FLOPPY DISK CONTROLLER

FOR

M6800 BASED SYSTEM

by

E. Fuad Ertüzün

B.S. in E.E., Boğaziçi University, 1981

Submitted to the Institute for Graduate Studies in Science and Engineering in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical Engineering

Boğaziçi University

1984

To Ayşın

ACKNOWLEDGEMENTS

I am grateful to Dr. Ömer Cerid for his guidance and helps till the beginning of the thesis, and for his cooperation in our obstinate work to operate the realized system. I would also like to appreciate the patience of my parents during the design of the thesis.

I wish to express my thanks to Sedat Yılmazer, who first gave me the idea for the thesis, Semih Pekol for his assembler, Mustafa and Ahmet Niksarlı for their work in developing the films, and Nuri Çolakoğlu for the supply of the missing components.

I also would like to thank Cüneyt and Fatih for their encouragement in our late works, and Gülsen Karşit for the typing of the manuscript.

ABSTRACT

The purpose of the thesis is to design and realize a single density floppy disk controller, in order to extend and increase the capacity and capabilities of a M6800 based microcomputer system. In order to create a powerful microcomputer, an operating system software is adopted to gain the features of a development system.

The floppy disk controller can be wieved as a module for an existing system; however, the aim of the thesis is not only to add a module, but to integrate the controller, the central processing unit, and the operating system to achieve the full microcomputer. A memory module is incorporated to increase capacity, and group memory elements on a single board. To lower component cost the hardware is minimized and all possible controls are left to the software.

The development system can be used for editing and entering programs, converting assembly or high level language programs to machine code by using assemblers and compilers, running these machine code programs, and linking separate programs to create the software of different kinds of microprocessor products.

ÖZETÇE

Günümüzde, mikroişlemcileri çok daha güçlü duruma getirmek ancak yazılım yoluyla olmaktadır. Merkezi işlem biriminin bellek sığası ne kadar geniş olursa olsun, her çeşit programın işlemci sisteminin içinde durması, sistemin esnekliği ve kullanımı açısından sakıncalar yaratmaktadır. Bilhassa besleme gerilimi kesildiğinde programlar kaybolmakta, dolayısıyla tüm yazılım bilgisinin salt oku belleklerde saklanması gerekmekte, bu da yazılımın değiştirilmesini engellemektedir. Bu tezin amacı, bir sistemin yazılımının taşınabilir bir ortama aktarılması ve gerektiği zaman kullanmayı sağlamak için bir esnek döner manyetik bellek denetleyicisini tasarlayıp gerçekleştirmektir. Bunun yanında, eldeki M6800 mikroişlemcisine dayalı sistemin bellek sığasını arttırmak ve bellek elemanlarını bir yerde toplamak amacıyla bir ünite geliştirmek, ve bir işletim yazılımını uyarlayarak bir yazılım geliştirme aracı gerçekleştirmek esastır.

Geliştirme araçlarında, programların değiştirilmesi ve girilmesi, alçak ve yüksek düzeyli dildeki programların çevirici ve derleyiciler vasıtasıyla makine diline çevrilmesi, bu programların denenmesi, ve her çeşit mikroişlemci mamulu için yazılım gerçekleştirilmesi olasıdır.

TABLE OF CONTENTS

CHAPTER 1

THE M6800 SYSTEM AND THE NEED FOR THE FLOPPY DISK CONTROLLER

Although the MC6800 was the first of the Motorola MPU family it still remains as a highly cost effective processor for a great number of process control and data-communications applications (2). Its instruction set and addressing modes give it a powerful capability, and a full range of compatible peripheral chips offer the widest possible latitude in system implementation. The floppy disk interface is in fact the result of such an extendable and flexible system designed previously (9).

It should be kept in mind that the family of parts surrounding the M6800 is not a chip set in the sense that the MPU operation is dependent upon other family elements. The M6800 is a self-contained microprocessor capable of operating with virtually any MOS or standard TTL device. The significant point is that the other family members merely add additional capability and/or flexibility. They provide excellent tools in configuring a full microprocessor system (19).

The starting point of this thesis was a readily designed M6800 based extendable, general purpose microcomputer (9). This system as all microcomputer systems is centered around an MPU unit which handles the four functional categories: data, address, control and supervisory. For communication with the outside world serial and parallel Input/Output channels are incorporated together with memory which makes the system a microcomputer.

For minimum cost, an 8 bit instruction format for the data bus is choosen which makes the architecture simple and flexible. The address bus of the system not only specifies memory, but it becomes a tool to specify I/O devices. By means of its connections to the data and control buses and selected address lines, the I/O interface is allocated an area of memory. As a result, the user may converse with I/O such as PIA, ACIA using any of the memory reference instructions, selecting the desired peripheral with a memory address (19). The resident program such as the system monitor program is kept in ROM. However, the RAM sections can be used to execute, debug or enter programs; another need for the RAM is that it can store the program variables and all kinds of data.

In such a system only small, self-operable programs can be entered, debugged and saved on paper tape using the TTY interface. When the program becomes a little "massy" the saving and reloading operations takes too much time.

Also there is no possibility to keep the source forms of programs, but the direct instruction code has to be entered in order to execute the program.

By using the floppy disk interface together with a sufficiently powerful operating system and its support programs the following are possible:

- Saving and reloading of programs to/from a magnetic media called "diskette".
- Entering programs in assembly language to get the source and code in form of a list file for a printer or in direct hexadecimal form to program the EPROM.
- Possibility of bringing together many different programs and forming their code all together (linkage of programs).
- Editing source programs - that is changing any part of a diskette file by using an editor program.
- Locating program code and data to any desired place.
- Usage of high level languages, compilers, disassemblers and all kinds of support software.

Although all this software support and the operating system need direct use of the diskette contents, an easy communication has been established between the controller

and the user programs. The controller can be easily installed and tested separately, then the operating system activated. Only entry points for the controller are fixed and there is no interest by the user of how the firmware achieves the desired diskette input/output. In this way, the controller software or hardware can be changed anytime without affecting the operating system at all.

CHAPTER 2

INTRODUCTION

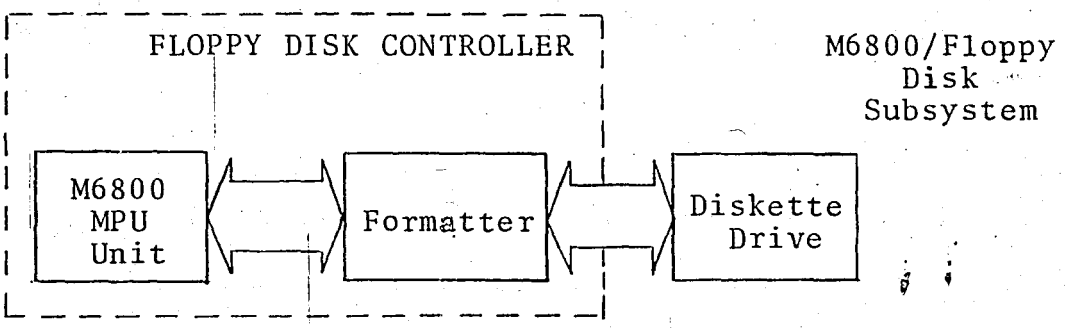
The floppy disk is fast becoming an important storage media. The promise of low cost and direct access has encouraged microcomputer users to select the floppy disk for mini-mass storage requirements. As microporcessing systems enter the market place, proposed applications for the floppy disk broaden to include the following:

- Program loaders for intelligent terminals and larger systems
- Key to disk key punch replacement
- Price look-up and credit card verification
- Message buffers for communication systems
- Word processing systems
- Minicomputer programs and auxiliary data storage
- Small business systems and batch terminal data storage
- Microprogram loading and error logging. (19, 30)

In these applications, the floppy disk will contend with cassettes and paper tape. The attractiveness of floppy disk over other means of mass storage rests in:

- Cost per bit of usable storage
- Cost of the floppy disk sub-system
- Reliability and maintainability
- Ease of media handling and transportability
- Compatibility of recorded data with other systems, large or small. (19)

The floppy disk itself (referred to as a diskette) is a removable magnetic storage media which is permanently contained in a paper envelope. The diskette drive is a low cost, random access storage device, which performs the electro-mechanical and read/write functions necessary to record and recover data on the diskette. Data is recorded serially on the floppy disk. Due to the high serial data rates, it is necessary to use auxiliary logic for the serial/parallel conversion, data recovery, and data error checkings when interfacing the floppy disk to the M6800 system. The hardware which performs this function is usually called a formatter. The formater also serves as a buffer between the M6800 system and the disk. It should be noticed that without the MPU unit no control is possible so that the formatter together with the MPU forms the controller (19, 30).



As used here the term "controller" includes not only the system hardware, but also those microprocessing system programs which directly or indirectly control the diskette drive. The program routines for the floppy disk are often referred to as floppy disk drivers or control modules.

Before describing the system it is of interest to discuss the trade-offs involved in microprocessor based floppy disk controllers. The decision to design a floppy disk controller using the M6800 depends upon:

- The way the disk system will be used in the overall system.
- The cost difference between the alternative design methods (such as hardwired logic).
- Both the short term and long term goals with respect to the use of the disk subsystem.

Due to the high data rates of the floppy disk, the microprocessor is, in effect, busy 100% of the time during data handling. This means no other peripherals can be serviced while in a disk read/write operation. This constraint is not too much important for a laboratory system, however, for a dedicated multi-purpose system Direct Memory Access (DMA) should be used.

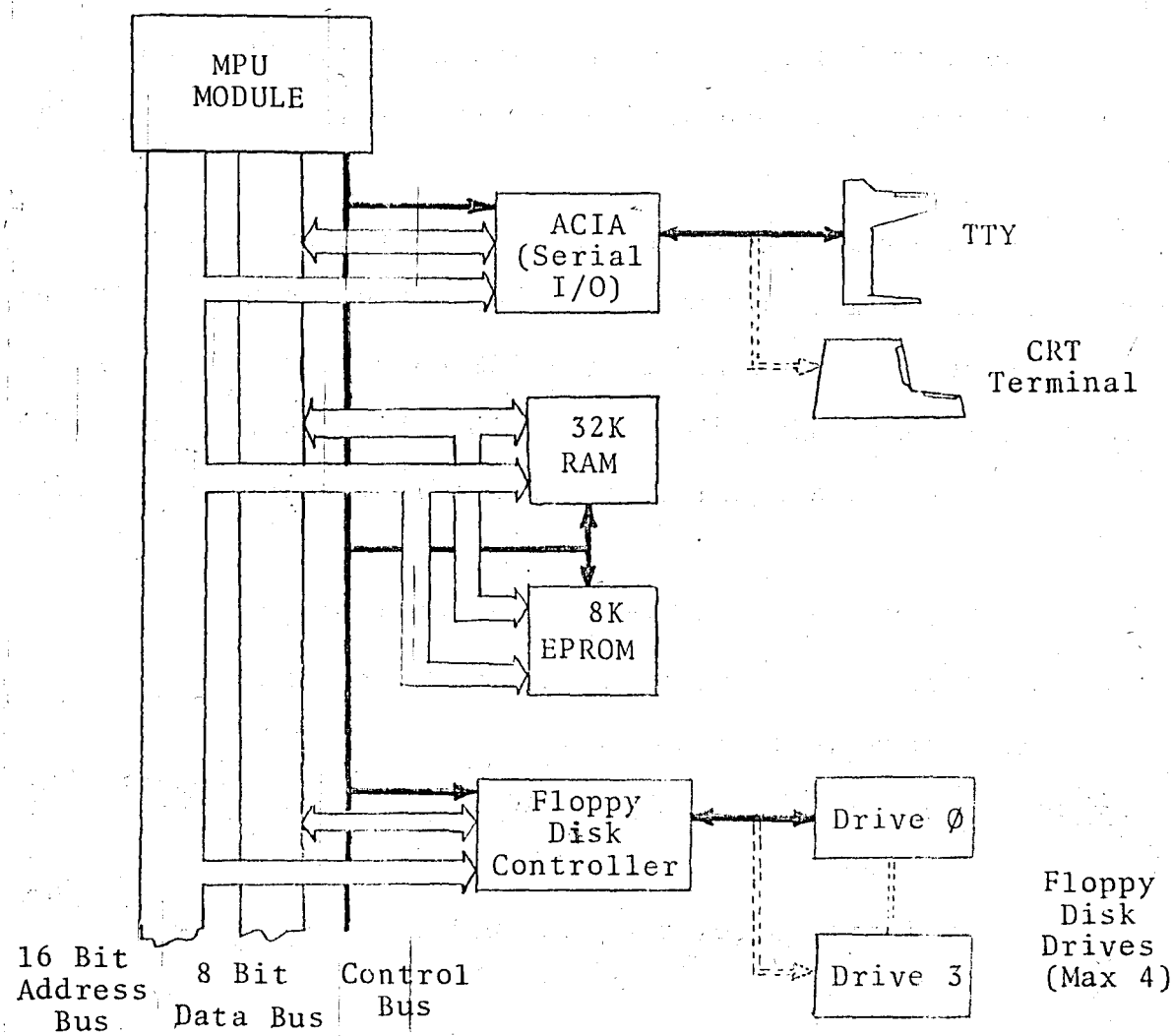
Since no other peripherals can be serviced, allowances must be made in the system design to permit 100% system dedi-

cation to floppy disk during read and write operations. To use a serial non-overlapping task system or to dedicate another processor can be a solution. However, these are cost effective solutions and in addition to that, as mentioned before, the system in hand has only the purpose of debugging and developing of software and hardware for the final product, so that the design is held at the minimum cost.

Only two new boards will be added to the system. These are the controller module which contains the formatter hardware and the dynamic RAM and EPROM module. The board contains all the system resident software which is the system monitor and the floppy disk firmware. Also 32K dynamic RAM is included on the same board which serves as a buffer for the CODOS operating system, itself being 8K.

The system consists of four modules; two of them are the previously built MPU module which has CPU, clock generator, single instruction hardware for step by step operation, bus drivers, and the asynchronous serial input/output (ACIA) module holding 4 ACIA's which provides the TTY interface. The TTY is of low speed and not much suitable for editing functions so that it can be replaced by intelligent CRT terminal whenever one is available. This replacement would affect a little bit the Input/Output software of the monitor but would provide many advantages to the user.

The controller module has a built in clock generator



SYSTEM BLOCK DIAGRAM

for data synchronization, a PIA for drive selection, 256 bytes of RAM for the operating system use, the floppy disk controller chip, and the data recovery circuit. A 40 pin connector should be provided for the interconnection with the diskette drive. Up to 4 drives can be attached in the so called "daisy chain" configuration. In this configuration some of the lines are shared and some are dedicated (such as the drive select line). Its disadvantage is that only one drive can be operational at onetime. On the other hand, in the "radial"

interface, all interconnections are dedicated to specific drives which isolates all drives but uses much more hardware.

The disk drive used for the design developped was the Siemens FDD 200-8. The double sided, removable disk cartridge will store up to 256 Kbytes for single-sided and 512 Kbytes for double-sided diskettes using the compatible IBM system 3740 format. The power requirements are as follows:

- 230V AC (180V to 253V)- 50 Hz ±0.5 Hz, for the diskette motor. Motor current at start: max 0.4A
  Motor current at run: max 0.2A
- +24V ±5% DC, 2.0A max for stepper-motor, read/write head.
- +5V ±5% DC, 1.0A max for logic circuitry.

The +24V supply should be provided separately since no +24V supply exists on the microcomputer system. However, +5V can be supplied from the built-in +5V supply.

The RAM and EPROM board, in effect replaces 9 boards of the old design. (8 x 4K state RAM modules + one 8K EPROM module). The RAM's used are the very low cost 16K x 1 bit 4116 dynamic RAM's so that 16 of them are used. These RAM's have the extra disadvantage of additional circuitry for refreshing and the need of 3 supplies. However, whatever is the point of view there is at least a cost reduction of a factor of four-five if the usage of a single +5V 2K byte 2716

EPROM's are considered instead of the three supply 2708's.

In addition to all that hardware the software support should be given. This software which should be referred as the resident driver firmware has to contain various fixed entry points (addresses) available to the user to perform specific operations, some of whose are listed:

- Bootleader for the operating system
- Hardware initialization of the controller
- Error checking routines
- Error printing routines
- Read routines
- Write routines
- Cyclic redundancy checking routines
- Head movement controlling routines
- Write test routines.

These routines will be contained in an half 2716 so that they will be of 1K byte length. These routines may be modified at all times but the operating system will not be aware of it since the inputs and outputs and even the call address will not be changed. Whenever this isolated 1K program operates with the system hardware there is no chance that the operating system is inoperable, provided that all the user routines meet the parameter requirements.

CHAPTER 3

SYSTEM LAYOUT

Some arrangements have been made on the system's allocation of memory space (memory map). As mentioned before the system uses memory mapped I/O so that some place should be reserved to I/O devices in the memory space. Also some restrictions should be considered because of Motorola's floppy disk I/O and firmware location and the map has to be realized accordingly.

The largest memory space is occuppied by the 32K RAM. But it's placement is not so difficult since you have only two alternatives: either lower memory locations (A15=0) or higher memory locations (A15=1). There are too many reasons to place it at the lowest memory locations; these can be mentioned as follows:

- Direct addressing mode which saves one byte, or 33% of the program memory space, in each instruction (9).
- Allocation of the 0000 to 001F addresses to the standard Motorola floppy disk firmware variables to communicate with the operating system.

- Program allocation of the CODOS operating system starting from 0020 with a length of about 8K bytes.

Taking into account these reasons the RAM is put from 0000 to 7FFF. Another restriction is brought by the CODOS operating system which requires a RAM area at locations EF00 to EFFF. This 256 bytes of RAM consists of two 2112 static RAM chips and is located on the controller board since this board and the CODOS is completing each other. This area is also used as the scratch pad for the monitor.

In the M6800 system, an interrupt sequence may be generated by the three hardware signals: reset, nonmaskable interrupt ($\overline{NMI}$), interrupt request ($\overline{IRQ}$) or by the software interrupt (SWI). These events obtain the address of the interrupt routine in locations FFF8-FFFF. That is why EPROM addresses should be placed in the highest memory location (9). The EPROM's are thus in the range E000-FFFF.

One should immediately notice that in the range EF00-EFFF 256 bytes of RAM and the last 256 bytes of the second EPROM overlap. Of course, this does not occur since care has been taken so that the second EPROM behaves as a 1K EPROM if the appropriate harware setting is made on the EPROM board. This 1K EPROM which ranges from E800 to EBFF contains the 1K floppy disk software.

In fact, the total EPROM is of 7K byte size because of this arrangement. However, if one does not use the flop-

py disk controller so that the 256 byte RAM is out of the system the board could be easily modified for 8K byte EPROM.

The last items that have to be placed are the floppy disk controller and the drive select PIA. These are placed on the region EC00 to EC7F since EC00, ED00, EE00 and EF00 had been already decoded on the same board. Next with a minimal hardware EC00 is decoded to EC3F and EC40 to EC7F. In this way the FDC is at EC00 to EC3F and the PIA at OC40 to EC7F. However, the PIA needs four locations and it is given here 64 locations; similarly the FDC needs 8 locations and it is given 64. The reason is that full decoding is not done, in other words A3, A4, A5 are not used for the FDC and A2, A3, A4, A5 are not used for the PIA. In this way the PIA has 64/4=16 images and the FDC has 64/8=8 images. EC00, EC08, EC10 ... etc, are the same locations of the FDC and EC40, EC44, EC48 ... are the same locations of the PIA.

As a general result it is observed that the locations E800 to EFFF contains all needs of the floppy disk controller including the firmware EPROM, the I/O ports and the scratch pad RAM. F000 to FFFF could be assigned to a well sophisticated monitor and E000 to E7FF will be a general purpose EPROM for further expansion. The buffer will occupy the lower half of the memory and the I/O modules could be located between this RAM and the EPROM.

MICROCOMPUTER ADDRESS DECODING CHART (MEMORY MAP)

• Connected to the Device Address or Register Select Lines
x Not connected

Device	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	From	To
16K RAM	0	0	•	•	•	•	•	•	•	•	•	•	•	•	•	•	0000	3FFF
16K RAM	0	1	•	•	•	•	•	•	•	•	•	•	•	•	•	•	4000	7FFF
Reserved for Other Devices																	8000	DFFF
First EPROM	1	1	1	0	0	•	•	•	•	•	•	•	•	•	•	•	EC00	E7FF
Second EPROM	1	1	1	0	1	0	•	•	•	•	•	•	•	•	•	•	E800	EBFF
	1	1	1	0	1	•	•	•	•	•	•	•	•	•	•	•	or if wanted E800	EFFF
Floppy Disk Controller	1	1	1	0	1	1	0	0	0	0	x	x	x	•	•	•	EC00	EC3F
																	But used as EC00	EC07
Drive Select PIA	1	1	1	0	1	1	0	0	0	1	x	x	x	x	•	•	EC40	EC7F
																	But used as EC40	EC43
256 Byte RAM	1	1	1	0	1	1	1	1	•	•	•	•	•	•	•	•	EF00	EFFF
Third EPROM	1	1	1	1	0	•	•	•	•	•	•	•	•	•	•	•	F000	F7FF
Fourth EPROM	1	1	1	1	1	•	•	•	•	•	•	•	•	•	•	•	F800	FFFF

CHAPTER 4

DISKETTE RECORDING METHOD

FORMAT NECESSITY

The data on the diskette is recorded on 77 circular tracks, so that the read/write head has to move back and forth to find a specified record, which are organized in the form of files and given a name. These file names are saved on the first track (this may depend on the operating system) and consecutively their position on the diskette are recorded; in this way first, a small search for the file name is made on track 0, the file location (track address and location on track) plus its length taken, and the file is transferred to the computer's RAM, using the FDC firmware.

Also tracks should be subdivided into sectors since the overlap of a record to the consecutive track may cause the wasting of a whole track or any means should be found to determine the end of a record, which will cause another search operation for the CPU. With sectors only a few bytes will be wasted compared to the whole diskette capacity. Also the sector approach will give the length of files in count of sectors.

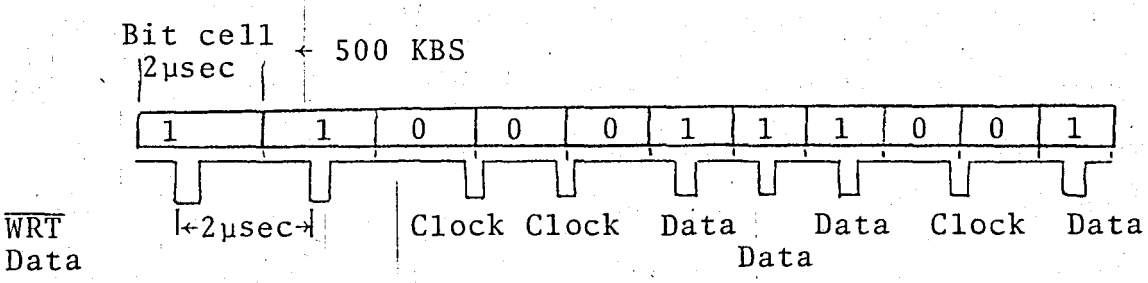
As explained, the floppy disk is in fact a sequential access device (SAM) so that the desired location on the diskette itself cannot be accessed directly. Some searches, some movements, such as positioning the head, finding the record have to be made. In such devices the access time cannot be specified but always care has been taken to reduce that time to minimum. Some other basic reasons for the format approach is to provide both an interval for switching the drive electronics from reading or writing and compensation for rotational speed, and other diskette-to-diskette, and drive-to-drive manufacturing tolerances to ensure that data written on a diskette by one system can be read by another (11).

Together with its advantages the format of a diskette has also many disadvantages:

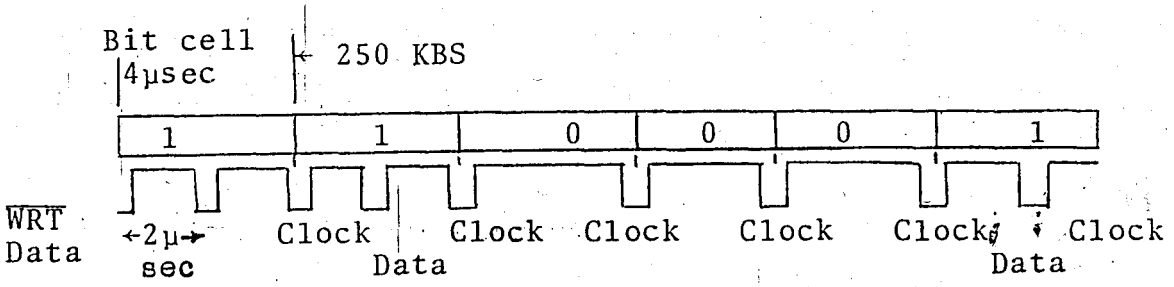
- About 35% of the recordable surface is reserved for the format.
- Each diskette should be pre-formatted and a format standard should be adopted.
- In the possibility of the format to be damaged this sector or track will be unusable.
- Controller logic is a little bit more sophisticated since sector, index or track identifiers should be found and compared.

IBM 3740 FORMAT FOR SINGLE DENSITY RECORDING

The recording format is dependent upon requirements of the controller, that is of the host system. Double-density or single-density recording schemes are possible for all drives but the read logic of the controller prevents all MPU's to handle double-density recording. In effect there are great differences between these schemes. The double-density bit cell is 2μsec compared to the 4μsec bit cell of the single-density; the main reason is that the recording encodings are different. MFM is for double-density (Modified FM) and only FM for single density. It is noted that although the minimum pulse spacing is 2μsec for each encoding - which does not affect the drive read/write logic and electronics - the decoding and encoding logics are very different which cause use of other hardware and software for each scheme (28).

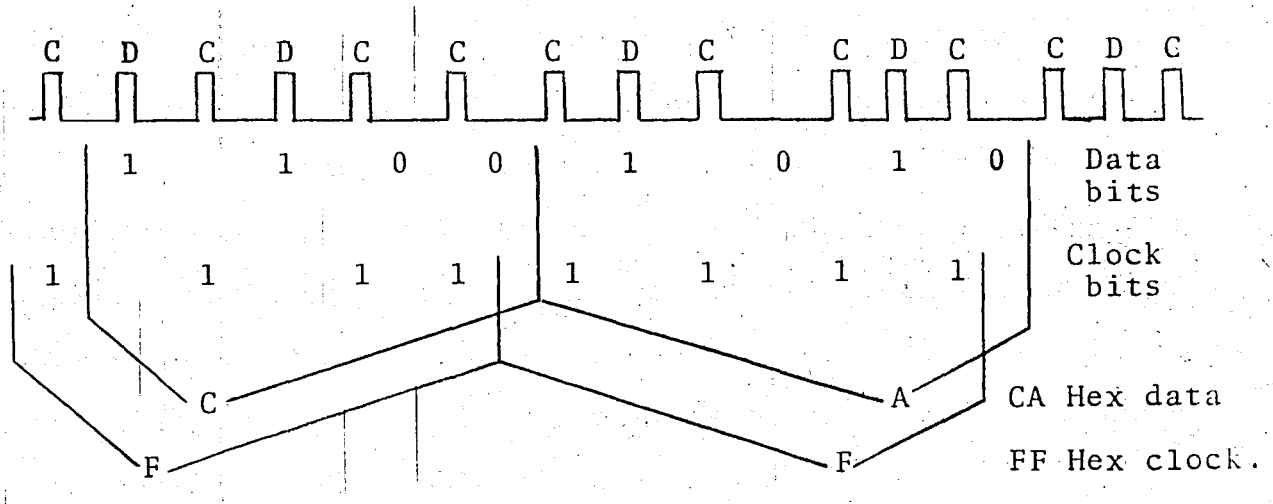


DOUBLE DENSITY ENCODING



SINGLE DENSITY ENCODING

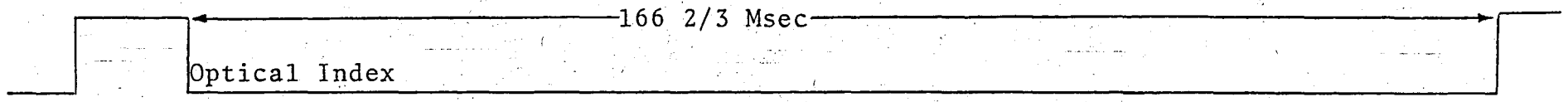
The single density is recorded on the diskette using frequency modulation as the recording mode, that is, each data bit recorded on the diskette has an associated clock bit recorded with it. A byte when referring to serial data, is defined as 8 consecutive cells. The MSB is transferred to the drive and from the drive first. The figure illustrates the relationship of bits within a byte and data and clock patterns.



The IBM 3740 Format is a soft sector format, that is, sectors are determined by software and not by any other hardware pulses. The only thing that initiates a track operation (formatting, read/write operation) is the index hole. All tracks on the diskette are referenced to this physical index mark and each time the hole passes a photodetector cell (one revolution of the diskette), the index pulse is generated to indicate the logical beginning of the track (11).

Each track is subdivided to 26 sectors each capable of saving 128 bytes of data. The first sector is numbered 01, and is physically the first sector after the physical index mark. The logical sequence of the remaining sectors may be non-sequential physically. The location of these is determined at initialization by CPU formatting software and no problems will occur if the controller or CPU is changed since a specific search for that sector is made. Each sector consists of an identification (ID) field and a data field. Normally preambles and postables containing a stream of coded bytes are written at the beginning/end of each sector, to separate fields and to provide data synchronization. The beginning of each field is indicated by 6 bytes of $(00)_H$ followed by a one-byte address mark. With this format 256K bytes of data can be stored to one side of a diskette which contains 77 track subdivided to 26 sectors (30). This amount is doubled with a double sided diskette.

Address marks are unique bit patterns one byte in length which are used in this typical recording format to identify the beginning of ID and Data Fields and to synchronize the deserializing circuitry with the first byte of each field. Address mark bytes are unique from all other data bytes in that certain bit cells do not contain a clock bit (all other data bytes have clock bits in every bit cell). There are four different types of address marks used: Index address mark (Clock D7, Data FC), ID address mark C7 clock, FE data), Data Address Mark (C7 Clock, FB Data) and Deleted Data Mark (C7 Clock, F8 Data) to show that a data field has



Track	Post Index	Sector 1	Sector 2	Sector 26	Trailing Gap (Pre-Index) 247 Bytes of Hex FF
Format	73	188	188	188	247 (Varies with speed)

Post Index Field	40 bytes	6 bytes	1	26 bytes
Format	FF	00		FF

Index address mark (FC Data, D7 clock)

Sector	ID Field	Gap FF	Data Field	Gap FF
Format	13	11	137 Bytes	27

ID Field	6	1	1	1	1	1	2
Format	00	FE		00		00	CRC

Cyclic redundancy Check characters

ID Address Mark C7 Clock Pattern

Track Number in Binary

Sector Number in Binary

Data Field	6	1	128	2
Format	00	FB	Data	CRC

Data Address Mark C7 Clock bit pattern

SINGLE DENSITY IBM 3740 FORMAT (11, 19, 20, 22,30,34)

been deleted (11, 19).

Each field on a track is separated from adjacent fields by a number of bytes containing no data bits. These areas are referred to as gaps and are provided to allow the updating of one field without affecting adjacent fields. There are four different types of gaps on each track: The post-index gap defined as the 32 bytes between the Index Address Mark and the ID Address Mark for sector one, the ID gap defined as the seventeen bytes between the ID field and the data field, the Data Gap defined as the Data Field and the next ID field and lastly the Pre-Index Gap defined as the three hundred and twenty bytes between the last Data Field on a track and the Index Address Mark. Except the Post-Index Gap the other gaps may vary in size after the Data Field has been updated. The Pre-Index Gap is 320 bytes nominally; however, due to write frequency tolerances and disk speed tolerances this gap may vary slightly in length in addition to the update of sector 26 (the last one). (11, 19)

Each ID and data field is appended by two Cyclic Redundancy Check (CRC) bytes which is generated from all characters (except the CRC itself) including the data in the address mark but not the clocks. During all read operations, these CRC bits are again generated and compared with the CRC bits that were appended to the field when it was written (8). The CRC bits may be generated by software or hardware, how-

ever, software generation needs too much time since for each byte about 550 machine cycles are needed; this corresponds to a maximum baud rate of 9600 bauds compared with 31,35 Kbauds in the floppy disk (2).

For this reason, fast hardware methods are used. The CRC is in fact, capable of detecting multiple-bit errors, unless a repetitive burst of errors. It is the most favorite method for error checking since it can easily overcome simple coding procedures like parity checks with minimal waste of bits.

The bits in any message may be pictured as being represented by the coefficients (0 or 1) of a polynomial of nth degree if there are n+1 bits in the message block. For the CRC generation a 16th degree generator polynomial G(X) will be used. For the IBM format the CRC-CCITT reverse generator $G(x) = X^{16} + X^{12} + X^5 + 1$ is used. The whole data stream is of degree (129 x 8) - 1 = 1031. However, after the division, the remainder will be of degree 15 so it will contain just 16 bits which forms the CRC. These 3 bytes are appended to the data stream and when read back now being a polynomial of 1031 + 16 = 1047'th degree together with CRC bytes, the remainder will be 0. Assume R(X) is the remainder, Q(X) is the quotient, B(X) is the 1031th degree polynomial then B(X) = G(X)Q(X) + R(X) and B(X) - R(X) = G(X)Q(X) so that the new polynomial B(X) - R(X) is divisible by G(X) exactly (35).

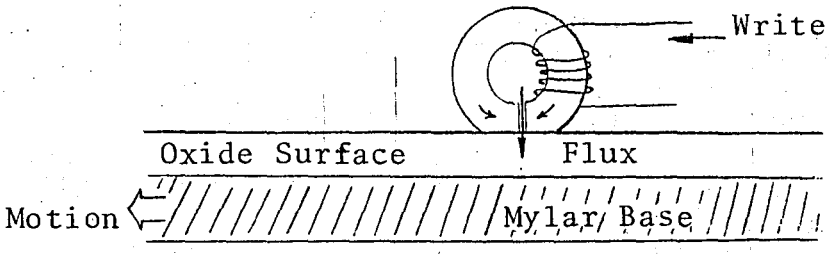
The division is performed as data cycles (hence the name of CRC) with serial 16 shift registers and feedback exclusive OR gates to achieve the substraction of the division. The location of these gates changes with the generator polynomial. This circuitry is generally included in controller IC's and shifted automatically after the data stream.

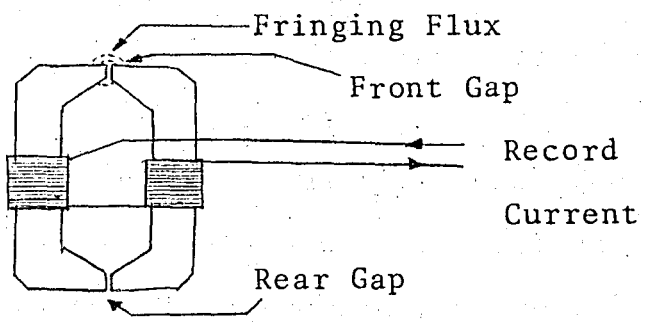
RECORDING THEORY

Magnetic recording which is used in the floppy disk has many advantages such as wide frequency range, wide dynamic range immediate and repeated playback, high density storage and low cost. The magnetic recorder/reproducer consists of four essential elements which are the head, the recording media, the record amplifier and the reproduce amplifier (13).

The recording takes place on a magnetic-oxide coated flexible mylar disk sealed within a plastic envelope for protection, self-cleaning, and ease of handling. The coating materials are particles of the gamma form of ferric oxide, $Fe_2O_3\gamma$. These particles are fixed to the backing material with a synthetic-resin binder. When the magnetic flux penetrates into the coating it reorients the vectors of the magnetic domains. This alignment has a magnitude and direction that is a function of the magnetizing field intensity. To have a high-frequency response a thin coat is needed since at high-frequencies penetration doesn't occur. The thin coat

also provides good signal to noise ratio. Since small particles can be easily oriented this will also cause high-frequency response and good signal to noise ratio. The recording density for a single density diskette varies from 1800 to 3200 bits per inch (1800 for track 00, 3200 for track 76) which corresponds to a recording speed of about 100-200 inch per second.





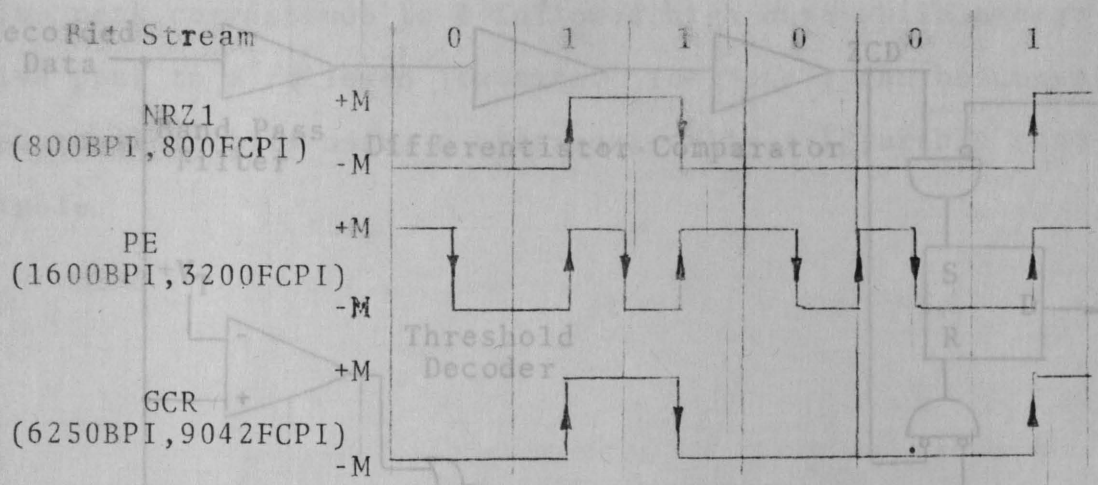
RECORD HEAD

Generally the head consists of two cores and there are windings around both the cores. The rear gap is kept minimum to avoid the fringing flux. Also the front gap should be kept parallel to maintain the frequency response of the head constant. In a record head the gap must be wide enough to permit the magnetic flux to fringe far enough away from the gap to penetrate into the oxide coating of the magnetic tape. On the other hand, during read, the gap must

be small enough so that sharp changes or gradients of the flux may be generated, to permit small changes in data level to be recognized (13).

Fundamentally, the process of magnetic recording is the magnetization of the magnetic particles in the disk. The magnetic field intensity produced is proportional to the head current. The reproduction from magnetic tape is based on the voltage induced in the reproduce head by the magnetic flux lines that emerge from the tape surface and pass through the reproduce Head core. The value of this voltage is based on the rate of change of flux. In this way, some kind of derivative is taken after magnetic recording occurs (13).

