET pulse.

E904 to E9FF - Undefined.

EA00 - Input Sense 2, read only. The individual bits contain the following input values:

Bit 0 - Contains the present value of the head load active one shot. The value is a one if the heads are still loaded on the selected drive.

Bit 1 - CRC value from the chip. Contains a zero when okay.

Bit 2 - Write Protect - side 1 from the selected drive, 0 when protected.

Bit 3 - Index pulse for the selected drive, 0 when index polse is present.

Bit 5 - Ready line from the selected drive, 0 when ready.

Bit 6 - T00 from the selected drive, 0 when drive is over track 0.

Bit 7 - Write Protect from the selected drive, 0 when protected.

EA01 - Output Control 3 register, write only. The individual bits of this register are used as follows:

Bit 1 - Enable the Restore line for the drives.

Bit 4 - Enable the Direction Select line for the drives. 0 causes the head to move out towards lower-numbered track.

Bit 5 - Enable the Drive Select 2 line.

Bit 6 - Enable the Step line.

Bit 7 - Enable the Drive Select 1 line.

EA02 - Output Control 4 register, write only. The individual bits of this register are used as follows:

Bit 0 - Enable the head load side 1

Bit 1 - Not used.

Bit 2 - MSB of the density select control. Used with LSB to select densities as follows:

00 - 250 kHz FM
01 - 500 kHz MFM .

Bit 3 - LSB of the density select control.

Bit 4 - Enable the Head Load line.

Bit 5 - Enable the Remote Eject - Side 0 line.

Bit 6 - Enable the Side Select for Persci Drives line, 1 selects side 1.

EA03 - Write only, used to configure the 8255 chip containing the above three locations. Must be loaded with 90 Hex after any RESET pulse.

EA04 to EAFF - Undefined.

EB00 - Read only, Address Mark Synchronous read input. Reading this address causes the CPU to be put into a Wait State until there is a compare in the clock value compare logic. When the comparison occurs, the data byte corresponding to the clock byte is input on the data lines.

EB01 to ECFF - Undefined.

ED00 - Read only, Byte Complete Synchronous read. Reading this address causes the CPU to be put into a Wait State until the next serial byte from the drive is ready for parallel input.

ED01 to EDFF - Undefined.

EE00 - Write only, Byte Complete Synchronous write. Loading this address causes the CPU to be put into a Wait State until the Controller is ready to accept the next parallel output byte.

EE01 to EEFF - Undefined.

EF00 - Write only. Loading this byte causes the head load active one shot to be triggered, independent of the data value.

EF01 to EFFF - Undefined.

DIO-D USER GUIDE

All IMSAI DIO-D disk controllers are designed to occupy 4K of address space in an IMSAI S-100 mainframe. The specific addresses are at the top of memory at locations E000 to EFFF hex. Each DIO-D uses two I/O Ports in the 8080/8085. The ports are used by the operating system to switch a specific DIO-D in and out of operation. Each DIO-D contains a DIP switch which allows the user to select several default options for the controller. Four of these switch positions are used to give the DIO-D board a port address of XE or XF where X is a value between 0 and F. The XE port is used to disable a DIO-D with address switch set to X and the XF port is used to enable a DIO-D with the address switch set to X. (the value of X is set in binary on the switch; this is explained later).

Thus up to 16 DIO-D controllers can occupy the same DIO-D address space (E000 to EFFF) and any one can be enabled at a specific time. During this time all remaining DIO-Ds are disabled by the operating system.

The DIO-D is designed so it can occupy its memory space along with real RAM memory, providing that the RAMs use A16 (backplane pin 16) to disable their selection logic. Alternately the DIO-Ds may exist without any RAM in their address space. Both these conditions are described below.

SINGLE DIO-D WITH NO OVERLAPPING RAM

In the situation where the DIO-D is to exist without overlapping RAM, when the DIO-D receives a RESET command (RESET on the front panel), the DIO-D is enabled and no further I/O instructions need be executed for the DIO-D. The I/O ports for the DIO-D should be switch selected to be I/O ports which are NOT used in the rest of the system.

SINGLE DIO-D WITH OVERLAPPING RAM

In the situation where the DIO-D is to exist with overlapping RAM when the DIO-D receives a front panel RESET command, the DIO-D is enabled. To disable the DIO-D (and thus enable the RAM locations at the same addresses) an OUT XE instruction should be executed. For the I/O instructions, the contents of the A register (i.e. the data value) is not used and can be any value. The value of X should be switch selected to be unique to that DIO-D and the appropriate output instruction executed to enable or disable the DIO-D for DIO-D and RAM references respectively.

CAUTION: Note that this definition precludes the transfer of data directly to or from the RAM locations E000-EFFF using the DIO-D Floppy Disk System.

MULTIPLE DIO-D INTERFACES

Multiple DIO-D Interfaces may be used in a single system by selecting a different set of I/O ports (e.g. a different value of X) for each DIO-D. Then the derived DIO-D can be enabled, or all of them disabled for referencing overlapping RAM memory. In addition, reset command jumpers (described later) are inserted so that the primary DIO-D board is enabled and the others are disabled by the Reset Command.

DRIVE SELECTIONS

The DIO-D board requires switch settings to delineate the type of mini-standard floppy drive (MPI or Micropolis) and the recording format for that drive for use by the Firmware. These switch values are read by the Firmware when bootstrapping and each time the system is initialized to determine the type of drive, default density, type of processor (8080 or 8085),and so on. The results are stored in specific DIO-D address locations. The value in the RAM memory is then used by the Firmware when transferring data to or from the disks. In this manner, the same physical drive can be used, under program control, to read and record in different formats. For more complete information on how to do this, the reader is referred to the Programming Options section of this guide.

JUMPER AND SWITCH SELECTIONS

This section gives the physical configuration requirements to accomplish the above alternatives. The switches at location U3 are used for the I/O Port selection and the Drive selection. The discrete jumper locations are called out alphabetically as shown on the assemble diagram.

Switches:

Address Assignment

The standard address assignments are selected by the top half of the DIP switch (switches 1 - 4). The assignment for the first DIO-D is E, D for the second, C for the third, and so forth in descending order. Note that a 1 corresponds to OFF on the switch.

Recording Density

The recording density used by the drives is selected by switch 6. It should be ON for single density FM and OFF for double density MFM. Number of sides Switch 8 selects 1 or 2 sided operation and is currently not implemented. Place in OFF position.

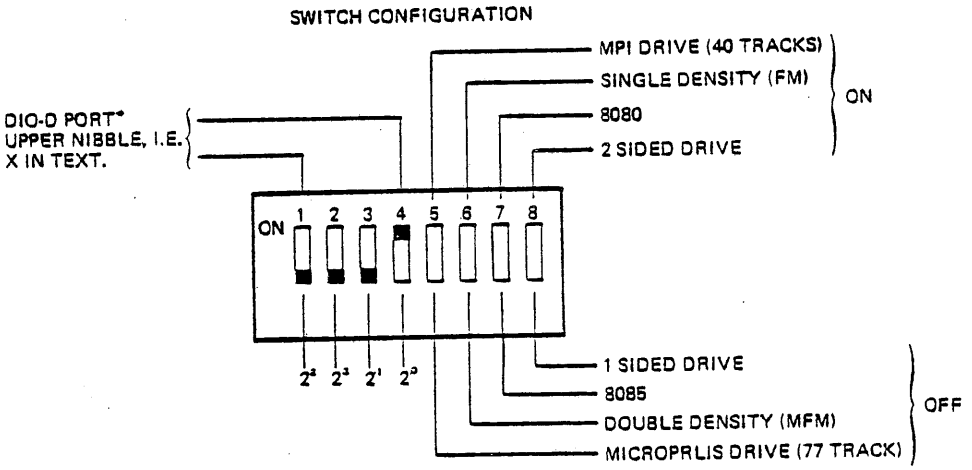
MPU Selection

Due to differences in system clock frequency the type of MPU being used must be set in switch 7. It should be ON for 2 mHz 8080 (MPU-A) systems and OFF for 3 mHz 8085 (MPU-B) systems.

Jumpers:

Reset Command Selection (L, M, N)

The DIO-D has a trace from L to M which causes the DIO-D to be enabled by a Reset Command (RESET is usually pressed to make a cold-boot). To cause the particular DIO-D to be disabled, cut the trace from L to M and insert a jumper from L to N. Note only one DIO-D in a group may be the default DIO-D that's enabled by RESET.



*NOTE: Binary values of this switch do not follow normal 3, 2, 1, 0 weighting. Weighting is actually 2, 3, 1, 0 as shown in the above example. Also the ON position is a binary 0 and the OFF position is a binary 1. Thus for port set to D in the above example, switch 3 on the DIP switch would be set on and switch 4 would be off. In the example above the switch is set to E.

FIGURE 4 DIO-D Switch Organization

SYSTEM BOOTSTRAP

A "bootstrap" is a short program which reads another program from some storage medium into system RAM and executes it. This simple, yet general procedure gives the user freedom to load in any kind of operating system desired. The IMSAI Floppy Disk System bootstrap reads sector 1 of track 0 from drive 0 into system RAM addresses at 0-127 (0-7F) and then jumps to location 0. Note: On all IMSAI diskettes, regardless of density and format, track 0 is always single density 18 sectors/track and 128 bytes/sector. This allows any IMSAI computer with IMDOS to always be able to read in a system. Drive 0 of the mini drive is used by the bootstrap starting at E003.

PROGRAM OPTIONS

Under program control, the main program can alter the recording format used on a standard disk drive, modify the timing used by the DIO-D Firmware and program input/output pins on the 34 pin flat cable to use options available on the disk drives which are not supported by the firmware.

ALTERING RECORDING FORMATS

The second four locations of the DIO-D RAM Memory are used to define the type of drive present and the recording format used for that drive. The locations are:

E804 - Specify drive 0 for the mini drives
E805 - Specify drive 1 for the mini drives
E809 - Specify drive 2 for the mini drives
E807 - Specify drive 3 for the mini drives

These four locations are loaded with the switch selected drive type and recording density each time Initialization Command or Bootstrap is executed.

The codes used to identify these parameter are given below:

SD	DD	
6H	7H	Shugart SA400 (35 tracks)
8H	9H	MPI B51 (40 tracks)
AH	BH	Micropolis 1015 (77 tracks)

Bit-4 is reserved for future expansion to include double-sided media specification; thus 1AH would specify a media organization of: mini- disk, double sided, 77 tracks, single density.

