MACRO DISC

MICROMATION

DUAL DISK DRIVE

INSTALLATION GUIDE

AND USER'S MANUAL

VER:0.04

PLEASE NOTE THE FOLLOWING SPECIAL PRECAUTIONS WHEN USING DISKETTES

There are a few special precautions you must observe when handling diskettes and files to avoid destruction of data and programs through misuse or mishandling:

1. Whenever you remove a diskette from a drive and replace it with another diskette, REBOOT the CP/M system BEFORE PERFORMING ANY SUBSEQUENT OPERATIONS. A "warm start" is sufficient (control-C) to cause CP/M to recognize that the diskettes have changed. A reboot is not necessary, however, if the replaced diskette is "read-only" and data or programs will not be written to the diskette.

2. Do not turn the mainframe or disk drive power off with a diskette in the drive. Many controllers (such as the MDS 800 controller) will engage the head and turn on the write electronics momentarily, thus destroying a track of data.

3. Always store diskettes in their protective jackets when not in the diskette drive. Otherwise, dust will gather on the recording surface causing drive head wear and reduced media life.

4. Store the diskettes in normal work areas where temperatures are not extreme (within the 50-125 degrees F), and do not allow them to get near magnetic influences (such as large power supply transformers) or allow them to be exposed to direct sunlight for any extended period of time.

5. Provide adequate archives for your programs and data. Regular and organized backup techniques are essential for protection against media, hardware, software, or operator failures in any computing environment.

TABLE OF CONTENTS

SECTION 1.0

PARTS LIST

QUANTITY	ITEM
1	Dual Disk Drive
1	S-100 Bus Controller Card
1	40 Conductor Connector Cable
1	Software Diskette
1	16 Pin DIP Header
1	Documentation Manual
1	Warranty Card
1	CP/M Registration Card
1	CP/M Licensing Agreement
6	CP/M Documentation Manuals
1	BASIC-E Reference Manual

SECTION 2.0

MINIMUM HARDWARE CONFIGURATION

The MICROMATION Dual Disk Drive system runs on an S-100 bus, 8080 or Z-80 microcomputer with a minimum of 16K of RAM. To run BASIC-E at least 20K is needed. The memory must be contiguously addressed from locations 0 through 3FFF. Additional memory must avoid certain locations due to the memory on the controller board The reserved locations are four pages of memory, F800 through FBFF.

A console communications device is also needed. A video terminal (CRT) such as the ADM-3 or Hazeltine 1500 or a hardcopy type such as the Decwriter II, teletype or the HyTerm II is adequate. The terminal must be able to communicate over a standard serial interface using RS232 conventions or a Teletype terminal (TTY) that uses a 20ma current loop.

Computers that do not have front panel switches or some other means of transferring control to a specified location (other than zero) will need a board that transfers control on a reset or power-on.

SECTION 3.0

HOW TO CONNECT THE HARDWARE

Be sure that all components of the computer system are unplugged before connecting the MICROMATION dual disk drive.

Place the controller card in an empty S-100 bus slot component side forward. The card must be firmly seated in the connector to make good electrical contact. The connector end of the 40 conductor cable extending from the MICROMATION disk drive unit should then be connected to the top row of pins on the controller board. A small number "1" is etched near the leftmost pin on the board. The side of the cable with the single red wire must align with this. In most computers the cable will extend toward the rear of the box.

Any board that executes a jump instruction on a reset or power-on, such as the MICROMATION JUMP-START (tm) board, should be disabled until the disk system has been tested (If you do not have front panel switches, you WILL need a JUMP-START board or equivalent to get started. See Section 5.1). The JUMP-START board will be a helpful addition to the system when it is up and running.

SECTION 4.0

BRINGING UP THE SYSTEM

Supplied with the MICROMATION disk system is a copy of CP/M, a microcomputer Operating System (OS). It provides a named file structure on diskettes and I/O routines for the system's peripheral devices. It includes system tools such as an assèmbler, text editor and dynamic debugger.

CP/M comes with six separate manuals describing its abilities and use. It is important that these manuals be read and studied.

This manual is a guide to the CP/M documentation and it provides a convenient summary for generating CP/M systems. It will not act as a substitute for a thorough reading of the CP/M documentation. The contents of this manual and the CP/M manuals should be read completely before attempting any action.

SECTION 4.1

SYSTEM CONSOLE COMMUNICATION

CP/M uses a two-way communications device called the System Console. Through it the user requests services from the operating system and the OS informs the user of its status. The console device is usually a Cathode Ray Tube (CRT) or a Teletype (TTY). In addition to the console device itself, two things are necessary. First is a serial I/O port that supports either the TTY interface or the RS232 terminal and the software routines to interface between this port and the CP/M operating system. Both are supplied with the MICROMATION system. The I/O port is implemented in an on-board PROM. The interface drivers are ready to run in the distributed version of the CP/M BIOS (See CP/M SYSTEM

ALTERATION GUIDE, page 12, for a discussion of the drivers).

The serial I/O port provides instant communication with the system. It is possible to start running right away. It also allows the use of the CP/M facilities to customize CP/M avoiding the laborious task of hand assembling and toggling in of I/O routines. It also avoids the undesirable practice of "hot patching" programs.

The primary purpose of the on-board sofware I/O port is to get the system running with a minimum of time and effort. It is not designed as a permanent replacement for hardware I/O support. Many hardware I/O boards provide more than one port and a wider range of communication disciplines than the on-board software port.