Magnetic recording on tapes had first used the "Non return to zero" (NRZ) encoding which was the most popular recording format before the Phase Encoded (PE) format. For NRZ recording, the magnetic surface of the tape is magnetized to saturation in one direction or the other each time a "1" is to be recorded. The magnetization remains unchanged for a stream of "0"s. In the PE format data is represented by transitions occuring in the middle of the data cell. A low to high flux transition is defined as a "1" and the reverse as a "0". Phase transitions are not required when the encoded data consists of 1-0 patterns. The encoding used in the floppy disk is referred as the Group Coded Recording and it is a high density recording scheme which uses the NRZ convention for "1" and "0"s but adds the restriction that

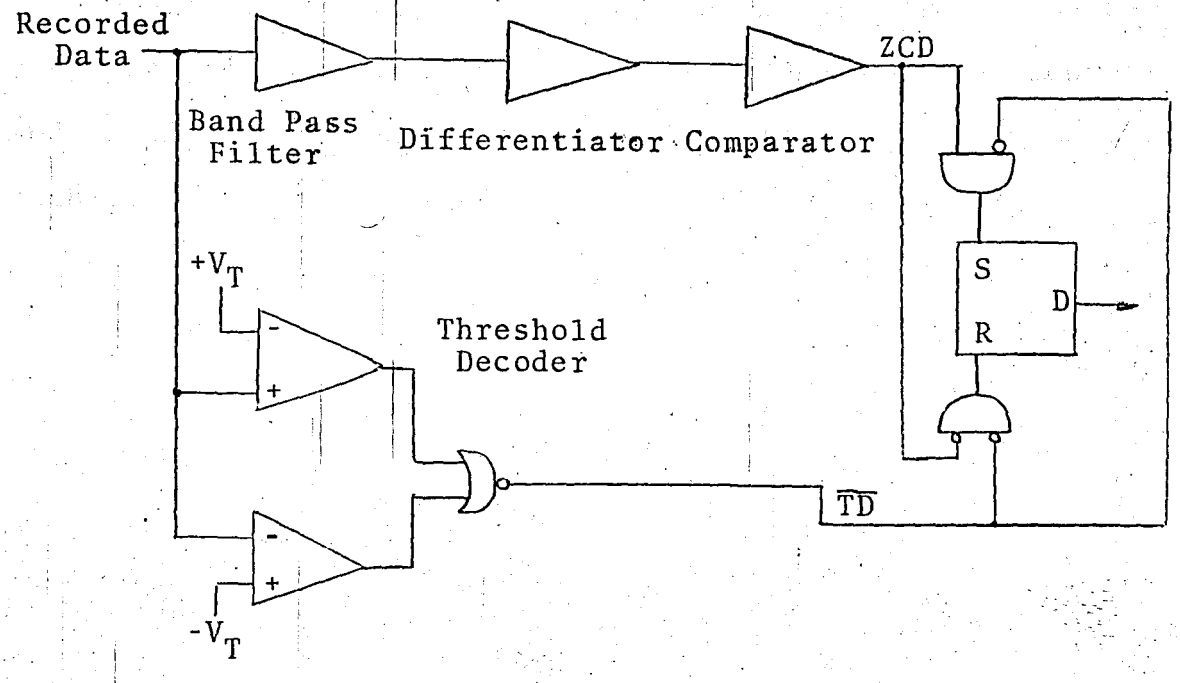


MAGNETIC RECORDING FORMATS

The disadvantage of the NRZ is that a long stream of 0's may occur; causing the read amplifier's low frequency to go to DC so a wide band amplifier is needed. The high frequency cut off will be about the flux changes per inches (FCPI) divided by 2. The disadvantage of the PE format is that FCPI are doubled compared to the GCR and NRZ. On the other hand, for this kind of recording the low frequency cut off is at least proportional to the FCPI/4. The GCR is the most advantageous because of two factors: the low FCPI compared to the PE, and the reduced FCPI/6 low cut-off frequency due to the three bit cell restriction (12).

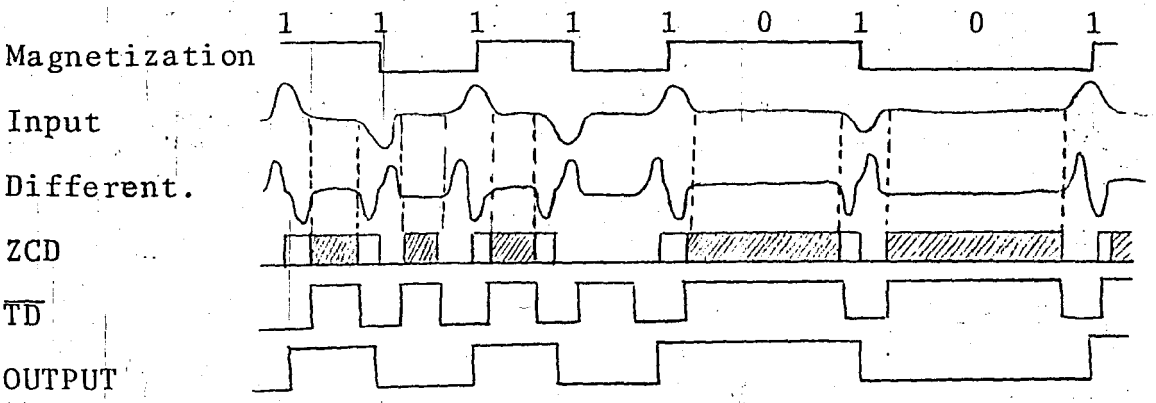
While writing, the incoming data pulses are converted to GCR data using a flip-flop and the recording occurs directly in the form of reversals on the disk. While reading a smooth derivative of the incoming pulses is taken, due to

the differentiating effect of the magnetic read/write process.



READ CIRCUIT BLOCK DIAGRAM

The input is a high-gain linear amplifier (about 100) followed by a linear-phase bandpass filter. The differentiating network provides a 90^{o} delay to convert the incoming read data signal peaks to zero crossings for the comparator.



WAVEFORMS SHOWING DATA DEMODULATION

The function of the threshold detector is to avoid recovery of data at critical points of the input. The comparator in fact detects positive or negative peaks: a positive peak corresponds to a followed high data while a negative peak to a followed low data. The "one"s can be converted to small bits using a bidirectionally triggerable monostable.

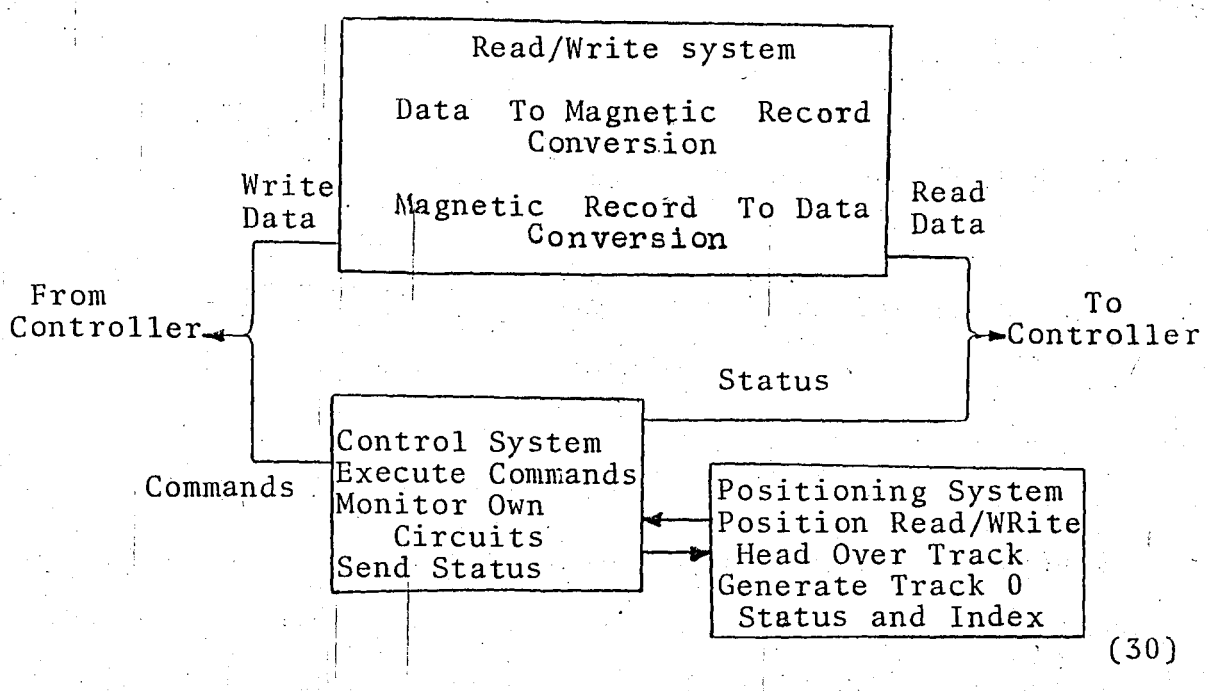
CHAPTER 5

THE DISKETTE DRIVE

GENERAL

The disk drive is comprised of three major assemblies being the printed circuit board (Electronics), the main deck assembly and the carrier assembly. The electronic circuitry required to convert the digital data input and output to, and from analog data for the read/write heads, and head positioning information is contained on one circuit board. The main deck assembly is the principal supporting assembly, and contains the spindle drive motor , drive belt and pulley to rotate the diskette at 360 rpm, the stepping motor, lead screw and head assembly to position the read/write heads, the single-gap magnetic recording heads, the disk cartridge guide and ejector, optical sensing LED's and phototransistors for index, track 00 and write protect sensing. The carrier assembly is the secondary frame which pivots from the main deck assembly and includes the disk centering cone, the head load solenoid together with the head assembly and push button latch release mechanism.

Operationnally the disk drive contains three major systems being the control, positioning and read/write systems.



(30)

DISKETTE DRIVE SIMPLIFIED BLOCK DIAGRAM

The control system provides the interface circuitry between the disk controller and the disk drive. The controller addresses a disk drive for on-line operation by activating a unique select line. This technique allows multiple disk drive units to share common interface lines while remaining individually selectable (daisy-chain operation).

Commands are received by the drive in the form of low level TTL interface signals which designates one of the following operations:

$\overline{\text{SELECT}}$ - Places disk drive on-line with controller by enabling all input/output gates.

$\overline{\text{STEP}}$ - Enables read/write head positioning.

$\overline{\text{STEP IN}}$ - Determines read/write head movement direction.

HEAD LOAD ($\overline{\text{HDLD}}$) - Places disk surface in contact with head.

$\overline{\text{SIDE SELECT}}$ - Selects the disk side for read/write operations.

$\overline{\text{WRITE GATE}}$ ($\overline{\text{WRT}}$)- Enables write current to turn on and inhibits read output.

The $\overline{\text{STEP}}$ signal moves the read/write heads to either a higher or lower track position depending on the $\overline{\text{STEP IN}}$ line. Since relative track positioning is used the controller maintains current track position and generates the number of pulses necessary to achieve a new track position. Once positioned and $\overline{\text{SELECT}}$, $\overline{\text{HDLD}}$ and $\overline{\text{SIDE SELECT}}$ commands outputed by the controller, a read or write command can be initiated.

Five drive status signals are gated to the I/O lines when the disk controller selects a drive:

$\overline{\text{WRITE PROTECT}}$ - indicates write data cannot be recorded on the disk (if write protect disk is used).

$\overline{\text{TRACK 00}}$ - Shows that the head is positioned at track 00. This signal is used to initialize the controller's track address register.

$\overline{\text{INDEX}}$ - Occurs once per disk revolution and shows physical beginning of a track.

$\overline{\text{READY}}$ - Indicates that disk is loaded and up to operating speed.

$\overline{\text{DOUBLE SIDED}}$ - Optional status showing whether the inserted diskette is double or single sided.

To accomplish the positioning of the head in response to $\overline{\text{STEP}}$ pulses from the controller three functions are required: Stepper motor control, stepper motor, and carriage assembly.

The stepper motor control converts serial $\overline{\text{STEP}}$ pulses to three sequential control signals. Each signal energizes one of the three motor windings, causing a 15° rotation which is one track position. The sequentially grounded control windings of the stepper motor causes the rotor shaft to rotate through detent positions at a maximum rate of 333 steps per second. A lead screw on the exposed rotor shaft converts rotary movements to linear movement, to drive the carriage assembly guided by a fixed way to prevent from skewing.

The read/write system records or retrieves data according to the $\overline{\text{WRT}}$ signal from the controller. The read/write head is essentially an electromagnet that can concentrate a high magnetizing force over a very small area of the adjacent recording surface. When recording, the flux field is alternated to magnetize the disk with the desired bit pattern. Each read/write head contains an addtional elecromagnet, the function of which is to erase the edges of the recorded track as data is being written. The width of the track is narrowed to 0.013 inch so that crosstalk between tracks are minimized.

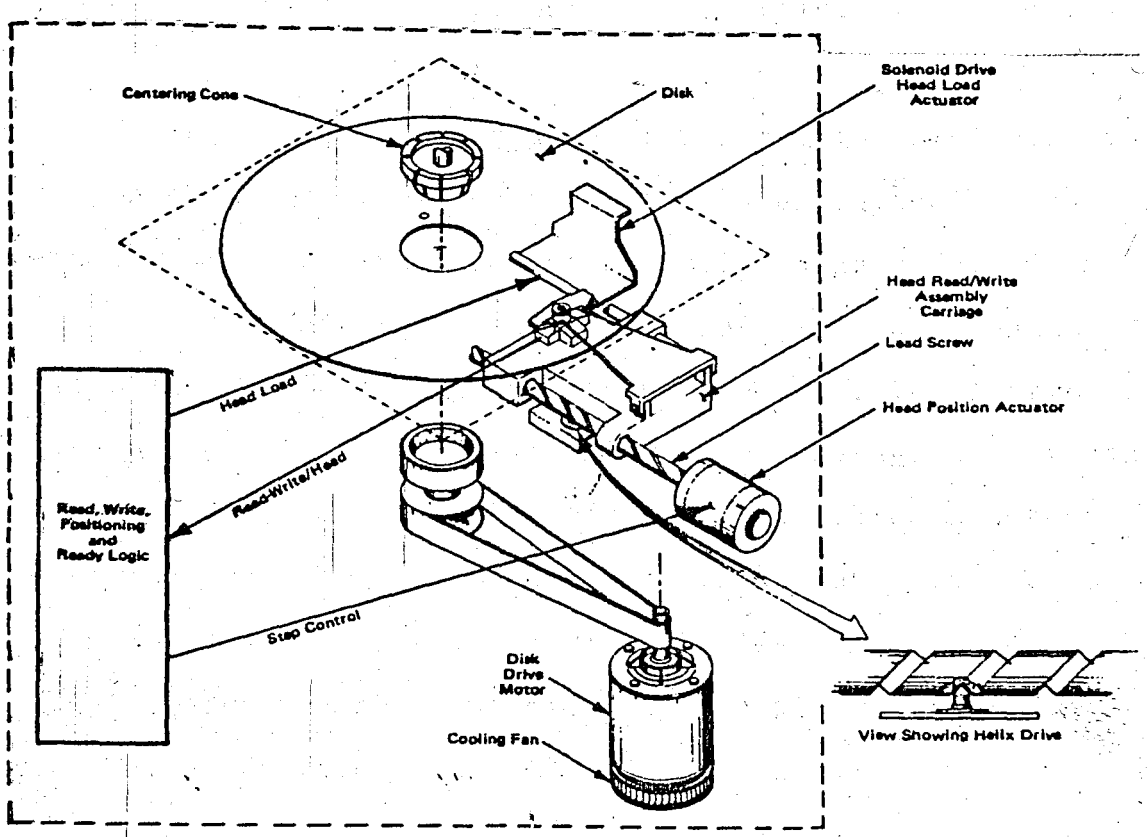
When reading, the read/write electromagnet acts as a sensor. A flux reversal on the recorded track induces a voltage across the electromagnet coils. This voltage is amplified and conditioned to recover the recorded information.

ELECTROMECHANICAL COMPONENTS

The input AC and DC power for the drive are supplied by the controller. The electromechanical elements and their interface to the host are shown on the figure. The drive comprises of the following functional systems:

- Spindle drive mechanism
- Read/Write Head positioning system
- Head load system.

The spindle drive system provides rotational movement of the spindle using a single-phase motor. Rotation of the spindle is provided by a belt and pulley connected to the drive motor rotor shaft. The drive pulley is selectable for either 50 or 60 Hz input power for rotational speed of 360 revolutions per minute. A floppy disk is engaged with the spindle drive hub by the spindle centering cone. The spindle system consists of a spindle mounted on the deck and a centering cone mounted on the carrier. In the unload position, the centering cone carrier is pivoted open creating an apperture through which the floppy disk is inserted. As the carrier is pivoted to the load position, the centering cone



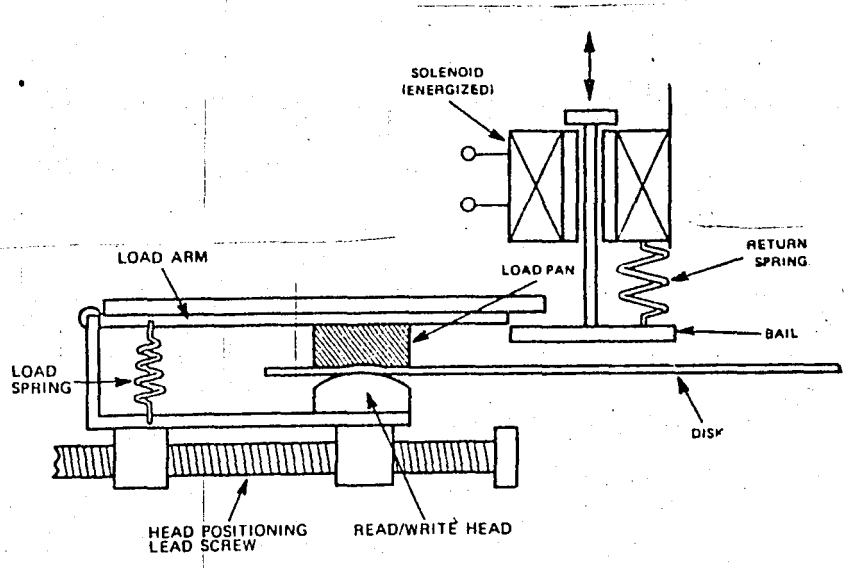
DETAILED FUNCTIONAL DIAGRAM

enters the floppy disk center. Just prior to the fully closed position, the centering cone expander is activated to expand the cone which grips and aligns the diskette to the spindle.

The positioning system comprises a carriage assembly containing a fixed read/write head and a second read/write head on a spring loaded arm, a bidirectional stepper motor and a lead screw. The stepper motor rotational movements are converted to linear motion by driving the lead screw and carriage assembly. When the stepper motor is pulsed,

the lead screw rotates clockwise or counter-clockwise, moving the carriage in or out respectively.

The variable reluctance stepping motor provides precision positioning of the head. It is energized by +24 Volts and operates in either Detent or Positioning mode. The stepper motor has three pairs of windings. In detent current flows in one winding and maintains the rotor in electromagnetic detent. For positioning, the windings are driven sequentially, causing the rotor to rotate through detent positions until the step commands are halted. The rotor then locks in that position with the last winding being driven. The sequence in which the stepper motor windings are pulsed dictates rotational direction (1).



HEAD LOAD AND CARRIAGE ASSEMBLY

The head load system is basically, a solenoid driver and a solenoid. When activated by the $\overline{\text{HDLD}}$ signal, the spring loaded arm is released and brings the recording surface of the floppy disk into conformance with both heads. At the same time a foam pressure is released exerting a force on the disk cartridge against the platen on the deck. The head load command needs a 25 msec execution time.

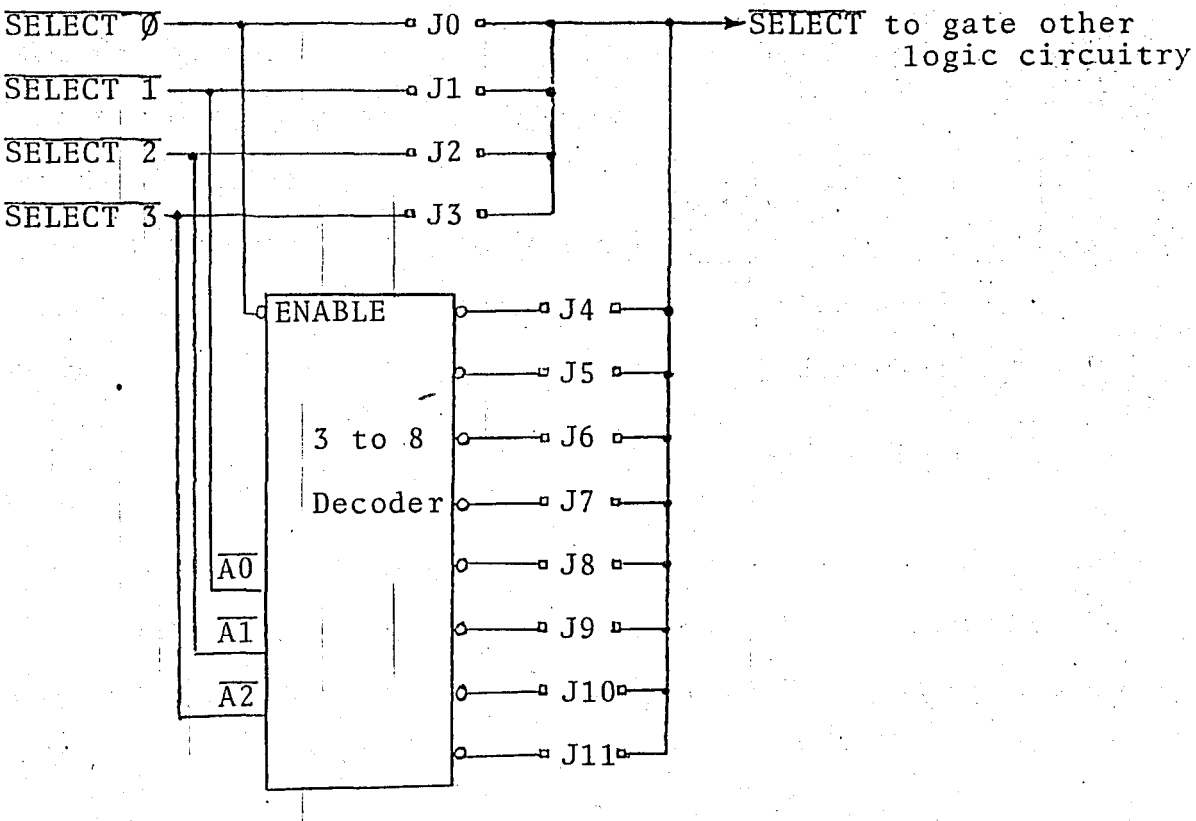
CONTROL LOGIC

The disk drive uses standard 5-volt logic, where a voltage more positive than + 2.4 volts is considered a logical one, and a voltage more negative than + 0.4 volts is considered a logical 0. Interface logic signals are active low and are inverted by line receivers and line drivers for use by the disk drive and the controller, respectively.

The control logic contained in the disk drive performs four prime functions:

- Accepts controller $\overline{\text{SELECT}}$ command and enables all interface logic.
- Detects and provides index pulses.
- Monitors floppy disk rotation to develop a ready status for the controller.
- Reports to the controller the type of floppy disk in use.

The $\overline{\text{SELECT}}$ line is used to activate only one of the drives connected in the daisy-chain fashion. Each drive may have a dedicated $\overline{\text{SELECT}}$ line or a binary selection logic for activation. When $\overline{\text{SELECT}}$is inactive, the select logic inhibits all interface input receivers and output drivers to and from the disk drive. For dedicated selection, jumpers J0, J1, J2 or J3 should be used and only one of them should be strapped for each drive, so that when that dedicated line is low the drive will be selected. With this feature up to four disk drives can be connected.

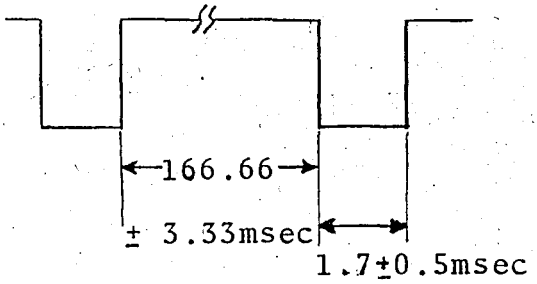


SELECT LOGIC

To address more drives an 3 to 8 line decoder is provided on some drives. In this fashion $\overline{\text{SELECT Ø}}$ line is used

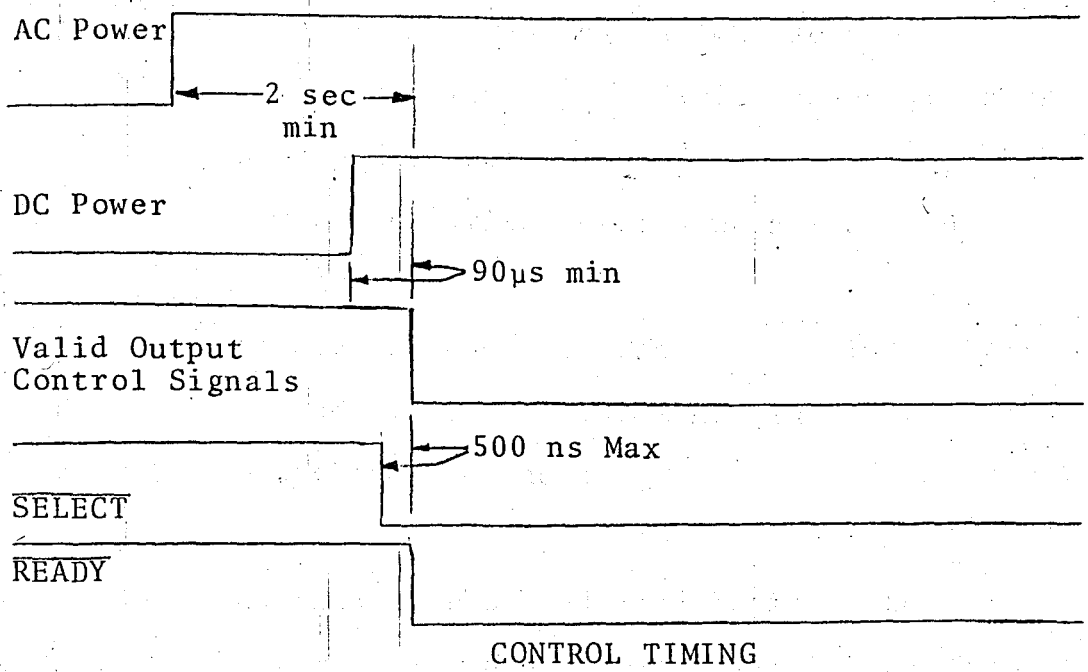
to enable the decoder and in this way only one of the outputs will be low. The decoder accepts a binary coded address on the other three select lines. To select a drive where J10 is installed, for example, the 0010 pattern should be given, $\overline{\text{SELECT } \emptyset}$ being the least significant bit and the decoder accepting inverted addresses.

The index detection logic is comprised of a LED, a phototransistor, a comparator and an output line driver. Since index holes are positioned differently for ouble-sided floppy disks, two LED/phototransistor pairs are included. When the media bocks the LED output, the output from the comparator is low. For each index hole sensed the comparator output goes high for 1.7 msec nominally. This comparator output is then inverted and sent to the controller.



INDEX TIMING

The ready logic is used to monitor the INDEX pulse for the rotational speed of the floppy disk. When the required speed is reached the READY status is sent to the controller after receiving two index pulses. Two seconds are required, however, after the AC power is applied, to send the ready signal.



CONTROL TIMING

The control logic also lights an activity indicator LED, mounted in the door open push button, when the SELECT line is active. This feature may also be used for the door lock option to prevent the operator from opening the door and removing the diskette when the drive is selected.

Another line used by the controller is the TWO-SIDED line which is active whenever a dual sided pack is installed in the disk drive. To generate this signal the two index pulses are used since two index pulses shows that the diskette is double sided.

READ/WRITE HEAD POSITIONING LOGIC

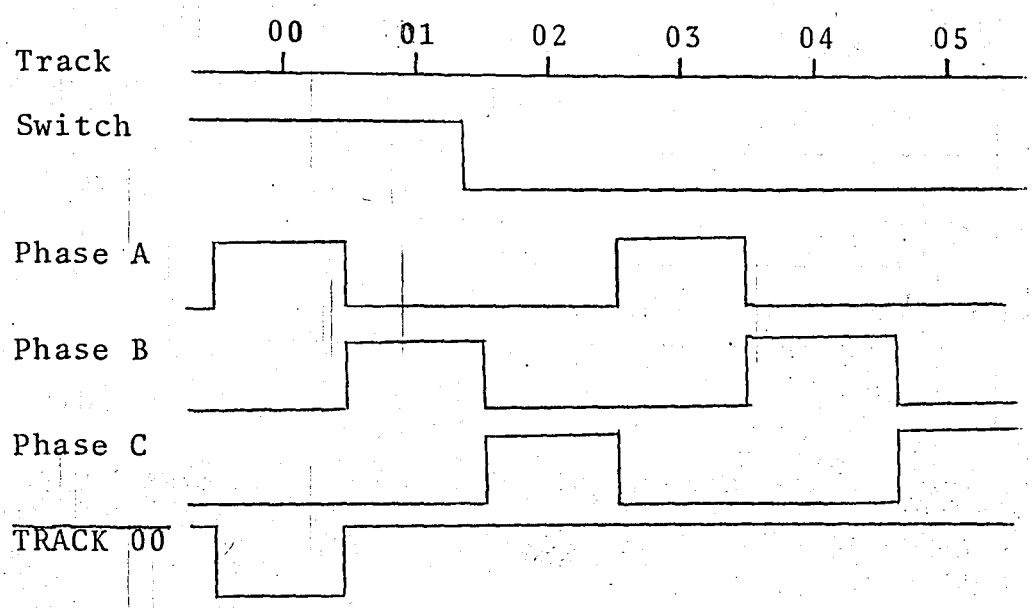
The read/write head positioning logic performs four prime functions:

- Activates the head load/unload solenoid
- Detects position of read/write heads at track 00 and signals controller
- Switches low write current when track 00 is encountered
- Activates stepper motor and determines direction of read/write head movement, in response to controller commands.

The function of the head load logic is to accept the $\overline{\text{HDLD}}$ command from the controller and energize the head load solenoid. The energized solenoid releases the spring loaded head arm and gently forces the media against the heads. As a condition on the loading of heads, the door must be closed. A photoswitch mounted on the deck assembly is activated by a tab on the floppy disk carrier blocking the LED output when the door is closed. This will provide an enable signal for the head load driver. Also, optionally the heads can be loaded with $\overline{\text{SELECT}}$ only, independently of $\overline{\text{HDLD}}$.

The track 00 logic monitors the position of the read/write head by means of a photoswitch and comparator, and signals the controller when the head is at track 00. The photoswitch is mounted on the deck assembly and is activated by a tab on the carriage blocking the LED output. However this is not sufficient since the switch may not be sensitive enough to strictly determine track 00. For this reason the switch output is gated with phase A of the stepper motor

control. The stepper motor is forced to phase A on power on reset so that the initial condition is always the head being on track 00 (this sould be done by the operating system on system reset and initialization).

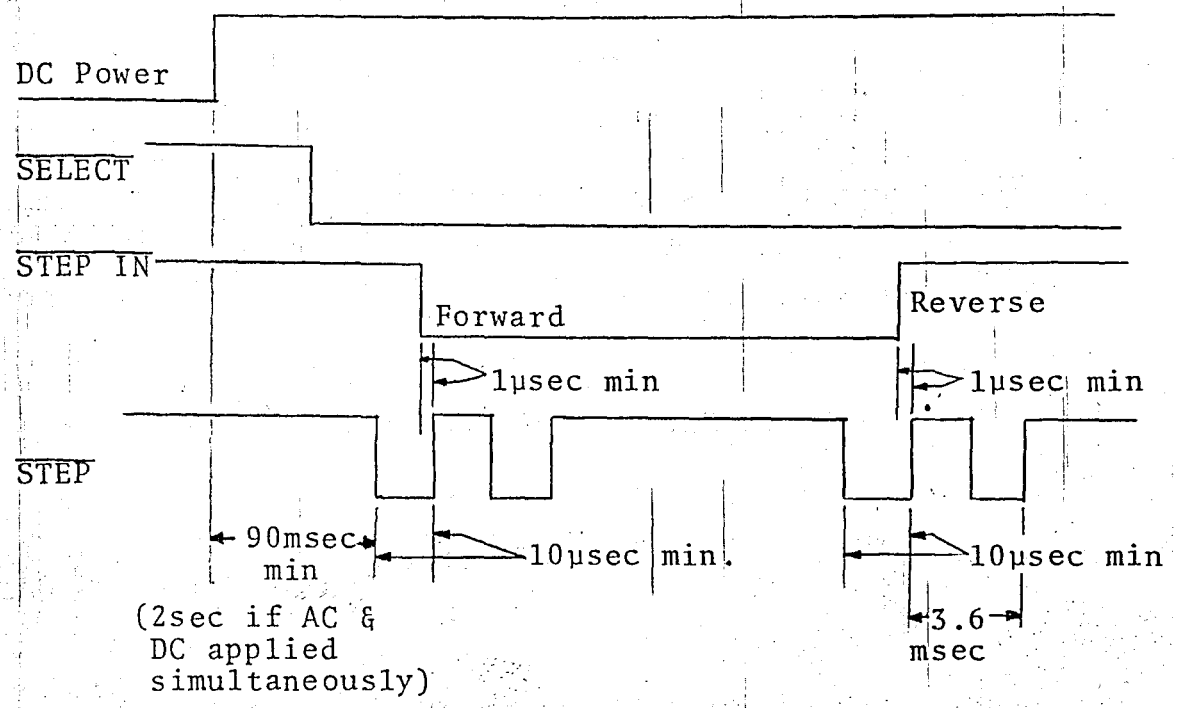


TRACK 00 GENERATION

The drive positioning logic causes the head to move one track distance for each active $\overline{\text{STEP}}$ command, and in a direction determined by the high or low state of the $\overline{\text{STEP IN}}$ command. The stepper motor drive logic comprises interface gates, the state counter which generates active stepper motor phases and the motor drivers.

The 3-state counter is triggered by the trailing edge of the $\overline{\text{STEP}}$ pulses which should have a pulse width of at least 10μsec. The $\overline{\text{STEP IN}}$ command should be issued minimally 1μsec before this trailing edge. The $\overline{\text{STEP IN}}$ lead also determines whether the track counter is in the up or down mode

and whether the state counter is in the forward or reverse mode.

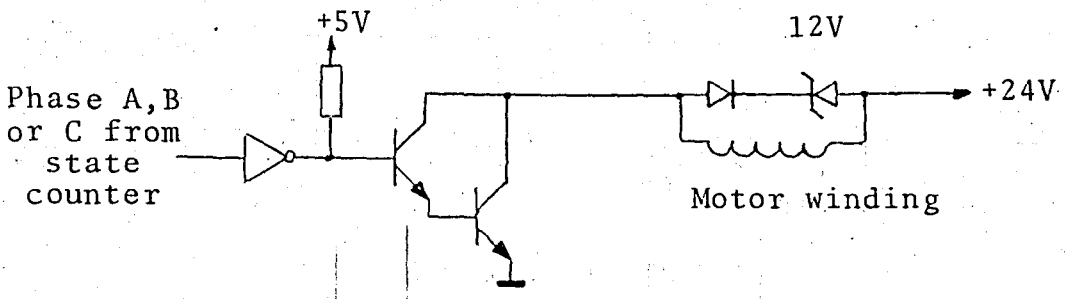


STEPPER MOTOR TIMING

The state counter is in fact a shift register only one position of which can be active concurrently. On power on reset phase A is forced. If the register is in the forward mode it will advance to Phase B when a pulse occurs then to C if another occurs, cycling each time. In the reverse mode the cycling is reversed from C to B to A then to C again.

All three phase motor drivers are identical and independent circuits. It consists of a driver, a darlington transistor pair and a flyback diode. When the driver input is high the transistors are cut off and the winding not energized. A low input on the driver turns the transistors on to energize the motor winding. In order to diminish current

flow to the transistor, due to the back EMF during the off switching, the diode conduction is let to a voltage source of 24 + 12 = 36V while the voltage at the collector is 48V. (7)

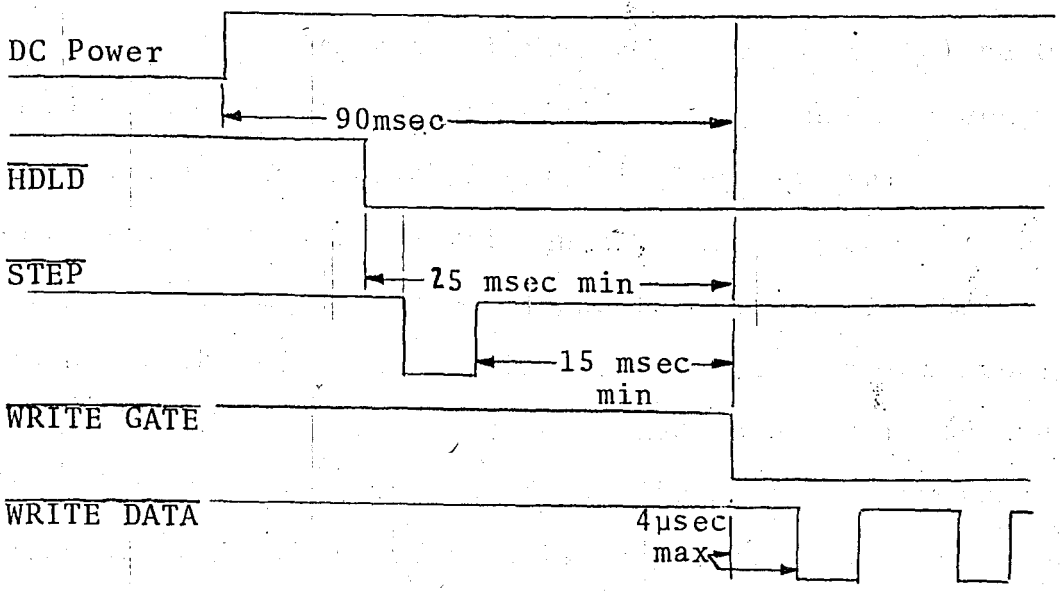


STEPPER MOTOR DRIVER CIRCUIT

READ WRITE LOGIC

The write logic concerts digitally encoded serial data from the controller to analog flux patterns that are magnetically recorded on the disk. The recorded data is sensed and decoded during a read operation and restored to digital read data for the controller (pp 23-28: Recording Theory).

To perform a normal write operation the disk cartridge's write-protect slot should be covered, otherwise any write command is disregarded and all write logic disabled. For this purpose, again a LED/ phototransistor pair is placed on the write-protect slot so that when the slot is covered, that is when a non-write protect cartridge is used, all write logic will be enabled.



WRITE INITIATE TIMING

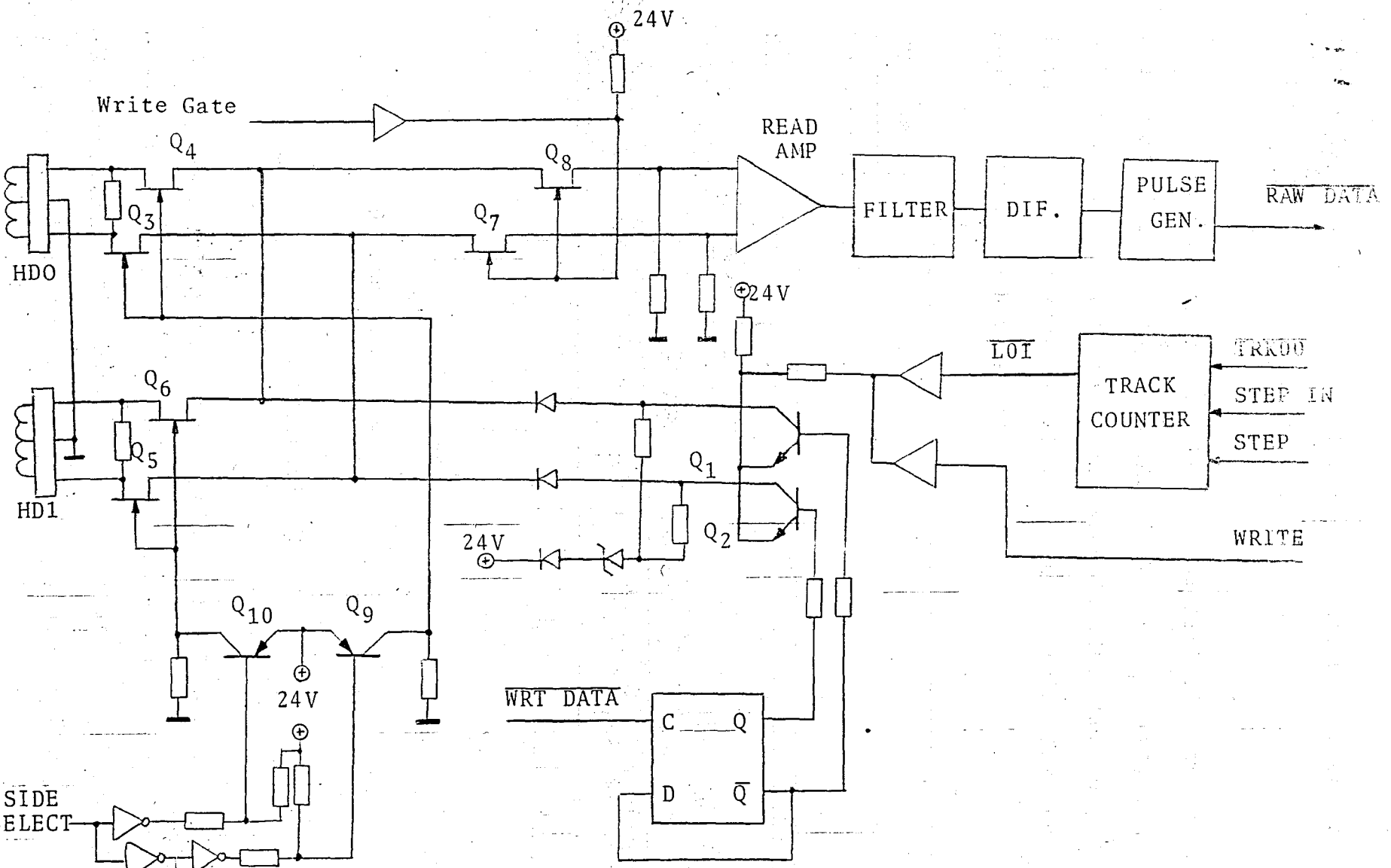
To begin a write operation the head should be loaded and step commands terminated; the contrary will cause the disabling of the operation. The $\overline{\text{WRITE GATE}}$ signals that the data to be written will be outputed in at least one bit time.