DISK DRIVE OPTIONS

The DIO-D uses memory mapped transfer for transferring its I/O data. There are four output registers and two input registers used for the control functions. For a complete description of all bits in these registers the reader is referred to the Firmware Reference Information below. When using the control ports, care must be taken to ensure that all bits of that port are set appropriately.

Register Name	Memory Address	RAM Address
Output Control 1 (OC1)	E900	E80F
Output Control 2 (OC2)	E902	E810
Output Control 3 (OC3)	EA01	E811
Output Control 4 (OC4)	EA02	E812
Input Sense 1 (IS1)	E901	None
Input Sense 2 (IS2)	EA00	None

MINI DRIVE OPTIONS

Motor On - Controlled by OC2, bit 4. Always turned on by DIO-D Firmware. 1 is on, 0 is off.

FIRMWARE REFERENCE INFORMATION

The DIO Floppy Disk Interface is designed to function with a PDS Programmable Data Separator to form a Floppy Disk Controller. The DIO uses 1000 Hex address locations beginning at E000 Hex for its self-contained ROM, RAM and Memory Mapped I/O. The following paragraphs provide a detailed description of the address use within the 1000 Hex locations. For jumper options available on this board, the reader is referred to the User Guide.

The format for the following discussion will be to give the address locations in hex followed by a description of the use of those locations. Locations labeled as undefined will cause indeterminate results if referenced with a read or write operation.

E000 to E7FF - Used to address the 2048 bytes of ROM contained on the board. The ROM contains all of the firmware required to operate all supported combinations of drives.

E800 to E8FF - Used to address the 256 bytes of RAM contained on the board.

E900 - Output Control 1 register, write only. The individual bits of this register are used as follows:

Bit 0 - Enable the CRC calculation on the CRC chip.

Bit 1 - Bit 0 (LSB) of the four bit clock pattern which is used for identifying the soft-sectored Address Marks.

Bit 2 - Sync Signal to PDS

Bit 3 - Bit 3 of the clock recognition pattern.

Bit 4 - Enable a write on the selected drive. Controls the Write Gate line for all drives.

Bit 5 - LSB of the Write Precompensation ROM group select. These two bits are used as follows:

00 - FM Recording Format
01 - FM Address Mark recording
10 - MFM Recording Format
11 - MFM Address Mark recording

Bit 9 - MSB of the Write Precompensation ROM group select.

Bit 7 - Enable the CRC bytes to be shifted out onto the data line (for recording CRC)

E901 - Input Sense 1, read only. The individual bits contain the following input values:

Bit 0 – Write Protect for selected Mini Floppy – 0 when protected.

Bit 1 – Contains the value of Switch 6.

Bit 2 – Contains the value of Switch 5.

Bit 3 – Contains value for switch for 2-sided version

Bit 4 – Contains the value of Switch 7.

Bit 5 – Ready from selected Micropolis Drive – 0 when ready.

Bit 9 – T00 from selected Mini Floppy – 0 when positioned over Track 0.

Bit 7 – Index pulse from selected Mini Floppy – 0 when index pulse is present.

E902 - Output Control 2 register, write only. The individual bits of this register are used as follows:

Bit 0 – Enable the Step line for Mini Floppy

Bit 1 – Enable the Drive Select 1 line for Mini Floppy.

Bit 2 – Enable the Drive Select 2 line for Mini Floppy.

Bit 3 – Enable the Drive Select 3 line for Mini Floppy.

Bit 4 – Enable the Motor On line for Mini Floppy

Bit 5 – Enable the Direction Select Line for Mini Floppy (0 causes head to move out towards lower-numbered track.)

Bit 9 – Bit 1 of the clock recognition pattern.

Bit 7 – Preset the CRC value in the CRC chip to all ones.

E903 - Write only, used to configure the 8255 chip containing the above three locations. Must be loaded with 82 Hex after any RESET pulse.

E904 to E9FF - Undefined.

EA00 - Input Sense 2, read only. The individual bits contain the following input values:

Bit 0 – Contains the present value of the head load active one shot. The value is a one if the heads are still loaded on the selected drive.

Bit 1 – CRC value from the chip. Contains a zero when okay.

EA01 - Undefined

EA02 - Output Control 4 register, write only. The individual bits of this register are used as follows:

Bit 0 - Enable the drive select 4 line for mini floppy

Bit 1 - Not used.

Bit 2 - MSB of the density select control. Used with LSB to select densities as follows:

00 - 125 kHz FM (for Mini)
01 - 250 kHz MFM (for Mini)

Bit 3 - LSB of the density select control.

Bit 7 - Enable the side select line for double side drives

EA03 - Write only, used to configure the 8255 chip containing the above three locations. Must be loaded with 90 Hex after any RESET pulse.

EA04 to EAFF - Undefined.

EB00 - Read only, Address Mark Synchronous read input. Reading this address causes the CPU to be put into a Wait State until there is a compare in the clock value compare logic. When the comparison occurs, the data byte corresponding to the clock byte is input on the data lines.

EB01 to ECFF - Undefined.

ED00 - Read only, Byte Complete Synchronous read. Reading this address causes the CPU to be put into a Wait State until the next serial byte from the drive is ready for parallel input.

ED01 to EDFF - Undefined.

EE00 - Write only, Byte Complete Synchronous write. Loading this address causes the CPU to be put into a Wait State until the Controller is ready to accept the next parallel output byte.

EE01 to EEFF - Undefined.

EF00 - Write only. Loading this byte causes the head load active one shot to be triggered, independent of the data value.

EF01 to EFFF - Undefined.

6-5 THEORY OF OPERATION

6-5.1 DISK I/O BOARD (DIO)

INTRODUCTION

The DIO Board is designed to operate with a data separator to form a complete floppy disk controller. It is used with the IMSAI Programmable Data Separator (PDS) Board to form the IMSAI Floppy Disk Controller which is capable of operating with minifloppy disks and standard floppy disks in either single density or double density. The minifloppies and standard floppies are connected to the DIO using flat cables as recommended by the drive manufacturers. The standard floppies connect to J4 using 50-conductor cable. There is a one-to-one correspondence between the pin numbers and signals on the DIO connectors and those called out in the drive manual. The reader is referred to the manual for the particular drive used to identify these signals for his system.

DIO AND PDS INTERCONNECTION

The PDS connects to the DIO using a 20-conductor flat cable attached to connector J2. All odd-number pins on this connector are signal ground. The signals contained on the even pins are as follows:

Pin 2 - CLK DATA

This signal is a high if there was a clock pulse in the previous bit cell. It is gated into the DIO on the low-to-high transition of the PLO shift pulse.

Pin 4 - PLO SHFT

This is the square wave output of the phaselocked oscillator. The low-to-high transition is used (one per bit cell) to shift the value of the data and clock lines into registers on the DIO.

Pin 9 - /CLK

This is a 2 mHz reference signal transmitted from the DIO to the PDS. It is used on the PDS for the self-adjust feature.

Pin 8 - CLK A

This is used with CLK B to define the format of the input data as follows:

CLK B	CLK A	Data Format (DIO-C)	Data Format (DIO-D)
0	0	Not used	FM Data at 125 KHz
0	1	Not used	MFM Data at 250 KHz
1	0	FM Data at 250KHZ	Not used
1	1	FM Data at 500KHZ	Not used

Pin 10 – /STD DATA

This is the Raw Data input from the standard drives. A high–to–low transition is used to signify a pulse. It must be high when not being used.

Pin 12 – /MINI DATA

This is the Raw Data from the mini drives. A high–to–low transition is used to signify a pulse. It must be high when not being used.

Pin 14 – CLK B

This is used with CLK A to define the format of the input data as described above.

Pin 16 – /CLR

When low, this causes the PDS to be in a clear state which in turn forces the PLO Shift Output signal to be a high.

Pin 18 – /SYNC

Input signal used by the PDS to properly phase itself during an input of zeroes on the raw data line.

Pin 20 – DATA IN

This signal is a high if there was a data pulse in the previous bit cell. It is gated into the DIO on the low–to–high transition of the PLO Shift Pulse.

DIO IMPLEMENTATION

With the exception of the data separator, all the logic required to interface with the floppies is contained on the DIO. There are two 8255 Programmable Peripheral Interface chips used to generate and receive all signals except read and write data from the floppies. They are also used to generate or receive other control signals for the interface. All I/O operations in the DIO Firmware are performed using Memory Mapped I/O. The reader is referred to the Firmware Reference Information for a definition of the addresses used and the bit assignments within the two 8255 chips.

The self–contained memory on the DIO consists of 2048 bytes of ROM (or EPROM) and 256 bytes of RAM. The ROM is implemented using a single 2316/2716 ROM/EPROM. The RAM is implemented using two 8111 chips (each is a 256 x 4 RAM). The serializing and deserializing of the data is accomplished using the two 74LS395 chips. These tri–state chips are used so an internal data bus can be used. The data bus provides bidirectional communication with the two 8255 chips, the two 8111 RAMs, the two 74LS395 chips and the S–100 Bus Interface 8216 chips. The 2316/2716 ROM/EPROM also gates its data onto the internal bus. The 8216 chips are selected by the /BD SEL signal (discussed below) while the direction of data flow is determined by the Backplane signal PDBIN which is high when data flow is from the DIO to the MPU Board.

There are three possible sources for an internal RESET signal for the DIO. Two are low active back plane signals /POC and /EXTCLR. These signals are ORed with an internal reset signal by the 74LS11 at U42. The internal signal is active (i.e. low) whenever the +5 Volts from the regulator chip falls below approximately 4.25 volts. This is detected by comparing the output voltage of the Zener Diode CR1 (which is 3V) with the voltage at the base of Q2 formed by the divider network of R9 and R10. When this is less than 2.3V (+5V bus is less than 4.25V), Q1 is turned on providing the low active signal.

ADDRESSING

As defined in the User Guide, the DIO may be enabled or disabled using two I/O ports with addresses of XE (for deselect) and XF (for select) where X is any hex digit. This is accomplished by comparing the I/O addresses with the switch settings using the 74LS85 Comparator at U39. The A=B Input is active when the 4 LSB's (bits) contain an E or F and the /PWR signal is low. The A=B output is ANDed with the SOUT signal at the 74LS08 gate located at U26 to form a clock signal for the 74LS74 at U1. This clock captures the value of AO and selects or deselects the DIO on a high or low respectively. Note that the M, N and P jumper configuration can be used to have the /RESET signal cause this flip-flop to be initialized in the set (selected, M to N) or reset (deselected, P to N) state.