SECTION 4.2

CONNECTING THE CONSOLE

The MICROMATION controller board has a 16 pin DIP socket in the upper right corner for connecting the system console device. The console device MUST have either an RS232 or 20ma current loop interface. A 16 pin DIP Header (plug) is provided with the MICROMATION system. It should already be inserted in the DIP socket. Remove it before soldering to it! The connecting wires from the terminal must be soldered to the top of the pins on the plug. The pin configuration is shown in Figure 4.6. If the terminal is a TTY or other current loop device connect only the pins shown in Figure 4.4. If it is an RS232 compatible device connect only the pins shown in Figure 4.3. Some RS232 devices require that lines other than the three shown here be pulled high or grounded. Check the specific manufacturer's documentation before hooking up any additional lines.

Data is sent through the port serially with no parity. There is one start bit and two stop bits on each data byte. Assure that the terminal is set for this type of communication.

Once the soldering is finished, plug the DIP header into the socket on the controller board. The notches on both components must be aligned.

FIGURE 4.3

RS232 CONNECTION SUMMARY

DIP PIN	NAME	EIA	RS232
15	GROUND	AB	7
2	SIG. OUT	BB	3
1	SIG. IN	BA	2

FIGURE 4.4

TELETYPE CONNECTION SUMMARY

DIP PIN	NAME	POLARITY
6	OUT	-
5	OUT	+
3	IN	+
4	IN	-

SECTION 4.5

BAUD RATE SELECTION

The serial data transmission speed of the software I/O port is regulated by a two byte constant held in memory in the CP/M CBIOS. It is set initially for ~~110 baud (10 cps)~~ 2400 BAUD, the speed of TTYs. Most CRTs can be set for this speed. If the terminal will operate at higher speeds it is possible to alter the speed constant. This should be changed only after the system is in an operational state. Once the system is up the speed can be altered with the facilities of CP/M. A description of how to change the baud rate constant is given in Section 6.1.

SECTION 4.6

DIP PIN LAYOUT

```
------------------------------------------------------------
I                                                          I
I    16     15     14     13     12     11     10      9   I
I                                                          I
I          RS-232                                          I
\           GND                                            I
 \                                                         I
  I                                                        I
 /                                                         I
/                                                          I
I   RS-232 RS-232  TTY    TTY     TTY    TTY               I
I     IN    OUT    IN+    IN -    OUT+   OUT-              I
I     1      2      3      4       5      6      7      8  I
I                                                          I
------------------------------------------------------------
```

SECTION 5.0

CP/M INITIATION PROCEDURE

When all of the hardware components are properly installed CP/M can be initiated and run. It is strongly suggested to make at least one backup copy of the system diskette immediately after determining that the system is functional and before ANY other processing. A program is included with the system for this purpose. The procedure for using it is described in Section 6.0. The following steps describe the exact procedure for starting CP/M.

A. All components must be interconnected. The 40 conductor cable should run from the disk controller to the disk drives. The console must be connected to the 16 pin DIP header.

B. All components must be plugged into a grounded, 115vac circuit. Be sure that the console device is plugged in.

C. Turn on the power to the computer and the disk drives. Depress the "RESET" switch on the computer's front panel. The "STOP" switch, if the computer has one, should be depressed before the "RESET" to prevent the computer from executing random instructions before the bootstrap operation. Allowing the computer to process "garbage" instructions can cause it to write garbage on the diskette! Do not turn on the power while the diskette is in the drive as power transients can destroy data also.

D. Insert the system diskette in drive "A". Make sure the diskette is facing in the proper direction (See Figure 5.2). CP/M will ALWAYS bootstrap from drive "A".

E. Examine memory location F800. This is the beginning of the program "SUPERBOOT" in PROM. Verify that the first byte of this routine is a 31H. The system will now be ready to execute the procedure to bootstrap the operating system into main memory.

F. Start processor execution with the "RUN" switch and the "SUPERBOOT" routine will bring in the bootstrap program from track zero, sector one of drive "A". The bootstrap program will then read in the remainder of the OS from tracks zero and one.

The system bootstrap will take approximately three seconds. When the operation is finished, the CP/M Console Command Processor (CCP) will type the system prompt message to the console. It looks like this:

```
A>
```

The prompt message is printed whenever CP/M is idling and awaiting a command from the console operator.

If some combination of one or two other characters appear on the screen it may indicate a communications problem. Check the console device to assure that it is set for the proper baud rate and framing pattern. If there is no response from the terminal, check the manufacturer's documentation. Some RS232 terminals require a "Clear To Send" or other signal to be pulled high before they will respond to any external communications. Some devices can automatically transmit a line-feed following a carraige-return. If this option is present, it must be disabled.

Briefly test the CP/M functions at this point. Type the command "DIR" followed by a carraige-return and the operating system should respond by printing the diskette file directory. See the CP/M FACILITIES manual for a more detailed description of the CCP functions.

Test the resident command "TYPE" by typing:

```
TYPE BOOT.ASM
```

The source file for the bootstrap program "BOOT" should be printed to the system console (long typeouts can be aborted by hitting any key on the console keyboard.).

The next test should be of a CP/M system transient program. Use the "STAT" transient for this. Type:

```
STAT
```

The response should be:

```
BYTES AVAILABLE: nnnK
```

To test the write function of the system type:

```
SAVE 1 X.COM
```

This will build a small file on diskette by the "X.COM". The "DIR" command should show that the file has been added to the directory.