At the same time the $\overline{\text{WRITE GATE}}$ activates the erase logic to clip the track width to 0.013 inches, to prevent contamination to neighbor tracks. If an auto-erase head is used a turn-on delay of 200 μsec and a turn-off delay of 530 μsec is given by using one-shots. In an straddle erase head no delays are needed and the $\overline{\text{WRITE GATE}}$ directly controls the erase head current. The $\overline{\text{SIDE SELECT}}$ interface line enables the related side of the head to erase the excess sides of the recorded data.

The write current control logic consists of the write command, the low write current input and the data input. The

write input drives the emitters of transistors Q1 and Q2 when a write command is issued. This input is used in conjunction with the $\overline{\text{LOI}}$ lead which is used to increase the write current when the data is to be written in a track greater than 43, since the diameter decreases and the recording density has to be increased. The $\overline{\text{LOI}}$ is derived from a track counter which is zeroed every time the $\overline{\text{TRK00}}$ is encountered and increments with step pulses if the direction is inwards, or decrements if the direction is outwards. This 8 bit counter is loaded with hex 6A when $\overline{\text{TRK00}}$ occurs and the 6th output is used as the $\overline{\text{LOI}}$ lead since this output is low when the count hex 3F or less is reached (6A-3F = 2B hex = 43 decimal). This output drives Q1 and Q2 additionally with the write input, so the write current is increased.

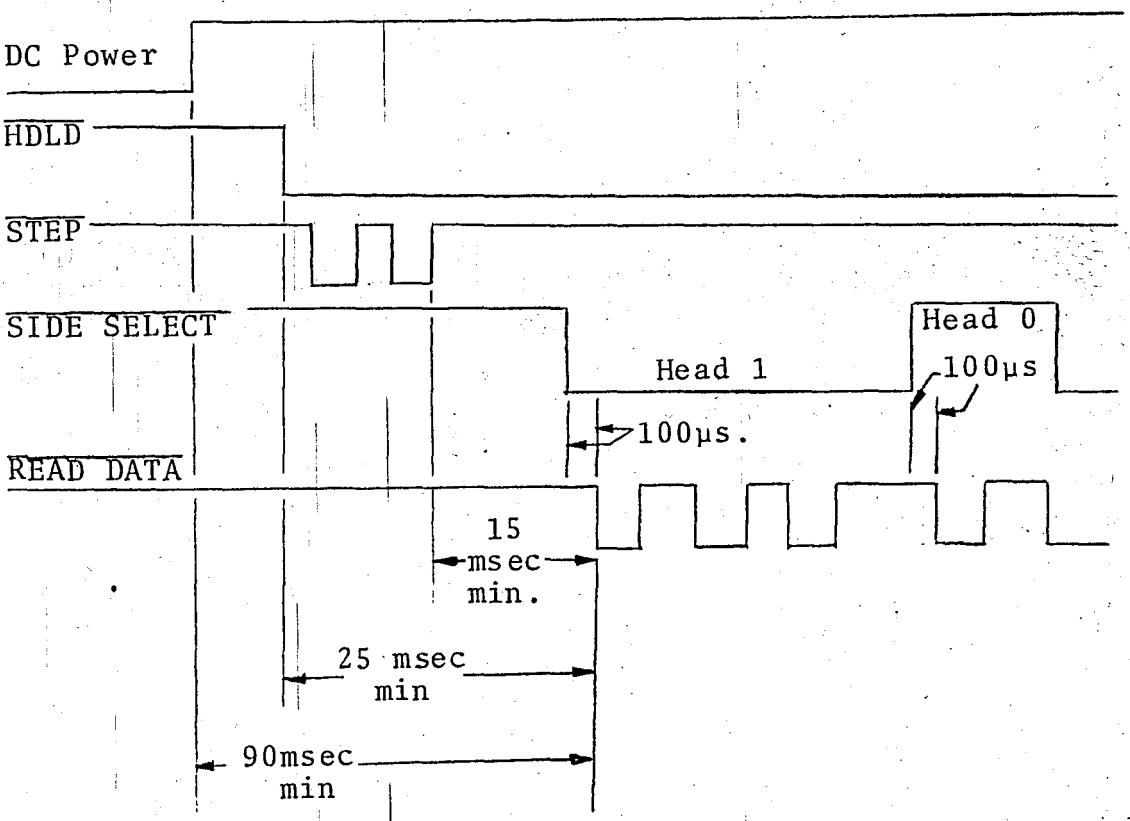
The write data flip-flop functions as a toggle switch for transistors Q1 and Q2. Every time a write pulse arrives from the controller, the flip-flop's Q qnd $\overline{Q}$ output changes state together with transistors Q1 and Q2, so that data is recorded in NRZ form. The FET switching network determines which head is selected and Q1 and Q2 current driver switches will drive the write current through the selected path controlling the magnetization on the diskette. The heads are selected according to the $\overline{\text{SIDE SELECT}}$ input: when the $\overline{\text{SIDE SELECT}}$ is high (which is the case for single sided diskettes) inverter drivers bias Q9 on and Q10 off. In this way, Q5 and Q6 are conducting and Q3 and Q4 non-conducting. With WRIGATE active Q8 qnd Q7 do not conduct so that the write current is gated to head 0 and blocked from the read amplifier. Similarly



READ WRITE LOGIC ELECTRONIC DIAGRAM

head 1 is enabled for flow of write current when side select is low.

For a read operation the read amplifier's way towards the heads will be ready when WRITE GATE is inactive, the only selection will again be the SIDE SELECT. However, this input has to be restricted with a certain timing for the head-amplifier path to settle.



READ INITIATE TIMING

The read amplifier is a gain of 100 linear amplifier cascaded to a 3-pole linear-phase bandpass filter of 800 kHz -3dB bandwidth. The decoding is made using a differentiator, a cross-over detector and a time domain filter (see page 28).

The time domain filter recovers the obtained NRZ data to a series of pulses for the controller using a bidirectional one-shot providing a positive pulse for each transition. The positive edge of each pulse triggers a last one-shot to output nominal 200 nsec pulses. The output is referred to as the $\overline{\text{RAW DATA}}$ since it not only contains data pulses but clock pulses as well. This pulse train should be separated by the controller.

CHAPTER 6

THE FLOPPY DISK CONTROLLER IC

GENERAL DESCRIPTION AND PIN OUTLINE

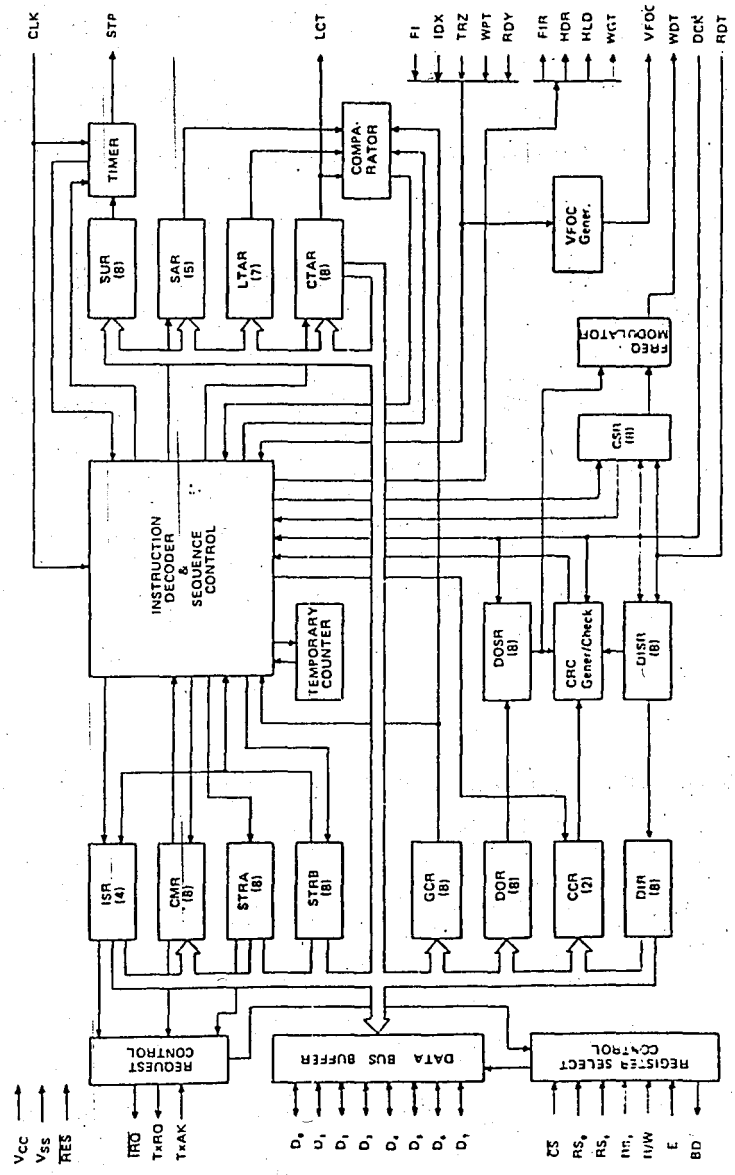
Essentially all one chip floppy disk controllers will interface to most drive manufacturers and are naturally IBM 3740 format compatible. They will provide at least the following:

- Automatic track-seek with verification.
- Read or write with single or multiple records, automatic sector search and entire track read or write.
- Programable controls such as track to track stepping time, head engage and settling time and three phase or step-plus-direction motor control.
- DMA or program transfers.

The differences between controller chips are generally the recording density and the level of the number of disk-drives that one chip will control simultaneously.

The MC6843 FDC is a single-density controller which

is IBM compatible. Data is clocked into the FDC by an external phase-lock loop oscillator. Internal synchronization is handled automatically. A 1 MHz clock is used as a timing signal for the internal functions of the FDC, such as head load and step, as well as shifting data serially to the drive. Status bits are provided to indicate varicus error conditions and drive status.



INTERNAL BLOCK DIAGRAM

The operation is based on the function of 4 main sections:

- The bus interface providing operation with M6800 bus which is comprised of the Data buffers, DMA control, and register select circuitry. The data buffers are usual bi-directional tri-state buffers enabled when FDC transfers data to the bus. The DMA control generally outputs an interrupt request together with a transmit request, and transfers data to or from the drive when acknowledged. The registers are selected using the three select lines, the R/W, $\overline{CS}$ and E pins. The BD (Bus Direction) lead should be used in conjunction with DMA transfers.

- The control and register section is internally connected to provide data flow and generates internal timing and control signals from the macro instructions issued by the MPU. There are twelve user accessible registers for controlling the drive.

- The serializing section handles the serial-to-parallel and parallel-to-serial conversions for Read/Write operations, as well as CRC generation/checking and the generation/detection of the clock pattern. The Data Output Shift Register (DOSR), Data Input Shift Register (DISR), CRC Generator/Checker and Clock Shift Register (CSR comprise the serializing section of the FDC. The Data Clock(DCK) input serves to clock data from the drive into the FDC on both positive and negative edges. The Read Data Input (RDT) is

the serial data stream including both the clock and data bits and must be presynchronized to the Data Clock (DCK). These two signals (DCK and RDT) are generated from the read recovery circuit. The write Data Output (WDT) is the double frequency modulated data from the FDC having a bit time of 4/f (f being the CLK frequency, 1MHz for IBM 3740 single-density format) and a pulse width of 1/f (1μsec for this case).

- The drive interface section, having the known Head Load (HLD), Step (STP), Head Direction (HDR), Low-Current Track (LCT may be generated by the drive itself), Write Gate (WGT) outputs, and the Index (IDX), Ready (RDY), Write Protect (WPT) inputs. All signals are active high and should be inverted and buffered for the drive interface, (see Chapter 5 for nominal timings of I/O signals).

REGISTER DEFINITIONS AND OPERATIONS

Register Select 0-Register Select 2 (RS0-RS2) Inputs, in conjunction with the R/$\overline{W}$ and $\overline{CS}$ inputs are used to select one of the user accessible registers in the FDC. There is an exception that when TxAK input is active the DOR and DIR can be selected without the activation of $\overline{CS}$, for DMA transfers.

Data Output Register (DOR): Write only; information contained in DOR is loaded to the DOSR and shifted out on

the WDT line.

TxAK	$\overline{CS}$	RS2	RS1	RS0	R/W	Registers
0	0	0	0	0	0	DOR
					1	DIR
0	0	0	0	0	1/0	CTAR
0	0	0	1	0	0	CMR
					1	ISR
0	0	0	1	1	0	SOR
					1	STRA
0	0	1	0	0	0	SAR
					1	STRB
0	0	1	0	1	1	GCR
0	0	1	1	0	0	CCR
0	0	1	1	1	0	LTAR
1	X	X	X	X	0	DOR
1	X	X	X	X	1	DIR

X: DONT CARE

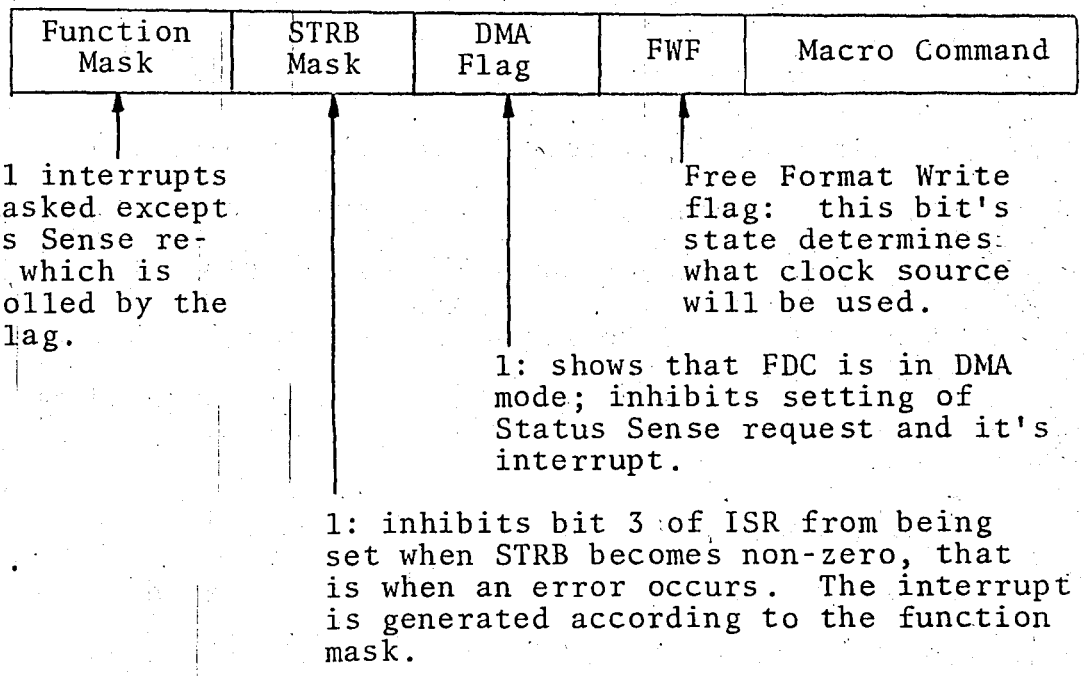
ADDRESS CODES FOR REGISTER SELECTION

Data Input Register (DIR): Read only; RDT is clocked into the DISR and when eight clock pulses have occured the eight bits in the DISR are loaded to the DIR where it can be read by the bus interface.

Current Track Address Register (CTAR): Read/write; Bits Ø-6 are used since the track address may be maximum 76. It shows the address of the track over which the R/W head is

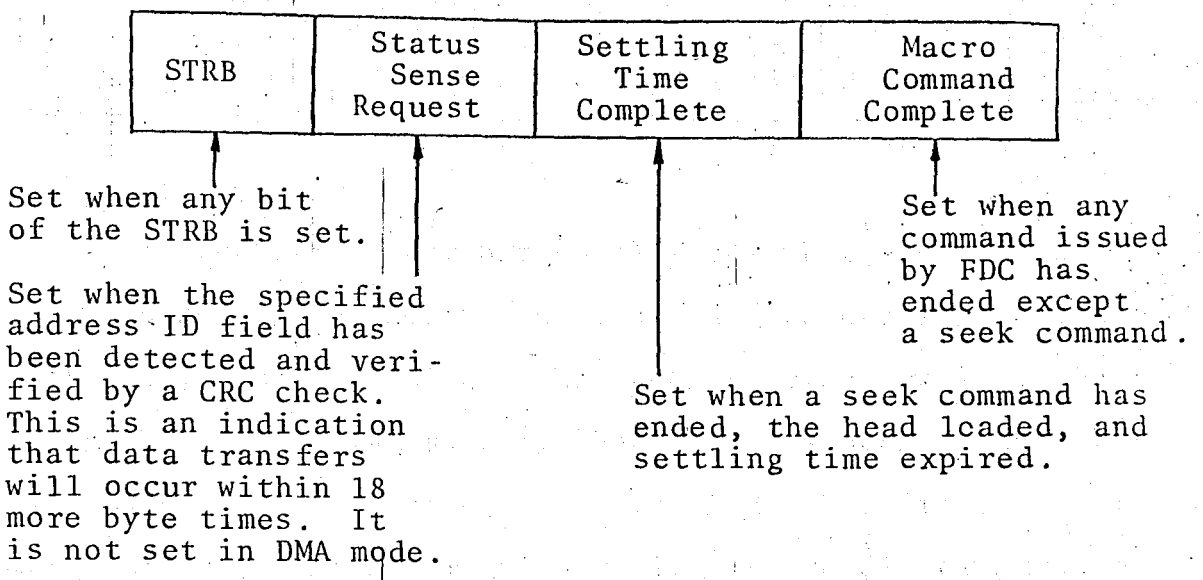
currently positioned. It is cleared when $\overline{TRK00}$ is encountered. CTAR is a read/write register so that the head position can be updated when several drives are connected to one FDC.

Command Register (CMR): Write only; the commands that control the FDC are loaded into the lower four bits of the CMR. Information that controls the data transfer mode and interrupt conditions are loaded into bits 4 through 7.



Interrupt Status Register (ISR): Read only; Bits 4-7 are unused and 0. The register is cleared when $\overline{RESET}$ occurs. ISR0, ISR1, ISR2 are cleared when the ISR is read but ISR3 is cleared only after STRB has been read.

Set-up-Register (SUR): Write only; bits 0-3 determines the head settling time after the head is placed in contact with the disk. The delay is equal to 4096.B/f, so it



varies from 4.096 msec to 61.44 msec for a 1MHz clock. Bits 4-7 determine the frequency of the STP pulses. The STP period equals A.1024/f with step pulses of width 32/f. So STP pulses of 32μsec width with period 1.024 msec to 15.36 msec can be obtained with a 1MHz clock frequency.

Sector Address Register (SAR): Write only; the address of the sector on which the operation will be performed should be written to the SAR. The sector address of the ID field is compared with the SAR. On multiple sector commands, the SAR will be incremented for each sector. Only bit Ø-4 are used.

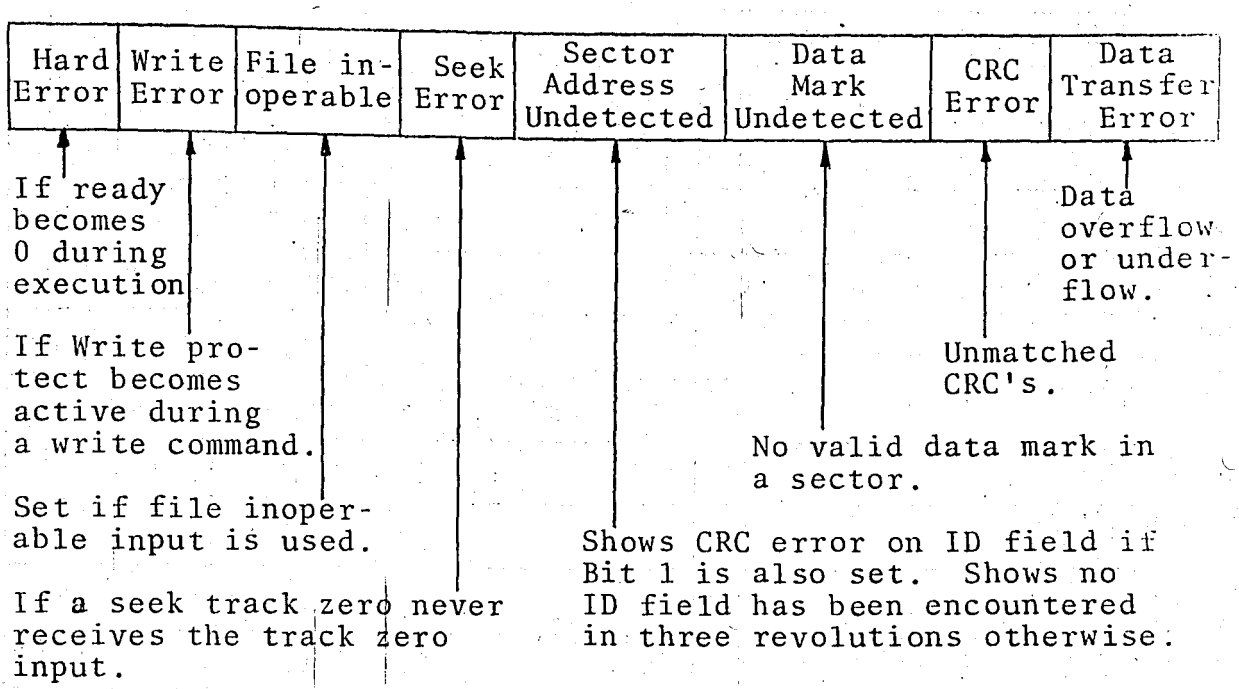
Status Register A (STRA): Read only; this register consists of eight statuses related with the drive and controller.

Busy	Index	Track not Equal	Write Protect	Track Zero	Drive Ready	DDM Detected	Data Transfer Request

The data transfer bit (DTR) shows that the DOR is ready to accept the next data to be shifted out in a write operation, and that the data has been shifted from the DISR to the DIR. The DTR is reset when data has been written to the DOR or it has been read from the DIR. The drive ready, write protect, track zero, index signals are direct active high state inputs from the drive. When a deleted data mark is encountered DDM is set, and when the ID field track address does not match with the logical track address, track not equal bit is set. When busy is a logic 1 the FDC is executing a command and no new commands can be issued. DTR and Busy are the LSB and MSB of the register for easy checking of these statuses.

General Count Register (GCR): Write only; contains the destination track address on a seek command and its contents are transferred to the CTAR at the end of the operation. In a multisector read or write operation the number of sectors to be read or written minus one should be written to the GCR. The MSB is not used.

Status Register B (STRB): Read only; represents possible error conditions that may occur during execution of macro commands.



Logical Track Address: Write only; the address of the track on which the operation is to be performed must be written into the LTAR. The track address in the ID field is compared with the LTAR.

CRC Control Register (CCR): Write only; it is used only in free format commands. Only 2 bits of the register is used: CRC enable starts the calculation of the CRC and stops when all data has expired. Shift CRC is used only in free format write operations and is active when CRC has been calculated for all data. After the data this calculated CRC will be shifted out and CRC calculation will stop.

MACRO COMMAND SET

The command set consists of ten macro commands and each is active when written into CMR Ø-3 bits.

		CMR Bits	Hex Code
STZ	Seek Track Zero	0 0 1 0	2
SEK	Seek	0 0 1 1	3
SSR	Single sector read	0 1 0 0	4
SSW	Single sector write	0 1 0 1	5
RCR	Read CRC	0 1 1 0	6
SWD	Single sector write with DDM	0 1 1 1	7
MSW	Multi sector write	1 1 0 1	D
MSR	Multi sector read	1 1 0 0	C
FFW	Free format write	1 0 1 1	B
FFR	Free format read	1 0 1 0	A

On a STZ command the FDC issues step pulses on the STP output until the TRZ input becomes a high level or until 83 pulses have been sent to the drive. If TRZ is again inactive an error has occured. After the settling time the head is loaded and the busy flag is reset.

On a SEK command the GCR is the destination and CTAR the source address, so that (CTAR-GCR) pulses are issued by the FDC and with a direction depending on the sign of the substraction. At the end of the command busy is reset, the head loaded, and the contents of the GCR are transferred to the CTAR.

FFW has two modes of operation and is generally used for formating diskettes. In the first mode data is directly written from the DOR and all clock bits will be ones; In

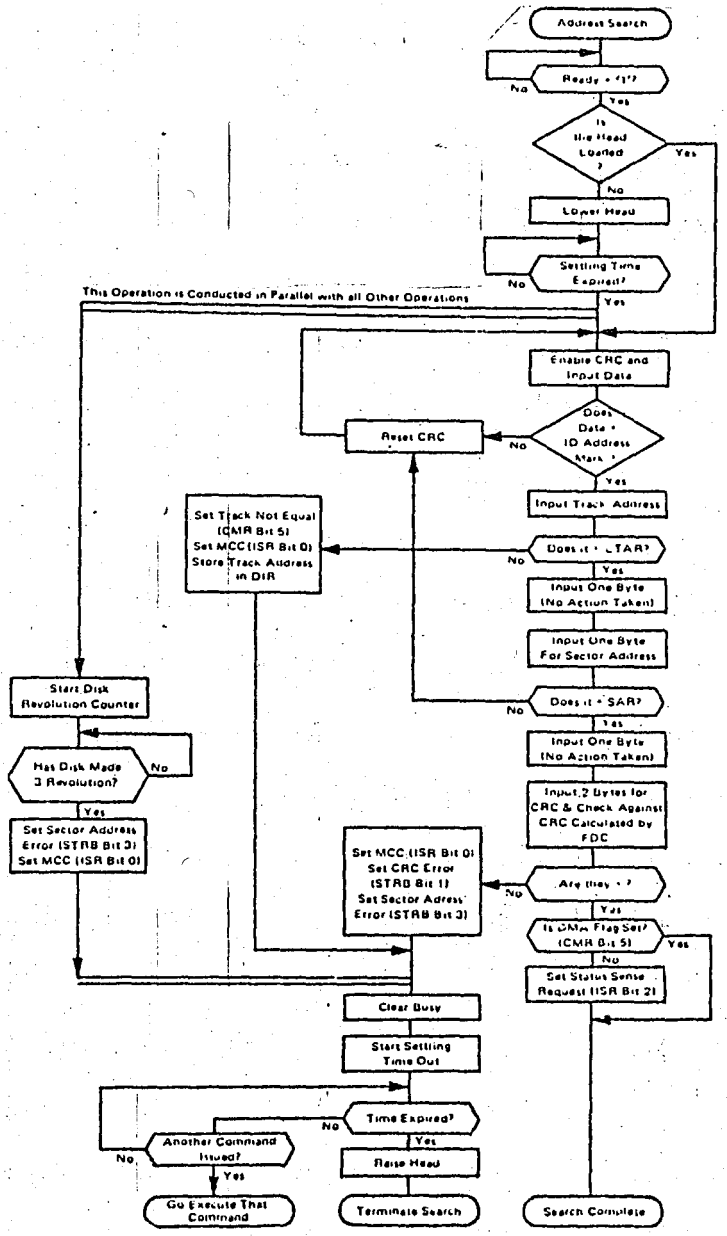
the second mode odd bits of the DOR are clock bits and even one's data bits, so that the clock of the DOSR should be twice a normal write operation. MCC is never set and only the user can stop the command by writing a 0000 into the CMR.

FFR is used to input all data (including ID marks) and should be synchronized with these patterns. This operation as in the FFW is stopped by writing a 0000 pattern into the CMR.

The single sector read/write commands (SSR, RCR, SSW and SWD) are used to read/write data from a single 128 byte sector on the disk. These type of instructions (including MSR/MSW since these are in fact multiple SSR/SSW commands) can be divided into two sections. The first section, which is common to all instructions, is the address search operation, while the second section is unique to the requirements of each instruction.

In an address search the incoming data is compared with the ID address mark until a match is found. This search is, however, restricted to three disk revolutions and when the ID mark is not encountered the sector address error is set.

When the ID mark is found the track and sector addresses are inputted and compared with the LTAR and SAR respectively. An unmatch in the first one sets track not equal



OPERATIONAL FLOW OF THE ADDRESS SEARCH SEQUENCE

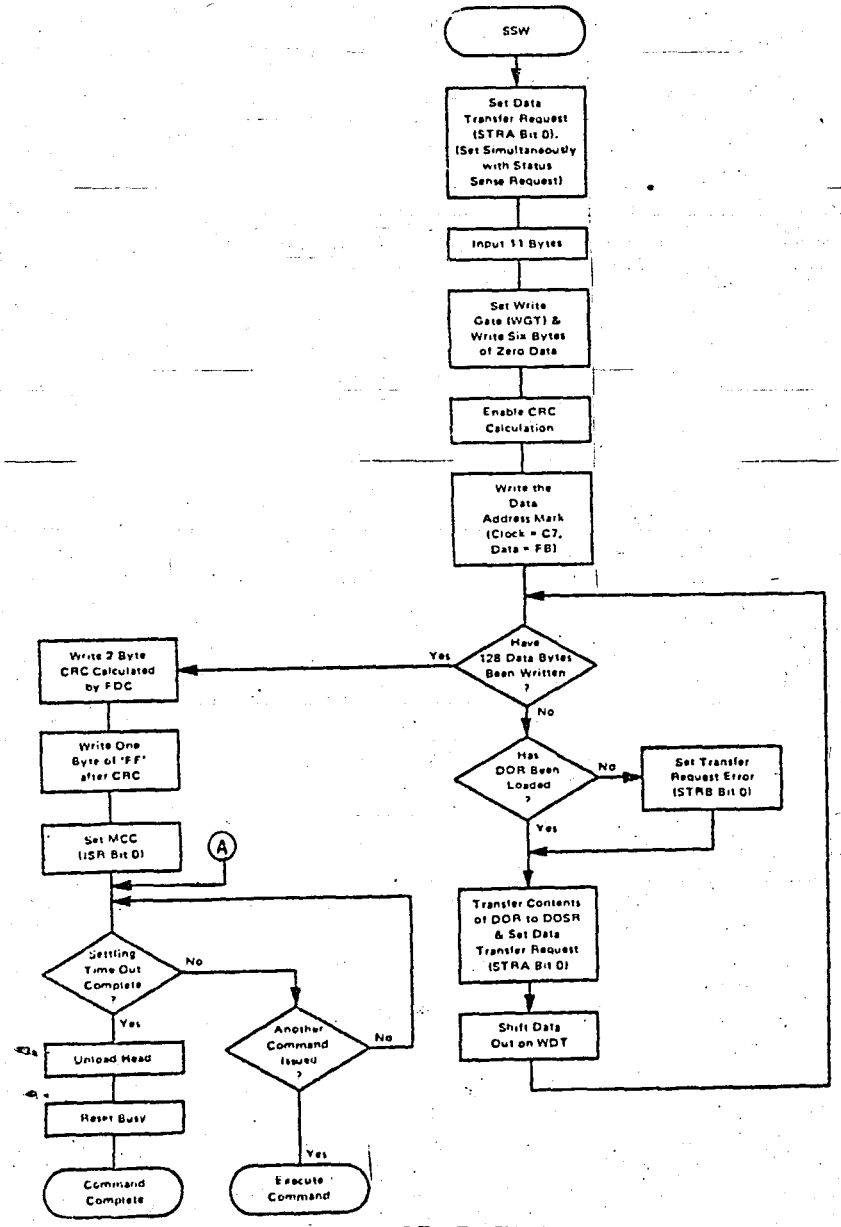
bit and an unmatch in the second one restarts the address search until the wanted sector address is found in three revolutions. When the match occurs the 10 address field CRC is inputted and compared against the CRC calculated by the FDC. An unmatch causes both the CRC error and Sector address

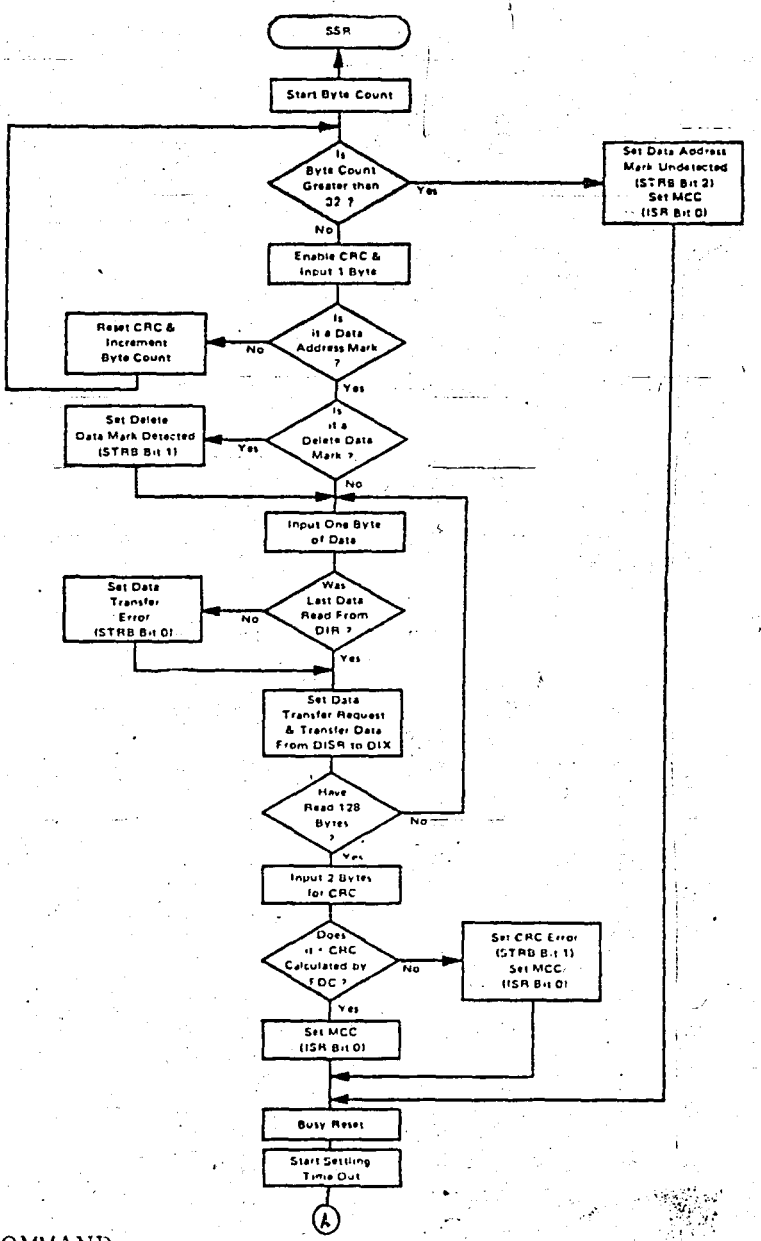
error bits to be set; the contrary causes the start of the operation on the data field.

The SSW is not immediately active after the address search, but 11 bytes of FF inputed and six bytes of zero is written. After this gap, the Data address mark and 128 bytes of data are written, together with the calculated two byte of CRC. Any delay in refilling the DOR while writing will cause the transfer request error to be set.

The SSR waits at most 32 bytes after the address search to find the data mark, but a failure causes the Data address mark undetected bit to be set. Whenever the data mark is detected the CRC is enabled, and the DDM detected bit is set if the data pattern is F8. After 128 bytes of data input the CRC is read and checked against the calculated CRC. If the system is late in reading the DIR the data transfer error will be set.

The MSR and MSW command are used for sequential read or write of two or more sectors. The sector read or write operations are exactly the same as SSR and SSW commands with the difference that the SAR is incremented after each sector command is executed. The detection of a DDM in any sector during a MSR command will cause the DDM bit to be set; when a multi sector instruction is issued the sum of the SAR and GCR should be less than 26. This overflow will show that the head should be moved one track further.





OPERATION FLOW OF THE SSR AND SSW COMMAND

The RCR command is mostly used to verify that correct data was written on a disk. The operation is the same as for the SSR command with the exception that the data transfer request is not set and no data transfer occurs. At the end of the command, if the CRC error bit is not set the operation will be successful.

After the execution of each command the head settling time will be checked and if it has expired the head will be unloaded and the busy flag reset unless another command is issued within this time.

CHAPTER 7

THE CONTROLLER MODULE

GENERAL

The controller module is developped around the 6843 FDC according to it's requirements. The module also incorporates circuitry to enhance the number of drives which can be connected simultaneously. The decoding is designed to fit operating system and disk controller standards of Motorola. 256 bytes of RAM is included on the board which is shared by the monitor, controller and operating system software.

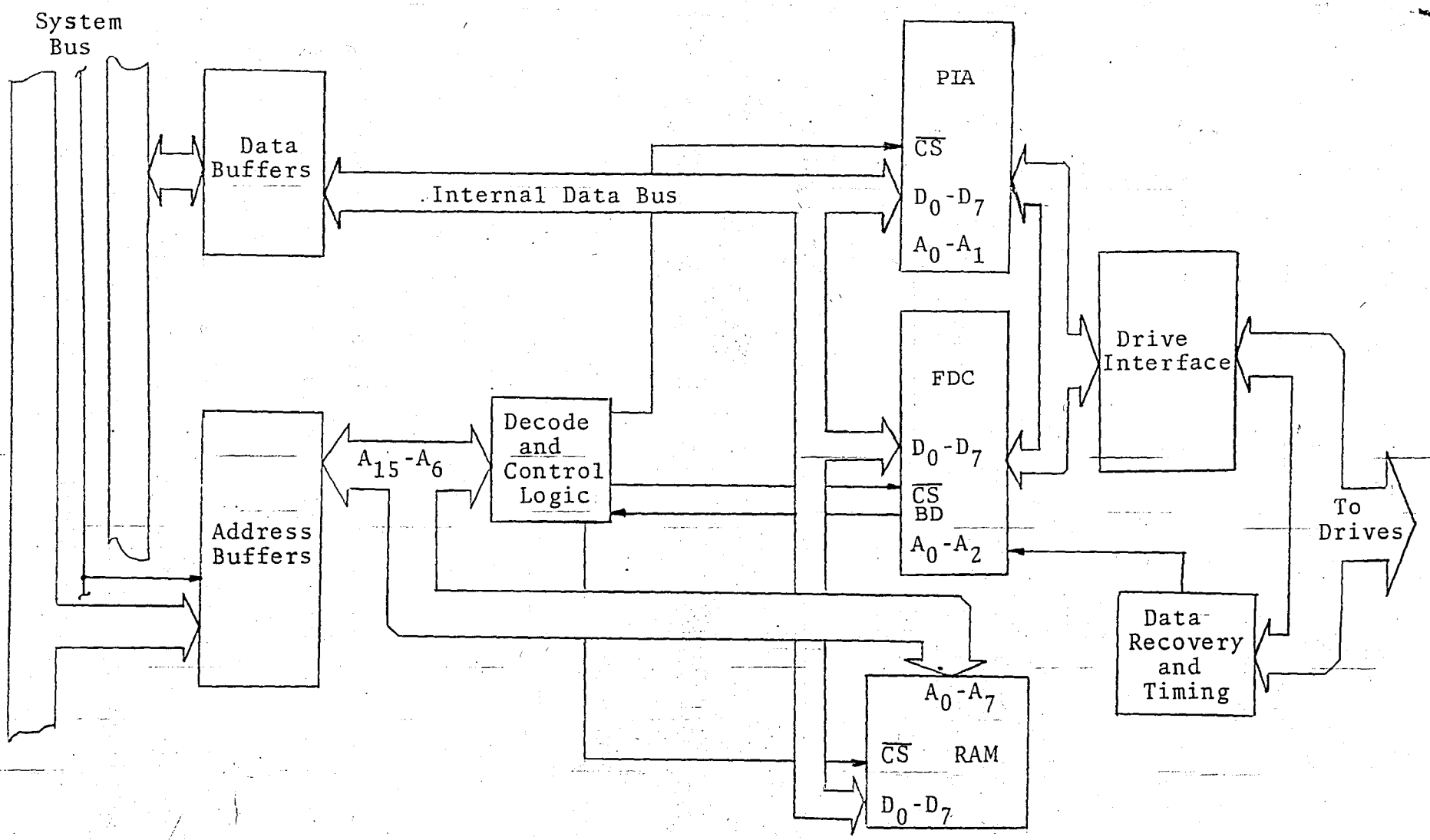
The module is interfaced to the processor bus by bi-directional buffers for the data bus and unidirectional buffers for the address bus. The address lines provide the decoding of the FDC, Parallel Interface Adapter (PIA) and the 256 byte RAM. The control logic is designed to determine data flow direction in normal or DMA applications. The FDC and PIA provide input and output signals to the drive via compatible buffers. An optional 1 MHz crystal oscillator is also incorporated into the module for use whenever the main CPU is running at a frequency different than 1MHz.