Figure 5 shows the DIO address Decode Logic. The standard (trace present) jumper configurations are shown with the solid curved lines. For operation with the IMSAI IMM Board these traces must all be cut and the jumpers shown with dotted curved lines must be added. The 74LS21 at U32 (output pin 6) is used to form the board select signal. In either case, the 4 MS bits of the address must be an E (Hex) and the other Backplane signals must be low (indicating that this is a memory reference). For the standard case, the select flip-flop must be set (U1-5) or with the IMM the four extra address bits must all be ones (thus putting the DIO in the topmost page) to complete the selection.

The /BD SEL signal is then used to enable the ROM (or EPROM) if A11 is low. This uses addresses E000 to E7FF Hex and is accomplished by the 74LS32 at U22. If A11 is high then /BD SEL enables the 8205 Decode chip at U23.

This selects eight 256-byte segments of the addresses from E800 to EFFF Hex. The Firmware Reference Information defines the use of each of these selections and defines the addresses used by the DIO Firmware.

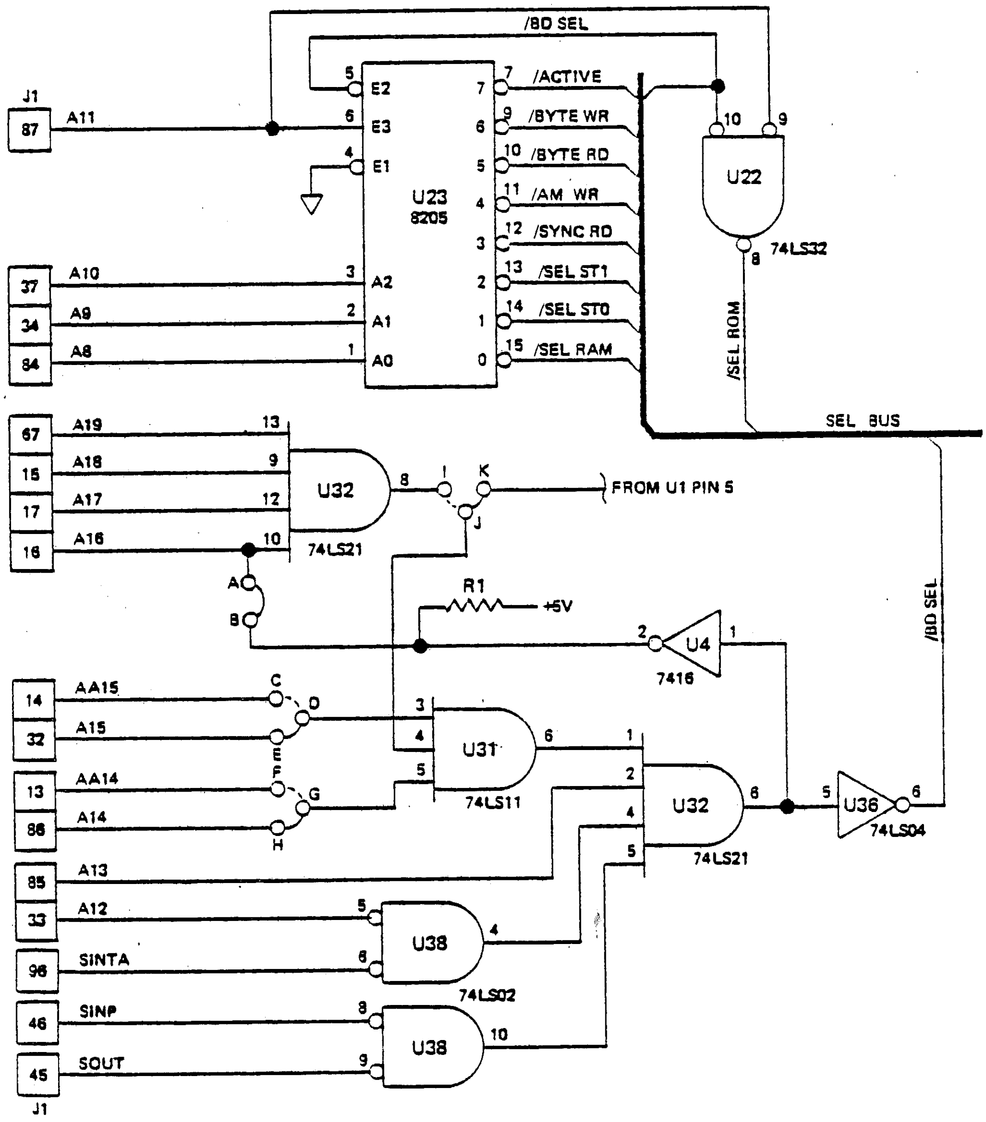


FIGURE 5 DIO Address Decode

BIT AND BYTE TIMING

Figure 6 shows the bit timing generation for the DIO. During a read operation the Write signal (U12 pin 7) is low causing the two counters and the flip-flop to be held in the clear state. The PLO SHFT signal from the data separator is used to generate the timing and to form the SHFT (internal shift) signal. When Write is high, the PLO SHFT signal is low and the DIO generates the timing. The two sections of the 74LS04 at U9 with the feedback resistors are used with the crystal to form a free-running 10 mHz oscillator. Capacitor C1 is to ensure that the crystal is not overstressed while C2 is used for pulse shaping.

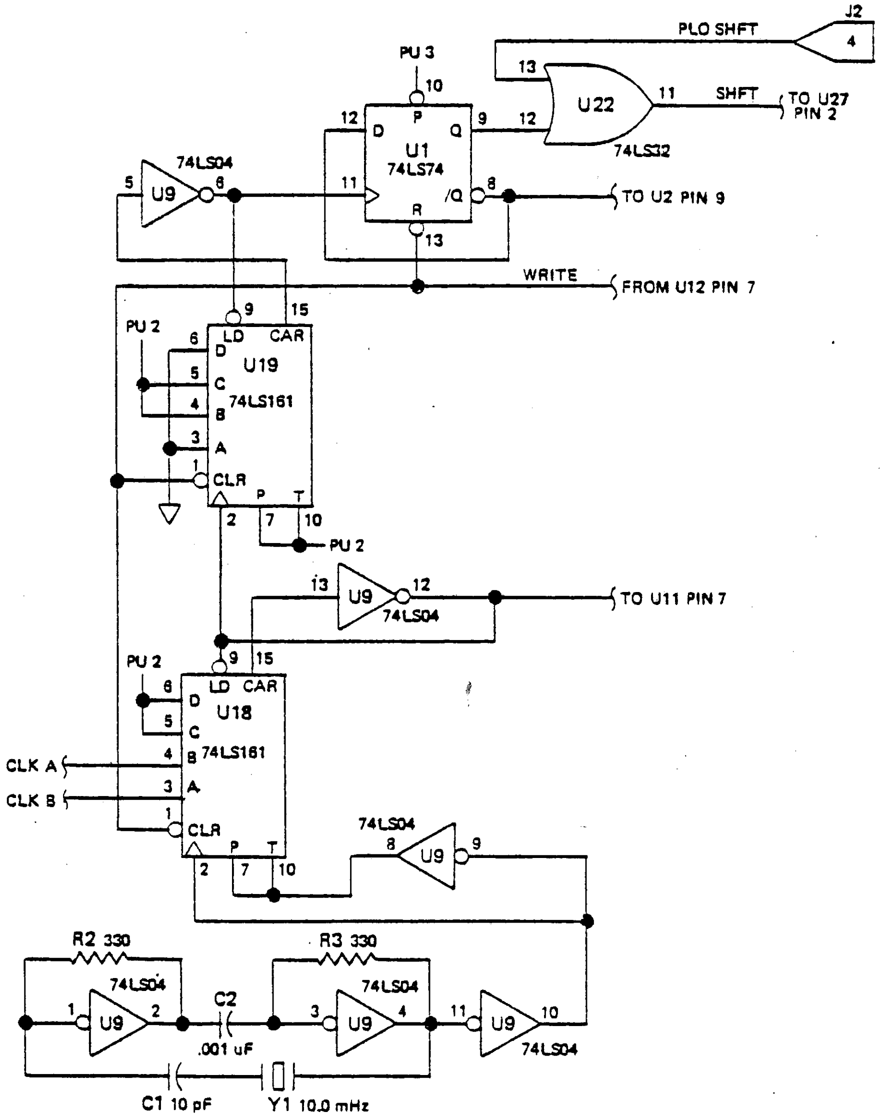


FIGURE 6 DIO Bit Timing Generation

The 74LS161 at U18 is then used to divide the 10 mHz signal for the required speeds as follows:

		Output Freq.		Bit Freq.	
CLK B	CLK A	DIO-C	DIO-C	DIO-C	DIO-D
0	0	Not used	2.5 MHz	Not used	125 KHz
0	1	Not used	5.0 MHz	Not used	250 KHz
1	0	5.0 MHz	Not used	250 KHz	Not used
1	1	10.0 MHz	Not used	500 KHz	Not used

The output frequency (from U9-12) is used to determine the amount of write precompensation and to keep it in proportion with the bit rate. The 74LS161 at U19 divides the output of U19 by ten; its output is divided in half by the 74LS74 at U1 to form the bit frequency.

Figure 7 shows the Byte Timing for the DIO. The SHFT signal, which is a square wave with a cycle time equal to a bit cell, is divided by eight using the 74LS161 at U27 to form the BYTE RDY signal. When reading, BYTE RDY is active when a full byte has been assembled in the shift registers (i.e., the two 74LS395 chips). This byte must be read from the interface during the next bit (as opposed to byte) time. When writing, BYTE RDY is active during the bit time AFTER the parallel data from the CPU has been loaded into the shift registers. Note that the division is accomplished by sequencing the counter from 1 to 8. The 74LS21 at U33 is used to decode a count of seven when a write is in progress. Its output goes to the mode control of the 74LS395 chips and causes a parallel load of these chips on the leading edge of the clock which counts the 74LS161 to the BYTE RDY state. (Note that the clock signal for the 74LS395 chips is the inversion of SHFT.)