At this point the system is functioning correctly. Before attempting any programming tasks at least one system backup disk should be created.

SECTION 5.1

BRINGING UP THE SYSTEM WITHOUT A FRONT PANEL

A power-on/reset jump start board is necessary to bring up the MICROMATION system if the computer does not have a front panel. Not having a front panel reduces the debugging facilities available but the initiation procedure is simplified.

Set the jump address on the board for F800, the address of Superboot. Follow the procedures described in section 5.0 except for step 'E' which will be automatically performed by the jump start board.

SECTION 5.2

HOW TO INSERT A DISKETTE

For Memorex systems, insert the diskette in drive A, the lower drive, with the label facing up. Push the diskette firmly until it engages in the drive, and close the door of the drive.

SECTION 6.0

BACKING UP THE SYSTEM

The program MMCOPY copies the entire contents of a diskette from drive A onto a diskette on drive B. Place a blank diskette in drive B. Be sure that the write protect notch is absent or has a tab over it. This will enable the write mechanism of the MICROMATION drive. Type the following in response to the system prompt:

```
        MMCOPY
```

To make more than one backup diskette, type:

```
        MMCOPY R
```

This causes the MMCOPY program to repeat the copying operation. When the copy program requests a return, make sure the diskettes are inserted and type return. When the diskettes have been copied, respond with a control-c to the "TYPE RETURN" message (See the MMCOPY documentation for a complete discussion of this program).

One backup copy of the system should be stored in a protected location and kept only in the event that all other diskettes are erased. Remember, if the last system diskette is accidently erased it will cost $25.00 to replace it. BACK IT UP!

SECTION 6.1

ALTERING THE SOFTWARE I/O PORT SPEED

If the system console device will run at a higher speed than the preset 110 baud rate, the speed constant held in reserved memory locations FA71 and FA72 can be altered. For baud rates of 110 or faster only the low order byte is significant. The high order byte is always set to zero. The single low order byte can be set to any new speed with the Dynamic Debugging Tool, DDT. After initiating DDT, use the 'set' facility to insert the proper speed constant in the low order location. The front panel can also be used. The speed constants are shown in Table 6.2.

Immediately after setting the constant, communications to and from the console will become garbled until the baud rate on the console device is changed.

Altering the speed constant with DDT is temporary. It will only last until the next "COLD" bootstrap operation. A cold boot will bring in a fresh copy of the CBIOS from diskette. It will contain the old speed constant for 110 and the system will instantly revert. A permanent change to the I/O port speed can be done when relocating CP/M.

SECTION 6.2

SERIAL I/O PORT SPEED CONSTANTS

	BAUD RATE					
	110	150	300	600	1200	2400
DECIMAL	171	125	62	31	15	7
HEX	AB	7D	3E	1F	0F	07

SECTION 7.0

RELOCATING YOUR SYSTEM

When the CP/M disk system is up and running it is possible to generate a system that will utilize all available RAM. To run BASIC-E or CBASIC a system running in at least 20K of memory is needed.

While adjusting the size of the operating system the CBIOS console I/O routines can be replaced. The new I/O routines can communicate through the normal I/O ports and hardware I/O support board(s). A driver routine to allow CP/M to output to a printer can be installed.

The procedures for generating, relocating and customizing the operating system are thoroughly described in the CP/M documentation manual:

CP/M SYSTEM ALTERATION GUIDE

The following section is a step-by-step summary of how to relocate and customize the system. This manual is not as detailed as the SYSTEM ALTERATION GUIDE. It is not a substitute but an aid and summary for the CP/M manual. The process of generating a new system is not complex but it can be a confusing procedure the first few times it is attempted. It is suggested that both this and the CP/M manuals be studied before building a custom system.

SECTION 7.1

OPERATING SYSTEM COMPONENTS

CP/M is composed of resident and transient programs. The transient programs need no modifications because they adjust themselves to the size of the current operating system. The resident components of the operating system must be modified for different sizes. They are:

CONSOLE COMMAND PROCESSOR
BASIC DISK OPERATING SYSTEM
BASIC INPUT OUTPUT SYSTEM
BOOTSTRAP PROGRAM

A detailed discussion of the organization of resident CP/M components is in the manual:

CP/M INTERFACE GUIDE

When I/O drivers are added to the Basic Input Output System (BIOS) a Customized BIOS or CBIOS is created. This CBIOS must be assembled for the desired system size and combined with the other portions of the OS including the bootstrap program. A standard system program, MOVCPM, will regenerate a new version of the operating system of any desired size without the customized portions and the bootstrap.

SECTION 7.2

REGENERATION PROCEDURE

Create the operating system (CCP and BDOS) for the size desired. The CBIOS and Bootstrap programs are distributed in source form. They need to be modified and reassembled separately. The three components are then gathered in the transient program area with DDT and saved on diskette as a COM type file.

Use the MOVCPM program to create a new copy of the operating system of the desired size. Save it as an ordinary COM file by typing:

```
        SAVE 32 CPMnnK.COM
```

Where 'nn' is the size of the new system. Use the PIP program to make a copy of the BIOS source that is distributed with the system. Call it 'CBIOSnnK.ASM'. Copy the bootstrap 'ASM' file with PIP also. Name it 'BOOTnnK.ASM'.

Make the following modifications to the source files of the CBIOS and Bootstrap. Use the CP/M Text Editor (ED).

The manual:

ED: A CONTEXT EDITOR FOR THE CP/M DISK SYSTEM

gives a thorough description of the use of this program.