The rest of the board is reserved for the data recovery circuit consisting of the phase-lock loop (PLL), voltage controlled oscillator (VCO), reference counters, and synchronizing elements.

THE DECODE AND CONTROL LOGIC

As mentioned before all I/O devices or shared devices such as RAM, ROM should have a location on the memory space in order to access the device from the CPU. All devices occupy different sizes of addressable area according to their function. The devices on the controller module are distributed over an area of about one kilobyte. However the whole area is not occuppied by the devices; the total area is 256 bytes for the RAM, 64 bytes for the FDC and 64 bytes for the PIA. Although the PIA needs 4 bytes and the FDC 8 bytes as memory location they both occupy 64 bytes because of the partial decoding used.

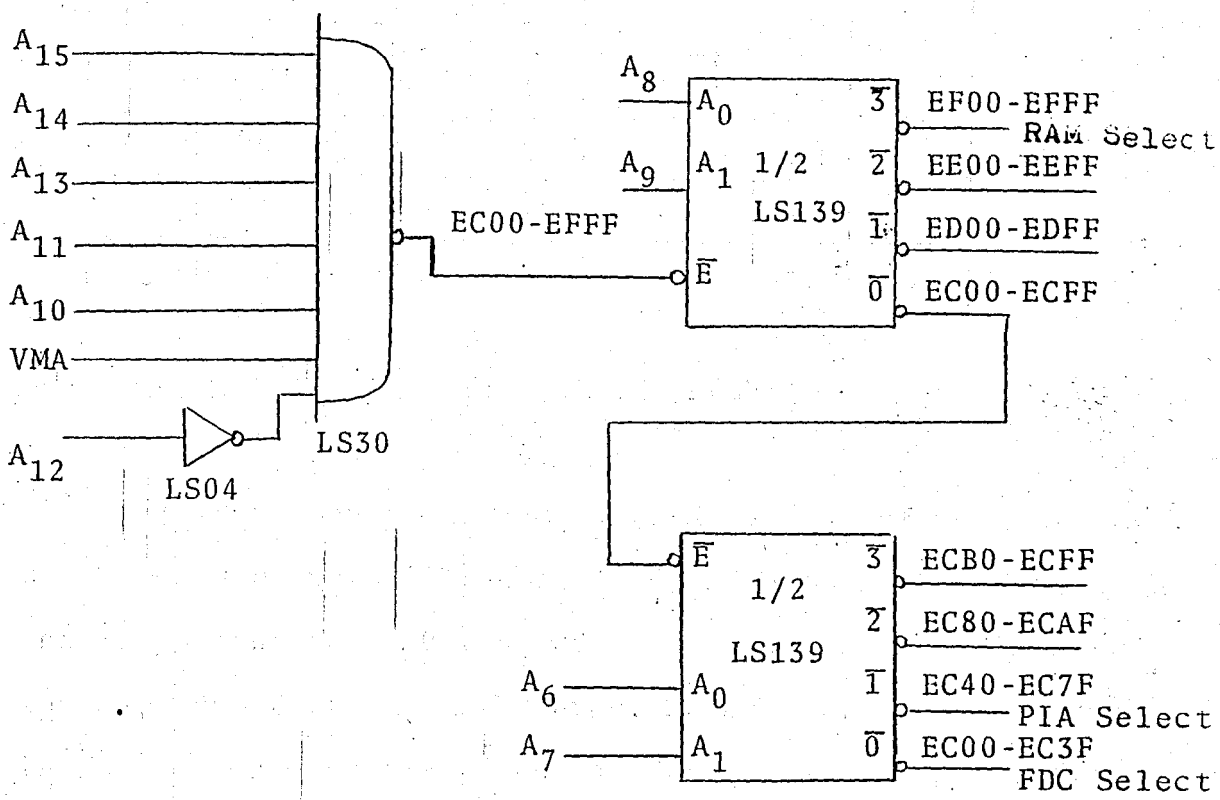
It is always preferable to group devices which have near addresses in the memory map. That is why the 256 byte RAM is incorporated in the FDC module, since it's base address EF00 is very near to the controller I/O devices' base address EC00. As mentioned, the area EC00 to EFFF defines one kilobyte of memory space. The reason to group devices to a minimal area is that the decoding becomes easier and cost effective. On the controller module first the address EC00 to EFFF is decoded.



CONTROLLER MODULE BLOCK DIAGRAM

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1	1	1	0	1	1	X	X	X	X	X	X	X	X	X	X
E				C-F				0-F				0-F			

X: Dont care condition



CONTROLLER MODULE DECODE LOGIC

One kilobyte of memory location is addressed by ten address lines. The left six address lines should be used to enable the decode circuit according to desired address. Assuming all ten lower address lines being low A_{15}, A_{14}, A_{13}, A_{11} and A_{10} should be high and A_{12} low to obtain the address FC00. With the ten address lines high the maximum obtainable address is EFFF. As a result, A_{12} should be inverted and

then AND'ed with the other lines, together with the valid memory address (VMA) line issued by the CPU when addresses had settled. The defined one kilobyte will be divided into four sections using one half LS 139 two-to-four line decoder.

$\overline{E}$	A_0	A_1	$\overline{Y0}$	$\overline{Y1}$	$\overline{Y2}$	$\overline{Y3}$
0	0	0	0	1	1	1
0	1	0	1	0	1	1
0	0	1	1	1	0	1
0	1	1	1	1	1	0
1	X	X	1	1	1	1

LS 139 TRUTH TABLE

If you connect A_8 and A_9 to A_0 and A_1 of the LS139 chip the $\overline{Y0}$ output will be low from EC00 to ECFF. When A_9 and A_8 switches to 10 from 00, the second quarter is selected (ED00-EDFF), the $\overline{Y1}$ output being low. The $\overline{Y3}$ output is low for address area EF00 to EFFF so this output defines the RAM select line. The 256 byte RAM is itself addressed by eight address lines (A_0-A_7) and selected by using all the left eight address lines (A_{15}-A_8). Such decodings, which use all address lines are referred to as full decoding.

The $\overline{Y0}$ output of the LS139 defines 256 bytes after the EC00 base address. However, this should be further divided to define the PIA and FDC. The second half of the LS 139 is well suitable for this job. The $\overline{Y0}$ output is con-

nected to the enable input of the LS 139 together with A_7 and A_6 to form four 64 byte areas. The first quarter which goes from EC00 to EC3F will define the FDC. The lowest three addresses A_0, A_1 and A_2 will be connected to RS0, RS1 and RS2 of the FDC to access the wanted registers. With this configuration, three address lines A_3, A_4 and A_5 are left not connected. So whatever their logic value the FDC will be selected. There is in fact, eight different combinations of A_5, A_4 and A_3 which will select the FDC. Similarly, for the PIA the unconnected address line number is four, so that sixteen different addresses will select the same location of the PIA. Such a decoding is called partial decoding, and with this technique two or more different addresses will define the same physical locations of a device, which are called the images of the device.

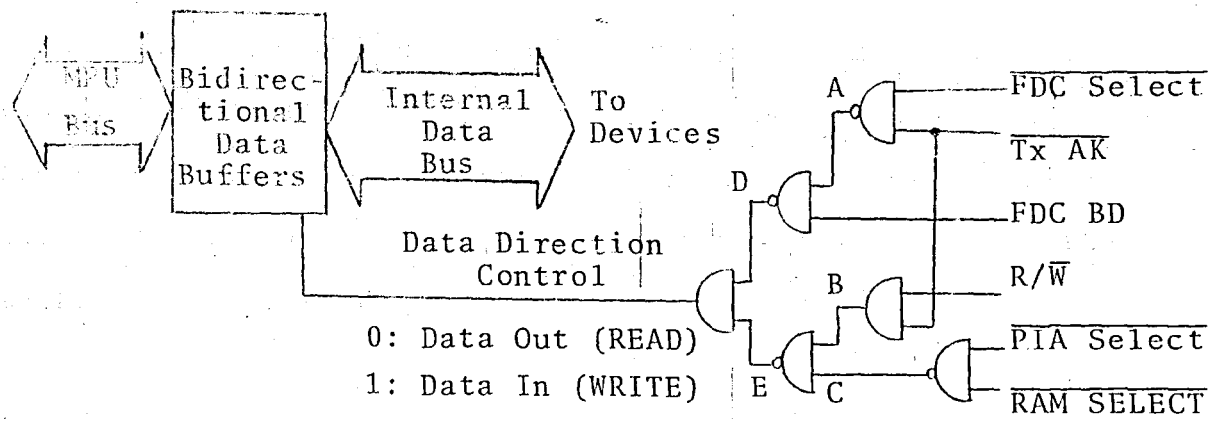
A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
1	1	1	0	1	1	0	0	0	0	X	X	X	RS2	RS1	RS0	FDC
1	1	1	0	1	1	0	0	0	1	X	X	X	X	RS1	RS0	PIA

One of the important considerations in a decoding circuit is the speed, that is the time which will ellapse from the settling of addresses to the selection of the devices. This delay should be calculated by taking the slowest path, in this case for example, the propagation of A_{12} to the selection of the PIA or FDC should be found. From TTL device specifications (25) it is found that the total delay is about 65 nsec (2 x 21 nsec for the LS 139 propagation de-

lay from the enable line, plus 2 x 10 nsec for the LS04 and LS 30 gates). This delay should be kept minimum in the design, since some devices (especially RAM's) add up some additional delays called the access time which may force the cycle time of the processor to increase because of the slow operation of the peripheral. The peripheral devices or modules have to finish their functions in the time between the address settling till the end of the cycle where the data bus should be set ready.

After the selection of the device the data bus is in the position to send or receive data to or from the peripheral module. However, since only one device is selected at a time and because all data lines are parallel, the data lines should be isolated from each other on different modules. 3-state output devices are used to interface a number of components, which is advantageous when they must communicate with each other on a common bus. The concept depends on the ability of all of the interconnected components to be switched to a third state that presents a very high impedence to the bus. In this way, one, and only one component is selected to drive the bus at one time; all unselected components must place their bus drivers in the high impedence state. Simultaneously, one of the other components on the bus is enabled to read the data, thus transferring or transmitting it in one direction; this 3-state switching is controlled by the R/$\overline{W}$ line and the chip-select lines which themselves depend on the VMA and addresses on the address bus. All components of the 6800 family include this 3-state

capability in their data output circuits (10).



$\overline{\text{Tx AK}}$	$\overline{\text{FDC SELECT}}$	BD
0	X	$\overline{\text{R/W}}$
1	1	0
1	0	R/W

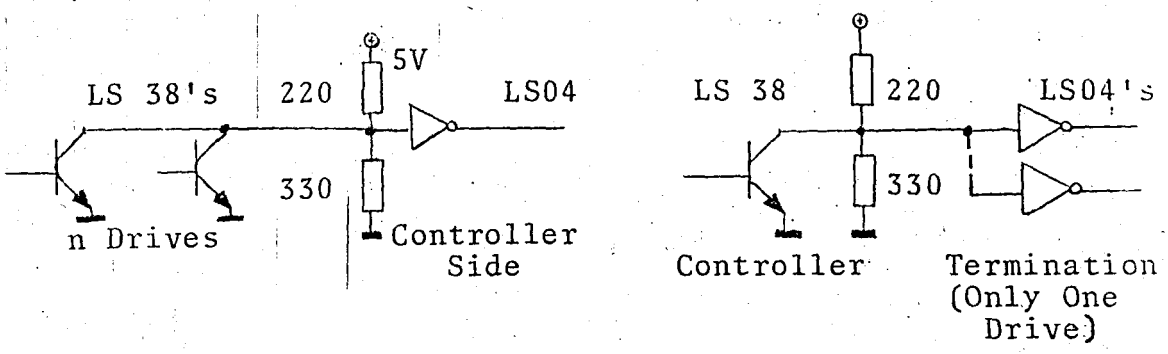
DATA DIRECTION CONTROL LOGIC

In modular operation, when none of the devices on the module are selected, data flow is let inside since every device is in the 3-state mode. This kind of application reduces the hardware, but has the disadvantage that the driving end has to drive all module's buffers. In non-DMA applications $\overline{\text{Tx AK}}$ is high so A and C are low, D and E become high so that the output is high, and data is left inside. Assuming non DMA applications when the PIA or RAM is selected, the control is on the R/$\overline{\text{W}}$ line, C being high and D being high, since the FDC can not be selected at the same time with the RAM or PIA. When the FDC is selected C is low since the RAM and PIA can not be selected, so E is high and the control is on the FDC's BD line, but BD is the R/$\overline{\text{W}}$ line itself on non-DMA applications. On DMA applications, when a transmit acknowledge ($\overline{\text{Tx AK}}$) is given to the FDC, no other

device can be selected since the MPU is halted, so E and A are high and the BD output which is the inverted R/$\overline{W}$ line provides the transfer as if the FDC was the MPU. The total delay after the BD has settled is two gate time, and after the device selection it is three gate time, each gate introducing a delay of 9-10nsec. Also the data buffer switching time should be considered and this is 35 nsec for the LS L242 buffers used.

THE DRIVING INTERFACE

For safety of the internal devices on both the controller and drive electronics, the interconnection should be isolated from both sides. This isolation is done by open collector drivers and compatible receivers. Use of open collector gates is indispensable since paralleling of TTL gates is only possible by this method.



INPUT/OUTPUT

When all LS 38's are in the high position current practically flows through the 220/330Ω termination resistors. Whenever one of the drives is selected, and only one LS 38

is driven to low position, all current flows through the output transistor and the 220 Ω resistor. In this way the LS 38 is driven to deep saturation by 5/220 = 22 mA (max. sink current is 24 mA). For receivers, LS 04 gates are used and terminations are on the controller board. For drivers only the last drive should be terminated with the 220/330 Ω network.

Similarly, for the select lines, which are dedicated lines, the related drive should be terminated with the resistor network according to its own select line. The select lines are specifically buffered by non-inverting 7417 drivers, because of undesired selection of drives during an MPU reset operation. When a reset is given to the system the PIA which controls the drive selection sets all it's output registers to the input mode by pulling the port lines up or putting them into high impedance state. If the select lines were driven by inverters instead of 7417 buffers all drives would have been selected simultaneously which is undesired, especially in daisy chain operation. Also the $\overline{\text{DOUBLE SIDE}}$ line is non-inverting for single-side operation when this line is not connected.

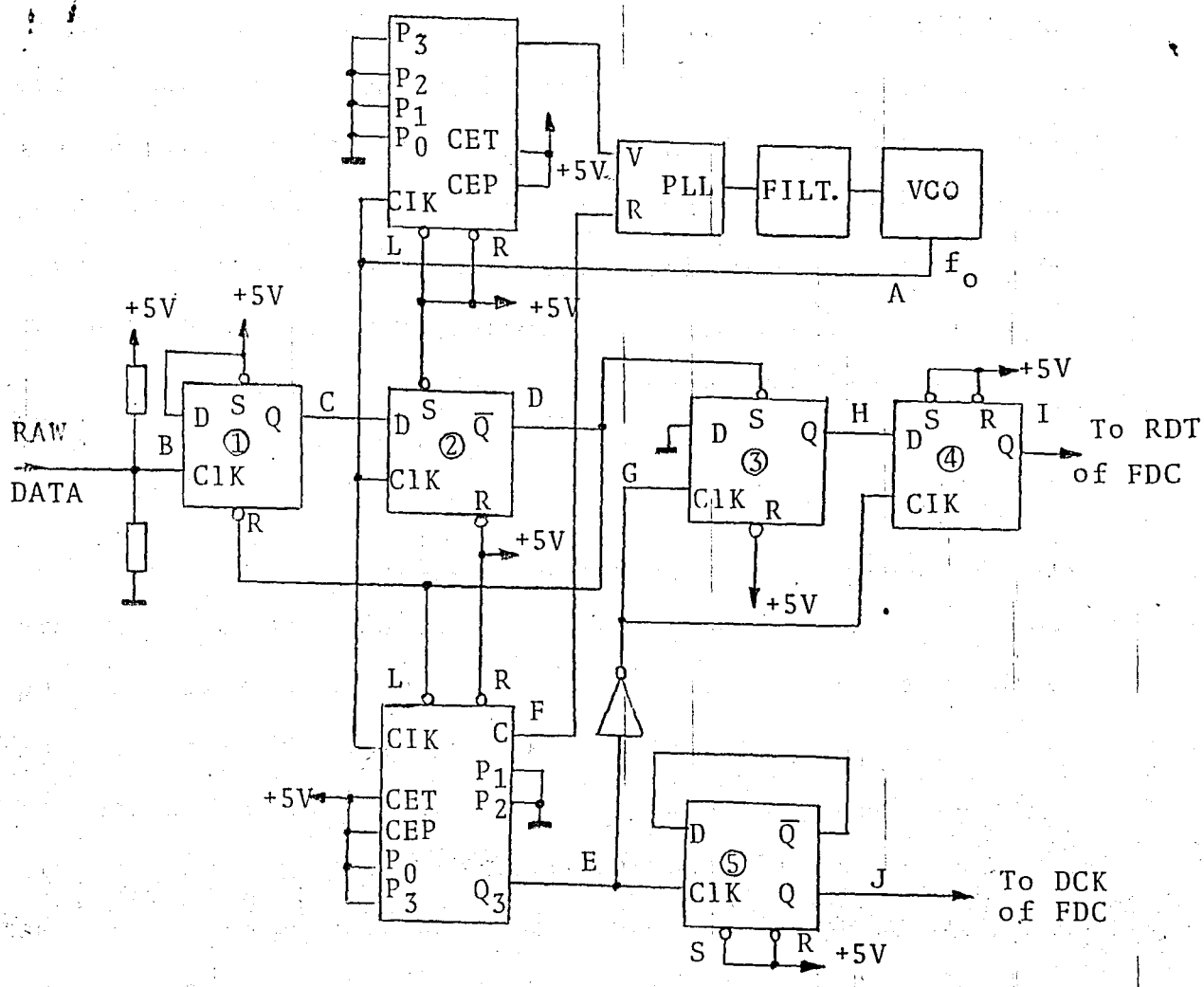
THE DATA RECOVERY CIRCUIT

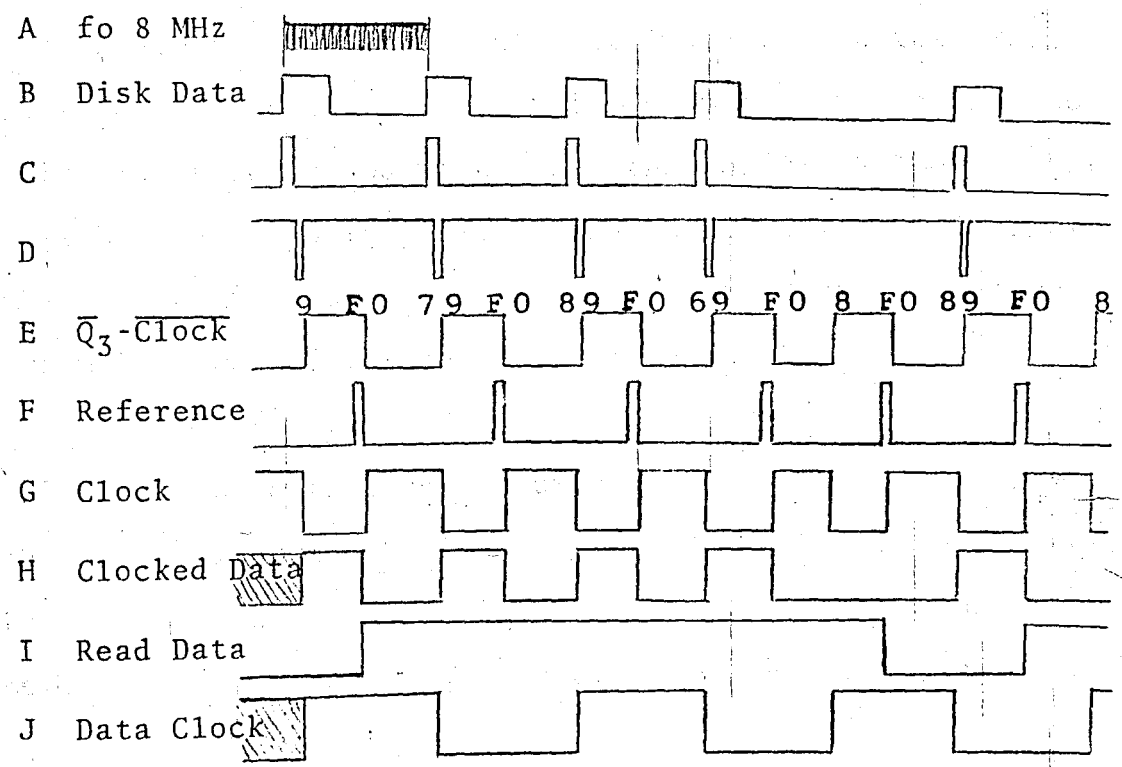
The data recovery circuit is designed to provide mainly three goals: separate and recover all recorded bits (both clock and data) in the form of NRZ data, generate a

continuous clock even when clocks are missing (that is address marks) and lastly track the long term changes in the data rate but not the short term data rate changes. The MC4044/4024 Phase Lock Loop (PLL) is the heart of the data recovery circuit. The analogue Voltage Controlled Oscillator (VCO) is phase and frequency locked to sixteen times the data rate (8 MHz). Other components of the circuit are two synchronous 4-bit counters, being the window counter (since it provides a window for the data even if the data does not exist) and the variable frequency counter (since it divides the VCO frequency), two flip-flops to gate the input data and three other flip-flops to divide or shift the data and clock outputs of the recovery circuit (19).

The raw data signal from the disk system (IBM 3740 format) is a combination of a 500 kHz clock and data signal with some missing bits. This signal is terminated and applied to an input circuitry consisting of two serially connected flip-flops. Flip-flop 1 goes high when a data bit occurs and the first positive going VCO pulse makes the $\overline{Q}$ output of the flip-flop low, thus loading the window counter with 9, resetting the input flip-flop and setting the first data flip-flop (flip-flop 3) to a logic high level. The duration of this $\overline{Q}$ pulse is about 1/8 MHz = 0.125 μsec nominally since in the next VCO pulse the D input of flip-flop 2 will be low.

In the data format used in the disk system, the incoming data stream can have only one consecutive pulse mis-





DATA RECOVERY CIRCUIT AND TIMING

sing. By loading the window counter with a 9, it will produce a positive output transition within 15 VCO pulse periods (1.875 μsec), thus generating a clock edge even if the data pulse is missing. This reference counter will also produce a carryout on the 16th VCO pulse. This pulse is then compared with the carryout output of the variable reference counter by the frequency/phase detector, thus providing a reference for the VCO (26).

The negative output transitions of the window counter are inverted and used to clock flip-flop 3, causing the output to change to a logic low level. If another data pulse is present in the incoming data stream, then the flip-flop is once again set by the input flip-flop circuit. However, if no data is present, then the output will remain low until set by a data pulse. When the clock input to flip-flops 3 and 4 goes high, the output of flip-flop 4 will be in the same state as flip-flop 3 at that instant. In this way, a continuous data state signal will be presented to the FDC which is synchronized with the output of flip-flop 5. Data will be read on both positive and negative transitions of the DCK input. That is why the 500 KHz clock is divided by two using the toggle-connected flip-flop 5.

In the data recovery circuit care should be taken in the PLL, filter and VCO design. The PLL reference frequency (R input of the MC 4044) is the carryout of the window counter. The reference frequency of the system is beat against

the VCO frequency divided by sixteen, which is applied to the variable V input of the MC 4044.

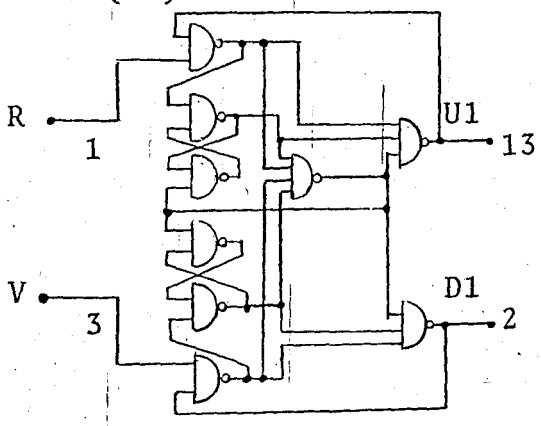
When the VCO frequency is too slow, the window counter carryout occurs before the VCO ÷ 16 carry out. This results in a "pump up" error voltage signal which causes the voltage at the VCO input to rise and the VCO frequency to increase. Likewise when the VCO frequency is too fast a "pump down" error voltage causes the frequency to decrease(19).

The "pump up" and "pump down" error voltages are filtered before being applied to the VCO input. The filter is designed to reduce the effects of the error voltage and gain long term stability. Long term stability is required to enable the system to remain in the lock frequency range when clock bits are missing. When a missing clock is encountered in the serial data stream, the window counter is not preset to 9 so it is not corrected. During this time the window counter acts like a flywheel generating the R input to the MC 4044 from the carryout signal. The PLL system tends to drift upward in frequency. But, because the active filter was designed with a low leakage, the increase in frequency is kept within the lock range. The tradeoff for long term stability is a longer capture time. Capture time refers to the amount of time it takes the PLL to lock to the data rate from an out of lock condition. The design goal for the floppy disk was to insure that the capture time did not exceed six bytes. For the worst case read data rate, six byte times is approximately 180 µs. Six byte is choosen because

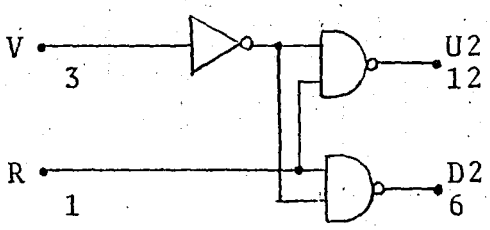
it is the minimum write gate turn on time prior to a data record field (19).

The phase/frequency detector MC 4044 consists of two digital phase detectors, a charge pump and an amplifier. The circuit as a whole generates an error voltage that is proportional to the frequency and/or phase difference of the input signal. The output voltage to input phase difference ratio may be specified as the phase detector transfer function, K_ϕ, of approximately 0.12 volt/radian if appropriate filtering is used.

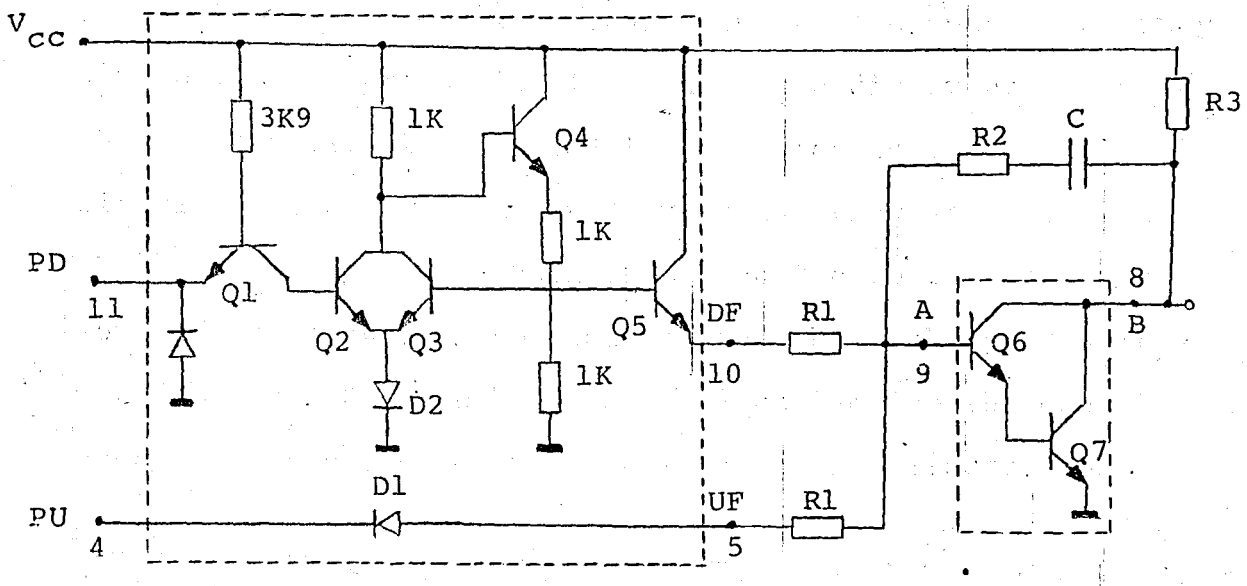
The two phase detectors may be used in different applications. When phase detector #1 is used, loop lockup occurs when both outputs U1 and D1 remain high. This occurs only when all the negative transitions on R, the reference input, and V, the variable or feedback input coincide. The circuit responds only to transitions, hence phase error is independent of input waveform duty cycle or amplitude variation (23).



PHASE DETECTOR # 1

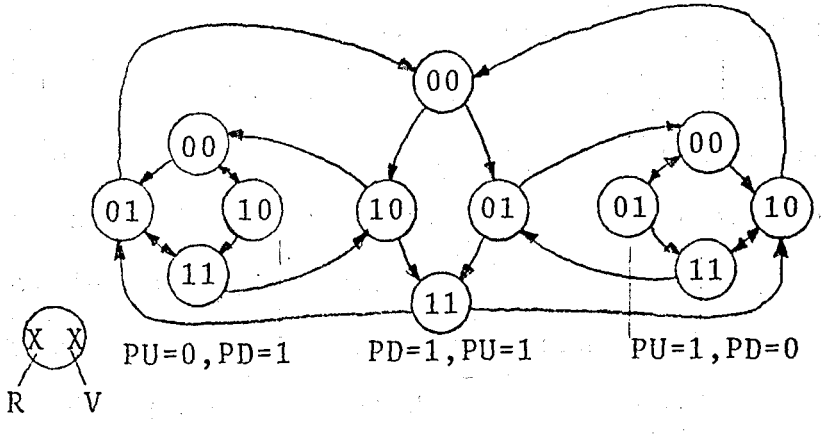


PHASE DETECTOR # 2



CHARGE PUMP, AMPLIFIER AND FILTER

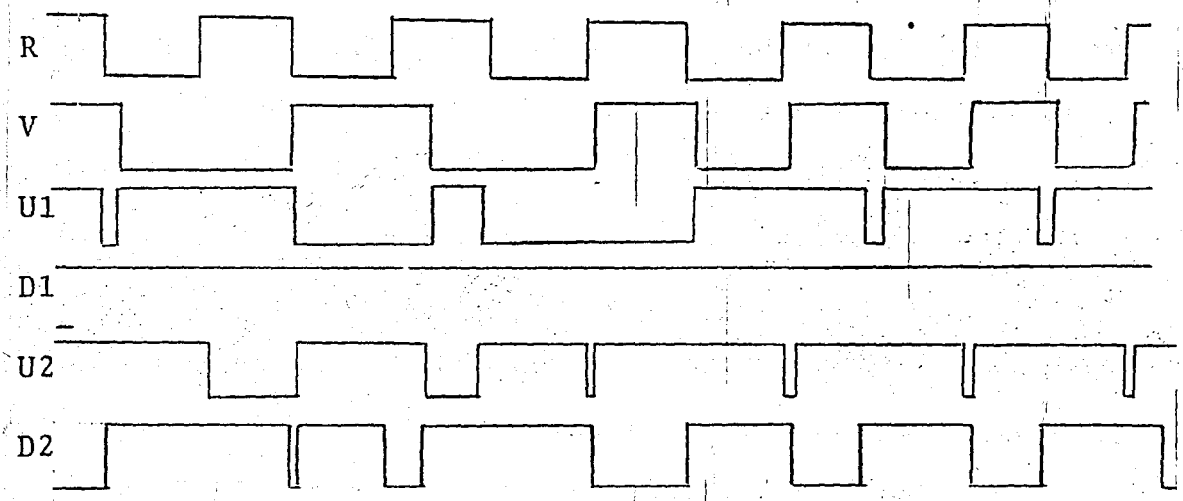
Phase detector #1 consists of sequential logic circuitry, therefore operation prior to lockup is determined by initial conditions. Three output states can be defined: lockup (high impedence: both PU and PD high), pump down and pump up. Considering the present output state, the state diagram shows the next output state, after a change in the P.V. inputs occur (33).



R	V	U2	D2
0	0	1	1
0	1	1	1
1	0	0	1
1	1	1	0

PHASE DETECTORS LOGIC OPERATION

In each case the average value of the pump up or pump down output is proportional to the phase difference between the two inputs. In a closed loop application, the error signal for the VCO is derived by translating and filtering these waveforms. Also both the U and D outputs can not be low and is an undefined output state; the lock up is determined by both U and D being high (inactive for the charge pump). This could be best expalined by an illustration.



PLL TIMING DIAGRAM DURING LOCKUP

Phase detector #2 consists only of combinational logic, therefore its characteristics can be determined from its truth table. Since circuit operation requires that both inputs to the charge pump either be high or have the same duty cycle when lock occurs using this detector leads to a quadrature relationship between R and V. Note that any deviation from a fifty percent duty cycle would appear as a phase

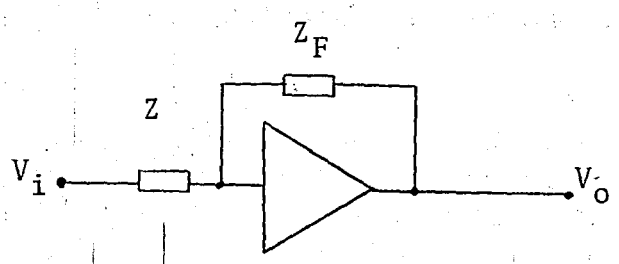
The operation of the charge pump is best explained by considering it in conjunction with the Darlington amplifier included in the package. The three possible states of PD and PU will be analyzed for the charge pump operation. When PD is low and PU is high, D1 will not conduct, so does Q2, since Q1 is in normal operation and does not supply its base current. So Q3 will conduct forcing Q4 to conduct which will cause the supply of current to Q6 by Q5. This will tend to lower the voltage at the collector of Q7, resulting in an error signal that lowers the VCO frequency as required by a "pump down" signal. When PU is low and PD high D1 is forward biased and Q5 is cut off since Q2 is saturated because of the reverse mode operation of Q1, the voltage at the base of Q4 being $V_{CE\ sat} + V_{D2} \approx 0.9$ V. Base current of Q6 is not supplied, the collector voltage of Q7 moves up, resulting in an increase in the VCO operating frequency as required by a "pump up" signal. If both inputs to the charge pump are high, both D1 and Q5 are reverse biased, and there is no tendency for the error voltage to change. The output of the charge pump varies from V_{BE} (if D1 conducts) to $3V_{BE}$ (if Q5 conducts) as the phase difference varies from minus 2Π to 2Π so the gain:

$$K_{\phi} = \frac{2V_{BE}}{4\Pi} = \frac{2 \times 0.75}{4 \times 3.14} = 0.12 \text{ V/rad}$$

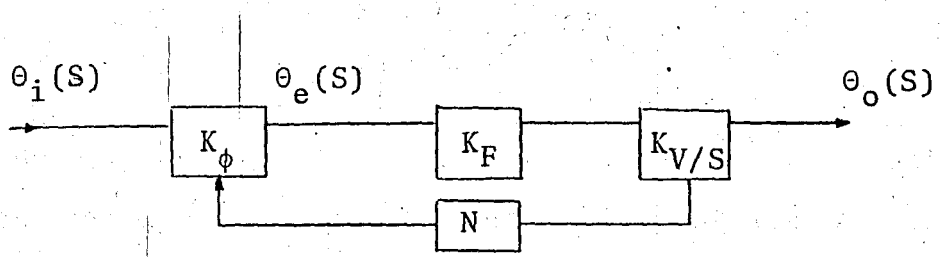
One can also easily find the filter transfer function assuming the amplifier gain being infinite. In this case:

$$K_F = \frac{V_o}{V_i} = \frac{Z_f}{Z} \qquad (a)$$

$$K_F = \frac{1/SC + R_2}{R_1} = \frac{1 + S\,R_2C}{SCR_1} = \frac{1 + ST_1}{ST_2} \qquad (b)$$



The equivalent Z is equal to R_1 since in each case (UF low, DF high or UF high, DF low) current flows through only one of the resistors. T_1 is defined as R_2C and T_2 as R_1C.



Assuming the VCO gain being given in rad/sec/V at the operating frequency, with a value K_V, the whole circuit transfer function can be found.

$$\Theta_o(S) = \frac{K_FK_V}{S}\,\Theta_e(S) \qquad \Theta_i(S) - \frac{\Theta_o(S)}{N} = \frac{\Theta_e(S)}{K_\phi} \qquad (c)$$

$$\frac{\Theta_o(S)}{\Theta_i(S)} = \frac{K_\phi K_V K_F}{S + \frac{K_\phi K_V K_F}{N}} \qquad (d)$$

The system's dynamics when in lock are determined by the amplifier/filter block. Fundamental loop characteristics such as capture range, loop bandwidth, capture time, and transient response are controlled primarily by the loop filter. That is why the loop transfer function should be analized after it is incorporated to the formula.

$$\frac{\Theta_o(S)}{\Theta_i(S)} = \frac{N\ (1 + T_1 S)}{\frac{S^2 N T_2}{K_\phi K_V} + T_1 S + 1} = \frac{N\ (1 + T_1 S)}{\frac{S^2}{\omega_n^2} + \frac{2\xi S}{\omega_n} + 1} \qquad (e)$$

$$\omega_n = \sqrt{\frac{K_\phi K_V}{N\ T_2}} \qquad (f) \qquad \xi = \sqrt{\frac{K_\phi K_V}{N\ T_2}} \left(\frac{T_1}{2}\right) \qquad (g)$$

$$R_1 = \frac{K_\phi K_V}{N \omega_n^2 C} \qquad (h) \qquad R_2 = \frac{2\xi}{\omega_n C} \qquad (j)$$

The system is of second order and the defined ω_n (loop bandwidth or natural frequency) and ξ (damping factor) are particularly important in the transient response to a step input of phase or frequency. ω_n and ξ should be choosen either from a transient basis (time domain response) or steady state frequency plot. Once these two design goals are defined, synthesis of the filter is relatively straight forward. Constants K_V, K_ϕ and N are usually fixed due to other design constraints, leaving T_1 and T_2 as variables to set ω_n and ξ. One can determine the required ω_n for a given ξ from plots of the overshoot and stability as a function of the normalized time $\omega_n t$, where t is the lock-up time (24).