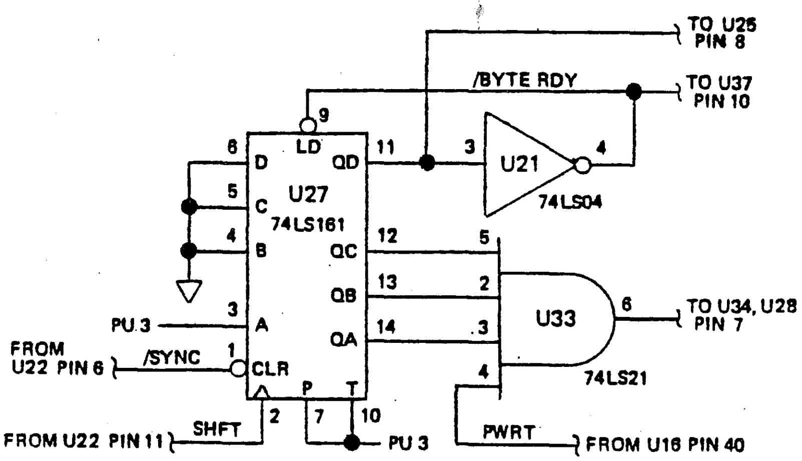


FIGURE 7 DIO Byte Timing

READY LINE SYNCHRONIZATION

As discussed above, there is a single bit time used to accomplish the reading or writing of the parallel data from the CPU. The CPU is put into the Wait State to perform the required synchronization for this timing. The logic used to generate the ready signal is shown in Figure 8. The one-shot (74LS123) at U43 is used to ensure that the DIO does not cause continuous Wait State if there is a hardware malfunction. Its time constant is set longer than any Wait State required for normal operation and it is triggered each time the DIO is referenced. The /Q output is then ANDed with the internal wait signal (by the section of U25 with pin 1 as its output) in order to form the wait (PRDY) signal for the MPU.

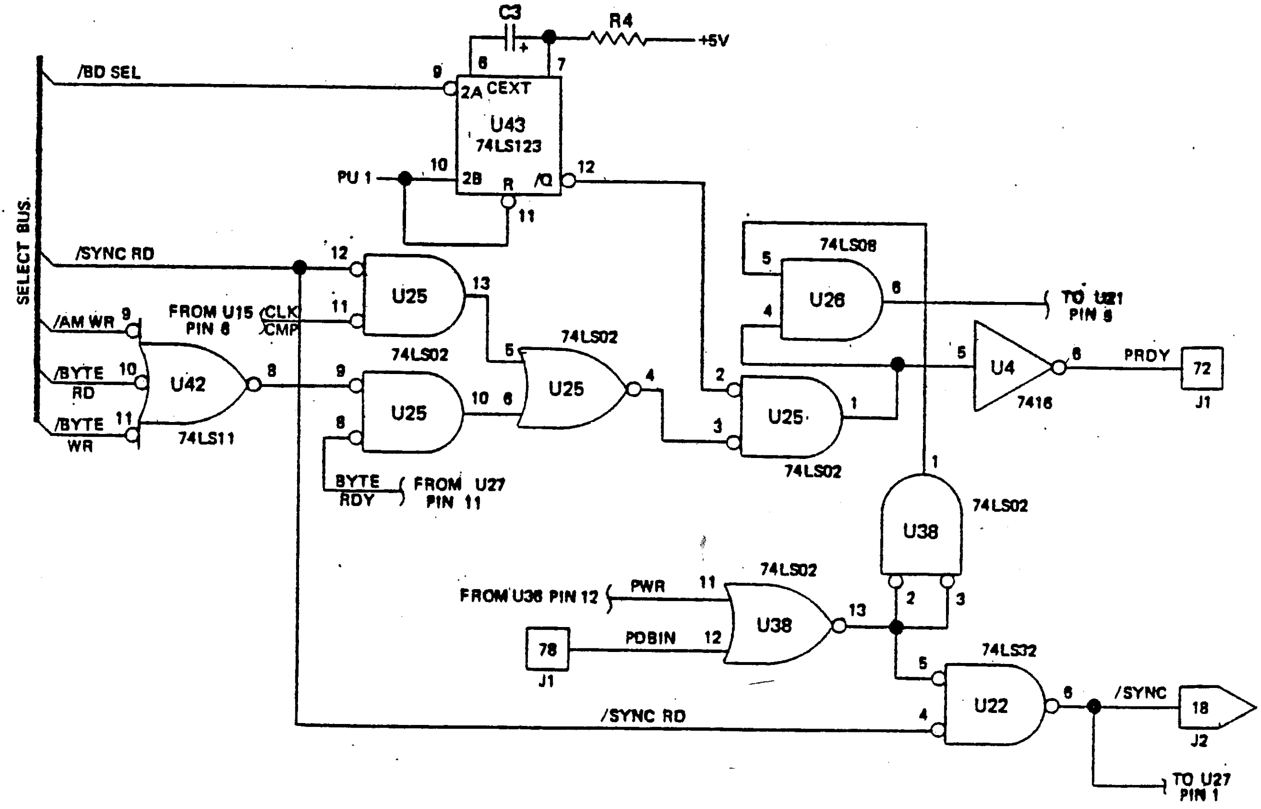


FIGURE 8 Ready Synchronization Logic

CRC TIMING

All other registers in the system shift on the low-to-high transition of the SHIFT signal. Due to the long setup and hold times required by the CRC chip it is set to shift on the high-to-low transition of this signal. The one exception is when the CRC value is to be shifted from this chip onto the Data Out Line. At this time the CRC chip must also shift on the low-to-high transition of the SHIFT signal. This is accomplished by having the CRC SHIFT signal select an alternate clock input. This clock is generated by the one-shot (74LS123) at U37 which is triggered on the low-to-high transition of SHIFT.

For writing, the output of the CRC chip is shifted into the 74LS164 at U13. This is required because five bits of data (two previous bits, the bit being written and the next two bits to be written) are needed to properly encode and compensate the data being written for the MFM format. Finally, the SHIFT signal is used to determine when a clock pulse (SHIFT is high) or a data pulse (SHIFT is low) is to be written. These seven bits are used to select one of 128 locations in the PROM located at U14 and the data stored in each location determines whether a pulse is required and how it is compensated.

The signals PWR and PDBIN are ORed and the result ANDed with the internal address decodes to prevent internal gating of signals when there is not a legitimate memory reference. The output of the 74LS08 at U26 is used to synchronize internal signals at the end of the Wait State.

AM SYNCHRONIZATION

There are two different synchronization requirements for the DIO. The first is waiting for the hardware to recognize the unique clock and data pattern associated with the Address Marks for the three formats. For a definition of these patterns, see Section 2. The recognition is performed by comparing the five clock bits (ignoring the MSB and two LSB) required to define the patterns. A 74LS85 at U15 is used to compare with the clock data value (deserialized by the 74LS164 at U20) with the value loaded in the 8255 by the firmware. When /SYNC RD goes low, the CPU is then put into the Wait State (U25 pin 13 goes high) until a comparison is found (U15 pin 6 goes high).

The other synchronization required is for the parallel transfer of bytes between the CPU and the DIO. The low active decodes of these signals are ORed by the 74LS11 at U42. Its output going low causes the Wait State to be entered until the BYTE RDY signal (from U27 pin 11) goes high indicating that output data has been taken or input data is ready.

CRC GENERATION

The remainder of the logic on the DIO is associated with the CRC generation and testing and forming the clock and data pulses for writing on the diskettes. A complete specification of the MC8505 CRC chip is attached, so this discussion will only describe how it is used. Figure 9 shows the logic involved in the CRC and write pulse generation. The serial input data comes from the 74LS395 shift register for both reading and writing. Therefore, when in the read mode one trailer byte (after the two CRC bytes) must be shifted into the shift register before sampling the /ALL ZERO output to determine whether there was a CRC error.

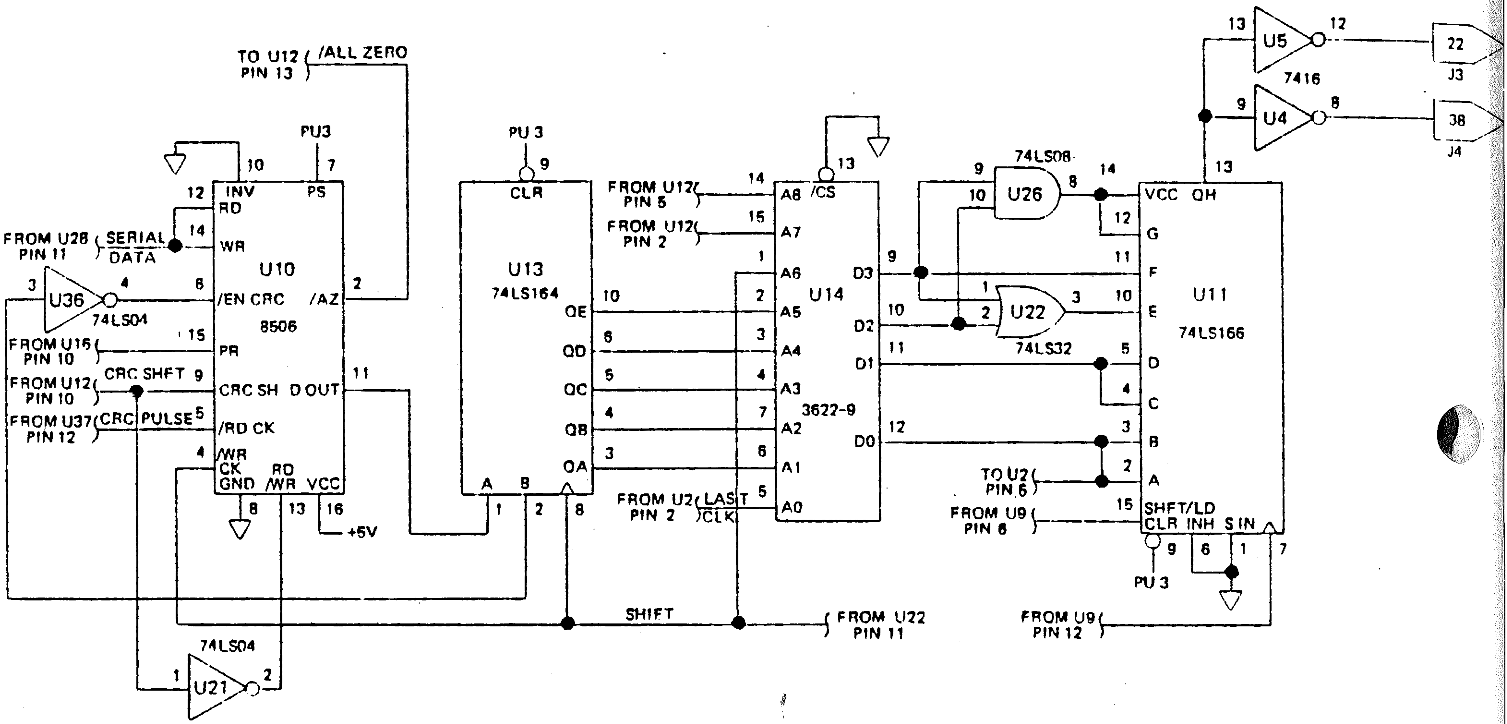


FIGURE 9 CRC and Write Pulse Generation (DIO-C)