Enter the Editor with the name of the CBIOS source copy. Alter the 'MSIZE' variable at the start of the program so that it indicates the proper size of the system. Put in comments describing any changes made in the CBIOS. Put the date of the change at the beginning of the code. Insert the I/O drivers for peripheral access after the names:

```
        CONIN:
        CONOUT:
        CONST:
```

A detailed description of the purpose of each routine can be found on page 15 of the SYSTEM ALTERATION GUIDE.

Be sure the I/O routines do not force the size of the CBIOS out of the alloted space.

For a list device, enter the driver routine after the name:

```
LIST:
```

Routines for a paper tape reader or punch can be added after the names:

```
READER:
PUNCH:
```

Unused routines should be terminated with a 'RET' instruction. If the hardware I/O board has UART chips that need programming, those procedures should be installed in the cold boot portion of the CBIOS marked with the comment:

```
;PLACE UART INITIALIZATION RTNS HERE
```

Sample I/O routines are shown in Section 7.3.

When all modifications have been made to the CBIOS, exit from the editor and assemble the new CBIOS. Detailed instructions on the CP/M Assembler are in the manual:

CP/M ASSEMBLER (ASM)

In addition to the CBIOS the bootstrap program must be reassembled for the new memory size. Using the editor, change the 'MSIZE' variable at the start of the bootstrap program. Exit the editor and reassemble the bootstrap.

CP/M can now be created out of its components: the relocated operating system (CCP, BDOS), the customized CBIOS, and the bootstrap. Uniting all of these parts is done with DDT. Start the DDT program and read in the new CP/M by typing:

```
DDT CPMnnK.COM
```

DDT will respond with its logon message followed by the next available address and the contents of the program counter:

```
DDT VERS 1.3
NEXT  PC
2100 0100
-
```

Now insert the name of the CBIOS file by typing:

```
ICBIOSnnK.HEX
```

This prepares DDT to read the CBIOS file. DDT normally reads programs into the memory locations for which they have been assembled. The operating system must be built in the TPA rather than the location where the OS will reside when it is running. They can be placed in the proper location by reading the files in with an "offset". The offset for the CBIOS is calculated from the size of the new system. The offset and calculating it is fully explained on pages 6-7 of the SYSTEM ALTERATION GUIDE.

The lowest page of memory is reserved for system communications and usable memory begins at location 100H. The SYSGEN program will occupy the 800H bytes ranging from 100H to 8FFH. The new operating system must be placed starting at location 900H where the SYSGEN program expects to find it.

Section 7.4 shows a chart of common offsets. For example, use the offset of A080 for a 32K system. To read the CBIOS with the 32K offset, type:

```
RA080
```

This causes the CBIOS to be properly inserted in relation to the BDOS forming the correct operating system configuration.

Insert the bootstrap program next. The bootstrap's location does not change from system to system, it is always loaded with the same offset and will always occupy the first sector on diskette. It must occupy the lowest portion of memory beyond 900H. The bootstrap is "org'ed" at location zero and must be loaded by DDT at an address 900H bytes away from its normal load address. This is done by specifying a 900H byte offset on the DDT "R" (READ) command:

```
IBOOTnnK.HEX
R900
```

The new customized CP/M is now properly organized in memory. Type a control-c to return to the monitor and type:

```
SAVE 32 CPMnnK.COM
```

This places a copy of the customized system onto the diskette under the name specified. The system must be "sysgen'ed" onto the first two tracks of a diskette for bootstrapping. The first two tracks are what the bootstrap program reads.

The program 'SYSGEN' is used for accessing the operating system tracks. SYSGEN performs two vital tasks. It reads a copy of the operating system off of tracks zero and one and places it into memory starting at location 900H in the Transient Program Area, and two, it will take any copy of the operating system that is already at location 900H and place it on the first two tracks of the specified diskette.

Use DDT to get the copy of the new system into the TPA, then use the second function of SYSGEN to place it on any diskette. The sequence is as follows:

```
DDT CPMnnK.COM
```

When DDT finishes loading the new OS and types the prompt character, enter a control-c to return to the monitor. Place the desired diskette in drive B and call SYSGEN to place the operating system on it. When SYSGEN types:

```
GET SYSTEM (Y/N)?
```

Type a 'N'; the system is already present. SYSGEN will then request:

```
PUT SYSTEM (Y/N)?
```

Respond with a 'Y' and the SYSGEN program will place the new OS onto the first two tracks of the diskette in drive B from the image of the operating system in memory.

When the SYSGEN is finished, the diskette in drive B is ready for rebooting. Remove the diskette from drive A and replace it with the one in B. Remember, in order to read in the ENTIRE new copy of CP/M, execute a cold-start (RESET) procedure. The control-c operation only performs a warm boot and will not read in the CBIOS.