Tolerating the system for an overshoot of 40% by taking ξ to be 0.3, the error is less than 2% of final value for all time greater than $\omega_n t = 14$. (This result arises from the examination of type 2 second order step response curves for different ξ's.) Taking the lock-up time to be six byte time (6 x 32 = 192 μsec) the ω_n of the system is found.

$$\omega_n t = 14, \quad t = 192 \times 10^{-6} \text{ sec} \quad \omega_n = \frac{14}{192 \times 10^{-6}} = 7.3 \times 10^4 \text{ rad/sec}$$

The value of C should be choosen arbitrarely to find the values of R1 and R2. Using (j) and taking C 0.01 μF the value of R2 is found:

$$R_2 = \frac{2x\xi}{\omega_n C} = \frac{2 \times 0.3}{7.3 \times 10^4 \times 1 \times 10^{-8}} = 822\Omega \quad \text{so } 820\Omega \text{ is taken.}$$

To find R1, K_ϕ and K_V should be known. The value of K_ϕ for the 4044 is roughly known to be 0.12 V/rad, but K_V is not known strictly. A value for K_V is assumed and then the VCO is designed and adjusted for the required value. A reasonable value for the VCO would be 12×10^6 rad/s/V. Using (h) and assumed values R1 is found:

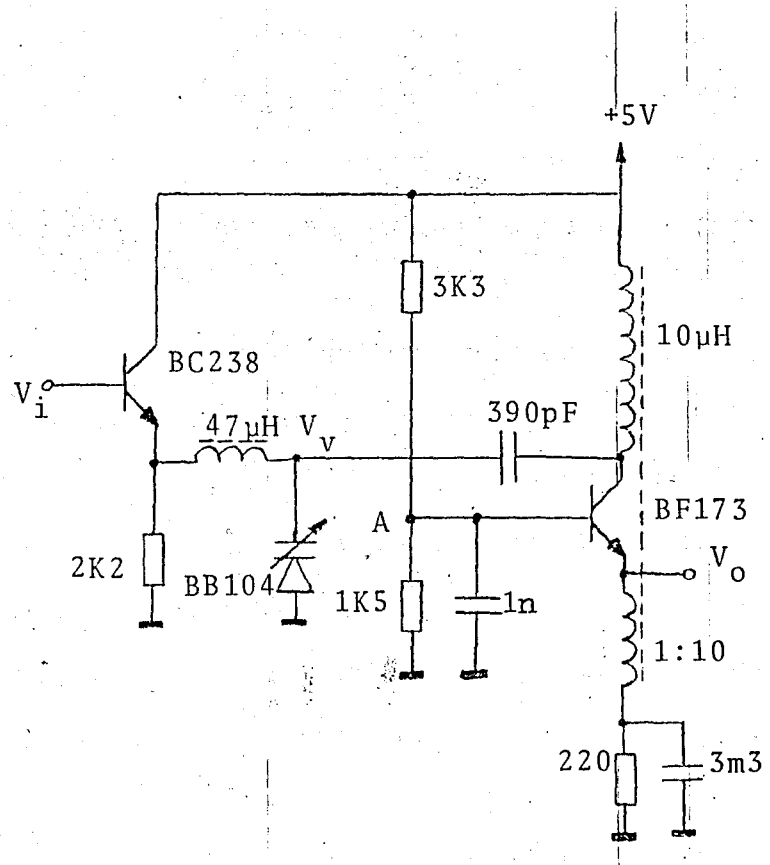
$$R_1 = \frac{K_\phi K_V}{N\omega_n^2 C} = \frac{0.12 \times 12 \times 10^6}{16 \times (7.3 \times 10^4)^2 \times 1 \times 10^{-8}} = 1690 \text{ so 1k8 is taken.}$$

The reason to force the VCO to have a gain of 12×10^6 rad/s/V is that the swing at the VCO input can be at most 1V, because of the maximum V_{BE} change at the charge pump

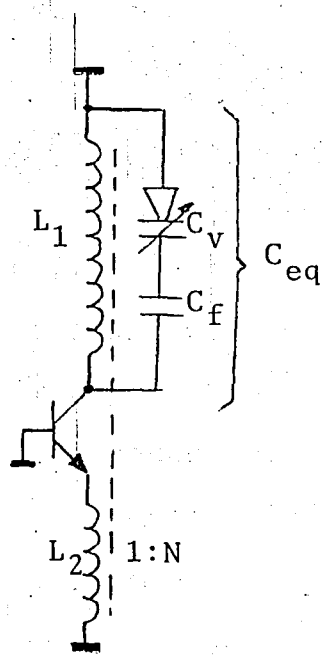
output. A maximum frequency swing of 2 MHz (from 7 to 9 MHz) is required for this application which restricts the VCO gain:

$$K_V = \frac{\Delta_\omega}{\Delta_V} = \frac{2 \times 2 \times \Pi \times 10^6}{1} = 12.5 \times 10^6 \text{ rad/s/V}$$

To realize this required VCO gain a common base oscillator is used, having a varactor diode as the voltage to frequency converter, since the diode is installed in the resonant LC circuit.



VCO CIRCUIT



AC EQUIVALENT OF THE VCO CIRCUIT

The VCO input consists of an emitter follower for isolation from the filter, and the varactor diode is in series with a choke to isolate RF across the varactor from reaching the emitter of the BC238. The voltage on the diode is about one V_{BE} less than V_i because of the emitter follower stage's base emitter drop. The DC analysis of the VCO shows that the quiescent output voltage is about 0.8V. In the DC analysis all coils are considered to be short circuits and capacitors open circuits. The voltage at point A is determined by the resistor ratio and the output quiescent voltage is equal to point A's voltage minus one V_{BE}. The quiescent point is choosen so low, to obtain a TTL compatible output where lower parts of the sinusoid are chipped off, and for the V_{CE} of the transistor to be high enough (about 4V).

The input voltage varies between 1 and 5V, so the varactor voltage varies between 0 and 4V approximately. In this manner, the center frequency should be at about 2V on the diode and the varactor capacitance should be found for a 2V value, and L1 calculated. This frequency is determined from the resonant requency of the tank circuit which consists of L1, Cv and Cf.

Cf is generaly choosen much greater (at least ten times) than Cv so that Ceq is approximately equal to C_V; L1 can be calculated since C_V is found to be about 40 pF from its characteristic CV curve. f is taken as the center frequency 8MHz.

$$f = \frac{1}{2\pi\sqrt{L_1 C_V}} \qquad L_1 = \frac{1}{4\pi^2 f^2 C_V} = \frac{1}{4\pi^2 x(8)^2 x40} = 9.9\mu H \simeq 10\mu H$$

The output to input feedback ratio is choosen quite large since the common base amplifier voltage gain is not very much. The loop gain is equal to βA_V where β is the feedback ratio and A_V the voltage gain. The output of the oscillator is the collector of the transistor, however, the VCO output is taken from the emitter for TTL compatibility. In fact, when an AC analysis is made the emitter is used as the feedback input of the oscillator. Assuming a voltage gain of at least ten, the feedback is taken with a turn ratio of one to ten to maintain a loop gain larger than one for oscillations. This ratio will be valid also for a larger voltage gain. The VCO gain K_V depends directly on the varactor diode characteristic and the output frequency is proportional to the inverse square root of the varactor capacitance. Only a numeric calculation can be possible and within 1.5 to 2V this gain is found to be about 7 to 8 Mrad/sec/V, which is quite an acceptable value for the PLL design steps.

CHAPTER 8

THE RAM AND EPROM MODULE

THE DECODE AND CONTROL LOGIC

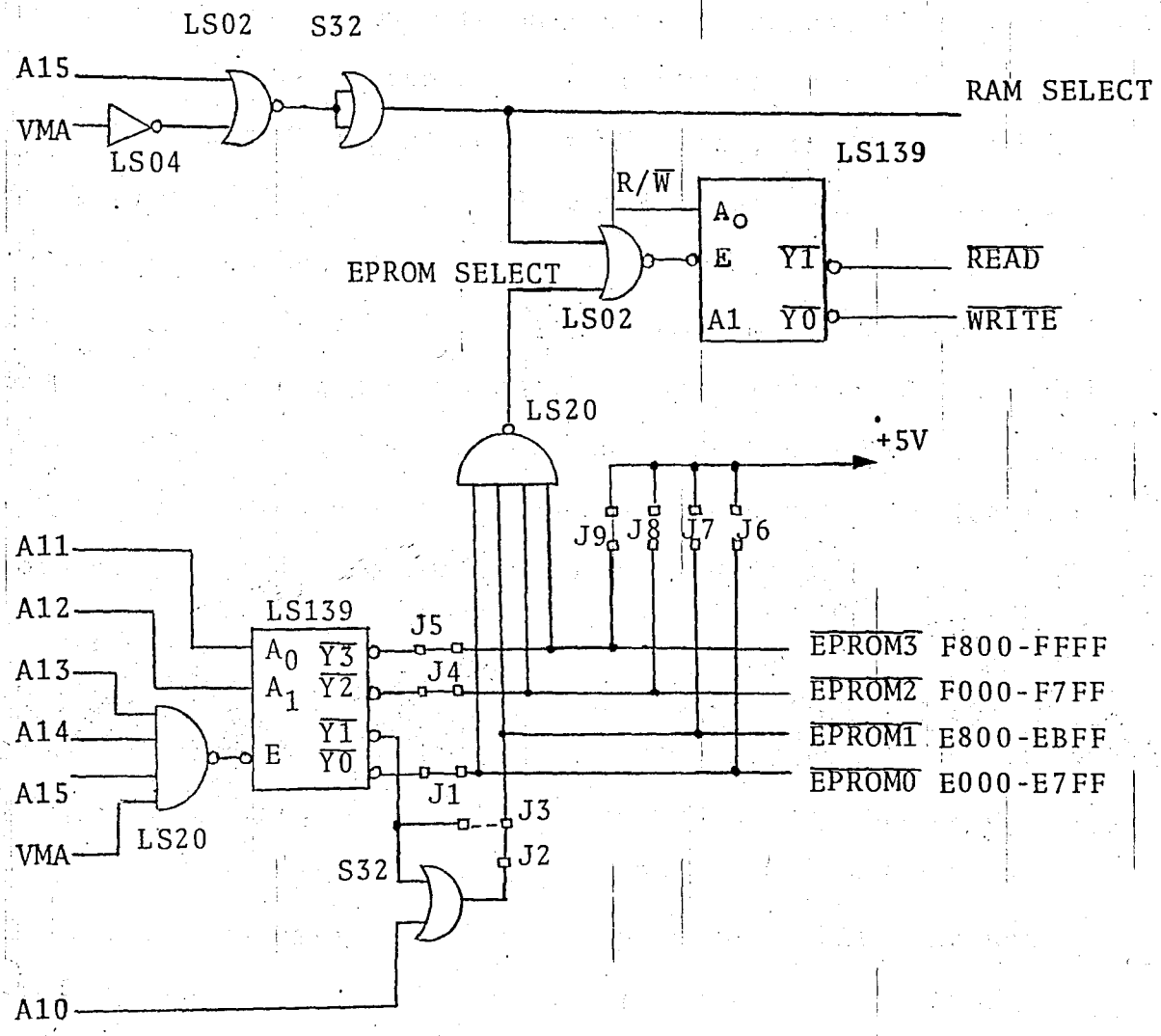
The additional RAM and EPROM module is designed to handle both the resident software (in EPROM) and the reloadable software (in RAM) of the system. The resident software will generally consist of the Input/Output driver routines for the tape, CRT, TTY, printer or Floppy Disk devices. It will also contain a monitor program to control the system externally. Any change in the system hardware can affect the resident software (for example substitution of a controller by another). The reloadable software, however, will be hardware independent but will use the I/O routines of the resident software. These programs are loaded into the RAM, executed in the RAM but saved in an external media to the system such as paper tape, floppy disk or cassette, and are under the control of an operating system which organizes the interface.

A monitor program together with a CRT as I/O device can occuppy up to 4K byte of EPROM. The floppy disk driver routines generally can go up to 1K byte. So the addition of other devices may require 6-7 kbyte of EPROM. For the

future use 8K bo EPROM is incorporated into this module from addresses E000 to FFFF to provide continiuty in the EPROM area up to the restart vectors. However, the use of the floppy disk controller prevent the access to memory locations EC00-EFFF since the FDC, PIA and scratch-pad RAM are in this area. For this reason an option is provided to disable this area. EPROM's used are four 2kbyte 2716 EPROM's since each may be modified separately (monitor, CRT controller or FDC controller softwares). 2716's were selected also because of their single + 5V supply. Half of the memory space is reserved for the RAM (32 kbyte) since only the operating system may occupy 8 K byte. The RAM location is from 0000 to 7FFF since higher locations are reserved to EPROM's, and its assymetric placement would cause more complicated decoding. There is also the advantage of direct addressing in locations 0000 to 00FF which reduces the software.

The RAM selection of the circuit is fixed and provided only by one address line, A15, together with the VMA line of the CPU, since A15 is the single line which can separate the low and high 32 kbyte memory. A15 low selects the low memory locations. To decode the EPROM's, again the TTL decoder 74LS139 is used since it is the most suitable for this application. Whenever the three highest address lines plus the VMA line is high the highest 8K of the memory is selected (Hex "E") which is used to enable the decoding. The A_0 and A_1 inputs are connected to address lines A_{11} and A_{12}, respectively, and the decoder subdivides the 8K area

to four 2K portions, each output enabling one EPROM.



RAM AND EPROM SELECTION

However, there is an option for the $\overline{Y1}$ output which should be installed when the FDC module is used, to disable the high 1K byte of EPROM 1. The $\overline{Y1}$ output defines the address E800-EFFF, and it is masked by OR'ing with A10, to deselect the EC00-EFFF portion. Whenever A10 is high, no matter what $\overline{Y1}$ is, EPROM 1 will be deselected and the CPU can

access other devices located at these addresses. To deselect an EPROM completely it's related jumper at the output should be removed, and a jumper to pull up it's select line should be installed. For example, when EPROM2 is not used J4 should be removed and J8 installed. In addition, if EPROM1 is to be deselected, jumpers J2 and J3 should be removed and J7 installed. On the other side, if one wants to use EPROM1 fully (from E800 to EFFF) J2 should be removed and J3 installed. The fast S32 gate is used in this circuit, for the Chip selection to be as fast as for other EPROM's. The selection time is found to be about 40 nsec in the worst case (15 nsec for the LS20, 21 nsec for the LS139, and 5 nsec for the S32).

The RAM select line is active high and has a delay of about 25-30 nsec from the settling of VMA. This line, in fact, defines the lower half of the CPU memory and is used to activate the refresh timing of the two bank of RAM, 16 Kbyte each, by using A14 for this purpose. This line has also the purpose of enabling the R/$\overline{W}$ buffer's to transfer data from or to the module. Also to use in the R/$\overline{W}$ buffer enabling all the EPROM select lines are NAND'ed together to obtain an active high EPROM select line. Whenever one of the select lines is low, the output of the gate is high; in steady position all inputs are high and the output is low. The RAM SELECT and EPROM SELECT lines are NOR'ed to enable the second half of the LS139, which is enabled whenever one of the lines is high. A1 is grounded so that only the $\overline{Y1}$ and $\overline{Y0}$ outputs are controlled by the A_0 input which is the

R/$\overline{W}$ line. $\overline{YO}$ is low on a write operation and $\overline{YI}$ low on a read operation. These $\overline{READ}$ and $\overline{WRITE}$ lines are used to enable in, or out the data buffers. The input buffers, that is the write buffers are not tied together with the read buffers to form a bidirectional internal bus in the module, since the EPROM's have to be read only and, no data should flow into the module if a write command to an EPROM is issued by the CPU, by accident. That is why discrete buffers and a separated R/$\overline{W}$ control is used. The total delay to enable data in or out from the settling of the VMA is about 100 nsec and this delay is not very important, since the data settles at the end of the cycle, especially on read operations.

REFRESH THEORY AND LOGIC

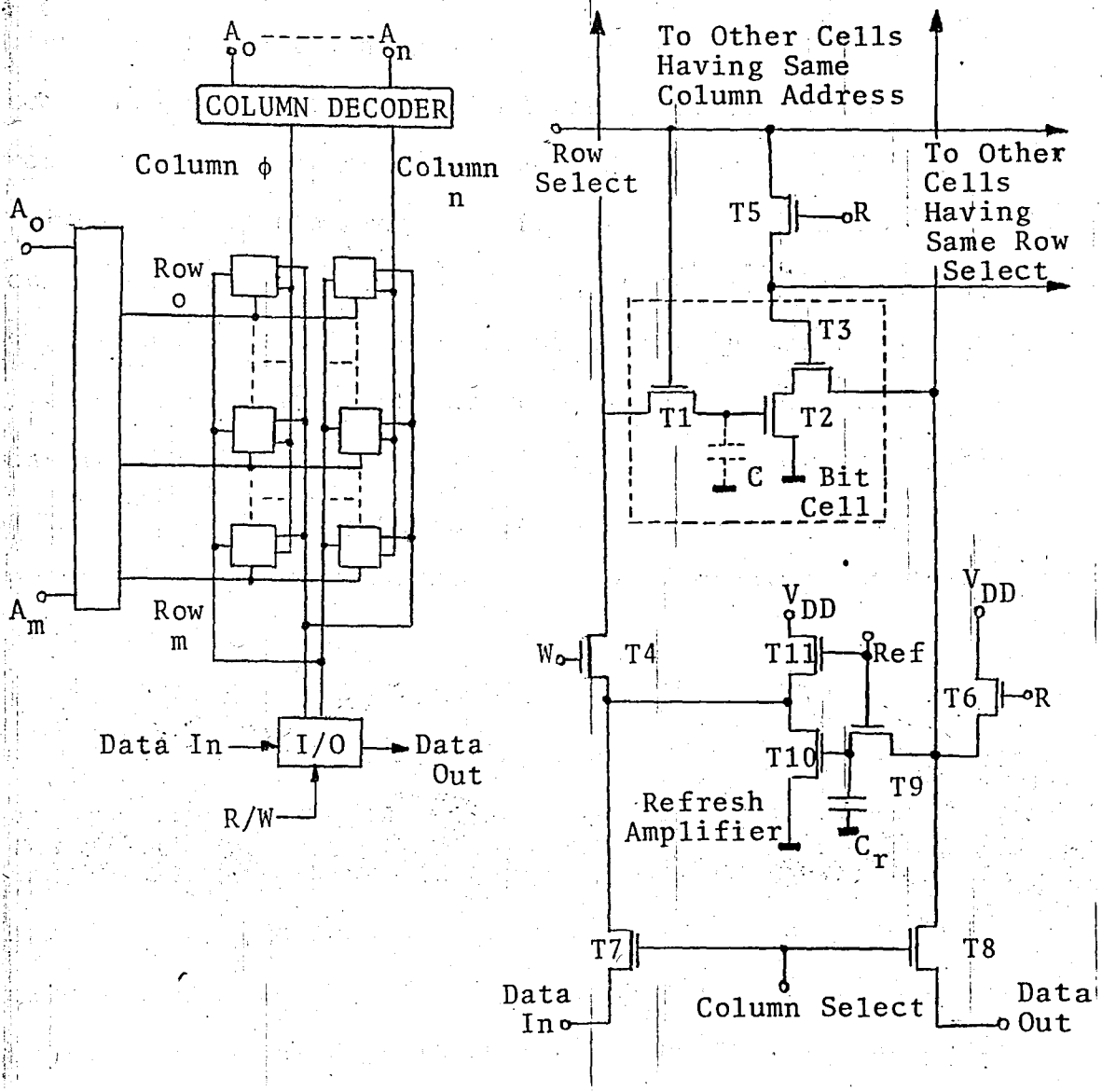
The 32 Kbyte RAM section has been choosen to be dynamic because of the large capacity required. Dynamic RAM's occuppy less surface on the board compared to static ones and they are more cost effective, although an additional refresh circuitry is required. For systems requiring more than 4 Kbyte memory the design can be optimized by using the high bit density and low cost/bit offered by the 4116 16Kx1 dynamic RAM. For this application sixteen 4116's will constitute the two 16 Kbyte area.

The minimal static MOS memory cell uses about six transistors. It uses a flip-flop, which consists of two

cross-coupled inverters plus one input and one output transistor. In the dynamic memory cell, the MOS transistor number is reduced to four, the reason being the use of stray capacitances to store the state of the cell. More developed cells use three transistors by storing the data on a single input capacitance. However, it should be kept in mind that if a write operation has not been performed for an extended time, the leakage of the capacitor charge may cause the information in the cell to be lost. It is therefore necessary to perform a refresh operation periodically so as not to lose the information (32).

Another difference of the dynamic RAM from the static one is its multiplexed address lines. In this way more dense storage is obtained using less number of external pins. The two set of addresses named Row and Column Addresses are decoded separately and their cross point define a memory cell in the mxn matrix.

The memory cell itself contains three transistors and the logic level on the cell depends on the capacitor C. That is why the memory cell should be designed to prevent discharge of C, especially during read operations. One way is to increase C, but while charging the capacitance during a write operation, the interval which needs to be devoted will be extended. For this reason separate paths for writing and reading C in addition to the refresh amplifier to update the charge on C are added.



DYNAMIC MEMORY ORGANIZATION AND THE MEMORY CELL (32)

To access and write the RAM, the cell is selected by putting its row and column select lines high together with the W line so that C is connected to the data-in input through T1, T4 and T7. Capacitor C charges to the state of the data input. To read the cell, R is set high so that T2 is connected to the Data Out line having T6 as its load since T3 and T8 are on. In this way an inverter is formed, and the state on C is inverted and read.

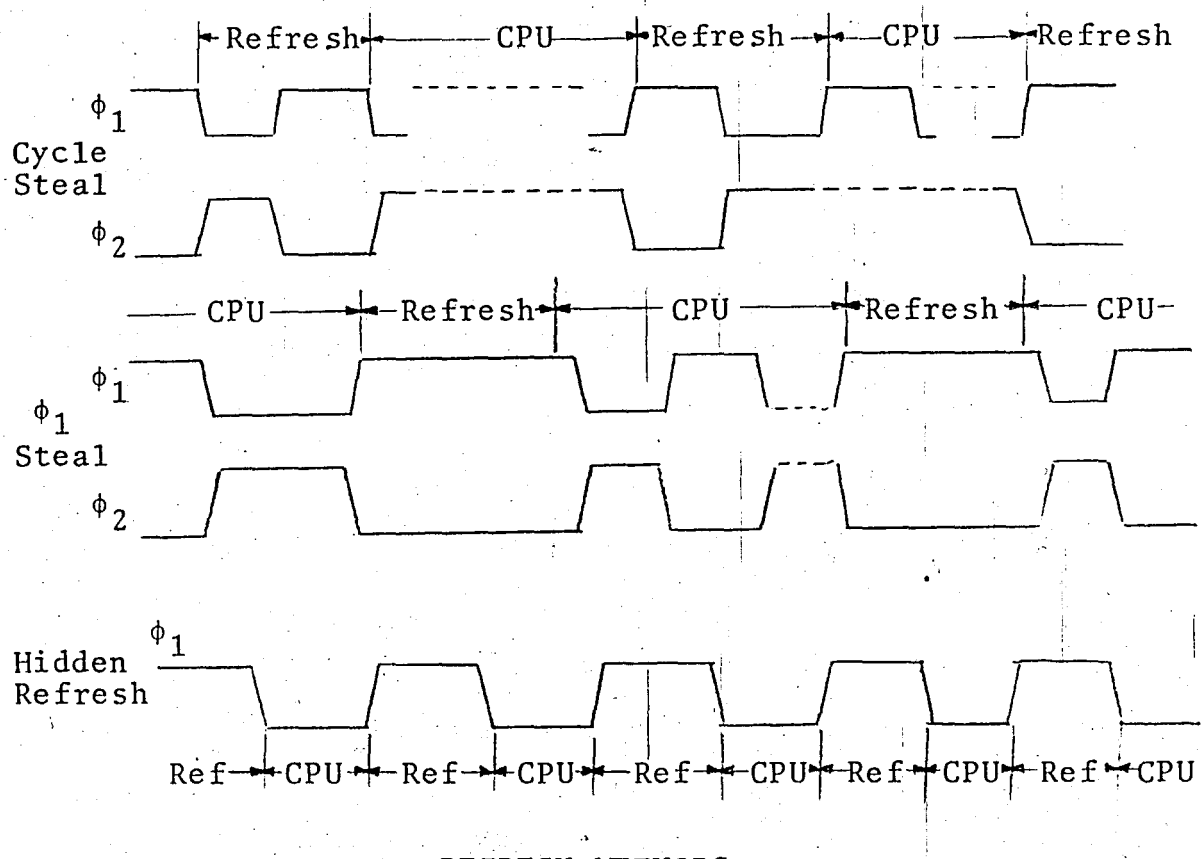
To refresh the cell only the row address line is set high together with the R and Ref lines. The data in and out lines are disconnected and the inverted data on C is read into the refresh capacitance C_r, since T3, T9 are on. This is called the precharge time since C_r precharges to the complement of the level on C. After this period write is set on and read off. The output of the inverter, T10 and T11, then refreshes the charge on capacitor C. This is to ensure that T11 does not initially load capacitor C until after C_r has been precharged to the correct level. If this precaution were not taken, capacitor C might be discharged erroneously.

The memory cells having the same column address are often refreshed sequentially so that in each refresh cycle one row is refreshed, in the next cycle the following row, etc. For each column of cells there is a single refresh circuit. In fact T4, T6, T7, T8, T9, T10 and T11 are shared by one column while T5 is shared by one row. If an M bit memory consists of m row and n columns the total number of transistors would be $m + 7n + 3M = m + 7n + 3mn$, since $M=mxn$ and there are 3 transistors per cell.

The 4116 is a 16384x1 bit dynamic RAM using an advanced N-channel silicon gate technology with a single transistor dynamic cell, organized in form of 128 x 128 memory array so that each address decoder accepts 7 address lines, the total being 14 address lines. The multiplexed addres-

ses permits the 4116 to be packaged in a standard 16-pin DIP. Refresh of the dynamic cell matrix is accomplished by performing a memory cycle at each of the 128 row addresses within each 2 msec time interval (21).

Any memory cycle will perform the refresh operation, however, this function is most easily accomplished with $\overline{RAS}$ cycles only, where only one row address is given, and the whole row refreshed at a time. This feature incorporated on most of the dynamic RAM's results in a substantial reduction in operating power. There are three major refreshing techniques while using an 6800 family CPU. The first one is to stop the system every 2 msec and perform the 128 refresh cycles or to distribute the 128 cycles over a 2msec time. Assuming one refresh cycle takes 1 CPU cycle (1 μsec for 1 MHz operation) the loss in the CPU time will be 128x1/2000 = 6.4%. Another method is to steal the ϕ_1 time by stretching the clock. The ϕ_1 time of the CPU, sets the address control signals in preparation for the data transfer during ϕ_2. By stretching or lengthening the ϕ_1 portion of the cycle, program execution is delayed, allowing memory refresh to take place. If ϕ_1 is stretched by 500 nsec the CPU time will be stollen by 3.2%. In most systems the reduction in program execution time may not be tolerated because of the real time and software requirements. For these type of systems, a "hidden" refresh configuration may be used.



REFRESH METHODS

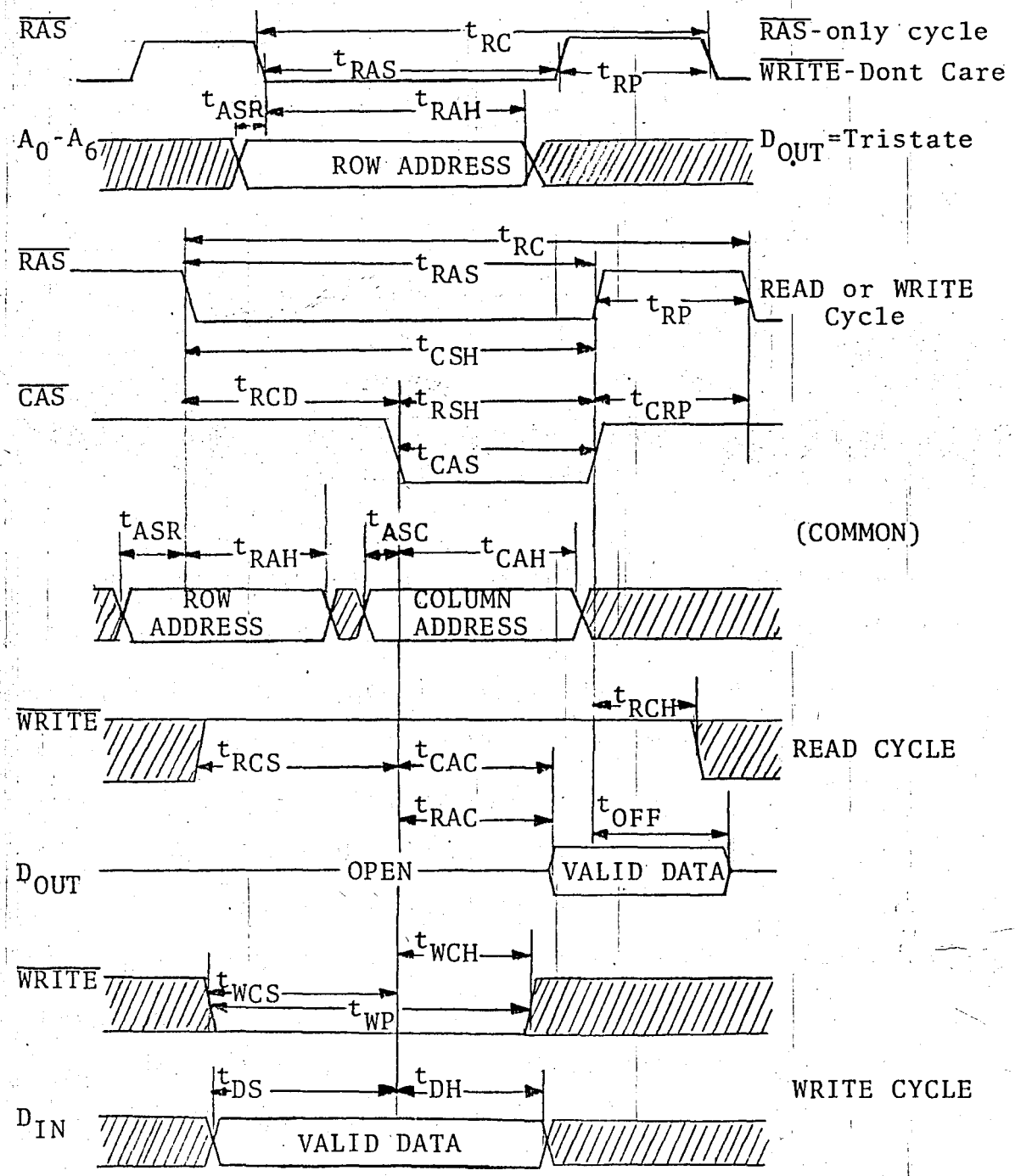
The place to hide or perform the memory refresh independent of the MPU program execution time, is during ϕ_1 as no data is being transferred between the MPU and memory or peripherals. This technique places the additional constraint on the dynamic memory system of being able to perform a complete refresh cycle during ϕ_1 (430 nsec minimum) and a complete read or write cycle during ϕ_2 (470 nsec minimum) if the MPU operates at 2 MHz. Using this concept only 128 of the ϕ_1 periods every 2 msec may be used for refreshing leaving the other ϕ_1 time periods open for other uses such as DMA transfer from external sources. In this mode, DMA and memory refresh would share the ϕ_1 portion of the cycle while the MPU would have access to the memory during ϕ_2 portion of the cycle (3). However, this requires an additional

hardware to determine the refresh time, so that an automatic refresh scheme on each ϕ_1 time has been designed which finishes the whole RAM refresh in 128 μsec instead of 2 msec. In this case, again DMA would have to halt the processor for access.

Whatever kind of refresh is used the hardware will consist generally of similar elements. The main element for 128 refresh cycles will be a 7 bit counter which constitutes the refresh counter. This counter increments in every refresh cycle to refresh all rows sequentially. Also the CPU addresses should be multiplexed with the necessary timing while reading or writing the RAM. In addition the CPU addresses should be isolated while refreshing and vice versa. Two flip-flops are the main circuitry to generate the Row Address Select ($\overline{RAS}$) and Column Address Select ($\overline{CAS}$) lines of the RAM. Additional circuitry is used to enable the counter, multiplexers, $\overline{RAS}$ and $\overline{CAS}$ lines.

The refresh cycle generally mentioned as the $\overline{RAS}$ - only refresh cycle is accomplished in the most easiest way. First $\overline{RAS}$ is held high and the row address to be refreshed is prepared. Afterwards $\overline{RAS}$ is lowered to let the address in; $\overline{RAS}$ is held low until the refresh occurs, then it is pulled up at the end of the cycle. In a read or write operation the sequence is more complicated. Similarly, the row address is prepared and latched in with a $\overline{RAS}$ signal, then the column address and R/$\overline{W}$ line is prepared (if it is a write

cycle the data should also be ready) and all the information is latched with the $\overline{CAS}$ line. Here the $\overline{CAS}$ line acts as a kind of chip select for the RAM. Then the CPU must wait till the access time expires, take the read data out if it is a read cycle and remove both the $\overline{CAS}$ and $\overline{RAS}$.



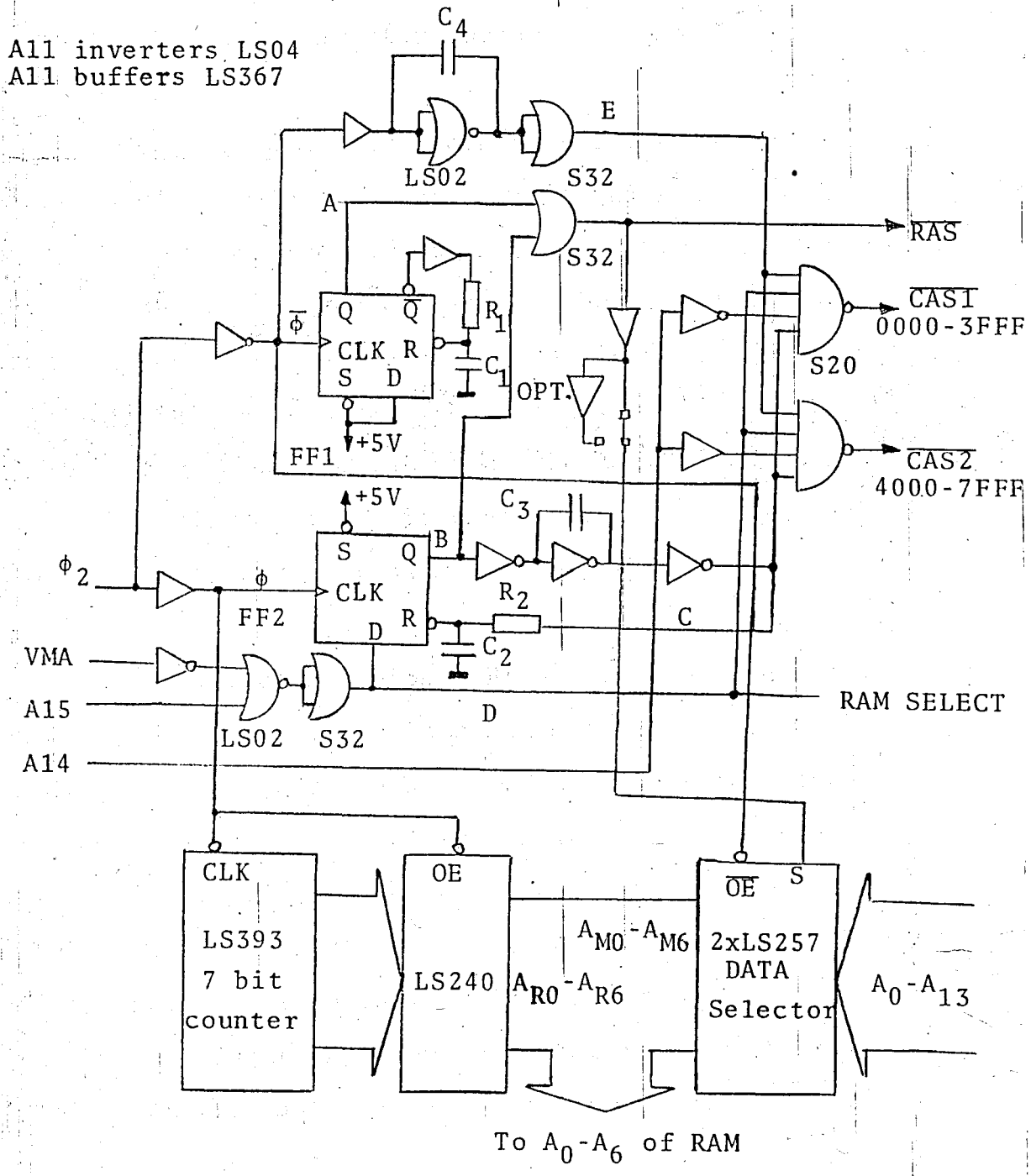
4116 READ/WRITE AND REFRESH TIMING (10)

Parameter	Minimum (nsec)	Maximum (nsec)	Parameter	Minimum (nsec)	Maximum (nsec)
t_{RC}	375		t_{RAH}	25	
t_{RAC}		200	t_{ASC}	10	
t_{CAC}		135	t_{CAH}	55	
t_{OFF}	0	50	t_{ar}	120	
t_{RP}	120		t_{RCS}	0	
t_{RAS}	200	10000	t_{RCH}	0	
t_{RSH}	135		t_{WCH}	55	
t_{CAS}	135	10000	t_{WP}	55	
t_{CSH}	200		t_{DS}	0	
t_{RCD}	25	65	t_{DH}	55	
t_{CRP}	- 20		t_{WCD}	- 20	
t_{ASR}	0		t_{REF}		$2x10^6$

TIMING VALUES FOR THE MK4116P-3 (250nsec Access, 375 nsec Cycle Time) (21)

'If the timing values are examined it is seen that the slowest 4116, which is the 4116-4 can be used for the hidden refresh application, since its cycle time is less than both the minimum ϕ_1 amd ϕ_2 cycles. The problem is to synchronize the RAM operations with ϕ_1 or ϕ_2. All the refresh timing, refresh address and CPU address multiplexing will be derived from ϕ_2. Since there are 32 Kbyte of RAM, the circuit is divided into two banks each having eight 4116's to form the 16 Kbyte of one bank. The banks are selected by address line A_{14}; when A_{14} is low the lower bank is selected, and when high the higher bank is selected.

The aim of the circuit is to perform a $\overline{RAS}$ cycle only during the ϕ_1 period that is when ϕ_2 is low. Whenever, a READ or WRITE command is issued this will be performed in the ϕ_2 cycle that is when ϕ_2 is high. In this application $\overline{CAS}$ can be low only during a ϕ_2 cycle since this cycle is reserved to select and read or write to the RAM.



HIDDEN REFRESH CIRCUITRY FOR TWO BANK OF 4116's

Two $\overline{CAS}$ outputs are given from the refresh circuitry for each of the two banks controlled by A_{14}. There is no CPU cycle where no refresh is performed, even in halt position, with the exception that ϕ_2 is not active. Every timing is based on ϕ_2 and the refresh occurs for the whole RAM area having the same Row Address. The $\overline{RAS}$ precharge time and the $\overline{CAS}$ delays are arranged by using two flip-flops connected as monostables and inverter-buffers to obtain wanted delays.

The ϕ_2 line is inverted and buffered separately to obtain equally delayed ϕ and $\overline{\phi}$ lines which control the primary functions of the refresh circuit. The ϕ line increments the refresh counter and enables the buffer to drive the RAM address lines. At the same time the inverse of ϕ, $\overline{\phi}$, disables all the CPU addresses. During the other cycle when ϕ is high, $\overline{\phi}$ enables the output of the 14 to 7 line data selector. The selector is additionally controlled by the delayed $\overline{RAS}$ output for CPU address multiplexing.