NOTE: DIO-D differs from DIO-C thus:

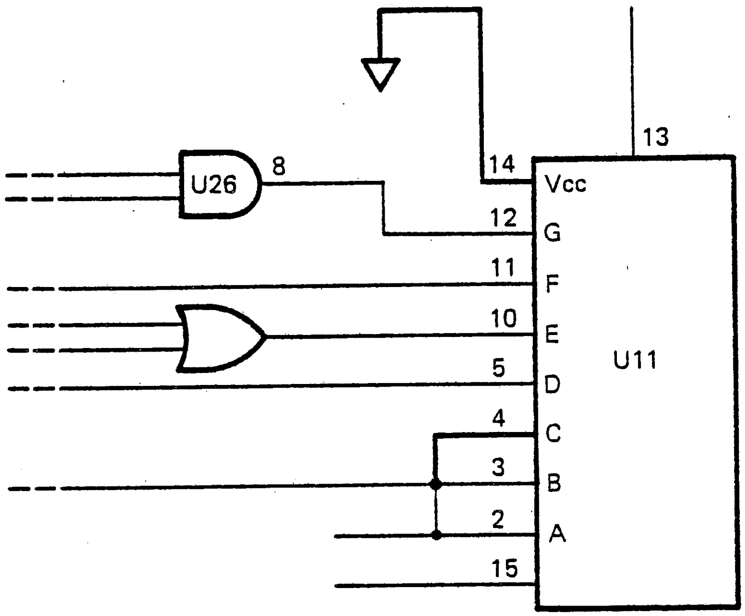


FIGURE 10

WRITE PRECOMPENSATION

The 3622 PROM has a 512 x 4 organization. The 512 locations are divided into four quadrants by A7 and A8 for the different encoding schemes as follows:

A8	A7	Used For
0	0	Encoding standard FM data
0	1	Encoding FM address marks
1	0	Encoding standard MFM data
1	1	Encoding MFM address marks

The 74LS166 shift register is parallel loaded twice per bit with the output of the PROM, once for the clock pulse and once for the data pulse. This is accomplished by having the output of the divide-by-ten chip at U19 (via U9 pin 6) activate the parallel load enable. There are 6 different values used in the PROM. The values and resultant compensation are as follows:

PROM Data	Pulse Compensation	DIO-C	DIO-D
0	No pulse		
1	Pulse, Compensated Heavy Late	(300 ns)	(400 ns)
3	Pulse, Compensated Light Late	(100 ns)	(200 ns)
7	Pulse, no compensation		
B	Pulse, Compensated Light Early	(100 ns)	(200 ns)
F	Pulse, Compensated Heavy Early	(300 ns)	(300 ns)

The times in parentheses are calculated based on the fact that compensation is only used for MFM encoding and for these formats the shift pulse from U9 pin 12 will be a 10 MHz signal on a DIO-C, a 5 MHz signal on a DIO-D.

Note that the PROM outputs together with the gates at U26 and U22 insure that whenever a data bit in the 74LS166 is parallel loaded with a one, all bits to the right of it will also be a one. Therefore, pin 13 of U11 will go high once (if at all) and stay high until all of the bits are shifted out (eight shifts at most). This in turn will cause negative going pulses at J3 and J4. All drives which interface with the DIO detect the leading edge of input data pulses and ignore the pulse width. Hence the difference in timing of the leading edge of the pulse generates the desired precompensation at the drive.

PDS II

INTRODUCTION

The PDS II board serves a supporting role to the DIO in the floppy disk interface process: it separates the clock and data pulses during a read from disk operation. The DIO can write straight to a diskette, but the ambiguities of the pulse stream from the disk drives must be resolved before the clock and data pulses can be separated.

The PDS II has two configurations: one for standard drives (PDS-II/S) and one for mini drives (PDS-II/M). The difference is in the values of those capacitors that establish the various timing parameters. The capacitors and their values are shown on the schematic.

FUNCTIONAL DESCRIPTION

Beside the power requirements derived from the motherboard connector (J1), the PDS II works entirely from inputs provided through the 20-conductor flat cable from the DIO. The list below identifies the signals. Note that all odd numbered pins are signal ground to provide shielding.

INPUT SIGNALS

CLK A (pin 8)

This signal is used to define the format of the input data as follows:

0	FM data at 125 KHz (PDS-II/M) or 250 KHz (PDS-II/S)
1	MFM data at 250 KHz (PDS-II/M) or 500 KHz (PDS-II/S)

Note that the aforementioned capacitors determine the frequency for each format.

/STD DATA (pin 10)

The raw data pulse stream input from standard drives. A high to low transition is used to signify a pulse.

/MINI DATA (pin 12)

The raw data pulse stream input from mini drives. As above, a high to low transition signifies a pulse.

/WRITE (pin 16)

Turns off the PDS II during a write to diskette.

/SYNC (pin 18)

Indicates to the PDS II that the DIO is searching for the next sync field and ID mark.

OUTPUT SIGNALS

CLOCK (pin 1)

This signal is high if there was a clock pulse in the previous bit cell. It is shifted into the DIO on the low to high transition of the PLO SFT pulse.

PLO SFT (pin 4)

The square wave output of the phase locking process. The low to high transition is used (one per bit cell) to shift the value of the data and clock lines into the appropriate register on the DIO.

DATA (pin 20)

This signal goes high if there was a data pulse in the previous bit cell. It is also shifted into the DIO on the low to high transition of the PLO SFT pulse.

Figure 1 illustrates the relationship between the disk drive(s), PDS II, and DIO.

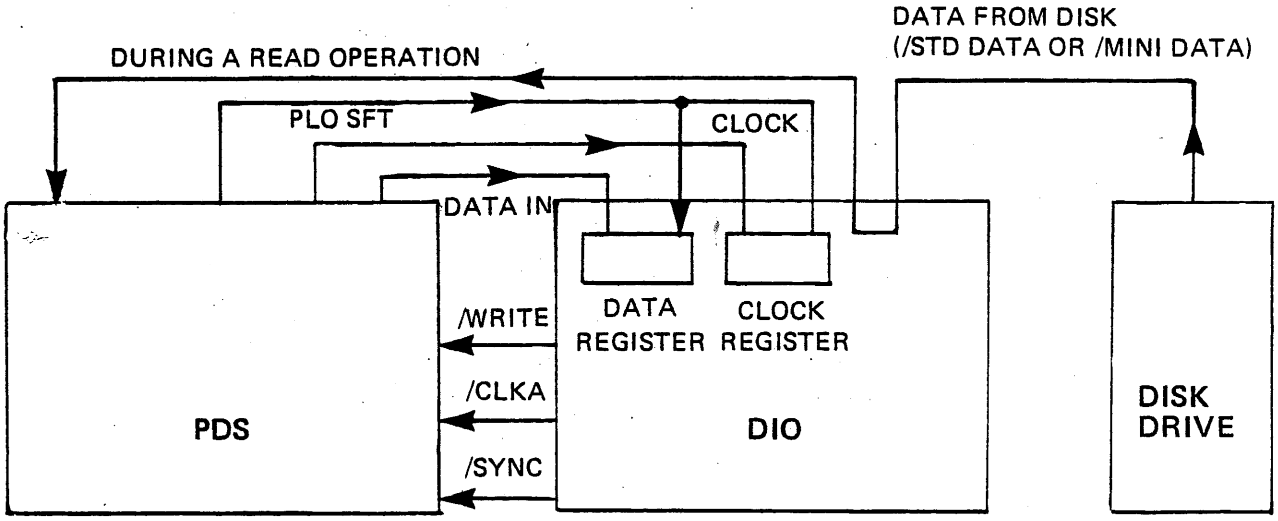


Figure 1

Note that the data from the drives (/STD DATA or /MINI DATA) is not processed by the DIO but sent directly to the PDS II.

There are several distinct functions performed on the PDS II that ultimately reduces the pulse stream to data and clock signals. Figure 2 identifies these functions and illustrates their interrelationship vis-a-vis signal processing.

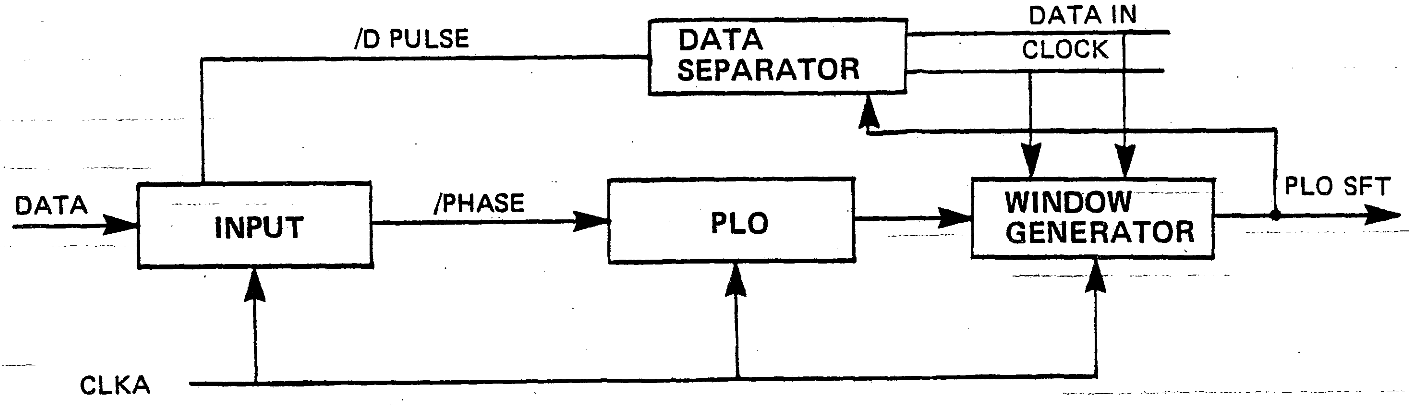


Figure 2
Block Diagram

The data from the Floppy Disk (/STD DATA or /MINI DATA) arrives first at the input section. Here the pulse stream is processed to yield two fixed duration signals (among others): /DPULSE and /PHASE.

The PLO section (Phase Lock Oscillator) is the next step for the /PHASE signal. Since each disk drive, even those of the same manufacturer, can have variations in motor speed, the PDS must self-adjust to accommodate these variations. This is the function of the PLO section. The VCF signal is the output of this process.