SECTION 7.3

SAMPLE CBIOS I/O ROUTINES

```
CONST:
        IN      0               ;GET PORT STATUS
        ANI     1H              ;IS A CHAR THERE?
        JNZ     WASTHERE        ;YES, SO JUMP
        XRA     A               ;NO, SO CLEAR FLAGS AND ACCUM
        RET                     ;ALL DONE.  EXIT
WASTHERE:
        MVI     A,0FFH          ;INDICATE THAT SOMETHING
        RET                     ;   WAS THERE THEN EXIT
;
CONIN:
        IN      0               ;GET PORT STATUS
        ANI     1H              ;IS A CHAR THERE?
        JZ      CONIN           ;IF NOTHING THERE, TRY AGAIN
        IN      1               ;GET THE INCOMING DATA BYTE
        ANI     07FH            ;CLEAR THE PARITY BIT
        RET                     ;ALL DONE.  EXIT
;
CONOUT:
        IN      0               ;GET PORT STATUS
        ANI     2H                      ;IS OUTPUT BUFFER CLEAR?
        JZ      CONOUT                  ;NO, SO TRY AGAIN
        MOV     A,C                     ;PUT OUTGOING DATA IN ACCUM
        OUT     1                       ;WRITE IT OUT
        RET                             ;ALL DONE.  EXIT
;
LIST:
        IN      2                       ;GET PORT STATUS
        ANI     2H                      ;IS OUTPUT BUFFER CLEAR?
        JZ      LIST                    ;NO, SO TRY AGAIN
        MOV     A,C                     ;PUT OUTGOING DATA IN ACCUM
        OUT     3                       ;WRITE IT OUT
        RET                             ;ALL DONE.  EXIT
;
PUNCH:
READER:
        RET                             ;PUNCH AND READER ARE NOT USED.
;
```

SECTION 7.4

COMMON OFFSETS

SIZE	OFFSET
IN K	VALUE
16	E080
24	C080
32	A080
40	8080
48	6080
56	4080
64	2080

SECTION 8.0

THEORY OF OPERATION

The MICROMATION disks are controlled by an S-100 bus compatible controller. The controller is managed by software in two pages (512 bytes) of on-board PROM. Data transferred is buffered in one page (256 bytes) of on-board RAM. An additional address page is reserved for use by the controller for communications. The memory is addressed as follows:

PROM	F800	-	F9FF
RAM	FA00	-	FAFF
I/O LOCATIONS	FB00	-	FBFF

The MICROMATION controller transfers information to and from diskette whenever one of two reserved memory locations is accessed. When the 'MARKPORT' byte is read the controller reads a sectormark from the diskette. When the same byte is written to the sectormark is written to diskette. A second 'pseudoport' is called 'DATAPORT'. When it is accessed it causes transfer of a single byte of data. The psuedoport can be considered as an output port to diskette. For example, if the instruction:

```
LDAX        DATAPORT
```

is executed, the transfer of data is to the accumulator from the diskette rather than from the memory byte itself. Conversely, if this is executed:

```
STAX        DATAPORT
```

The byte in the accumulator is written to the current disk location. Each sector of data is arrayed on diskette in the following format:

FIELD	BYTES
INTER-RECORD GAP	
ADDRESS MARK	1
TRACK	1
ZERO	1
SECTOR	1
ZERO	1
ID FIELD CRC	2
ONES	11
ZEROS	6
DATA MARK	1
DATA FIELD	128
DATA FIELD CRC	2
ZERO	1

The status of the disk controller can be read into the accumulator in the same manner as data is transferred. By reading the 'statusport' location, eight bits of information are placed in the accumulator.

D7	D6	D5	D4	D3	D2	D1	D0
READY	SEEK DONE	HEAD LOADED	INDEX	SECTOR	WRITE PROTECT	SERIAL INPUT	TRACK ZERO

The controller is given instructions by writing to the 'CONTROLPORT' memory location. The eight control bits are:

D7	D6	D5	D4	D3	D2	D1	D0
UNIT SELECT			SELECT A	SELECT B	RESTORE	DIR	STEP HEAD

SECTION 8.1

HOW CP/M IS INITIATED

A small program, called 'SUPERBOOT' is burned into PROM at the reserved memory location F800. This program has the single function of reading in a single sector of data from track 0 sector 1 of drive A. It places the 128 bytes of data at location zero in main memory. Execution is then transferred to location zero. The 128 byte program that SUPERBOOT loads is the cold bootstrap loader for CP/M. SUPERBOOT is the same for any MICROMATION version of CP/M. When initiating CP/M for the first time, either a manual operation or a power-on/reset triggered circuit must jump to location F800 where the SUPERBOOT program resides.

MMCOPY

This program is a generalized full-disk copy program that is designed to run in a CP/M environment. MMCOPY will copy the entire contents of a diskette on drive A to a diskette on drive B.

The program is invoked by typing the transient name with two optional parameters. For example:

```
A-MMCOPY RS
```

If the optional 'R' parameter is specified the program will repeat execution indefinitely or until a control-c (warm boot) is entered from the console in response to the mount message. The mount message is issued before every copy and is of the form:

```
SOURCE ON A, DESTINATION ON B, THEN RETURN
```

It gives the operator a chance to change either one or both of the diskettes. After the diskette has been copied or when a control-c is detected MMCOPY will issue a reboot message giving the operator the opportunity to mount a system diskette in drive A.

If the optional 'S' parameter is entered anywhere on the command-line, the copy program will stop copying when it encounters a full track of 'E5's. When a diskette is initialized it is padded with the hexadecimal byte configuration of 'E5's. The 'S' parameter will thus allow a diskette with only a few tracks used to be copied in significantly less time than if the entire 77 tracks of unused data area were copied.

All data copied is automatically verified on disk reads and writes. If an error is detected the entire track (26 sectors) will be recopied and a message will be printed indicating the hex address of the track and sector in error. If the error persists, MMCOPY will retry the track for 10 times. After 10 unsuccessful retries a 'PERMANENT' message will be printed and the program will continue, ignoring the bad data.