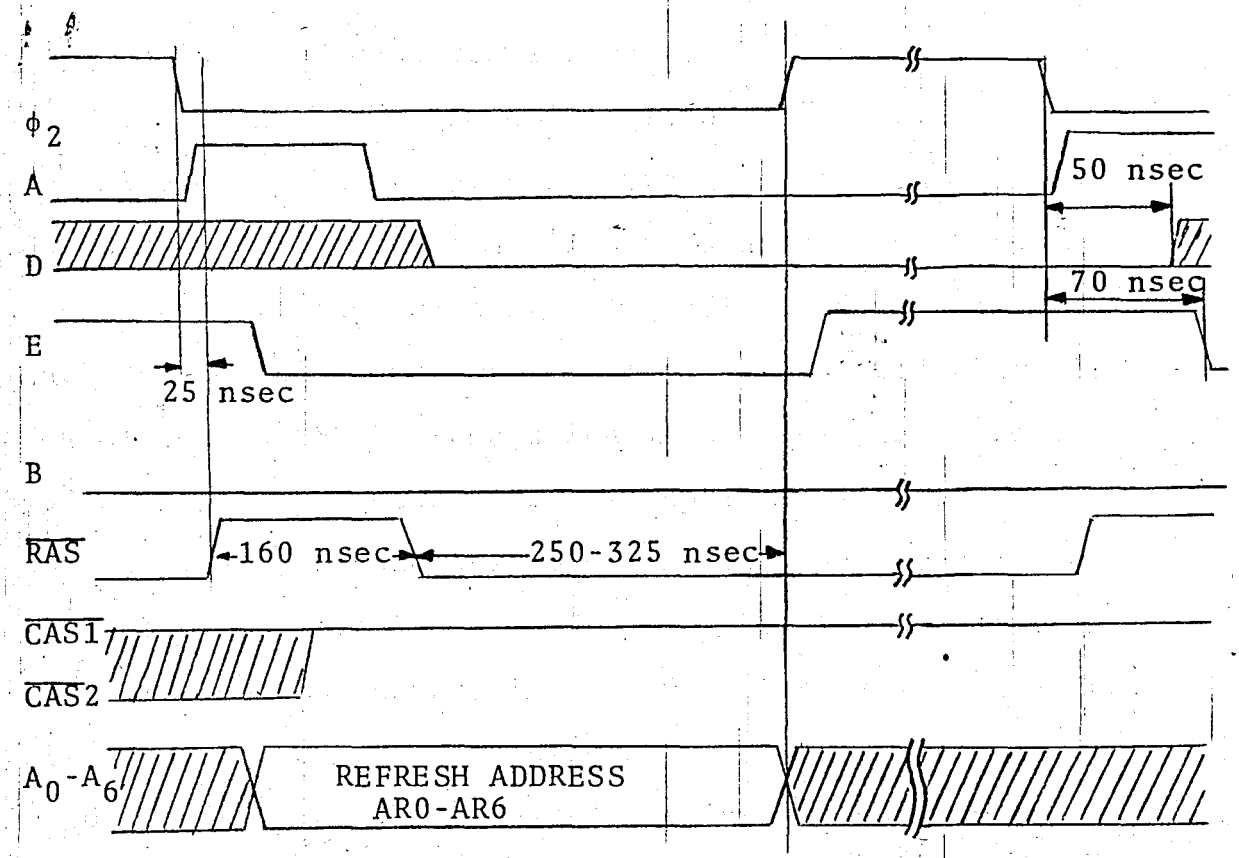
On power up position, FF1's output Q assumes either a low or high state. If it assumes a high state $\overline{Q}$ is low so C1 discharges through R1 and the R input goes low to reset the Q output. When Q becomes low, or it assumes a low state initially, $\overline{Q}$ is high and charges C1 to a high state pulling up the R input to prevent continuous reset condition. In fact FF1 is a monostable which generates the $\overline{RAS}$ precharge time on positive going $\overline{\phi}$ pulses, that is negative going ϕ_2 pulses. At the high going edge the Q output is set and it

remains high until C1 discharges through R1 and the R input becomes logic low. The value of R1 and C1 can be roughly calculated by using the 4RC constraint for the discharge time and fine adjusted by experiment. The $\overline{RAS}$ precharge time (t_{RP}) should be greater than 150 nsec and the $\overline{RAS}$ should be held low for 250 nsec (t_{RAS}). Considering that the total half cycle can be minimum 430 nsec the monostable should be adjusted to 150-180 nsec. FF2 operates with the same logic, however, there are some modifications. The D input of the flip-flop is not alwasy high, but only when the RAM is selected, since the $\overline{RAS}$ and $\overline{CAS}$ should not be generated otherwise.

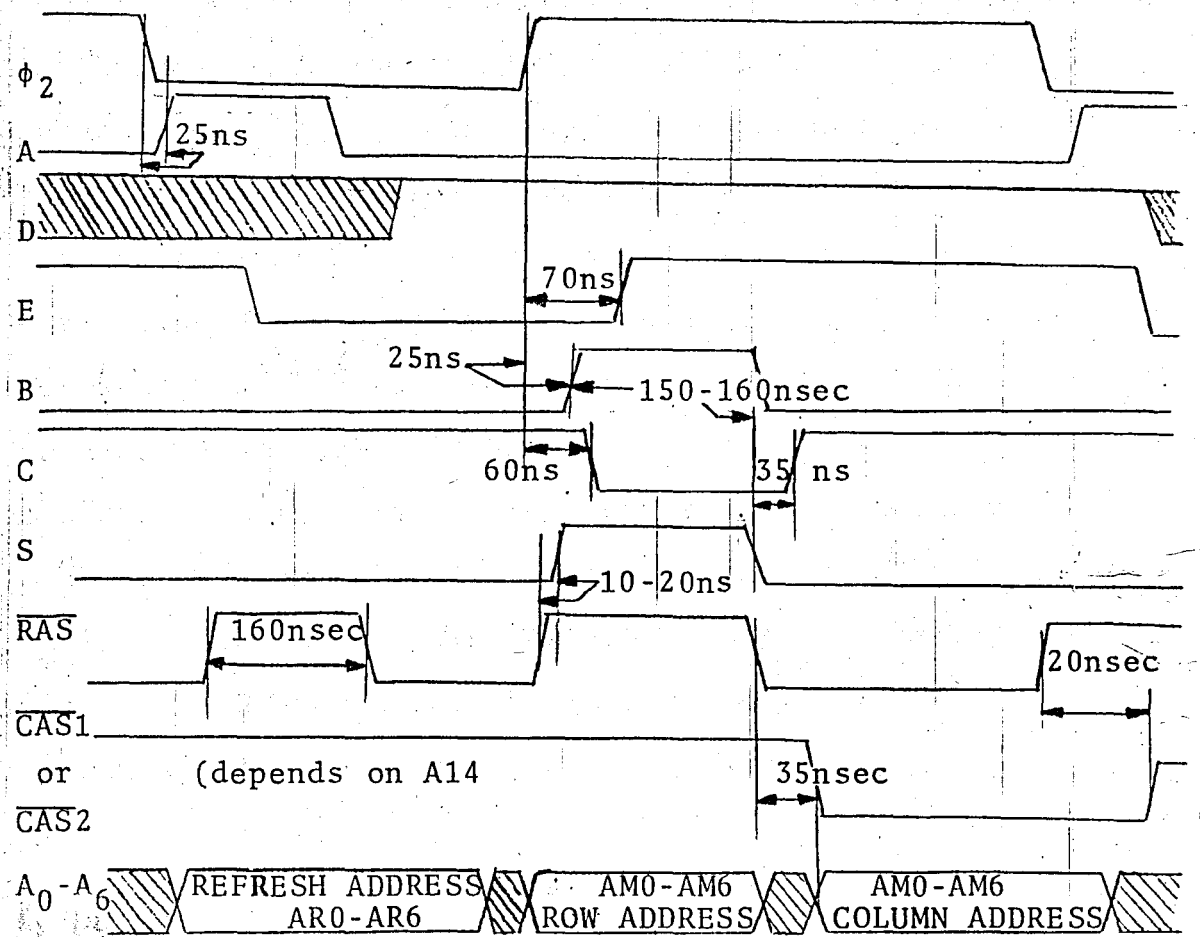
The big difference is the addition of a delay circuit between the Q output and the reset circuitry. The delay is obtained by the slow action of the second inverter which is obtained by the paralleling capacitor. Assuming the first inverter's output is high, than C3 will be charged from the left. If a transition occurs, C3 will discharge into the first inverter causing a certain delay until the second inverter accepts the signal as a logic low. When the second inverter goes high it charges C3 from the right. Now if a transistion occurs C3 will discharge into the second inverter and cause the same delay. The delay introduced can again be arranged experimentally. It should be noted that the total pulse length is equal to the delay introduced by the three inverter circuit plus the discharge time of C_2 to the third inverter. The third inverter output (C) will be used to generate the $\overline{CAS}$ outputs since it is delayed from $\overline{RAS}$.

The $\overline{CAS}$ outputs are enabled only when the appropriate bank of RAM is selected. That is why the RAM SELECT line is gated with A_{14} and $\overline{A_{14}}$ to select the two $\overline{CAS}$ outputs separately, producing a $\overline{CAS}$ for each bank of RAM, only when selected from the CPU. Another signal which disables the $\overline{CAS}$ outputs is the delayed ϕ_2 line, since no $\overline{CAS}$ can occur during the low state of ϕ_2. The reason to delay ϕ_2 is to prevent the occurrence of a glitch between the high going edge of ϕ_2 and the low going edge of the (C) output. That is, the delay introduced by C4 should be larger than the delay introduced by C3 plus the flip-flop propagation delay. The $\overline{RAS}$ to $\overline{CAS}$ delay is arranged to be 40 nsec so the delay introduced by C4 should be greater. Also C2 should discharge in about 100-110 nsec to add up with the delay of the three inverters to form the $\overline{RAS}$ precharge time.

During a refresh cycle the S input is pulsed since this input is in fact the $\overline{RAS}$ output delayed. However, this has no destructive effect, since during the ϕ_1 cycle only the refresh addresses are enabled and no CPU address can access the RAM, until the ϕ_2 cycle, where address lines are left to the CPU, under the control of the S line. The S line is left low during the second half of a refresh cycle and the low CPU addresses are directed into the RAM. This has no refresh or read/write effect since $\overline{CAS}$ is high and the refresh addresses have been latched in the first half of the cycle.



REFRESH ONLY CYCLE (RAM NOT SELECTED DURING CPU CYCLE, D=0)



REFRESH AND READ/WRITE CYCLE

During a read/write plus refresh cycle the S line becomes active on the second cycle since a $\overline{CAS}$ output occurs. The S line delay should be well arranged since the row addresses should be held for at least 35 nsec. For this reason the switching of the row address to column address should be delayed by this same amount. The address hold time will be equal to the LS257 propagation delay plus the introduced LS367 buffers (the second LS367 is optional if the delay is insufficient). The address hold time may even overlap 20nsec after the $\overline{CAS}$ activation according to the 4116-4 specifications. The column address is held until the end of the CPU cycle. The $\overline{RAS}$ output goes high about 25-30ns after ϕ_2's low going edge since gate delays are introduced (2 Schottky gates and one low power Schottky gate delay, assuming 15nsec for LS gates, 6 for S gates). The high going edge of $\overline{CAS}$ is controlled by the E or D output. The D output goes low if either VMA goes low or A15 goes high, that is after 30-50 nsec from ϕ_2's low going edge since the CPU holds the VMA for about this period. The E delay is about 60-70 nsec so the control of $\overline{CAS}$ for the low to high transition is on the VMA line. In this way $\overline{CAS}$ goes high about 10-20 nsec after $\overline{RAS}$ goes high for precharge. The refresh addresses or column addresses are held until the end of each half cycle. The $\overline{RAS}$ output is held low for more than 250 nsec and $\overline{CAS}$ for about the same time (this time depends on ϕ_2 pulse width, assuming a minimum ϕ_2 pulse of 430 nsec); in this way the timing will be enough for the access time of the 4116. The disadvantage of the Read/Write cycle is that the bank of RAM

which is not selected is subject to a refresh cycle since the $\overline{RAS}$ is common to both banks. Supposing A14 is low, the first bank is pulsed with $\overline{RAS}$ and $\overline{CAS1}$ but the second bank only with $\overline{RAS}$, since A14 is low and $\overline{CAS2}$ high. In this way bank 1 is accessed and bank 2 refreshed with the row address of bank 1 during the second half of the cycle.

It should be noted that only Schottky TTL devices are used for the input, $\overline{RAS}$ and $\overline{CAS}$ circuitry since no big delays are permitted for the refresh logic, these delays causing the decrease of the access time. Also only a Schottky gate could drive the sixteen $\overline{RAS}$ inputs of the RAM's. In the design, attention is given to the fan outs of LS and S gates; that is a reason for buffering ϕ_2 to be capable of driving the Schottky flip-flops. Also the D output is buffered with a OR gate to be able to drive the three Schottky gates. The E output is also buffered in the same manner for the same reason.

The big difficulty to design the circuit is that the resistor and capacitor values can be determined only by experiment since the delays introduced may even depend on the chip used and a new calibration could be required if the chip is replaced. However, it is possible to design a reliable circuitry considering reasonable tolerances.

CHAPTER 9

THE FDC FIRMWARE

FIRMWARE ORGANIZATION AND ENTRY

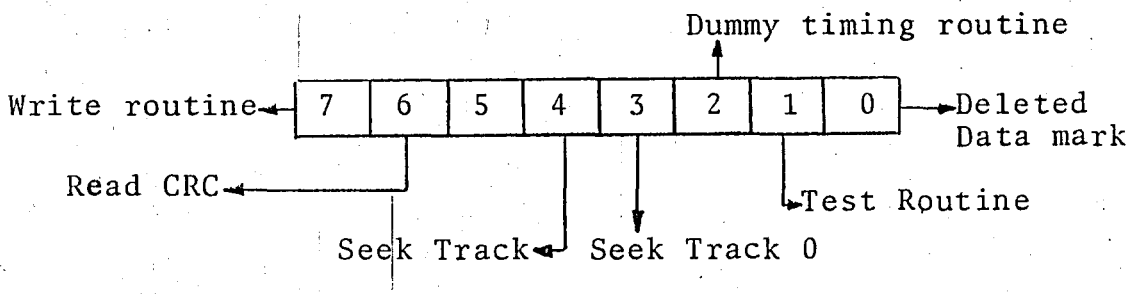
The firmware contains fifteen subroutines where four of these perform some special functions, and the other eleven perform the basic disk operations, according to the parameters set by the user (the CODOS operating system for this case). The special four routines are the operating system loader, the hardware initialization (PIA and FDC), the error checking routine, and the error display routine. These four routines have no parameter to be passed by the user; however the diskette routines need some parameters to perform the operations on these indicated variables such as track, sector or RAM address. The parameters are passed to the firmware by a firmware-operating system shared RAM area, consisting of nine bytes as explained below:

$0000	DRIVE	This byte contains the drive number where the operation has to be performed (0 to 3).
$0001-0002	BEGSCT	These two bytes contain the physical sector where the operation has to begin (0 to $7D2 for single sided, 0 to $FA4 for double-sided diskettes).

$0003-0004	NUMSCT	These two bytes contain the number of sectors where the operation will be performed (the sum of BEGSCT and NUMSCT cannot exceed $7D2 for single and $FA4 for double sided diskettes).
$0005	SCTCHR	During read into memory operations, this byte contains the number of bytes to be read from the last sector used, since file may not fill all of the 128 bytes of the last sector. The firmware may not stop on the byte specified since it segments the last sector into 8-byte blocks and stops loading after reading the block containing the last byte requested. If this number is greater than 128 an error will occur.
$0006-0007	BEGADR	These two bytes contain the first address from/to which data is to be read/written during disk read/write operations. The firmware updates this entry after each sector is read/written.
$0008	ERSTAT	Whenever the firmware sets the carry bit of the condition code register (CCR) this byte will contain an ASCII number ($31 to $39) indicating the error type. If no error occurs, then the carry is reset and this byte contains $30.

In addition to the shared RAM the firmware has also its own RAM, consisting of 18 bytes. This area is used to save some of the user registers such as the stack pointer (SPRSAV), NMI vector (NMISAV). Some of these variables are LOGSCT which shows the physical track sector, TRKADR which shows the track on which the head is positioned, RESECT

which shows the number of sectors left for operation, DBLSDF which shows whether the diskette is double or single sided, RTRIAL which shows the number of retrails for the operation. There is also a block of four bytes (CRTRK0-3) which saves the head position of each drive, since each drive is accessed at separate times, and the last position of the head should be known on each track (the operating system is not aware of the head positions, it only gives commands to position the head on a specified track). The most important byte in this area is FRMPAR which saves the firmware subroutine parameter. The firmware dedicates an hex byte to the different 11 subroutines (this is reduced to 10 subroutines since the basic read and partial read can be differentiated from the parameter SCTCHR). The basic read routine immediately loads SCTCHR with 128 so that these two routines are identified by setting FRMPAR to 0. The other commands are differentiated by some of the bits of FRMPAR.



For Seek Track, Seek Track 0, Read CRC, Basic Write and the dummy timing routine, only one of the related bits should be set. However, it is possible to set two or three bits when the routine is a write routine. Setting bit 7 and 0, means write with deleted data mark, setting bit 7

and 1 means perform a write test (to check write protection for example), setting bit 7 and 6 means write then check CRC, and lastly setting bit 7, 6 and 1, perform a write and read CRC test. The firmware is so organized that, as an example, the read CRC routine is a subprogram of a write and check CRC routine.

Upon each entry to one of each FDC firmware disk routines the B accummulator is loaded with the related control word FRMPAR which is saved later. All routines are activated by a branch or jump to subroutine command after all parameters are stored to the shared RAM area. The entry points are organized in a different form to avoid vastage of EPROM area:

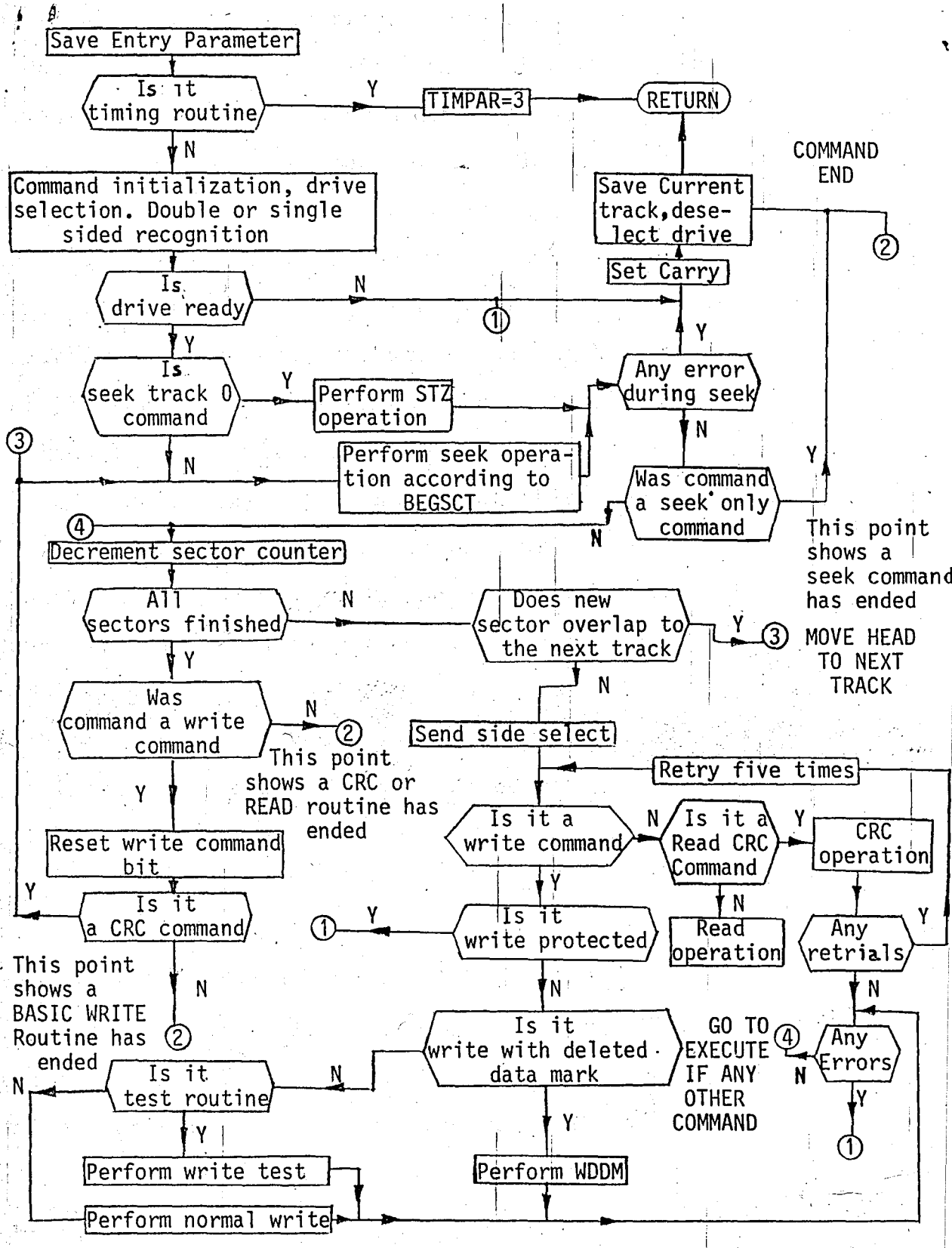
```
                                    :
E886   C6  40  REDCRC  LDA  B  #$40      READ CRC  Entry
E888   8C              FCB     $8C       WRITE AND CHECK CRC
                                            ENTRY
E889   C6  C2  WTHENR  LDA  B  #$C2
  :                                 :
                                    :
E8A8   D7  OE          STA  B  FRMPAR    STORE FIRMWARE
                                            PARAMETER
                                    :
```

If an entry to WTHENR is made, the B accummulator is loaded with $C2, and assuming the B accummulator is not altered later, this parameter is stored to FRMPAR. If an entry occurs to REDCRC, the B accummulator is loaded with $40 and then, a 8C C6 C2 is encountered which means CPX #$C6C2, so the Index Register is compared immediately with

$C6C2. At that point the B accummulator contains $40 and FRMPAR is stored with this same value since the program in between is similar after the WTHENR point. In this way a branch or jump instruction after the loading of the B accummulator is avoided, that is at least one byte (two bytes if an extended jump is needed) is saved for each entry, with the disadvantage of dummy "compare immediate index register" instructions which increases execution time. The same reasoning is found on error exit operations which has to store the error number on ERSTAT and set the carry bit of the condition code register. These error exit points loads the accummulator with the error number, performs a series of "BIT" instructions and ends with a "STA ERSTAT, SEC" instruction.

After the entry is performed and the firmware parameter is saved, the execution begins by looking if the command was to pass the timing parameter to CODOS, since this routine does not activate any drive hardware. If the command is a drive operation the drive is selected, the double side input is read to identify the diskette type, the stack pointer saved, error conditions cleared, the track where the head of the related drive is positioned is found, and a check for the ready signal from the drive is made.

If the operation is not a seek track zero command, immediately a seek operation is performed to the track which contains the sector number written in BEGSCT. This operation is performed at all times since the head need to be



FIRMWARE CONTROL PROGRAM FLOWCHART

positioned for all operations. The seek track zero command is mainly used for initialization. Whenever an only seek command is issued (bit 3 or 4 is set) the command ends after the seek, otherwise the sector count is decremented to find if the command has ended. Whenever the sectors are finished bit 7 is tested to see if the command included a write operation. If so the CRC bit (bit 6) is tested and the write bit resetted. This reduces the write and check CRC operation to a single read CRC operation and the exit will occur in the previous check of bit 7 after the read CRC is performed. If the CRC bit is not set at that point this means that only bit 7 was set and a basic write or a write test occurred. To perform the secondary CRC operation the seek operation is repeated. During an operation if the sector number exceeds 26 for single sided or 52 for double sided the head is moved to the next track. Before a write operation the write protect input is checked and an error occurs if the write protect input is active.

If the command is a write command it is executed in three different forms: write with deleted data mark, write test and basic write 128 bytes into sector. The read operation is also divided to read CRC, read 128 bytes and partial read operations. If an error occurs after a disk operation and the error is recoverable a retry is made up to five times. If the error is not recoverable or five retries are made the command ends by writing the error type and setting the carry bit.

FIRMWARE SUBROUTINES

The firmware routines occupy about 750 bytes beginning from the address E800. It also uses about 32 bytes of RAM and 20 bytes of input/output registers. A brief explanation of each routine and some flowcharts are given in the following section:

FOSLOD $E800 — This routine initializes the drive electronics, calculates the timing parameter, puts the head of drive zero on track zero and loads sector 23 and 24 into memory beginning at $0020. The check for any error is made and if an error occurred the error is printed on the system console or TTY. After the loading, control is given to the boot-loader by jumping to that location. It, in turn, loads the rest of the operating system into memory. This function does not return control to the calling program. At least $120 bytes of memory are required starting at location 0 for execution of this entry.

FDHWNI $E039 — This subroutine initializes the PIA and FDC. It first initializes the set up and interrupt status registers of the FDC; then the A output register of the PIA is initialized with $FF to prevent selection of any drive when the register is switched to output mode.

FDFMER $E86A — This subroutine checks for a disk error if called immediately after return from a disk operation by checking the carry flag. The subroutine just returns to the user if no error occurred (carry clear). If an error did occur the subroutine prints an E followed by the ASCII character in

ERSTAT and two spaces to the system console and disk operation is stopped.

ERTOCO
$E871

This routine sends an E, the contents of ERSTAT and two spaces and is used by FDFMER. The only parameter to be passed is ERSTAT. No other parameter is modified.

SKTK00
$E88C

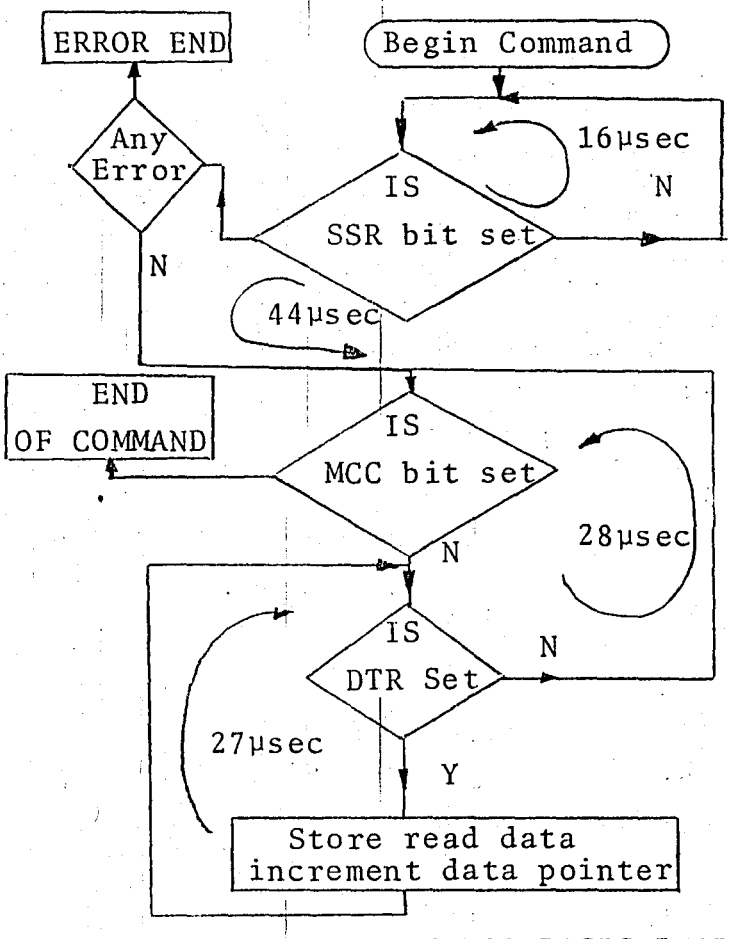
This entry causes the head of the DRIVE location to be restored to track 0. The drive must be ready or an error will occur. The subroutine loops around the settling time complete bit, controlling the error status at the same time. If the track zero input is not active after 83 pulses a seek error occurs. The head is loaded after the command has ended and remains loaded until a new command is issued or the drive deselected. All parameters are left unchanged with the exception of ERSTAT, in case of an error occurrence.

SEEKTK
$E88F

This routine causes the head of DRIVE to be positioned at the track containing BEGSCT and waits until the settling time is complete. Before the command is issued, BEGSCT and NUMSCT are added to find if the sector is in an allowed track (less than 77). The residue sector (between 1 and 26 for single, 1 and 52 for double sided) is stored to LOGSCT. The current track position is taken from the drive's CRTRK and stored to the FDC's CTAR. The wanted track is stored to the FDC's GCR. When the command ends without error, CRTRK is updated and the FDC's GCR is transferred to CTAR. The routine does not change any parameter except ERSTAT in case of any error. The head is in loaded position at the end of the command. The track number is found by dividing the given sector number by 26 for singled-sided and 52 for double-sided diskettes.

BSCRED
$E880

This routine reads the number of sectors contained in NUMSCT, beginning with BEGSCT from DRIVE into memory starting at the address contained in BEGADR. BEGADR is updated to the next address that is to be written into, after each sector is read. This entry sets SCTCHR to 128 since it reads the whole sector, and accordingly the basic read routine is separated from the partial read routine which has to compare the number of bytes read with SCTCHR. The read loop is a critical loop, and should be capable of handling all the operation for one byte in 32 µsec.



MCC: Macro command complete
SSR: Status Sense Request
DTR: Data Transfer Request

FLOWCHART FOR BASIC READ LOOP

The first event that a read sector command encounters, in a normal operation is the setting of the SSR bit which shows that the address ID field has

been detected and verified by a CRC check. During the search for SSR, DTR can not be set since data arrives after the ID field. If MCC is set before SSR, probably a sector address or CRC error occurred. In this way a 16µsec loop searches for SSR and when found the first data transfer will occur after 18 byte time (576µsec). The SSR bit service only checks for any hard error or the detection of a deleted data mark. When the SSR is serviced the search for any DTR begins, and when found, including the DTR bit test, it is serviced in 27µsec, which is less than one byte time, that is 32µsec. As noticed there are two loops in the read operation. The 28µsec loop occurs when DTR is not set consecutively. If DTR is found set it is immediately checked again so as to not loose the next data by making a 28+27µsec loop. When all the 128 bytes are transferred, the MCC bit is set and a check for any errors should be made.

REDLST
$E884

This entry is very similar to the previous BSCRED entry with the exception that not 128 bytes but a multiple of 8 bytes are read into the memory and BEGADR updated. If 128 is written into SCTCHR this routine is exactly the same as the BSCRED routine. The exception, in point of view of the software is that a comparison for SCTCHR is introducted which increases the 27µsec loop to 30µsec, which is again less than 32µsec; also when SCTCHR is reduced to 0, dummy read operations should be made and not placed into the RAM to prevent the data transfer error to be set.

REDCRC
$E886

This entry point reads the number of sectors contained in NUMSCT beginning from BEGSCT to check their CRC without reading the contents of the sectors. The only thing checked after the command is issued, is the MCC bit. When MCC is set the

errors are checked and in case of a CRC error the operation is repeated five times for one sector.

BSCWRT
$E89B

This routine writes NUMSCT of sectors beginning at BEGSCT from memory location contained in BEGADR. BEGADR is updated to the address of the next byte to be read from memory after each sector is written. The software logic is completely the same as the basic read loops, but data is transferred from memory into the floppy disk DOR, instead of the transfer from the DIR to the memory. The busy bit is checked and the STAB 0,X instruction increases the loop by one cycle and 28μsec is increased to 29μsec for the write loop.

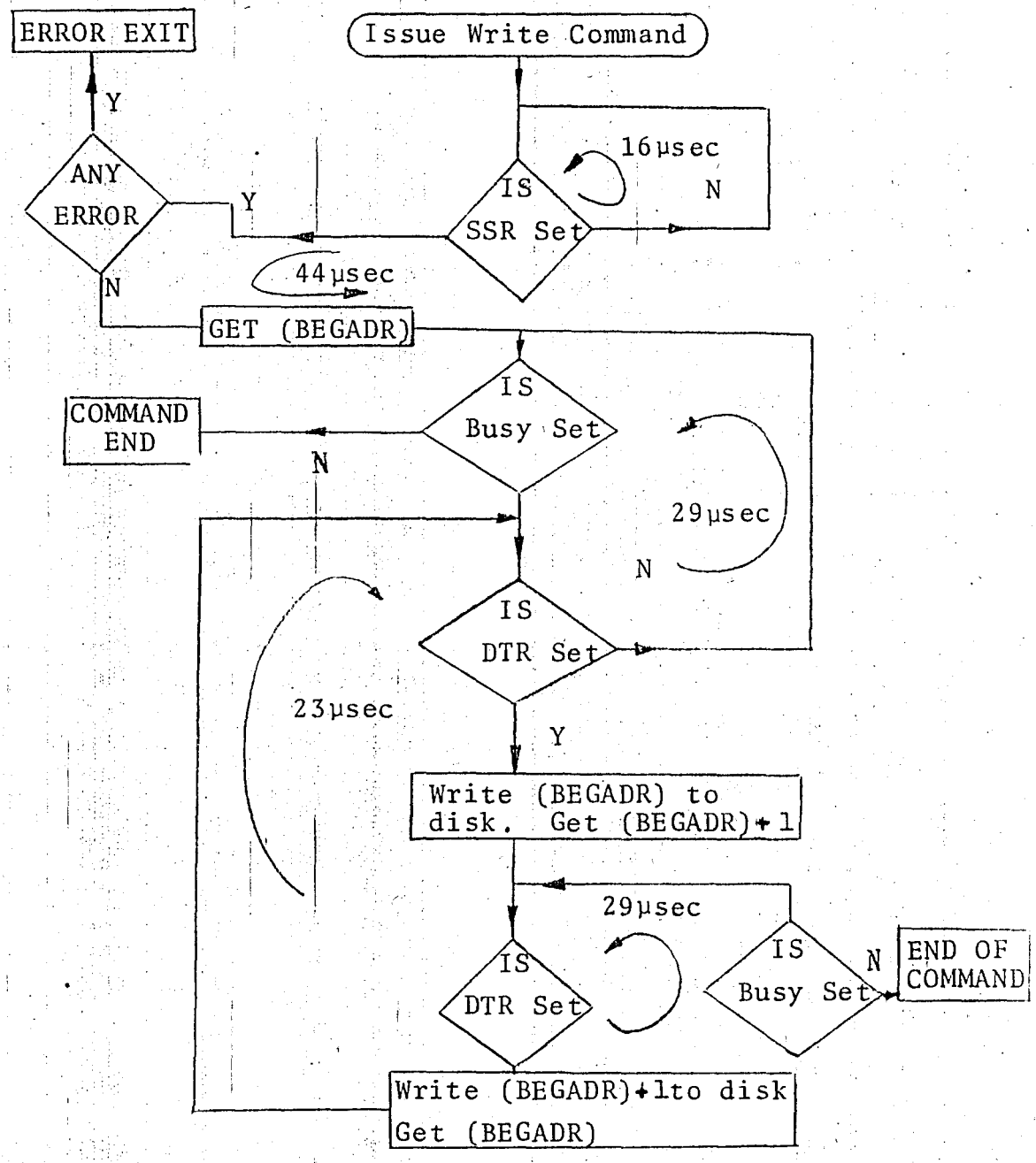
WANDRD
$E898

This entry is the combination of the BASWRT and REDCRC routines. First a basic write for all the sectors specified is performed, when all sectors are written the control program resets the write bit in FRMPAR and the operation is reduced to a read CRC operation. The control program jumps to the beginning of the program to perform a seek a command if necessary, followed by the CRC read command execution, and then the exit operation of the command occurs.

WRCHEK
$E892

This entry point writes the two bytes located at the address contained in BEGADR into alternating bytes of NUMSCT sectors beginning with BEGSCT. The address in BEGADR is not updated. The routine is used to check if a write operation on the diskette is permissible.

The write test loops are in fact two write loops, one to write even locations, the other for odd locations. These loops are the same as the basic write loop with the exception that the incrementa-



BEGADR: Address stored in the two locations of BEGADR.
(BEGADR): Contents of memory location pointed by BEGADR.

WRITE TEST FLOWCHART

tion of the index register is omitted so that the service routines are 23µsec. The search for MCC and DTR lasts at most 29µsec as in the write loop. The even loop gets the odd data to be stored and

jumps to the odd write loop. The odd loop gets the even data and jumps to the even loop. In this manner the loops and the written data are changed consecutively after each byte write operation.

WRWDDM $E895 — This routine writes a deleted data mark pattern to the beginning of the data field. The data written are the two consecutive bytes of the address shown by BEGADR. This routine is used to omit the usage of a specific sector.

WTHENR $E889 — This routine is a combination of the WRCHEK and the read CRC routine. First consecutive two bytes are written to the number of sectors specified. Then the written sectors are read back to check their CRC's. Since the entry is a test entry the parameter BEGADR is not updated.

TIMING $E89E — This entry point saves a timing parameter as a function of the system clock frequency for the operating system routines. This is done automatically if the system is started at the FOSLOD point. The number for a 1 MHz clock value is equal to 3 and it is stored into the memory location TIMPAR.

ERROR CONDITIONS

The error conditions are saved in the ERSTAT output parameter. If an error occurred during a disk operation, the carry bit will be set on return to the caller, and this byte will contain a number indicating the type of error in ASCII code. If no error occurs, then the carry will be reset and this byte will contain ASCII 0. Some errors are

submitted to retrials before they are written into this location. In multi-sector operations the operation is aborted when an error occurs and the number of the sector in error can be calculated by the following formula:

PSNE = BEGSCT + NUMSCT - RESECT - 1

Where PSNE is the number of the sector in error, and RESECT is the number of sectors left until the end of the operation. Initially RESECT is set equal to NUMSCT, and it is immediately decremented at the starting point of the operation; that is why a 1 should be subtracted in addition to RESECT. The value of ERSTAT and their meaning are given below.

$31 DATA CRC ERROR: This error occurs when CRC ERROR (STRB1) is set and Sector address undetected (STRB3) is reset (see Chapter 6). It shows that the data field CRC is incorrect because of a possible miswriting or misreading. This error will occur after the last sector byte is written in memory, but the BEGADR parameter will not be updated to the next address. If the error occurred during a write test or check, data should be rewritten. The firmware will attempt to read the sector CRC five times before returning this error.

$32 WRITE PROTECTED: This status is returned when the write protect hole of the disk envelope is not covered with an opaque material. A write operation to such a diskette is not permitted by the firmware.

$33 DISK NOT READY: A possible cause for such an error include the drive unit door not being closed, and the

diskette not being up to speed.

$34 DELETED DATA MARK DETECTED: This error is set when the data field address mark contains a deleted data mark. The BEGADR parameter is not updated and the sector number can be found from the given general PSNE formula. Any operation is aborted when this error occurs, except a CRC check and WDDM.

$35 TIMEOUT: This error occurs when a bad track, or sector, or unformatted disk is read since the sector address mark is not detected. An overrun error while reading or writing may also cause a time-out to occur. The operation is retried five times.

$36 INVALID DISK ADDRESS: This error occurs when the sum of the BEGSCT and NUMSCT parameters is greater than the number of the sectors on the disk.

$37 SEEK ERROR: This error occurs if a SKTK00 is not completed on track 0, or a seek operation puts the head on a track where the ID field track address is not equal to the LTAR of the FDC.

$38 DATA MARK UNDETECTED: This error occurs when a data mark is not found after a status sense request. Data can not be written into memory and the sector is not read. The firmware attempts five times before returning this error.

$39 ADDRESS MARK CRC ERROR: When both a CRC error and a sector address undetected error occurs simultaneously, this means that the CRC of an address mark is incorrect. The firmware attempts five times, for this error to be valid.

After each disk operation is executed, a check for errors should be made and if possible, the error type, the erroneous sector or track should be displayed on the system console. Generally errors occurs because of bad formatting, the destruction of the format by magnetic fields, or because of bad handling of diskettes.

CHAPTER 10

THE OPERATING SYSTEM

OPERATING SYSTEM FEATURES

The term operating system denotes those program modules within a computer system that govern the control of equipment resources, such as main storage, I/O devices, and information. The operating system can also be viewed as a collection of utility routines, like peripheral handlers or compilers under the control of an executive often called the kernel, that supervises resource allocation (31). In multi-user systems, the main function of the operating system (OS) becomes to resolve conflicts among the users and to see that the resources are used efficiently. For these reasons, it has to determine who gets what, when and how much, allocate and reclaim the resource when needed (15).

For a single user system, which is the case for the Codos operating system used in the project, the OS merely converts the source language program into machine language, assigns storage allocations to program and data, and makes sure the user retains control over execution. The OS has also to provide an environment in which particular software packages can execute, such as a compiler, which is a program

that translates a source program in a high-level language to a corresponding machine language program, or objet program. Most compilers interact closely with the operating system, both during the compilation of source programs and the execution of the object code. During compilation the compiler needs the operating system to read the source program, print its listing and punch its output decks (15).

The OS provides a simple command language that allows the user to state what he wants done and what device or file he wants it done to. For example a disk drive and a line printer have very little in common, however, the OS lets the user to copy data to either one with the same command. Programs to perform specific functions remain on the disk until a command that calls them is entered. The OS is itself stored on a floppy disk and loaded on the system reset. That is, only required programs are loaded into the memory and executed. This technique gives the full capabilities of the operating system and lets the user to reserve the majority of the memory space for his work.

The Codos OS includes a set of utility programs and commands to manipulate data on diskette, process data, execute utility programs, activate user-developed functions. Compilers and assemblers may easily be incorporated into the OS. To manage user programs, functions to create and change files, delete files, copy files, change name of files, list all files, combine program parts, convert object code to and

from hexadecimal formats are provided by Codos, including a job control facility to enable the sequential execution of commands automatically.

FILE AND DISKETTE ORGANIZATION

A file is a set of related information that is recorded on the disk or diskette. This information can be the actual machine instructions that make up a command or program; this information can also be a textual data, object program data, or any of a number of other forms. Each file on a disk has a name. Each file created is named by the user, or by the OS commands. The file is created on the indicated drive; if the drive is not specified, the appropriate default values are used in operating the file (generally drive Ø). Existing files are unaffected by the creation of a new file. File deletion is explicitly controlled by the operator using the "DEL" command. Some commands create an intermediate work file to perform the command's function; these intermediate files are then deleted by the command as a clean-up process(5).