Once phase locked, a square wave signal of the appropriate frequency must be generated for comparison against the pulse stream (/DPULSE). The Window Generator provides this service. Supplemental to this is the requisite logic to distinguish a clock from a data pulse.

The Data Separator is now prepared to perform its role. The reference (PLO SFT) is compared against the pulse stream (/DPULSE). The resultant clock and data signals are sent on separate traces back to the DIO and shifted into the appropriate register using the same PLO SFT signal used for separation.

CLK A and /SYNC are sent from the DIO firmware to define two conditions. The former states for the PDS II whether the pulse stream is single or double density. The latter tells the PDS II whether or not it's sync'ed. (The DIO determines this by comparing the clock and data registers looking for the unique address marks found before Index, ID and data fields).

THEORY OF OPERATION

The above block diagram identified the functions and sections necessary for accurate pulse analysis. The following text is an in–depth description of each section: the components and their interralationship.

DATA SEPARATOR

Actual data and clock pulse separation takes place at the 7474s at locations U10 and U11. The signals involved are /DPULSE (from U6), /PLO SFT and PLO SFT (inverted /PLO SFT).

The /PLO SFT signal is derived from the divide by two function of the 7493 at U9. /VCF is the controlling signal. SHIFT, a square wave signal, leaves U9 and becomes /PLO SFT after passing through U5. U5 is a participant of the Window Generator circuitry (see below) performing as an inverter.

Figure 3 below illustrates the wave forms yielded by the interaction of these signals within these two ICs. The signal on top is a hypothetical pulse stream (/DPULSE). The next two waves are /PLO SFT and PLO SFT. They are used as the reference (clock input) for clock/pulse separation. The sequence illustrates all four possible combinations of clock and data pulses. The last three lines represent the Q output of U10 (Q1) and U11 (Q2 = pin 6; Q3 = pin 9). The rules for Q1 and Q2 are:

1) any time there is a pulse on /DPULSE, the output lines (pins 9 and 5, respectively) will go high;

2) the line will return low upon the rising edge of /PLO SFT (Q1) or PLO SFT (Q2).

All other inputs to U10 and U11 (the lower section) are held constant. The DATA signal is derived from U10. Note that there are some extraneous pulses on this line, but pulses are only shifted into the data registers on the DIO upon the rising edge of PLO SFT. The arrows at the bottom of the figure indicate when this occurs. Notice the Os and 1s above the DATA line indicating the status at that moment and the elimination of clock pulses.

The Q2 output is not used for the CLOCK signal. Instead, this output is fed into the other section of U11 as the D input. The signal is then clocked by /PLO SFT and yields Q3. Again all other inputs to this section of U11 are held constant. The resulting signal is then shifted into the clock registers of the DIO upon the rising edge of PLO SFT.

If you compare the DATA and CLOCK lines with the arrows at the bottom of the figure (denoting the rising edge of PLO SFT), you will see that the resulting signal levels of each represent the appropriate clock and data pulse from /DPULSE. Notice that Q2 does not accurately separate the clock pulses.

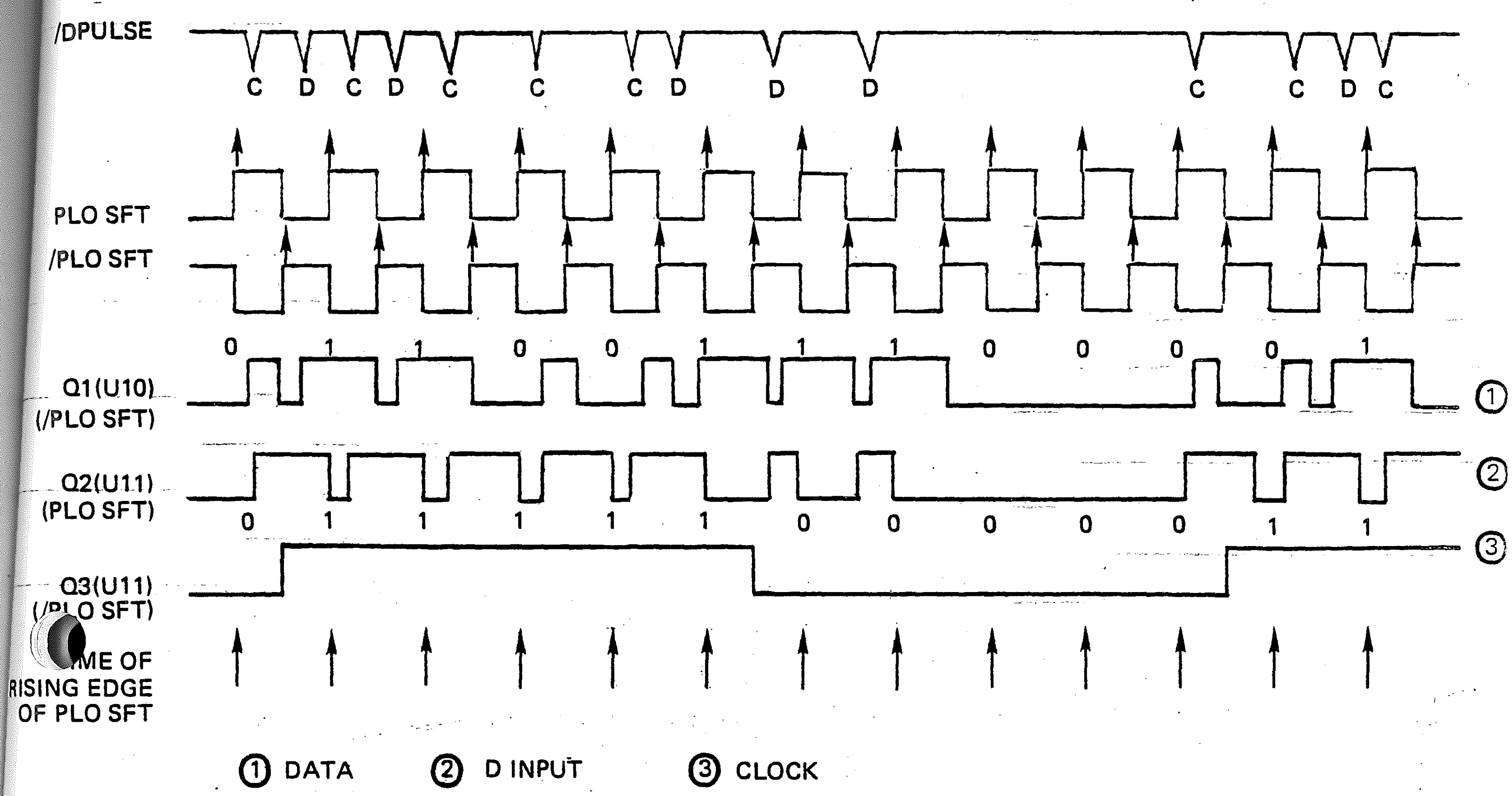


Figure 3

WINDOW GENERATOR

The Window Generator has two functions: first it must generate the reference for the Data Separator and DIO (PLO SFT); and second, it determines the identity of a pulse (i.e., whether it is a data or clock pulse).

The primary function is the first described above. The frequency of the /VCF signal from the Phase Lock Oscillator is divided by two at U9. For instance, the frequency of /VCF for a standard drive, double density recording format is 500 KHz. SHIFT, the preliminary output of the Window Generator, is generated at 250 KHz. Referring to the wave diagram, Figure 3, note that this is the desired frequency of PLO SFT (half that of /DPULSE).

The second function of the Window Generator is to determine whether a pulse on /DPULSE represents a data or clock bit. As the pulse stream is processed by the PDS II there is no reference indicating which is which. The PDS II must have a procedure for first identifying the pulse then adjusting if the pulses are being misinterpreted. The PDS II is assisted in this task by the data format on the disks.

Recall from section 2 – 2.5 that each Index, ID and DATA Address Mark is preceded by 6 or 12 (depending upon the encoding frequency) bytes of 0s. In both single and double density recording formats, this will register in the pulse stream as a sequence of clock pulses only. This sequence provides the reference needed to determine the nature of the pulse.

The only situation that need be checked for is a sequence of data pulses within this field. (If a sequence of clock pulses occurs, no adjustment is necessary.) Consequently, the PDS II circuitry tests the pulses on the DATA signal (again shifted by PLO SFT) at the 74LS163 at U12 for a sequence of 1s. Note that any pulse from the clock (in this case /CLOCK) will reset the counter. Consequently, only a sequence of data pulses with no clock pulses will set S ERR (for single density) and D ERR (for double density). The appropriate signal is chosen by the CLK A input to U7 and PH ERR results. The final affect of this process is to invert the SHIFT signal at U5 which begets /PLO SFT.

INPUT

The other contributor to the Data Separator is the Input Section. As well as generating /DPULSE for analysis, it provides a proper width signal for the Phase Lock Oscillator (/PHASE). Both of these signals are derived from the pulse stream from the floppy disk drive (/STD DATA or /MINI DATA).

As the signal arrives from the drive, it passes through an inverting Schmitt trigger to reduce the inherent "noise". (In fact, all inputs to the PDS II but CLK A pass through a Schmitt trigger for the same reason.) Notice that the two inputs (/STD DATA and /MINI DATA) are tied together. The source of the signal is irrelevant regarding disk type since the PDS II is manufactured for either standard or mini drives exclusively.

The single shot at U6 receives the pulse stream and generates a pulse just wide enough to activate the driver circuitry in the Data Separator. The other output, DPULSE is used by the Phase Lock Oscillator (see below). The width of the output pulse is determined by the resistor–capacitor (RC) pair at pins 9, 10 and 11. In this case, there is no resistor and the capacitor is a very low value. Consequently, a relatively short pulse exits that is just long enough for the data separator section.

The other function of the input section is to provide the signal that assists generation of PLO SFT. The Phase Lock Oscillator (PLO) intercedes between the Input section and Window Generator to create the signal phase locked with /DPULSE for the Window Generator. The input to the PLO is /PHASE. It is important that this signal be 1/4 bit cell wide for optimum centering of the /DPULSE pulses at the Window Generator. This length is established by the RC pairs attached to U1 and U2 (R1, R5 and C1; R2, R4 and C2 respectively). These ICs generate two signals, one for single density (/SD WINDOW) and one for double density (/DD WINDOW). Finally, the appropriate signal frequency is selected by the CLK A input from the DIO at U7. The selected signal, /PHASE, then goes into the Phase Lock Loop.