MEMTEST

MEMTEST is a program designed to give your RAM memory an extensive read and write test. It will record all errors found while running on the system console. It is designed to test only RAM memory and will not test disk I/O or disk DMA. Other programs are available to test those functions.

The program begins by requesting three addresses from the user that must be entered in hexadecimal form. Leading zeros are required. It will request a starting address, a test length and a test increment. The program will begin at the starting address and perform three tests (called phases) in a single block of memory that is the size specified by 'increment'. At the end of these three tests the starting address is increased to the next increment and the phases are repeated. This continues until enough increments have been tested to equal the test length. For example, if you specify a starting address of 4000 (remember, this is in hex), a test length of 4000 and an increment of 1000, the program will start at 4000 (16K) and test through 4FFF outputing error statistics at the end. It will then repeat the procedure starting at 5000 and testing through 5FFF. This repeats a total of four times or until it has tested 4000 (16K) bytes.

In order to thoroughly test a memory board, we recommend two tests. The first time through the increment should be equal to the amount of memory addressed by one bank of RAM chips (this is 4K on most 16K boards). The second test should be run with an increment equal to the total board memory size. Thus on a 16K system with the test board strapped for 4000H through 7FFFH the test specifications would be:

	TEST ONE	TEST TWO
BEG. TEST LOCATION:	4000	4000
TOTAL TEST LENGTH:	4000	4000
TEST INCREMENT:	1000	4000

Note that this is a very thorough test. Test one will take close to six hours to run to completion and Test two will take around 24 hours to execute!

The actual test consists of three phases. For Phase One the test area is written with a bit pattern and then examined to see if the pattern is still there. The program repeats this test 256 times checking all possible bit patterns.

In Phase Two the test area is initially filled with zeros. A byte containing a single '1' bit is then written to the first location of the test area. The entire increment is checked to see if it is still zeros. The program repeats this test routine through all eight single bit patterns. It will then write the test byte in the second location in the test increment and repeat the above loop. It will thus 'walk' the eight bit patterns through every byte in the test area.

The Phase Three test procedure is identical to Phase Two except that both the testing field and the walking bit pattern are complemented. All of memory is filled with FF's and the bit patterns that are walked are the eight patterns containing a single '0' bit.

Upon completion of the three phases on each test increment, the program prints a table that consists of a row of eight four-digit hex numbers that are a count of the total number of errors found during the three phases. The left most number corresponds to the most significant bit (7) of the chip and right most to the least significant bit (0) of the chip.

One row is printed for each 0400H bytes tested. If a board containing 1K chips was tested, each row corresponds to one block of chips and each number in the row to a specific chip. However, if the board contained 4K chips then the first four rows correspond to one block of chips and the total errors attributable to a given chip would be the sum of the four numbers in the individual column (of four rows) belonging to that bank.

MEMTEST was originally designed as a stand-alone memory test program but has now been upgraded to run in a CP/M environment. Therefore, care must be taken not to give the memory test program addresses that will cause it to overlay itself or the BDOS (operating system) with any of the test patterns. The minimum beginning test location is 1000H (4K) and the total test length should never extend into the CP/M BDOS.

The most convenient method of operation is to strap your memory so that the board to be tested has addresses that are contained completely within the TPA (Transient Program Area). A second method is to strap the board to be tested with addresses completely ABOVE the operating system. For example, if the system contained two 16K RAM boards a 16K version of CP/M could be used and the board to be tested should be strapped for 4000H (16K) through 7FFFH (32K-1).

If neither of the above options are possible on your system it may be possible to run the test on one 4K block at a time and then restrap the board so all blocks can be tested.

The MEMTEST program will ask during the setup procedure whether you want detail error information by printing:

```
        RECORD EACH ERROR ON CONSOLE? (Y OR N)
```

If you respond with a 'Y', each time a byte is found to be in error a line will be printed at the system console in the form:

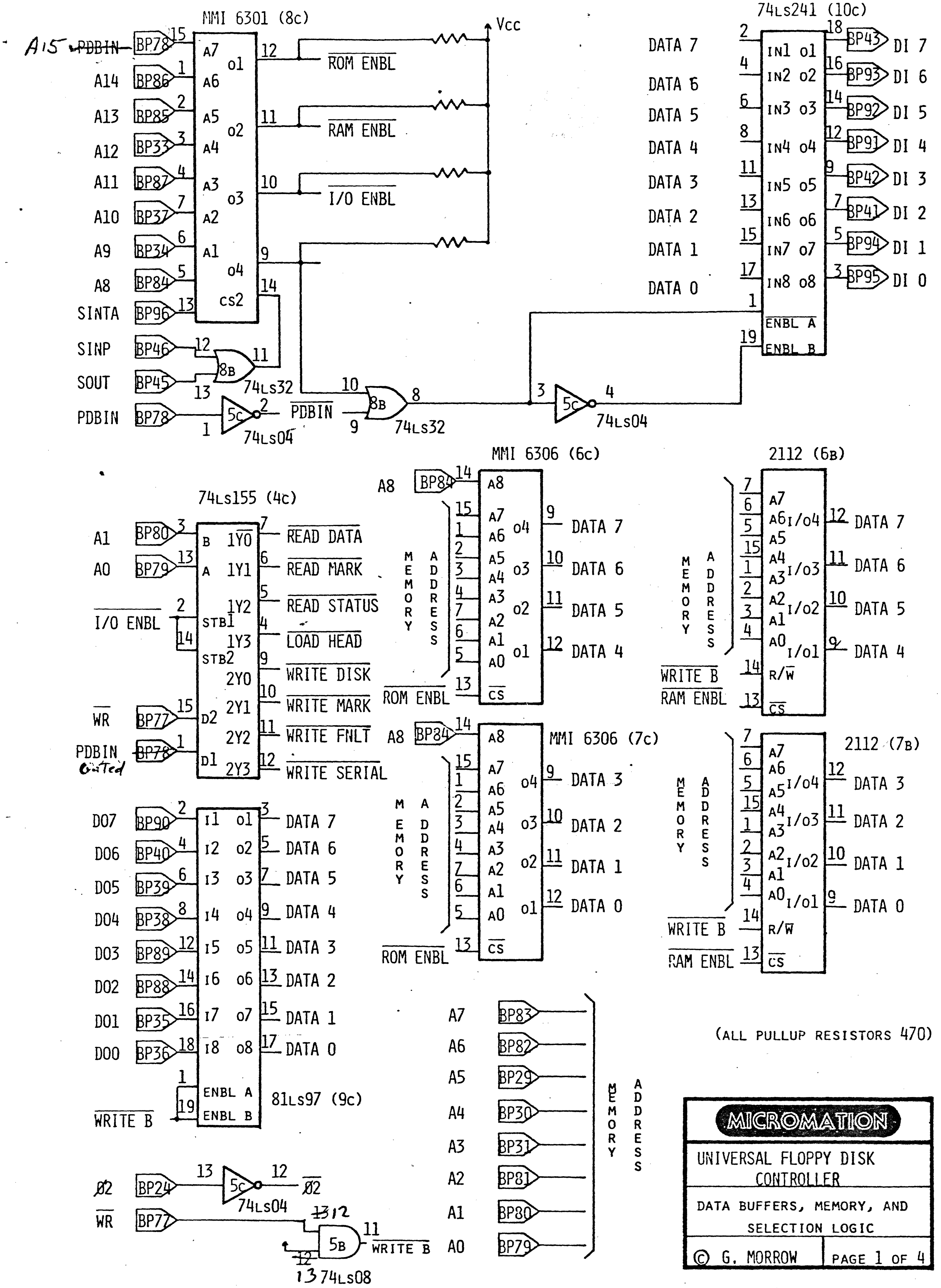
```
        A= aa aa  GB= gg bb  W= ww ww
```

Where aa aa is the address in memory in hexadecimal form of the bad byte.
Where gg is what pattern the program expected to find.
Where bb is what the program found.
Where ww ww is the address currently containing the walking byte.

During setup, the MEMTEST program will ask:

```
        REPEAT TEST? (Y OR N)
```

If the response is 'Y' the entire test procedure will repeat indefinitely.



(ALL PULLUP RESISTORS 470)