The operating system keeps a directory of all files of a disk. The directory is necessary to know the file's position on the disk. To access a file, only it's name is given; the OS finds the entry on the directory, gets the beginning address of the file and loads it on an empty location of the memory space. A "DIR" command is supplied to list the names of all files on a specified disk. A "DEL" command

only removes the file from the directory; actually the file resides on the disk and is not destroyed until a new write command is issued. All files can be specified with delete protection, write protection, or no protection. Write protection prevents any command from writing to that file as well as preventing deletion of the file. The "NAME" command can be used to set or change a file's protection.

A file name specification in Codos consists of three parts: A file name, a suffix, and a logical drive number. File names can be from one to eight alphanumeric characters in length, the first of which must be alphabetic. In most cases the suffix and the logical drive number are usually given appropriate default values by the various commands. Some commands, however, require the full name of the file. The suffix can be either one or two characters in length. Like file names, suffixes must begin with an upper case alphabetic character. Some suffixes have specific meanings and they are required to be so. The most frequently used one's are listed below:

.AL	Assembly listing file
.CM	Codos command file
.RO	Relocatable object file
.SA	Ascii source file (text file)
.SY	System file
.LO	Loadable object

Codos commands usually supply an appropriate default suffix when dealing with specific files. If specified the logical drive number must be separated from the file name or from the suffix by a colon (:). The equal (=) sign shows that Codos is ready to accept commands from the operator. The command line entered by the operator must indicate which command is to be executed, and the file names that may be required by the command must be specified. The options that are allowed by some commands should also be entered on the command line and separated by a semi colon (;). The command name is separated from the file names by a space and each file name is separated from each other by commas. The command line should be terminated by the "RETURN" key (5).

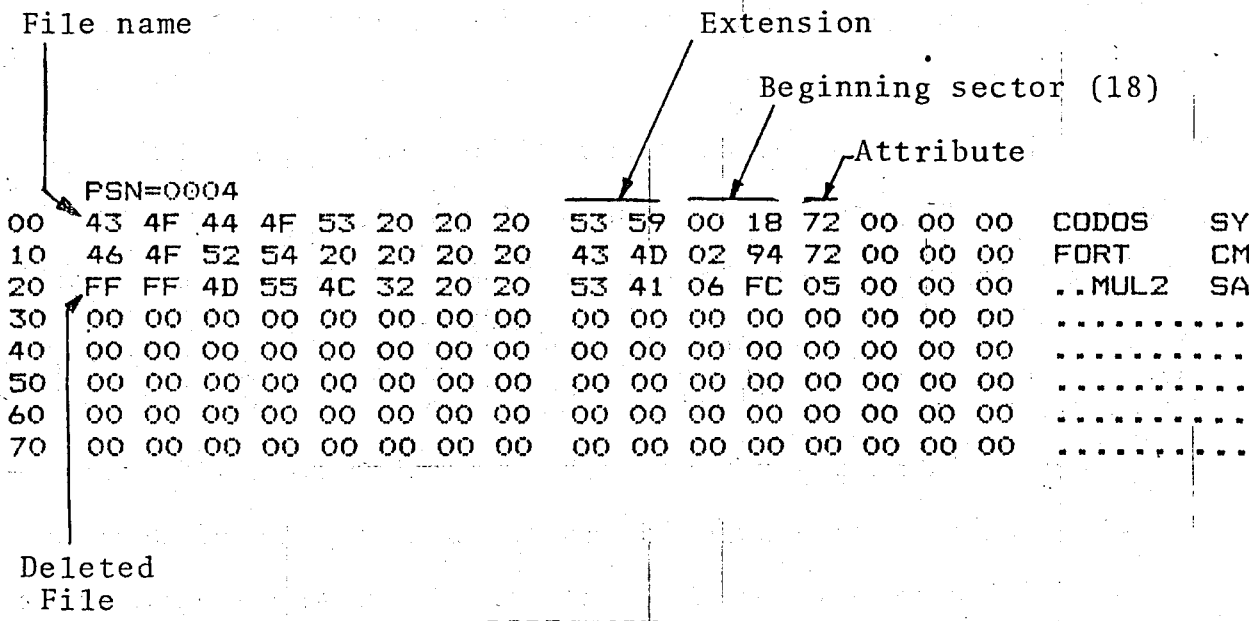
A Codos diskette has some reserved sectors, especially for the directory and system functions. In a Codos diskette the first three sectors are reserved as the identifier of the diskette. The next coming twenty sectors keeps the directory entries. Each entry uses sixteen bytes so a total of one-hundred-sixty entries are possible. If the diskette is a system diskette containing the Codos OS itself, the portion after the directory is reserved for the OS, and consists of about one-hundred sixty sectors. Then comes other files with order of entry to the diskette. New files are created by finding and reserving enough unused blocks to contain the data in the file. Two file allocation methods are currently in widespread use. The first method, partly used by Codos, allocates blocks for a file from a sequential

blocks on the disk. Unfortunately, as files are created, updated, and deleted, these free block pools become fragmented. When this fragmentation occurs, it often becomes impossible for the OS to create a file even though there is a sufficient number of free blocks on the disk. At this point, special programs must be run to squeeze or compact the disk, in order to re-create a single contiguous free-block pool.

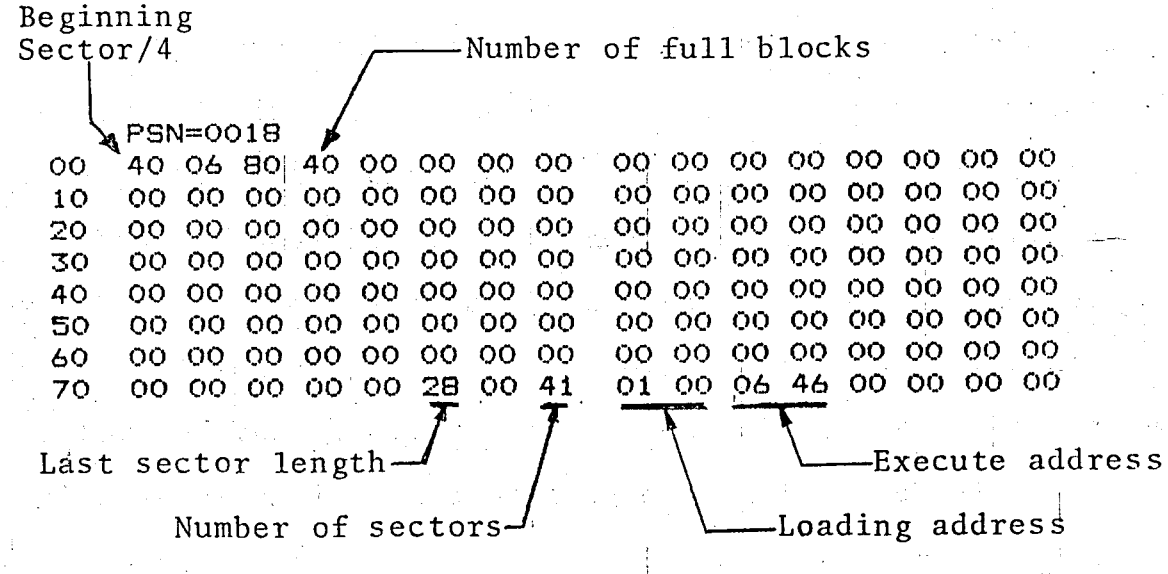
The second file allocation method uses a more flexible technique in which individual data blocks may be located anywhere on the disk. With this technique, a file directory entry contains the disk address of a file pointer block rather than the disk address of the first data block of the file. This file pointer block contains pointers for each data block in the file. Some files may require multiple pointer blocks. For this reson each pointer block contains the disk address of the following pointer block when necessary.

Although the second method uses all blocks of the diskette effectively, the read and search operations may take time since files will be spread all over the diskette. Codos, uses a combined method where each file contains a header sector which gives the last sector length, the execute address, the number of sectors to be loaded. Only the address of this sector is kept on the directory. So, the OS finds the entry, gets back the address of the header block and loads the file according to the information in the header block.

The directory entry contains only the eight character name, the two character extension, the two byte sector address of the header and the attribute of the file. If the first byte of the entry is 00 the entry has never been used; the value of FF at that location indicates that the file has been deleted ; an Ascii character indicates the presence of a file name (29).



DIRECTORY



HEADER

The attributes are saved on a single byte and three different bits are reserved; one for write protection, one for delete protection and one for system files. The beginning sector address is obtained from the first two bytes of the header by shifting them left circularly. So

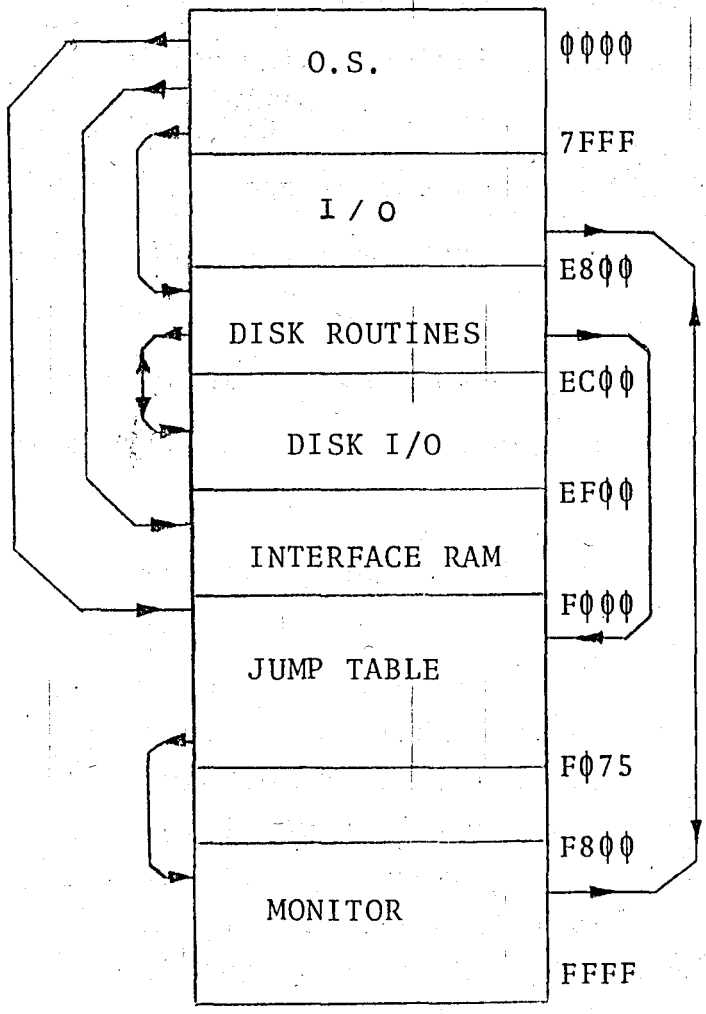
$$4\phi\phi6 = \phi1\phi\phi\phi\phi\phi\phi\phi\phi\phi\phi11\phi$$
$$= \phi\phi\phi\phi\phi\phi\phi\phi\phi\phi\phi11\phi\phi1 = 19$$

In this example the file begins just after the header.

Another method used by Codos to speed up especially directory searches for a file name, is to use a hashing function to map a file's name into one of the directory sectors. As a result, the number of sectors that has to be read before a match is found or not found is minimized. All then bytes of the file name and suffix are used by the hashing function. The function computes a number which when added to the physical sector number of the start of the directory is the sector number of the first sector used in a linear search of the directory. A directory search begins in the sector identified by the hashing function. If no entries within this sector contain zero in their first byte, and if no match is found, the next sector in the directory is searched. The sectors are searched in this round-robin fashion until a match or an entry with a first byte of zero is found, or until all sectors have been examined. The only time all sectors of the directory are searched is if every entry contains a valid file name or a deleted file name (5).

OPERATING SYSTEM RESIDENT SOFTWARE INTERFACE

The OS uses the monitor also, rather than duplicate functions that already exist there. These functions are in general the Input/Output routines to the teletype and some other monitor functions. Other functions used by the OS are of course the diskette I/O functions. These routines were explained in detail in the previous chapter. A slight difference while using the monitor routines occur, since the user (in this case the OS) can not give the direct routine address of the monitor.



INTERFACE OF SYSTEM PROGRAMS

The monitor program is subject to changes while new routines are added or updated; the monitor program itself may be changed or rearranged. With these modifications the addresses of the routines used by the OS will shift every time, and the OS would have to be updated at all times. To prevent this, a jump table is provided at locations F000 to F075 where a jump operation to the specified routine is performed. The OS calls this fixed address, jumps from this location to the monitor routine and returns from the monitor when it encounters an "RTS" instruction. In this way an update of the monitor will force the modification of the jump table but not the OS. However, in contrast to this flexibility, the disk routine beginning addresses can not be changed since a jump table for the disk I/O is not provided. Some flags or data are transferred between the OS and the monitor using the interface RAM area located at EF00-EFFF.

The most important routines used by the OS, their input and output conditions are listed below:

F000 START — Restarts the monitor by jumping to the warm start address of the monitor.

F015 INCHNP — Receives one character from the system console and masks the parity bit. A return to the caller is made only when a key is pressed from the console, and the inputted character is kept in accummulator A. The inputted character is echoed to the console if the ECOFLG (EF99) is equal to zero, otherwise it is not.

F018	OUTCH	Sends an Ascii character contained in accummulator A to the system console.
F021	PCRLF	Displays a CR and LF to the system console.
F024	PDATA	Displays a CR, LF and a string pointed by the index register, until an EOT (04) character is encountered.
F027	PDATA1	Displays a data string pointed by the index register until an EOT is encountered.

CHAPTER - 11

DISCUSSION

The disk interface software is a major contributor to the efficient and reliable operation of a floppy disk subsystem. This software must be a well-designed compromise between the needs of the application software modules, often referred to as the "logical interface level" and the capabilities of the floppy disk controller, referred to as the "physical interface level". System and application software modules often specify disk operation parameters that are not directly compatible with the FDC device. These software and hardware incompatibilities were caused and may be caused in the future by the following:

- The change from an existing FDC to a functionally, equivalent design: this caused to locate the software in a fixed area of the memory map (both the ROM and EPROM sections). This restriction was of course a consequence of the Codos OS which has already predefined the disk routine addresses. As mentioned in the previous chapter the monitor programs are not subject to the same restriction because of the jump table incorporated. This jump table has in fact a disadvantage too; it just re-

sides between the monitor and the disk routines, because again the OS restricts the jump table address.

- The upgrade of the FDC subsystem to a higher capability design: the capability of the system for that instant is four single-sided or three double-sided drives. The expansion for the future may be possible only by increasing the used pins of the PIA which is actually four.

- The expansion from a single density to double density, or from floppy to hard disk is not possible, because of the controller specifications. The restriction stands on the recording format, since all ID marks, gaps are different in a MFM recorded surface compared to the FM format. The 6843 can only handle FM data and the format is compatible with the IBM 3740 format. In fact, in many systems these parameters are never changed during the life of the system.

Besides all these restrictions and disadvantages often encountered in floppy disk systems, the system contains the capability for DMA. The DMA mode has the capability of permitting the processor to continue executing instructions while a disk transfer is in progress. This capability is especially useful in multi-programming when the operating

system is designed to permit other tasks to execute while a program is waiting for I/O. In polling or interrupt driven modes, referred to as non-DMA modes, there is the advantage of significantly lower system cost, but these modes are performance limited for double-density systems where data bytes must be transferred to/from the FDC every sixteen microseconds.

One of the largest portion of the access time in a floppy system is covered by the seek operations, especially on the head settling time which takes 12 msec when the head is loaded on a specific track. Some systems never unloads the head and performs the seek command in that position; in this way the settling time problem is removed. However, this kind of application may be hazardous for the disk surface. The 6843 never loads the head during a seek operation, but only when a read or write command is issued, and the head is raised whenever there is no other command issued. Another timing factor is the track to track seek time, which can be only minimized by the performance of the drive's stepper motor. In any case the track access time can not be specified strictly because of the initial head position. In some systems, after all operations, the head is positioned at the middle of the disk for easy access to both directions. In a system that directory searches are important at the initiation of the command, the head is positioned on the track which contains the directory, as it is for the Codos OS.

CHAPTER 12

CONCLUSION

With the design of the floppy disk controller module and the memory module the system has become more compact. Four boards constitute the microcomputer, the two other boards being the ACIA and MPU boards. These two boards may be reduced to a single board by incorporating an ACIA to the MPU board or using another M6800 family MPU such as the M6803 which contains serial and parallel I/O lines. In this case the minimum system will be only three boards but with great software capability similar to that of a development system. To reduce cost the hardware is kept minimal and especially interrupt or DMA modes are omitted and replaced by software polling and transferring methods.

Using the adopted operating system the following software applications could be realized:

- High-level and low level language programming using assemblers, compilers, cross-assemblers.

- Writing programs and debugging on the same machine by using the generated object codes of assemblers and compilers and converting them to program the

code in EPROM.

- Editing, linking files together and developing of a system software in the form of different modules.

The following hardware changes may be done to increase the user facilities:

- Interchanging the system console, which is a TTY, with a CRT device.

- Addition of an EPROM programmer module to easily program the developed software on the system. This could be a direct module or a communication module for an already existing programmer using the ACIA module hardware.

- Addition of a line printer. This will need only to write the interface software routines since a PIA module already exists to perform the hardware functions.

BIBLIOGRAPHY

1. Airpax, North American Philips Controls Corp., Stepper Motor Handbook 1979.

2. Alekar, S.V., "M6800 Program Performs Cyclic Redundancy Checks", Electronics, USA, Vol. 52, No. 25, December 6, 1979, p. 167.

3. Brinch, Hansen, P., Operating System Principles, Prentice Hall Inc., New Jersey, 1973.

4. Clarke, Kenneth K., Hess Donald T., Communication Circuit Analysis and Design, Addison-Wesley Publishing Company Inc., Second Printing, September 1978.

5. Codos Disk Operating System Through Version 5.0, Doc. Number 72771a, Second Edition, Codex Corporation.

6. Comer, David J., Modern Electronic Circuit Design, Addison-Wesley Publishing Company Inc., Second Printing, March 1977.

7. DC Motors, Speed Controls, Servo Systems, Electro-Craft Corporation, Third Edition, Pergamon Press, USA, 1977.

8. Diskette Operating System, Microcomputer Development System, MDS-DOS Hardware Reference Manual, Intel Corporation, 1975.

9. Ekinci, Firuzan, Design of M6800 Based Extendable, General Purpose Microcomputer, Boğaziçi University, September 1981.

10. Greenfield, Joseph D., Wray, William C., Using Microprocessors and Microcomputers: The 6800 Family, John Wiley and Sons Inc., Canada, 1981.

11. Intel Component Data Catalog, January 1981, Intel Corporation 1980.

12. Linear Integrated Circuits, Semiconductor Data Library, Motorola Semiconductions Inc., USA, 1976.

13. Lowman, Charles E., Magnetic Recording, McGraw-Hill Inc., USA, 1972.

14. M68SFDC, Floppy Disk Controller Module User's Guide, Second Edition, Motorola Incorporated, June 1978 First Edition.

15. Madnick, Stuart E., Donovan, John J., Operating Systems, McGraw-Hill Computer Science Series, International Student Edition, 1981, Tokyo, Japan.

16. Mahan, K. Douglas, Micro-Winchester Fits Right-in to Systems of the Future, Electronic Design, March 31, 1981.

17. MCU/MPU Applications Manual, Motorola Inc., 1982, Switzerland.

18. MEX 6816, MEX 6832, MEX 6858, MEX 6864, 16K/64 Hidden Refresh Memory Module User's Guide, Motorola Incorporated, First Edition 1978.

19. Microprocessors Applications Manual, Motorola Semiconductor Products Inc., 1975.

20. Microprocessors Data Manual 1982, Motorola Inc., Switzerland.

21. Mostek 1977 Memory Products Catalog, Mostek Corporation, USA, 1977.

22. Motorola Data Sheets:

DS 9471	MC 6800
DS 9837	MC 6843
DS 9520-R1	MC 6844
191-177/5	MC 6820
192-177/5	MC 6850

23. MTTL Complex Functions, MC 4344/4044 Frequency Phase Detector, Motorola Inc., 1977.

24. Nash Garth, Phase Locked Loop Fundamentals, AN-535, Motorola Inc., 1975.

25. National Semiconductors TTL Data Book, National Semiconductors Corp., West Germany 1978.

26. Parker, Larry A., A Floppy Disk Controller Using MC6852 SSDA and other M6800 Microprocessor Family Parts, AN-764, Motorola Inc., Semiconductor Systems Engineering.

27. Philips Data Handbook, High Frequency Transistors, Switching Transistors, Semiconductors and Integrated Circuits, Part 3, April 1976.

28. Rossi, Tom, An Intelligent Data Base System Using the 8272, Application Note AP-116, Intel Corporation, March 1981.

29. Rossi, Tom, Software Design and Implementation of Floppy Disk Subsystems, Application Note AP-121, Intel Corporation, June 1981.

30. Siemens, OEM Floppy Disk Drive FDD 200-8, Technical Manual, Volume 1, Model 200-8P.

31. Shindler, Max, "Microcomputer Operating Systems Branch into Mainframe Territory", Electronic Design, March 19, USA, 1981.

32. Taub, H., Schilling, D., Digital Integrated Electronics, International Student Edition, McGraw-Hill Inc., Kosaido Printing Co. Ltd., Tokyo, Japan, 1977.

33. The European CMOS Selection, Technical Information Center, Motorola Inc., 1979, Switzerland.

34. The IBM Diskette for Standard Data Interchange,
IBM Document GA 21-9182-0
IBM Document FA 21-9176-0.

35. Weissberger, Alan J., "Control Chip Handles Error Checking and Character Based Protocols Easily", Electronics, USA, Vol. 53, No. 7, March 27, 1980.

36. Williams, Tom, "Daisy Chains Put Winchester Drives into Small Systems", Electronics Design, March 5, 1981.

37. Zarella, J. "Operating Systems: Concepts and Principles", Microcomputer Applications, California, 1979.

APPENDICES

A P P E N D I X A

1.	Components	42,000.-
2.	Film and Printed Circuit Costs	100,000.-
3.	Disk Drive (Approximate Cost; two are needed for the full system)	500,000.-
4.	Engineering Cost	600,000.-
5.	Miscellaneous	10,000.-
	TOTAL	1,252,000.-

A P P E N D I X B

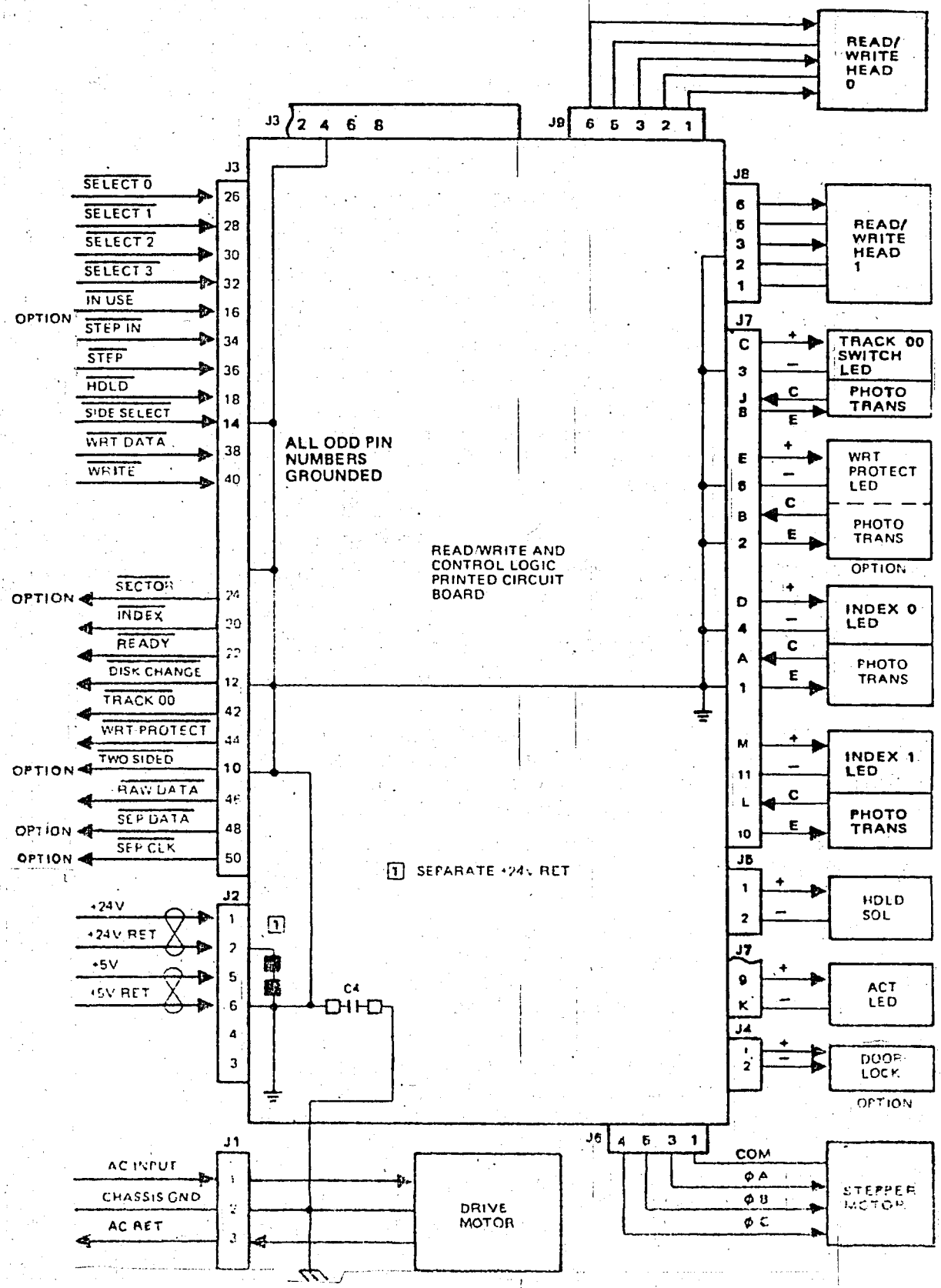
Function	Characteristics	
	Single-Density	Double-Density
Disk Type	ANSI Standard	ANSI Standard
Storage Capacity (Unformatted)		
Per Disk	6.4 megabits	12.8 megabits
Per Data Surface	3.2 megabits	6.4 megabits
Per Track	41.7 kilobits	83.4 kilobits
Tracks	154	154
Track Density	48 Tracks Per Inch	48 Tracks Per Inch
Recording Density		
Track 00 (Outside)	1836 bpi (3672 fci)	3672 bpi (3672 fci)
Track 76 (Inside)	3268 bpi (6536 fci)	6536 bpi (6536 fci)
Recording Method	FM	MFM
Rotational Speed	360 rpm ±2.5%	360 rpm ±2.5%
Rotational Latency		
Average	83.33 milliseconds	83.33 milliseconds
Maximum	175.6 milliseconds	175.6 milliseconds
Access Time		
Track-to-Track	3-6 milliseconds	3-6 milliseconds
Track 0 - Track 76	228 milliseconds	228 milliseconds
38-Track Move	111 milliseconds	111 milliseconds
Settling Time	12 milliseconds	12 milliseconds
Head Engage Time	25 milliseconds	25 milliseconds
Data Transfer Rate	250 kilobits/sec	500 kilobits/sec
Erase/Write Recovery Time		
Tunnel-Erase	580 microseconds (req'd for read to stabilize after write completed)	580 microseconds (req'd for read to stabilize after write completed)
Straddle-Erase	50 microseconds	50 microseconds
Read/Write Head	Single-gap with tunnel-erase or straddle-erase	

Function	Characteristics
	Single-Density / Double-Density
Read/Write-to-Erase Gap Spacing	0.035 inch (Tunnel Erase)
Track Width	0.013 inch
Erase Width	0.006 inch (on either side of track)
Spacing Between Tracks	0.02083 inch
Track Centerline Radius	$2.029 + \frac{76 - N}{48}$ inch Side "0" $1.9457 + \frac{76 - N}{48}$ inch Side "1" where N = track number
Logic Levels Disk Drive	Logical 1 (True) = +2.5V to +5.5V Logical 0 (False) = 0.0V to +0.4V
Interface	Logical 1 (True) = 0.0V to +0.4V Logical 0 (False) = +2.5V to +5.5V
AC Input Power Standard	115V (90-127V) 60 Hz ± 0.5 Hz
Optional	115V (90-127V) 50 Hz ± 0.5 Hz 230V (180 to 253V) 60 Hz ± 0.5 Hz 230V (180 to 253V) 50 Hz ± 0.5 Hz
Voltage Dropout	100%, 10 milliseconds once each 600 seconds
Motor Current (Max) Start	0.8 ampere for 115 volts AC 0.4 amperes for 230 volts AC
Run	0.4 amperes for 115 volts AC 0.2 amperes for 230 volts AC
DC Input Power	+24 volts ± 5%, 2.0 amperes maximum* +5 volts ± 5%, 1.0 amperes maximum

*Nominally 1.4 amperes

Function	Characteristics	
Reliability		
MTBF	6000 hours (after initial 200 hours)	
MTTR	Less than 20 minutes	
Read Errors		
Recoverable	Less than 1 in 10^9	
Non-recoverable (after 10 tries)	Less than 1 in 10^{12}	
	Operating	Non-Operating
Environmental		
Temperature	40° to 115°F (4.4° to 46.1°C)	32° to 150°F (0° to 65°C)
Relative Humidity	20% to 80% without condensation	5% to 90%, without condensation
Altitude	-1000 to +10,000 feet	-1000 to +45,000 feet
Heat Dissipation	300 BTU/Hour (worst case)	NA
Dimensions and Weight		
Dimensions	See Figure 1-6	
Weight	12.5 pounds	

APPENDIX C


READ/ WRITE HEAD 0
J3 2 4 6 8
J9 6 5 3 2 1
J3
SELECT 0
SELECT 1
SELECT 2
SELECT 3
OPTION IN USE
STEP IN
STEP
HDLD
SIDE SELECT
WRT DATA
WRITE
26
28
30
32
16
34
36
18
14
38
40
ALL ODD PIN NUMBERS GROUNDED
READ/WRITE AND CONTROL LOGIC PRINTED CIRCUIT BOARD
OPTION SECTOR
INDEX
READY
DISK CHANGE
TRACK 00
WRT PROTECT
OPTION TWO SIDED
RAW DATA
OPTION SEP DATA
OPTION SEP CLK
24
20
22
12
42
44
10
46
48
50
1 SEPARATE +24V RET
J2
+24V
+24V RET
+5V
+5V RET
1
2
5
6
4
3
C4
J8
6 5 3 2 1
READ/ WRITE HEAD 1
J7
C 3 J 8
TRACK 00 SWITCH LED
PHOTO TRANS
E 8 B 2
WRT PROTECT LED
PHOTO TRANS
OPTION
D 4 A 1
INDEX 0 LED
PHOTO TRANS
M 11 L 10
INDEX 1 LED
PHOTO TRANS
J5
1 2
HDLD SOL
J7
9 K
ACT LED
J4
1 2
DOOR LOCK
OPTION
J6 4 5 3 1
COM
ΦA
ΦB
ΦC
STEPPER MOTOR
J1
AC INPUT
CHASSIS GND
AC RET
1
2
3
DRIVE MOTOR

A P P E N D I X D

PIN NUMBER	SIGNAL MNEMONICS	SIGNAL NAME AND DESCRIPTION
1	+5 VDC	+5 Vdc Power - Output voltage supplied to Disk Drive Unit for logic circuits.
3	+5 VDC	+5 Vdc Power - Output voltage supplied to Disk drive Unit for logic circuits.
5	$\overline{\text{DIRECTION}}$	DIRECTION - This signal output is used in conjunction with the STEP signal to move the recording head from track to track. When this signal is a logic low level, the recording head is moved to the lower numbered tracks (out). When this signal is a logic high level, the head moves to the higher numbered tracks (in). This signal must remain in the desired logic state during the duration of the STEP signal.
7	$\overline{\text{STEP}}$	STEP - This signal output is used in conjunction with the DIRECTION signal to move the recording head from track to track. A logic low level pulse causes the head to be moved one track (step) in the direction indicated by the DIRECTION signal.
9	D.S.	DOUBLE-SIDED INDICATION (damage may result if connected to other systems).
11,13		Not used.
15	$\overline{\text{SELECT 0}}$	SELECT 0 - The logic low level output signal used to select disk drive 0.
17	$\overline{\text{TRK 00}}$	TRACK 00 - Low level input signal used to indicate when the recording head is positioned over track 00. When low, the stepper motor drive circuits are inhibited to prevent further outward movement of the head.
19	$\overline{\text{HEAD LOAD}}$	HEAD LOAD - This low level output signal is used to position the flexible diskette against the recording head.
21	$\overline{\text{READY 0}}$	READY 0 - This low level input signal is used to indicate that the flexible diskette is inserted correctly into disk drive 0, and that the dc voltage levels and disk speed in this drive are correct.
23	$\overline{>\text{TRK 43}}$	ABOVE TRACK 43 - This output signal is used to control the amplitude of the write current in the recording head. When recording on tracks 0 thru 43, this signal must be a high level. Conversely, when recording on tracks 44 thru 76, this signal must be a low level.
25	$\overline{\text{INDEX}}$	INDEX - A low level input pulse used to indicate the beginning of a track. This pulse occurs once per revolution of the diskette.
27	$\overline{\text{HEAD 1}}$	HEAD 1

PIN UMBER	SIGNAL MNEMONICS	SIGNAL NAME AND DESCRIPTION
29	$\overline{\text{WRITE ENABLED}}$	WRITE ENABLED - A low level input signal used to indicate that writing may take place.
31	$\overline{\text{WRITE DATA}}$	WRITE DATA - This output signal consists of low level pulses representing data to be recorded on the flexible diskette. Write current reverses direction on the leading edge of each pulse.
33	$\overline{\text{EN WRITE}}$	ENABLE WRITE - A low level output signal used to enable recording of data on the flexible diskette. When this signal is a high level, reading data from the flexible disk is enabled.
35	$\overline{\text{SEL 2}}$	SELECT 2
37	$\overline{\text{RAW DATA}}$	RAW DATA - This input signal contains the unseparated data read from the flexible diskette.
39	$\overline{\text{SELECT 1}}$	SELECT 1 - A low output signal used to select disk drive 1.
2-40	GND	GROUND (all even numbered pins are GROUND)

A P P E N D I X E

PIN NUMBER	SIGNAL MNEMONICS	SIGNAL NAME AND DESCRIPTION
A,B,C	+5 V	+5 Vdc - Used for the module's logic circuits.
D	$\overline{IRQ}$	INTERRUPT REQUEST - This low level sensitive input signal requests that an MPU interrupt sequence be generated within the machine. The MPU will wait until it completes the instruction being executed before it recognizes the request. At that time, if the interrupt mask bit in the MPU condition code register is not set, the MPU will begin the interrupt sequence.
E	$\overline{NMI}$	NON-MASKABLE INTERRUPT - This low-going, edge-sensitive input signal requests that a non-maskable interrupt be generated within the machine. The MPU will wait until it completes the instruction being executed before it recognizes the $\overline{NMI}$ signal. At that time, the MPU will begin its non-maskable interrupt sequence.
F	VMA	VALID MEMORY ADDRESS
H		Not used.
J	Ø2	Phase 2 clock signal.
K		Not used.
L	MEM CLK	MEMORY CLOCK - This signal is an ungated TTL level Ø2 clock used to refresh any dynamic memory used in the system.
M	-12 V	-12 Vdc
N	$\overline{TSC}$	TRI-STATE CONTROL - This active-low input signal works with MEM CLK to generate TSC GNT.
P	BA	BUS AVAILABLE - This output signal will normally be a low level. When activated, it will go high to indicate that the MPU has halted and the address bus is available. This will occur when the Halt line is in the Halt (low) state or the MPU is in the Wait state as a result of executing a Wait instruction. At such time, all the MPU three-state output drivers will go to their off (high impedance) state, and other outputs to their normally inactive state. A maskable interrupt or non-maskable interrupt removes the MPU from the Wait state.
R	$\overline{MEM\ RDY}$	MEMORY READY - This signal permits Monoboard Microcomputer 1D to work with slow memory modules. When this command is a low level signal, the Ø1 and Ø2 clock signals are stretched with the Ø2 clock held high. The Ø2 clock signal remains at a high level until the initiating module completes its memory operation. When completed, the initiating module returns MEM RDY to a high level, enabling the clock to again generate the Ø1 and Ø2 clock signals.

PIN NUMBER	SIGNAL MNEMONICS	SIGNAL NAME AND DESCRIPTION
T	+12 V	+12 Vdc
U-Z		Not used.
$\overline{A}$-$\overline{F}$		Not used.
$\overline{H}$	$\overline{D3}$	DATA bus (bit 3) - One of 8 bi-directional data lines used to provide a two-way transfer between Monoboard Microcomputer 1D and any other plug-in modules in the system.
$\overline{J}$	$\overline{D7}$	DATA bus (bit 7) - Same as $\overline{D3}$ on C1-$\overline{H}$.
$\overline{K}$	$\overline{D2}$	DATA bus (bit 2) - Same as $\overline{D3}$ on C1-$\overline{H}$.
$\overline{L}$	$\overline{D6}$	DATA bus (bit 6) - Same as $\overline{D3}$ on C1-$\overline{H}$.
$\overline{M}$	A14	ADDRESS bus (bit 14) - One of 16 address lines used to select a memory location either on this module or external to this module.
$\overline{N}$	A13	ADDRESS bus (bit 13) - Same as A14 on C1-$\overline{M}$.
$\overline{P}$	A10	ADDRESS bus (bit 10) - Same as A14 on C1-$\overline{M}$.
$\overline{R}$	A9	ADDRESS bus (bit 9) - Same as A14 on C1-$\overline{M}$.
$\overline{S}$	A6	ADDRESS bus (bit 6) - Same as A14 on C1-$\overline{M}$.
$\overline{T}$	A5	ADDRESS bus (bit 5) - Same as A14 on C1-$\overline{M}$.
$\overline{U}$	A2	ADDRESS bus (bit 2) - Same as A14 on C1-$\overline{M}$.
$\overline{V}$	A1	ADDRESS bus (bit 1) - Same as A14 on C1-$\overline{M}$.
$\overline{W}$,$\overline{X}$,$\overline{Y}$	GND	GROUND
1,2,3	+5 V	+5 Vdc - Used for module's logic circuits.
4	$\overline{HALT}$	HALT - When this input is in the high state, the MPU will fetch the instruction addressed by the program counter, and start execution. When low, all activity in the MPU will be halted. This input is level sensitive. In the Halt mode, the MPU will stop at the end of an instruction, Bus Available will be at a high level, Valid Memory Address will be at a low level, and all other three-state lines will be in their off or high impedance state. The Halt line must go low with the leading edge of the phase one clock (Ø1) to ensure single instruction operation. If the Halt line does not go low with the leading edge of the phase one clock, one or two instruction operations may result, depending on when the halt line goes low relative to the phasing of the clock.
5	$\overline{RESET}$	RESET - This buffered input/output signal, when used as an output, provides a reset output to any external devices requiring a $\overline{RESET}$ signal. An external switch closure to ground may be applied to this pin to manually restart Monoboard Microcomputer 1D.