PHASE LOCK OSCILLATOR

Floppy Disk drives are not highly regarded for the consistency of the frequency of the pulse stream generated. Generally, this signal drifts and jitters due to variations in the spindle velocity and power. Ultimately, one wants to derive a reference signal from the input pulse stream. This signal is then compared with the pulse stream to distinguish the two types of pulses. The process to implement this distinction is somewhat analogous to a flywheel. The outcome of this process is to accommodate drifts but to ignore jitter. The diagram below (figure 4) illustrates the components of this process.

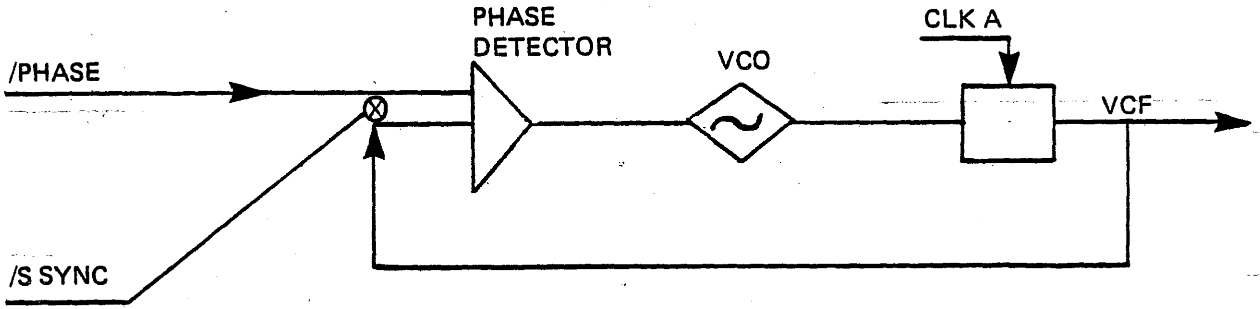


Figure 4

This section will discuss each aspect function by function. First, the PHASE DETECTOR.

PHASE DETECTOR

The dual D flip flop at U8 creates two independent outputs: /UP and /DOWN. Basically, these outputs reflect whether the /PHASE signal is before or after the VCF. The rules for each are:

/UP:	VCF=0 /PHASE	-> set
/DOWN:	/PHASE=0 VCF	-> set

Note that the pulses within /PHASE are 1/4 bit cell wide as determined by the 74121 at U1 or U2.

Figure 5, below, illustrates when a pulse will be sent on the /DOWN line. The pulse length is a function of the amount of phase difference. When either /PHASE or VCF go high, there is a reset.

Note that the width of /DOWN represents the degree of discrepancy.

Conversely, analysis of the /UP signal would indicate that the /PHASE pulse occurs before the VCF. Thus, the lines are mutually exclusive.

At this point the /S SYNC signal is introduced to create two levels of the pulses of /DOWN and /UP. A digression is in order at this point to define /S SYNC.

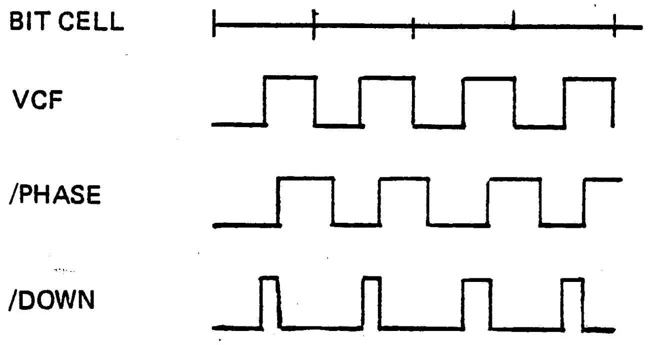


Figure 5

The /SYNC signal from the DIO is low active when sychronizing and inverted at U3. At U13, this signal is clocked by the DPULSE signal from U6. The /Q output, /S SYNC, then is high or low, depending upon whether or not the input is synchronized or not (as determined by the DIO firmware). /SYNC is clocked by DPULSE to prevent any transition of /S SYNC in the middle of a phase measurement.

The effect of /S SYNC on the /UP and /DOWN signals is to increase the amplitude of either of these signals when /S SYNC is low (to about 12 V) or decrease either when it is high (to about 3 V) depending upon whether the DIO has encountered the ID mark or not (/S SYNC is low until the ID mark has been read, then high).

The rest of the circuitry leading to the Loop Filter (U15) is used to translate the pulses from PUP and PDOWN into levels (performed by the RC pairs at R25/C4 and R23/C5) and to set a free running voltage to allow for chip to chip differences of the VCO (the pot at R3).

LOOP FILTERS

The Loop Filter (U15) is an active low pass filter. Its function is to provide a single output from the two inputs FUP and FDOWN by using the differential inputs of the amplifier. The output is the voltage which will be used by the VCO (U4). Remember that this voltage is originally a function of the amount of difference (the pulse width of /UP or /DOWN). The pulses are then translated into a precise level and finally processed by the loop filter to yield a signal for the VCO.

VOLTAGE CONTROL OSCILLATOR (VCO)

As the difference between the VCF and /PHASE signals are converted to a voltage level, the 74LS124 at U4 adjusts the frequency output (signal VCO) to rectify the difference. All inputs to the VCO chip are held constant except for the control voltage (input pin 1). Thus the VCO frequency is matched with the frequency of the pulse stream from the drives.

The VCO output is clocked at 2MHz for standard drives and 1MHz for minis. (This is, of course, the ideal state. The actual frequency is relative to the input from the drive.) However, the frequency of the pulse stream is 125 KHz single density, 250 KHz double density from mini drives or 250 KHz single density, 500 KHz double density from standard drives. Consequently, the VCO output is divided by the 7493 at U9, a 4-bit binary counter. The resulting signals, SD VCF and DD VCF, represent the proper division of the VCO.

The loop is completed at U7 where either SD VCF and DD VCF is selected by the CLK A signal. VCF results and goes back through the comparison with /IPHASE for further adjustments.

This completes the Phase Lock Loop circuitry. When the two signals are locked, /S SYNC goes high, decreasing substantially the voltage changes to the VCO chip. The self-adjusting features will now follow any "drifts" in the frequency.

PDS II ALIGNMENT

The settings of the potentiometers at R1, R2, and R3 can only be performed by a factory-authorized technician. However, once set the board will not wander out of alignment. DO NOT attempt to re-adjust or replace these or any other components. If you have any problem with the PDS II, contact IMSAI Customer Service.

CLOCK AND DATA SEPARATION

The leading edge of the PLO SHFT pulse is used to gate the present value of the two flip–flops into the DIO Board. It also triggers the one–shot (74LS123) at U11 to make CLK DATA and DATA in both low. If a RAW DATA pulse occurs while PLO SHFT is high then CLK DATA (U18 pin 9) will be set high. The exclusive or gate at U25 (in this case used as an OR gate since the inputs are mutually exclusive when RAW DATA occurs) is used to cause CLK DATA to remain high if a RAW DATA pulse also occurs when PLO SHFT is low — in that case this bit cell would have both a Clock and Data Pulse in it.

SELF ADJUST LOGIC

Figure 16 shows the test pulse generation used for the self adjust features on the PDS. The four switches are used to determine adjustments with Switch 1 (S1) on for normal operation and all others off. For adjusting S1 is off and S4 is on while S2 and S3 (via U10 pin 12) determine what is to be adjusted as follows:

DIO–C	SW2	SW3	Adjustment	Test Rate
	Off	On	1 Microsecond OS	125 KHz
	On	Off	2 MHz VCO Chan.	250 KHz
DIO–D				
	Off	Off	2 Microsecond OS	62.5 KHz
	On	Off	1 MHz VCO Channel	125 KHz

The test rate is the frequency at which the RAW DATA input oneshot is triggered. The LED on the PDS is lit whenever the 555 timer chip is not triggered. This occurs when the output of reference selector (Figure 16) and the signal selector in this figure have the opposite polarity. The timing delays of the logic loop account for the light not responding at the precise point where the potentiometers are correctly adjusted.