PIN NUMBER	SIGNAL MNEMONICS	SIGNAL NAME AND DESCRIPTION
6	R/W	READ/WRITE - This MPU output signal indicates to external plug-in modules whether the MPU is performing a memory read (high) or write (low) operation. The normal standby state of this signal is read (high). Also, when the MPU is halted, this signal will be in the read state.
7,8,9		Not used.
10	VUA	VALID USER'S ADDRESS
11	-12 V	-12 Vdc - Same as -12 Vdc on C1-M.
12	$\overline{RR}$	REFRESH REQUEST - This input signal, when low, initiates a memory refresh cycle of the dynamic memory modules. During the refresh operation, the clock circuit is inhibited from generating its Ø1 and Ø2 clock signals, but still generates the MEM CLK signal.
13	RG	REFRESH GRANT - This output signal, when high, instructs the dynamic memory modules to refresh their memories.
14		Not used.
15	TSC GNT	TRI-STATE GRANT - This signal, when high, causes BA to be low. Then the device requesting the bus takes control of VMA/VUA.
16	+12 V	+12 Vdc - Same as +12 Vdc on C1-T.
17,18		Not used.
19	VXA	VALID EXECUTIVE ADDRESS
20-28		Not used.
29	$\overline{D1}$	DATA bus (bit 1) - Same as $\overline{D3}$ on C1-$\overline{H}$.
30	$\overline{D5}$	DATA bus (bit 5) - Same as $\overline{D3}$ on C1-$\overline{H}$.
31	$\overline{D0}$	DATA bus (bit 0) - Same as $\overline{D3}$ on C1-$\overline{H}$.
32	$\overline{D4}$	DATA bus (bit 4) - Same as D3 on C1-$\overline{H}$.
33	A15	ADDRESS bus (bit 15) - Same as A14 on C1-$\overline{M}$.
34	A12	ADDRESS bus (bit 12) - Same as A14 on C1-$\overline{M}$.
35	A11	ADDRESS bus (bit 11) - Same as A14 on C1-$\overline{M}$.
36	A8	ADDRESS bus (bit 8) - Same as A14 on C1-$\overline{M}$.
37	A7	ADDRESS bus (bit 7) - Same as A14 on C1-$\overline{M}$.
38	A4	ADDRESS bus (bit 4) - Same as A14 on C1-$\overline{M}$.
39	A3	ADDRESS bus (bit 3) - Same as A14 on C1-$\overline{M}$.
40	A0	ADDRESS bus (bit 0) - Same as A14 on C1-$\overline{M}$.
41,42,43	GND	GROUND

A P P E N D I X F

OPERATIONS	MNEMONIC	IMMED OP	IMMED ~	IMMED =	DIRECT OP	DIRECT ~	DIRECT =	INDEX OP	INDEX ~	INDEX =	EXTND OP	EXTND ~	EXTND =	IMPLIED OP	IMPLIED ~	IMPLIED =	BOOLEAN/ARITHMETIC OPERATION (All register labels refer to contents)	5 H	4 I	3 N	2 Z	1 V	0 C
Add	ADDA	3B	2	2	9B	3	2	AB	5	2	BB	4	3				A + M → A	↕	•	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	5	2	FB	4	3				B + M → B	↕	•	↕	↕	↕	↕
Add Acmltrs	ABA													1B	2	1	A + B → A	↕	•	↕	↕	↕	↕
Add with Carry	ADCA	89	2	2	99	3	2	A9	5	2	B9	4	3				A + M + C → A	↕	•	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	5	2	F9	4	3				B + M + C → B	↕	•	↕	↕	↕	↕
And	ANDA	84	2	2	94	3	2	A4	5	2	B4	4	3				A · M → A	•	•	↕	↕	R	•
	ANDB	C4	2	2	D4	3	2	E4	5	2	F4	4	3				B · M → B	•	•	↕	↕	R	•
Bit Test	BITA	85	2	2	95	3	2	A5	5	2	B5	4	3				A · M	•	•	↕	↕	R	•
	BITB	C5	2	2	D5	3	2	E5	5	2	F5	4	3				B · M	•	•	↕	↕	R	•
Clear	CLR							6F	7	2	7F	6	3				00 → M	•	•	R	S	R	R
	CLRA													4F	2	1	00 → A	•	•	R	S	R	R
	CLRB													5F	2	1	00 → B	•	•	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	5	2	B1	4	3				A − M	•	•	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	5	2	F1	4	3				B − M	•	•	↕	↕	↕	↕
Compare Acmltrs	CBA													11	2	1	A − B	•	•	↕	↕	↕	↕
Complement, 1's	COM							63	7	2	73	6	3				M̄ → M	•	•	↕	↕	R	S
	COMA													43	2	1	Ā → A	•	•	↕	↕	R	S
	COMB													53	2	1	B̄ → B	•	•	↕	↕	R	S
Complement, 2's (Negate)	NEG							60	7	2	70	6	3				00 − M → M	•	•	↕	↕	①	②
	NEGA													40	2	1	00 − A → A	•	•	↕	↕	①	②
	NEGB													50	2	1	00 − B → B	•	•	↕	↕	①	②
Decimal Adjust, A	DAA													19	2	1	Converts Binary Add. of BCD Characters into BCD Format	•	•	↕	↕	↕	③
Decrement	DEC							6A	7	2	7A	6	3				M − 1 → M	•	•	↕	↕	④	•
	DECA													4A	2	1	A − 1 → A	•	•	↕	↕	④	•
	DECB													5A	2	1	B − 1 → B	•	•	↕	↕	④	•
Exclusive OR	EORA	88	2	2	98	3	2	A8	5	2	B8	4	3				A⊕M → A	•	•	↕	↕	R	•
	EORB	C8	2	2	D8	3	2	E8	5	2	F8	4	3				B⊕M → B	•	•	↕	↕	R	•
Increment	INC							6C	7	2	7C	6	3				M + 1 → M	•	•	↕	↕	⑤	•
	INCA													4C	2	1	A + 1 → A	•	•	↕	↕	⑤	•
	INCB													5C	2	1	B + 1 → B	•	•	↕	↕	⑤	•
Load Acmltr	LDAA	86	2	2	96	3	2	A6	5	2	B6	4	3				M → A	•	•	↕	↕	R	•
	LDAB	C6	2	2	D6	3	2	E6	5	2	F6	4	3				M → B	•	•	↕	↕	R	•
Or, Inclusive	ORAA	8A	2	2	9A	3	2	AA	5	2	BA	4	3				A + M → A	•	•	↕	↕	R	•
	ORAB	CA	2	2	DA	3	2	EA	5	2	FA	4	3				B + M → B	•	•	↕	↕	R	•
Push Data	PSHA													36	4	1	A → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
	PSHB													37	4	1	B → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
Pull Data	PULA													32	4	1	SP + 1 → SP, M_{SP} → A	•	•	•	•	•	•
	PULB													33	4	1	SP + 1 → SP, M_{SP} → B	•	•	•	•	•	•
Rotate Left	ROL							69	7	2	79	6	3				M	•	•	↕	↕	⑥	↕
	ROLA													49	2	1	A	•	•	↕	↕	⑥	↕
	ROLB													59	2	1	B	•	•	↕	↕	⑥	↕
Rotate Right	ROR							66	7	2	76	6	3				M	•	•	↕	↕	⑥	↕
	RORA													46	2	1	A	•	•	↕	↕	⑥	↕
	RORB													56	2	1	B	•	•	↕	↕	⑥	↕
Shift Left, Arithmetic	ASL							68	7	2	78	6	3				M	•	•	↕	↕	⑥	↕
	ASLA													48	2	1	A	•	•	↕	↕	⑥	↕
	ASLB													58	2	1	B	•	•	↕	↕	⑥	↕
Shift Right, Arithmetic	ASR							67	7	2	77	6	3				M	•	•	↕	↕	⑥	↕
	ASRA													47	2	1	A	•	•	↕	↕	⑥	↕
	ASRB													57	2	1	B	•	•	↕	↕	⑥	↕
Shift Right, Logic	LSR							64	7	2	74	6	3				M	•	•	R	↕	⑥	↕
	LSRA													44	2	1	A	•	•	R	↕	⑥	↕
	LSRB													54	2	1	B	•	•	R	↕	⑥	↕
Store Acmltr.	STAA				97	4	2	A7	6	2	B7	5	3				A → M	•	•	↕	↕	R	•
	STAB				D7	4	2	E7	6	2	F7	5	3				B → M	•	•	↕	↕	R	•
Subtract	SUBA	80	2	2	90	3	2	A0	5	2	B0	4	3				A − M → A	•	•	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	5	2	F0	4	3				B − M → B	•	•	↕	↕	↕	↕
Subtract Acmltrs.	SBA													10	2	1	A − B → A	•	•	↕	↕	↕	↕
Subtr. with Carry	SBCA	82	2	2	92	3	2	A2	5	2	B2	4	3				A − M − C → A	•	•	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	5	2	F2	4	3				B − M − C → B	•	•	↕	↕	↕	↕
Transfer Acmltrs	TAB													16	2	1	A → B	•	•	↕	↕	R	•
	TBA													17	2	1	B → A	•	•	↕	↕	R	•
Test, Zero or Minus	TST							6D	7	2	7D	6	3				M − 00	•	•	↕	↕	R	R
	TSTA													4D	2	1	A − 00	•	•	↕	↕	R	R
	TSTB													5D	2	1	B − 00	•	•	↕	↕	R	R
																		H	I	N	Z	V	C

ADDRESSING MODES

COND. CODE REG.

LEGEND:

- OP Operation Code (Hexadecimal);
- ~ Number of MPU Cycles;
- # Number of Program Bytes;
- + Arithmetic Plus;
- − Arithmetic Minus;
- · Boolean AND;
- M_{SP} Contents of memory location pointed to be Stack Pointer;
- + Boolean Inclusive OR;
- ⊕ Boolean Exclusive OR;
- M Complement of M;
- → Transfer Into;
- 0 Bit = Zero;
- 00 Byte = Zero;

Note — Accumulator addressing mode instructions are included in the column for IMPLIED addressing

CONDITION CODE SYMBOLS:

- H Half-carry from bit 3;
- I Interrupt mask
- N Negative (sign bit)
- Z Zero (byte)
- V Overflow, 2's complement
- C Carry from bit 7
- R Reset Always
- S Set Always
- ↕ Test and set if true, cleared otherwise
- • Not Affected

JUMP AND BRANCH INSTRUCTIONS

OPERATIONS	MNEMONIC	RELATIVE OP	RELATIVE ~	RELATIVE #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	IMPLIED OP	IMPLIED ~	IMPLIED #	BRANCH TEST	COND. CODE REG. 5 H	4 I	3 N	2 Z	1 V
Branch Always	BRA	20	4	2										None	•	•	•	•	•
Branch If Carry Clear	BCC	24	4	2										C = 0	•	•	•	•	•
Branch If Carry Set	BCS	25	4	2										C = 1	•	•	•	•	•
Branch If = Zero	BEQ	27	4	2										Z = 1	•	•	•	•	•
Branch If ≥ Zero	BGE	2C	4	2										N ⊕ V = 0	•	•	•	•	•
Branch If > Zero	BGT	2E	4	2										Z + (N ⊕ V) = 0	•	•	•	•	•
Branch If Higher	BHI	22	4	2										C + Z = 0	•	•	•	•	•
Branch If ≤ Zero	BLE	2F	4	2										Z + (N ⊕ V) = 1	•	•	•	•	•
Branch If Lower Or Same	BLS	23	4	2										C + Z = 1	•	•	•	•	•
Branch If < Zero	BLT	2D	4	2										N ⊕ V = 1	•	•	•	•	•
Branch If Minus	BMI	2B	4	2										N = 1	•	•	•	•	•
Branch If Not Equal Zero	BNE	26	4	2										Z = 0	•	•	•	•	•
Branch If Overflow Clear	BVC	28	4	2										V = 0	•	•	•	•	•
Branch If Overflow Set	BVS	29	4	2										V = 1	•	•	•	•	•
Branch If Plus	BPL	2A	4	2										N = 0	•	•	•	•	•
Branch To Subroutine	BSR	8D	8	2											•	•	•	•	•
Jump	JMP				6E	4	2	7E	3	3				See Special Operations	•	•	•	•	•
Jump To Subroutine	JSR				AD	8	2	BD	9	3					•	•	•	•	•
No Operation	NOP										01	2	1	Advances Prog. Cntr. Only	•	•	•	•	•
Return From Interrupt	RTI										3B	10	1		①				
Return From Subroutine	RTS										39	5	1		•	•	•	•	•
Software Interrupt	SWI										3F	12	1	See Special Operations	•	•	•	•	•
Wait for Interrupt*	WAI										3E	9	1		•	②	•	•	•

*WAI puts Address Bus, R/W, and Data Bus in the three-state mode while VMA is held low.

① (All) Load Condition Code Register from Stack. (See Special Operations)
② (Bit 1) Set when interrupt occurs. If previously set, a Non Maskable Interrupt is required to exit the wait state.

INDEX REGISTER AND STACK POINTER INSTRUCTIONS

POINTER OPERATIONS	MNEMONIC	IMMED OP	IMMED ~	IMMED #	DIRECT OP	DIRECT ~	DIRECT #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	IMPLIED OP	IMPLIED ~	IMPLIED #	BOOLEAN/ARITHMETIC OPERATION	COND. CODE R 5 H	4 I	3 N	2 Z	1 V
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	6	2	BC	5	3				$X_H - M, X_L - (M + 1)$	•	•	①	↕	②
Decrement Index Reg	DEX													09	4	1	$X - 1 \rightarrow X$	•	•	•	↕	•
Decrement Stack Pntr	DES													34	4	1	$SP - 1 \rightarrow SP$	•	•	•	•	•
Increment Index Reg	INX													08	4	1	$X + 1 \rightarrow X$	•	•	•	↕	•
Increment Stack Pntr	INS													31	4	1	$SP + 1 \rightarrow SP$	•	•	•	•	•
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	6	2	FE	5	3				$M \rightarrow X_H, (M + 1) \rightarrow X_L$	•	•	③	↕	R
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	6	2	BE	5	3				$M \rightarrow SP_H, (M + 1) \rightarrow SP_L$	•	•	③	↕	R
Store Index Reg	STX				DF	5	2	EF	7	2	FF	6	3				$X_H \rightarrow M, X_L \rightarrow (M + 1)$	•	•	③	↕	R
Store Stack Pntr	STS				9F	5	2	AF	7	2	BF	6	3				$SP_H \rightarrow M, SP_L \rightarrow (M + 1)$	•	•	③	↕	R
Indx Reg → Stack Pntr	TXS													35	4	1	$X - 1 \rightarrow SP$	•	•	•	•	•
Stack Pntr → Indx Reg	TSX													30	4	1	$SP + 1 \rightarrow X$	•	•	•	•	•

① (Bit N) Test: Sign bit of most significant (MS) byte of result = 1?
② (Bit V) Test: 2's complement overflow from subtraction of ms bytes?
③ (Bit N) Test: Result less than zero? (Bit 15 = 1)

CONDITION CODE REGISTER INSTRUCTIONS

OPERATIONS	MNEMONIC	IMPLIED OP	IMPLIED ~	IMPLIED #	BOOLEAN OPERATION	COND. CODE REG. 5 H	4 I	3 N	2 Z	1 V	0 C
Clear Carry	CLC	0C	2	1	$0 \rightarrow C$	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	2	1	$0 \rightarrow I$	•	R	•	•	•	•
Clear Overflow	CLV	0A	2	1	$0 \rightarrow V$	•	•	•	•	R	•
Set Carry	SEC	0D	2	1	$1 \rightarrow C$	•	•	•	•	•	S
Set Interrupt Mask	SEI	0F	2	1	$1 \rightarrow I$	•	S	•	•	•	•
Set Overflow	SEV	0B	2	1	$1 \rightarrow V$	•	•	•	•	S	•
Acmltr A → CCR	TAP	06	2	1	$A \rightarrow CCR$	①					
CCR → Acmltr A	TPA	07	2	1	$CCR \rightarrow A$	•	•	•	•	•	•

R = Reset
S = Set
• = Not affected

① (ALL) Set according to the contents of Accumulator A.

A P P E N D I X G

PROGRAMMING REFERENCE DATA

Table 7 is a summary of the information in the data sheet and can be used as a reference when programming the MC6843.

Register	Hex Address	R/W Mode	Data Bits							
DOR	0	WO	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
			8 Bits of Data Used for a Disk Write Operation							
DIR	0	RO	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
			8 Bits of Data Used for a Disk Read Operation							
CTAR	1	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
			Not Used	7 Bit Track Address of Current Head Position						
CMR	2	WO	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
			Function Interrupt Mask	ISR3 Interrupt Mask	DMA Flag	FWF	Macro Command			
MSR	2	RO	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
			Not Used				STRB	Status Sense Request*	Settling Time Complete*	Macro Command Complete*
SUR	3	WO	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
			Track to Track Seek Time				Head Settling Time			
STRA	3	RO	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
			Busy*	Index	Track Not Equal*	Write Protect	Track Zero	Drive Ready	Delete Data Mark Detected*	Data Transfer Request*
SAR	4	WO	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
			Not Used			5 Bit Sector Address				
STRB	4	RO	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
			Hard Error*	Write Error*	File Inoperable	Seek Error*	Sector Address Undetected*	Data Mark Undetected*	CRC Error*	Data Transfer*
SCR	5	WO	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
			Not Used	7 Bit Count for Track Number on SEK or Sector Count for MSR or MSW.						
CCR	6	WO	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
			Not Used						Shift CRC	CRC Enable
STAR	7	WO	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
			Not Used	7 Bit Search Track Address						

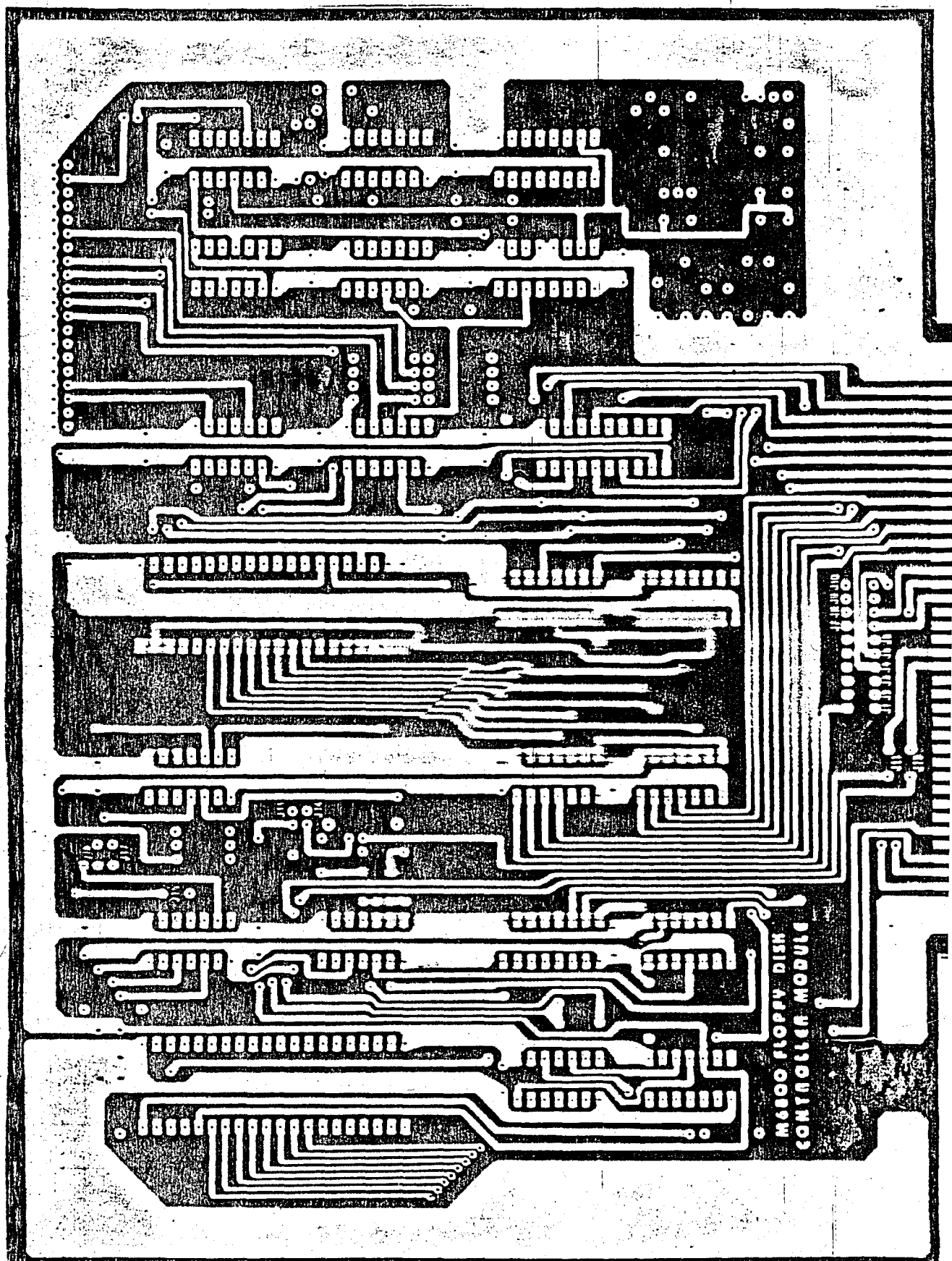
RO – Read Only
WO – Write Only
R/W – Read/Write

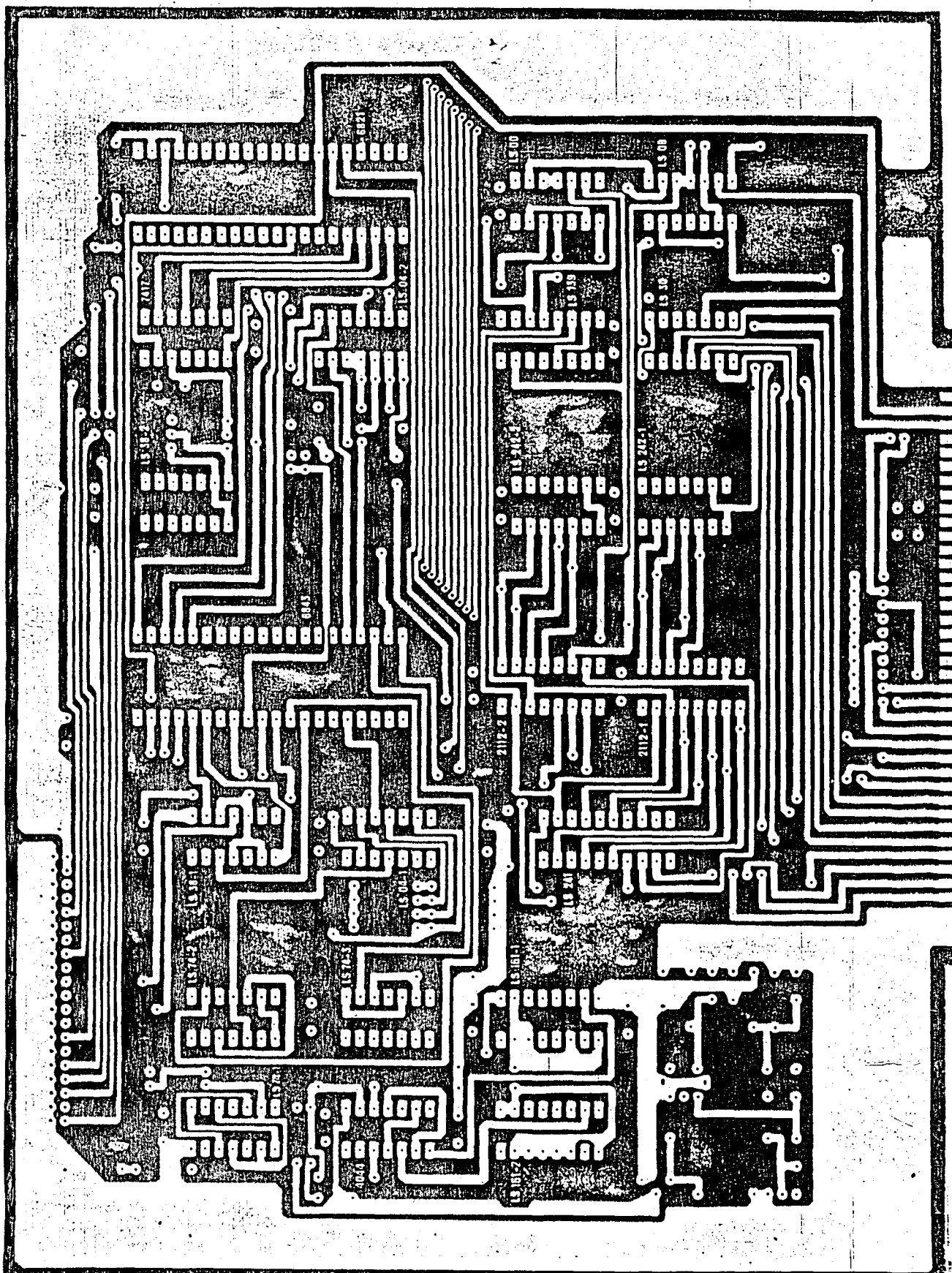
*Cleared by Reset

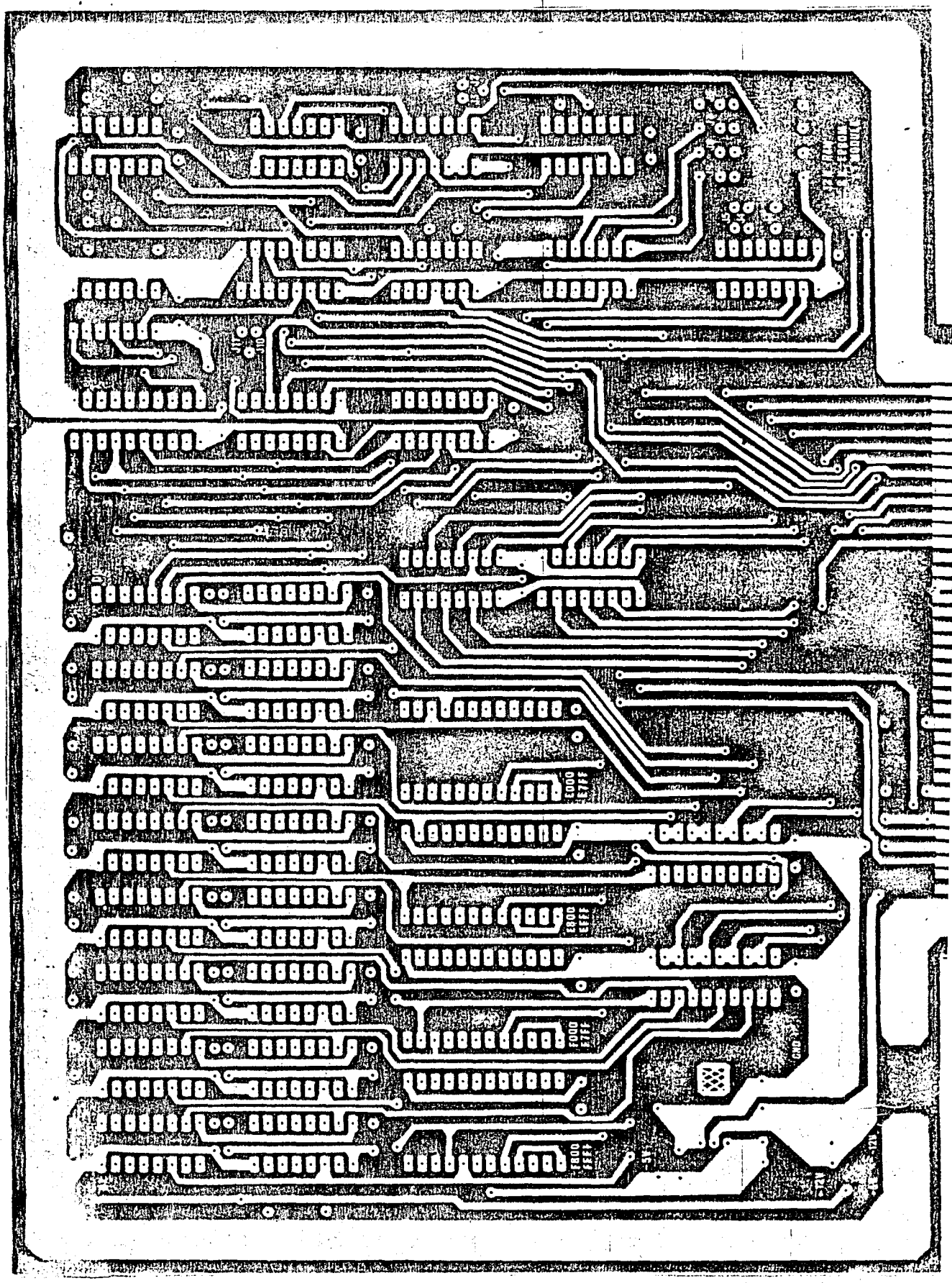
MACRO COMMANDS

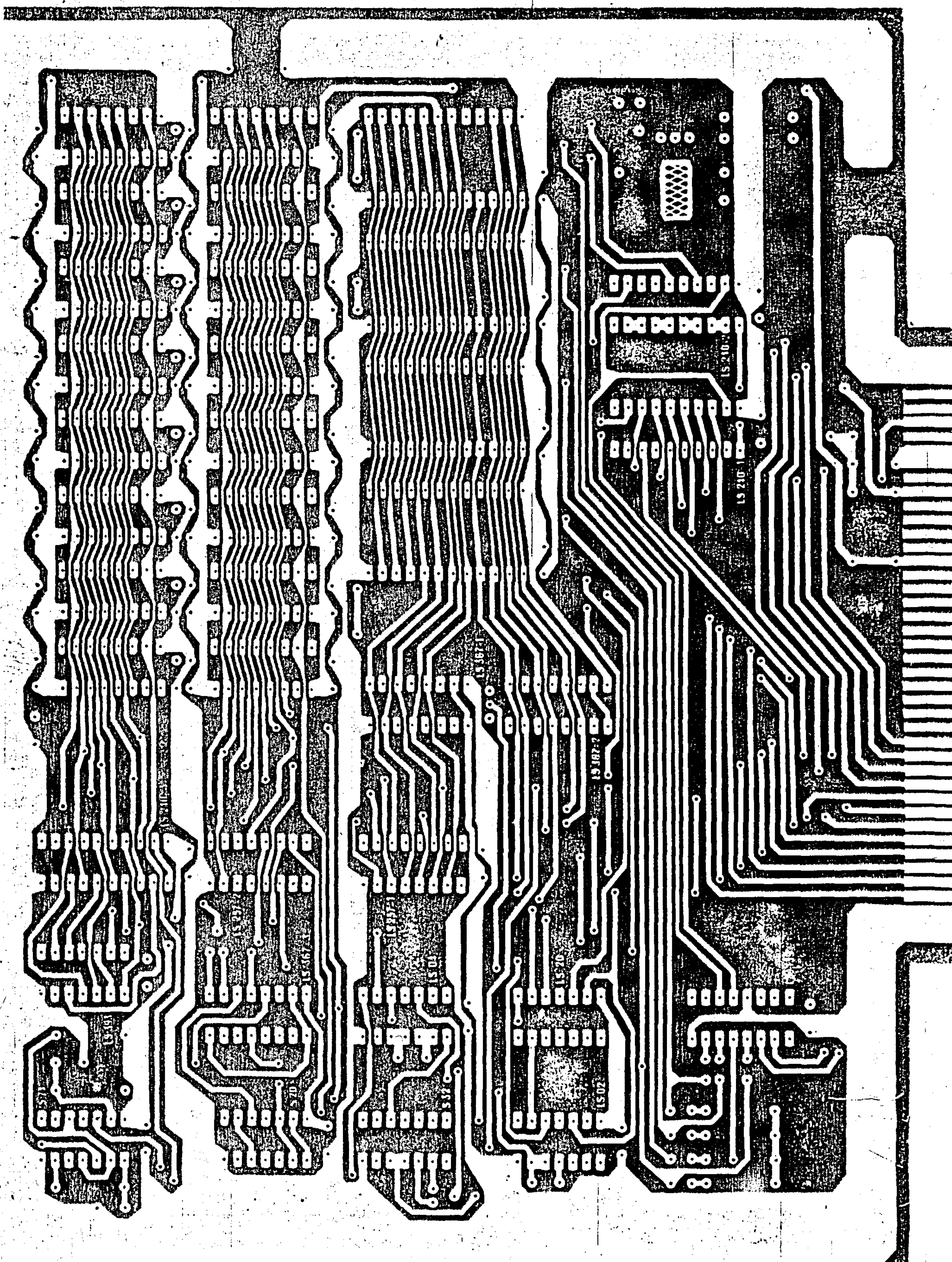
Hex Code	Instruction	Hex Code	Instruction
2	STZ	A	FFR
3	SEK	B	FFW
4	SSR	C	MSR
5	SSW	D	MSW
6	RCR		
7	SWD		

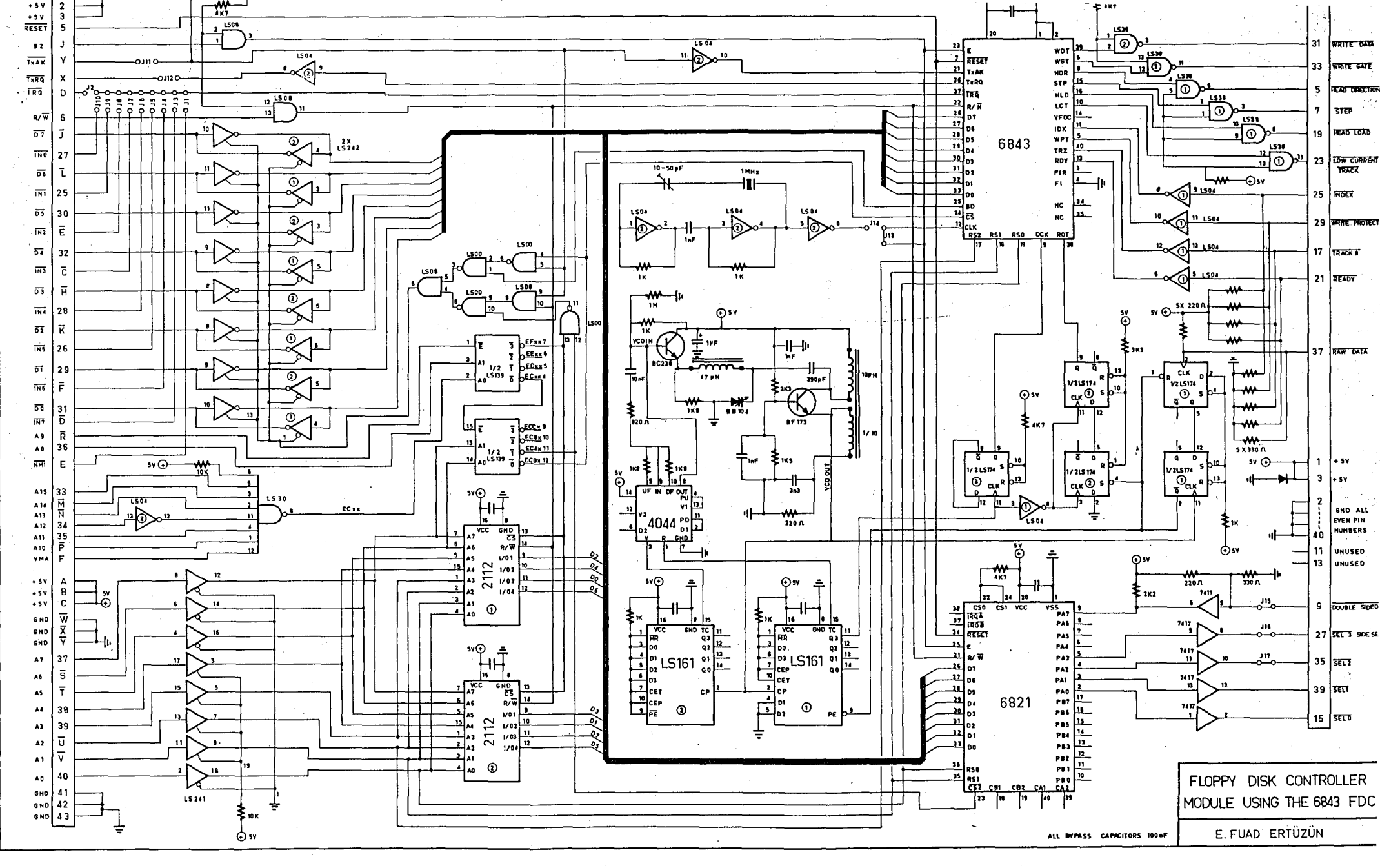
APPENDIX H

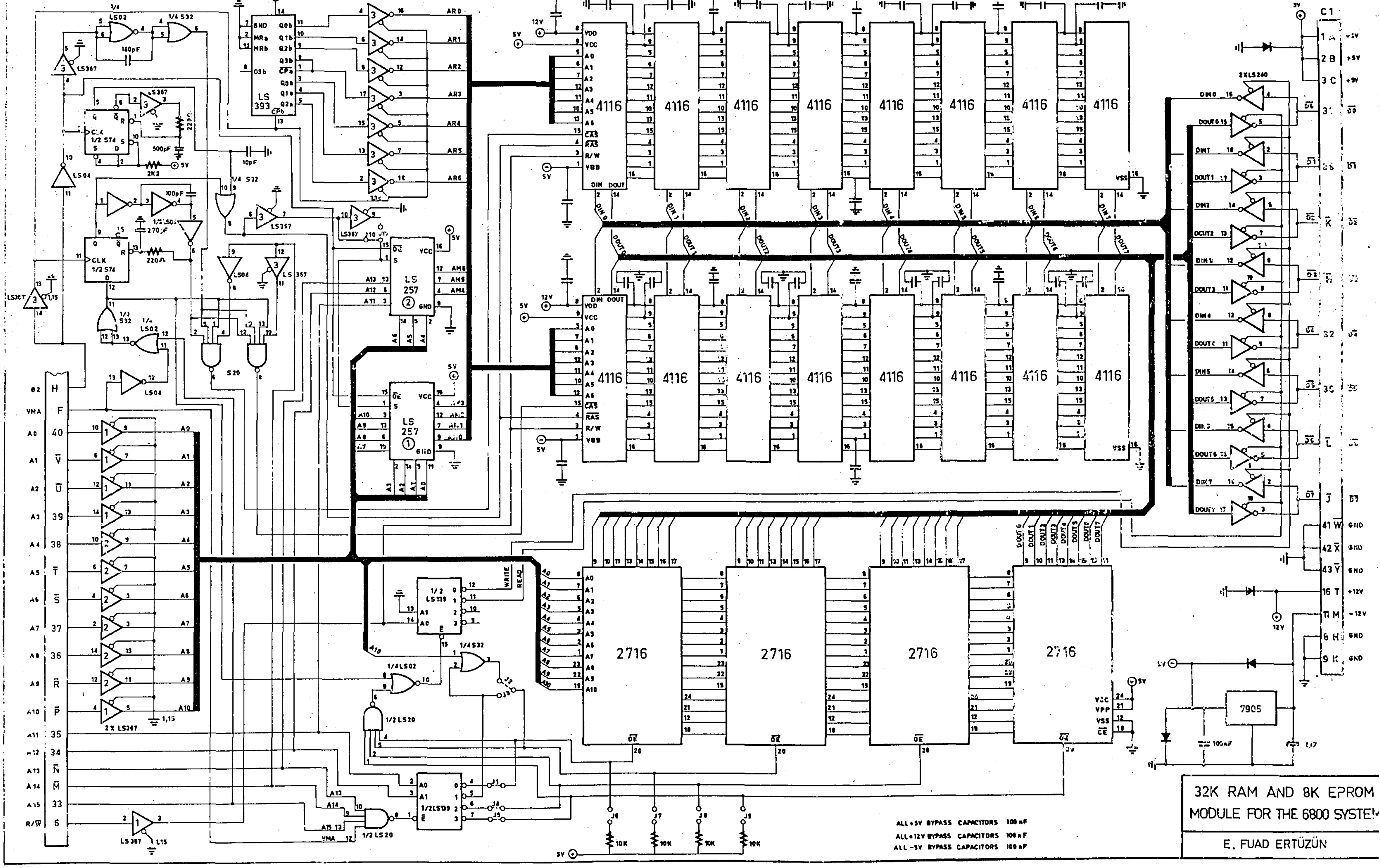





E000
FFFF
E000
FFFF
E000
FFFF
GND




FLOPPY DISK CONTROLLER
MODULE USING THE 6843 FDC
E. FUAD ERTÜZÜN
ALL BYPASS CAPACITORS 100nF
6843
6821
4044
LS161
2112
WRITE DATA
WRITE GATE
HEAD DIRECTION
STEP
HEAD LOAD
LOW CURRENT TRACK
INDEX
WRITE PROTECT
TRACK 0
READY
RAW DATA
GND ALL EVEN PIN NUMBERS
UNUSED
UNUSED
DOUBLE SIDED
SEL 3 SIDE SE
SEL 2
SEL 1
SEL 0
+5V
RESET
R/W
D7
IN0
D6
IN1
D5
IN2
D4
IN3
D3
IN4
D2
IN5
D1
IN6
D0
IN7
A9
A8
NMI
A15
A14
A13
A12
A11
A10
VMA
GND
TxAK
TxRQ
IRQ
Ø2
EC xx
LS 241
LS 242
LS 30
LS 04
LS 08
LS 00
LS 38
LS 39
LS 174
LS 139
BC238
BF 173
BB 104
VCO OUT
VCO IN



32K RAM AND 8K EPROM MODULE FOR THE 6800 SYSTEM

E. FUAD ERTÜZÜN