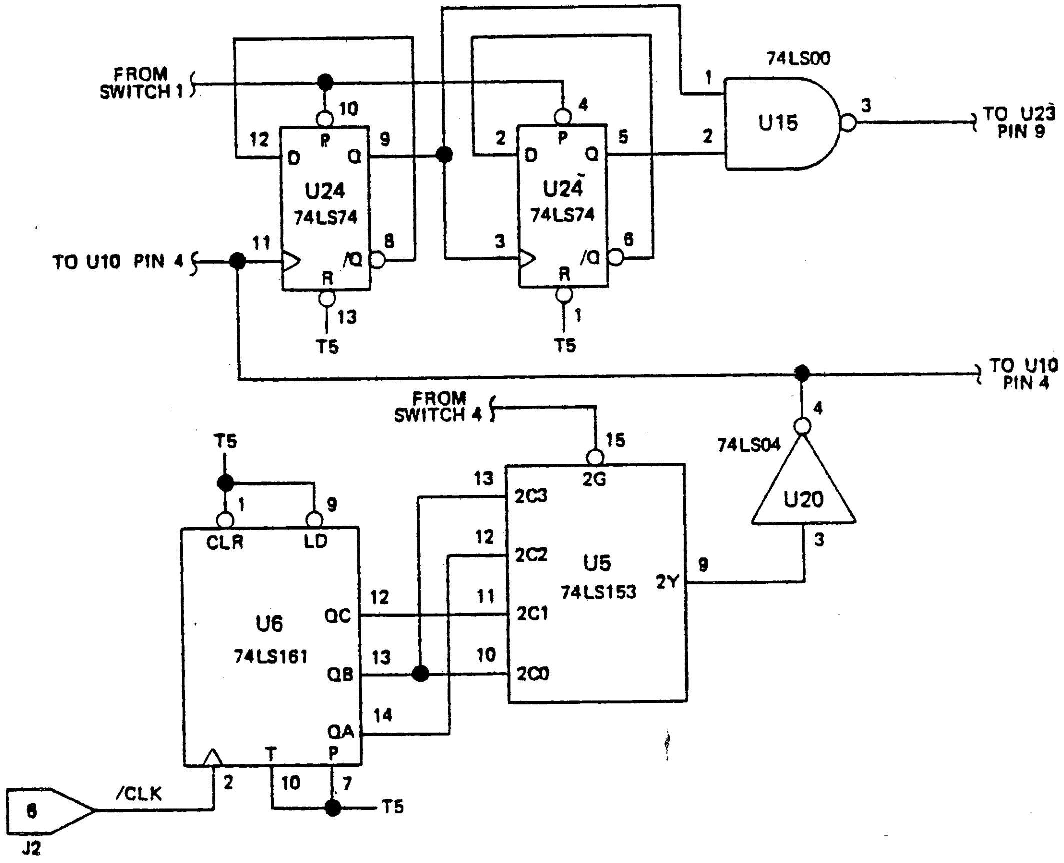


FIGURE 16 Pulse Generation

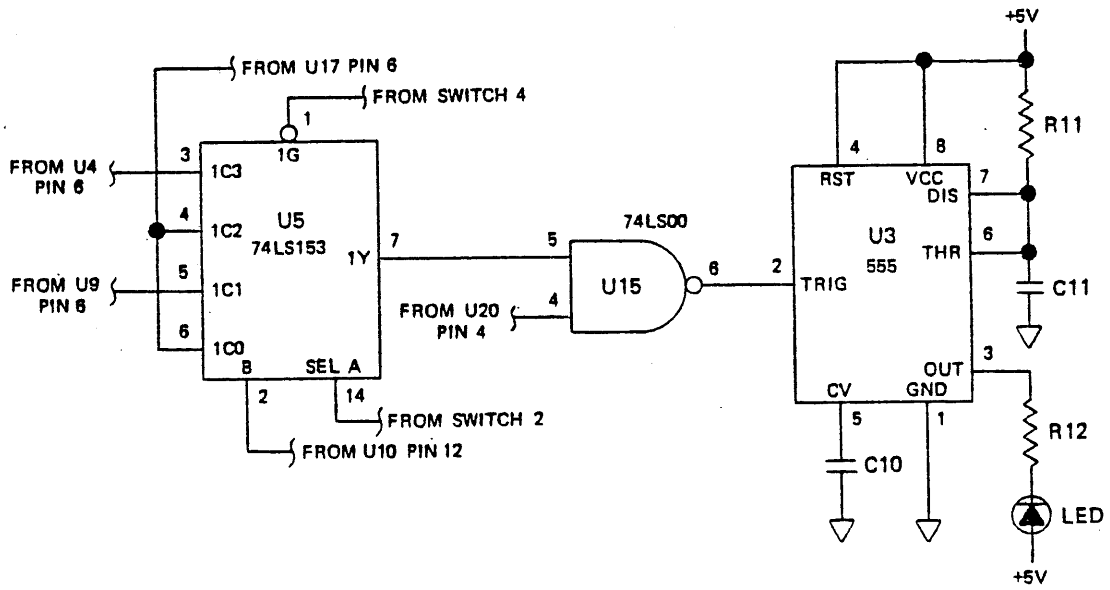


FIGURE 17   Test LED Indication

PDS ALIGNMENT PROCEDURE

The PDS board is designed to operate with the DIO board to form a floppy disk controller.

The four position DIP switch S1 (at location U2) is used to control the self-adjust features. For normal operation, switch 1 should be on with all other switches off. To perform the internal adjustments the following procedure should be followed. The discussion assumes DIO-C configuration. For PDS alignment is a DIO-D configuration, use references in brackets, [ ], instead.

1. Attach the PDS board and the DIO board with the 20-conductor flat cable.
2. Remove the the standard floppy interface cable from J4 on the DIO board.
3. Set S1 and S2 off and S4 on [S1, 2 and 3 off and S4 on] on the PDS board.

Our experience is that the factory prior to shipping of the DIO/PDS is very stable. Therefore, this procedure is to be used only if there is a high frequency of data synchronization errors (type 93 - refer to Section ...).

The set up will include the following items, though not all of them will be in use at any one time.

PCS-80 series chassis
MPU-A or B
DIO (known good)
Cable AG
S-100 Extender Board
Oscilloscope

The oscilloscope used in this procedure is a Tektronix Model 465. Its screen size is 10 cm long by 8 cm high. Most measurements refer to this particular screen. This test is easily modified for other oscilloscopes.

These adjustments set up both the intervals of two time-base one shots and the feedback bias of two voltage controlled oscillators.

ONE [TWO] MICROSECOND ONE SHOT ADJUSTMENT

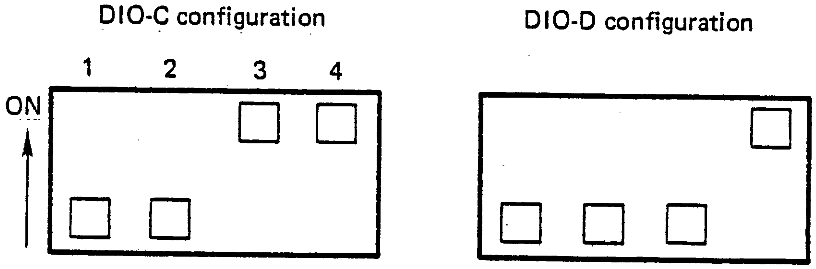
Initial Set Up

Attach Channel-1 scope probe ground lead to the ground loop on the extender board (S-100, pin 50).

Attach cable AG to J2 of the PDS board, making sure that conductor 1 of the cable matches pin 1 of J2. The other end of the cable is attached to J2 of the

DIO board. No other cables should be connected to the DIO.

PDS S1 should be set to the configuration shown below.



The scope controls should be set as follows:

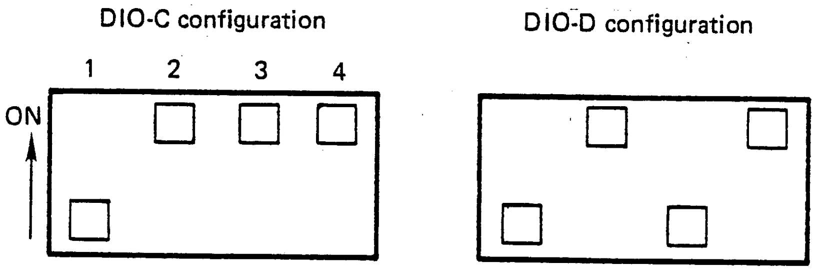
MODE	CHAN-1
Volts/Div	2
Time/Div	0.1 [.2] microseconds
Trig Source	CHAN-1
Trig Couple	AC, Normal
Trig Slope	+

At this point turn on AC power. To adjust the two microsecond one shot, monitor U4-6 (IC at U4, pin 6) [U9 - 6] with the channel-1 scope probe. Turn R6 [R3] clockwise (CW) or counterclockwise (CCW) until the pulse is exactly 1 [2] microsecond (i.e., the trace fills the entire scope screen).

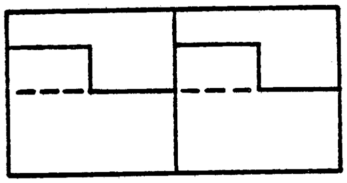
TWO [ONE] MHZ VCO CHANNEL ADJUSTMENT

Set Up Changes

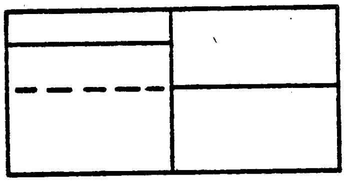
Reset S1 on the PDS board to the following.



Monitor U13-7 with the CHAN-1 probe. Adjust R36 [R35] CW or CCW until the following waveform appears. Failure to attain this waveform indicates a bad VCO chip.

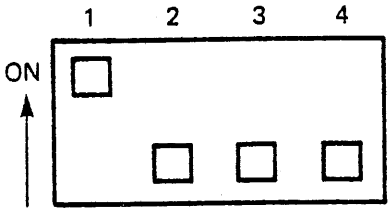


Next monitor U17-5 with the probe. Readjust R36 [R35] CW or CCW until the following waveform appears. Note that the adjustment should continue until the falling edge of the waveform coincides with vertical center of the scope screen.



This completes this set of tests. If these procedures do not succeed and the desired waveforms are not achieved, more detailed troubleshooting will be required.

Before reinstalling the PDS into your computer, reset S1-S4 on the PDS to the configuration shown below.



6 - 6 SCHEMATICS AND I/O INTERCONNECT

DIO-C TO DRIVE INTERCONNECTION

J4 PIN #	SIGNAL NAME	SIGNAL DIRECTION	NOTE
2	DSK SEL	——>	
4	HD LD 1	——>	
6	RDY 1	<——	
8	INDEX	<——	PERSCI ONLY
10	SEEK COMPL	<——	PERSCI ONLY; 2-SIDED FOR OTHER DRVs
12	RESTORE	——>	PERSCI ONLY
	DSK CHG	<——	OTHER DRIVES
14	REM.EJ0	——>	PERSCI ONLY; SIDE SEL FOR OTHER DRVs
16	LO CUR	——>	> TRK 43
18	HD LD	——>	
20	INDEX	<——	NON-PERSCI DRIVE
22	READY	<——	
24	MOTOR ON	——>	
26	DS 1	——>	SELECT DRIVE 1
28	DS 2	——>	SELECT DRIVE 2
30	PS 3	——>	NON-PERSCI DRIVE
	WT PROT	<——	PERSCI DRIVE
32	REM EJ1	——>	PERSCI ONLY; DRV SEL 4 FOR OTHER DRVs
34	DIR SEL	——>	DIRECTION
36	STEP	——>	
38	WT DATA	——>	
40	WT GATE	——>	
42	TRK 0	<——	
44	WT PROT	<——	
46	RAW DATA	<——	
48	——		
50	——		

NOTE: All signals are low true, except DIR SEL which is bi-polar. All odd-numbered J4 pins are signal ground.

——> means: control signal to drive
<—— means: sense line from drive

DIO-D TO DRIVE INTERCONNECTION

J3 PIN #	SIGNAL NAME	SIGNAL DIRECTION
2	——	
4	——	
6	READY	<——
8	INDEX	<——
10	DS 1	——>
12	DS 2	——>
14	DS 3	——>
16	MOTOR ON	——>
18	DIRECTION	——>
20	STEP	——>
22	WT DATA	——>
24	WT GATE	——>
26	TRK 0	<——
28	WT PROT	<——
30	RAW DATA	<——
32	——	
34	DS 4	——>

NOTE: All signals are low true, except DIRECTION which is bi-polar. All add-numbered J3 pins are signal ground.

——> means: control signal to drive
<—— means: sense line from drive

THE STANDARD OF
EXCELLENCE IN
MICROCOMPUTER
SYSTEMS

IMSAI

IMSAI MANUFACTURING CORP.
14860 Wicks Boulevard
San Leandro, CA 94577
(415) 483-2093
TWX: 910-366-7287

IMSAI EUROPE
Europa Terrassen
8 Rue Jean Engling
Dommeldange, Luxembourg
43-67-46 TWX: 1428