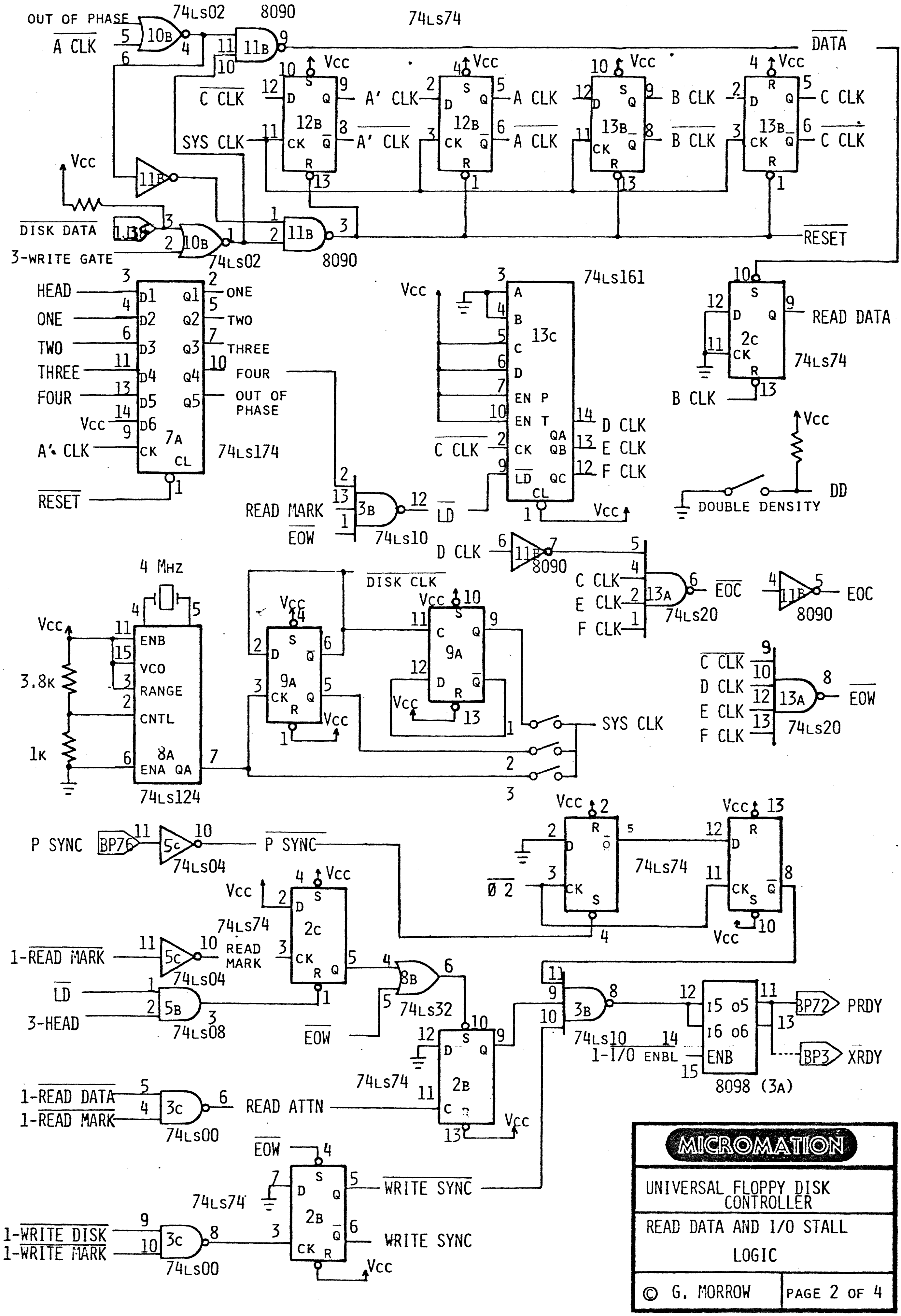
MICROMATION

UNIVERSAL FLOPPY DISK CONTROLLER

DATA BUFFERS, MEMORY, AND SELECTION LOGIC

© G. MORROW

PAGE 1 OF 4

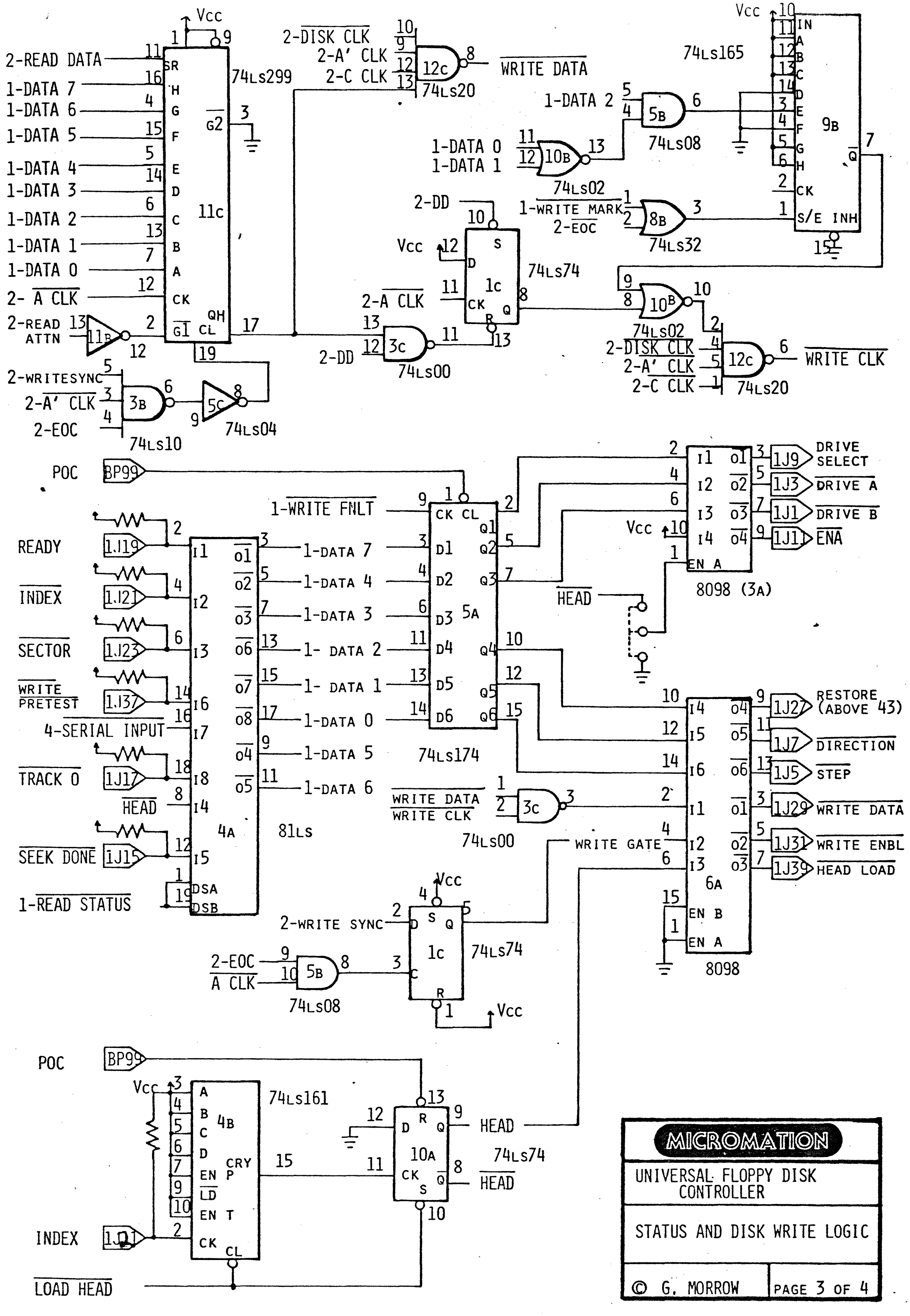


MICROMATION

UNIVERSAL FLOPPY DISK CONTROLLER

READ DATA AND I/O STALL LOGIC

© G. MORROW



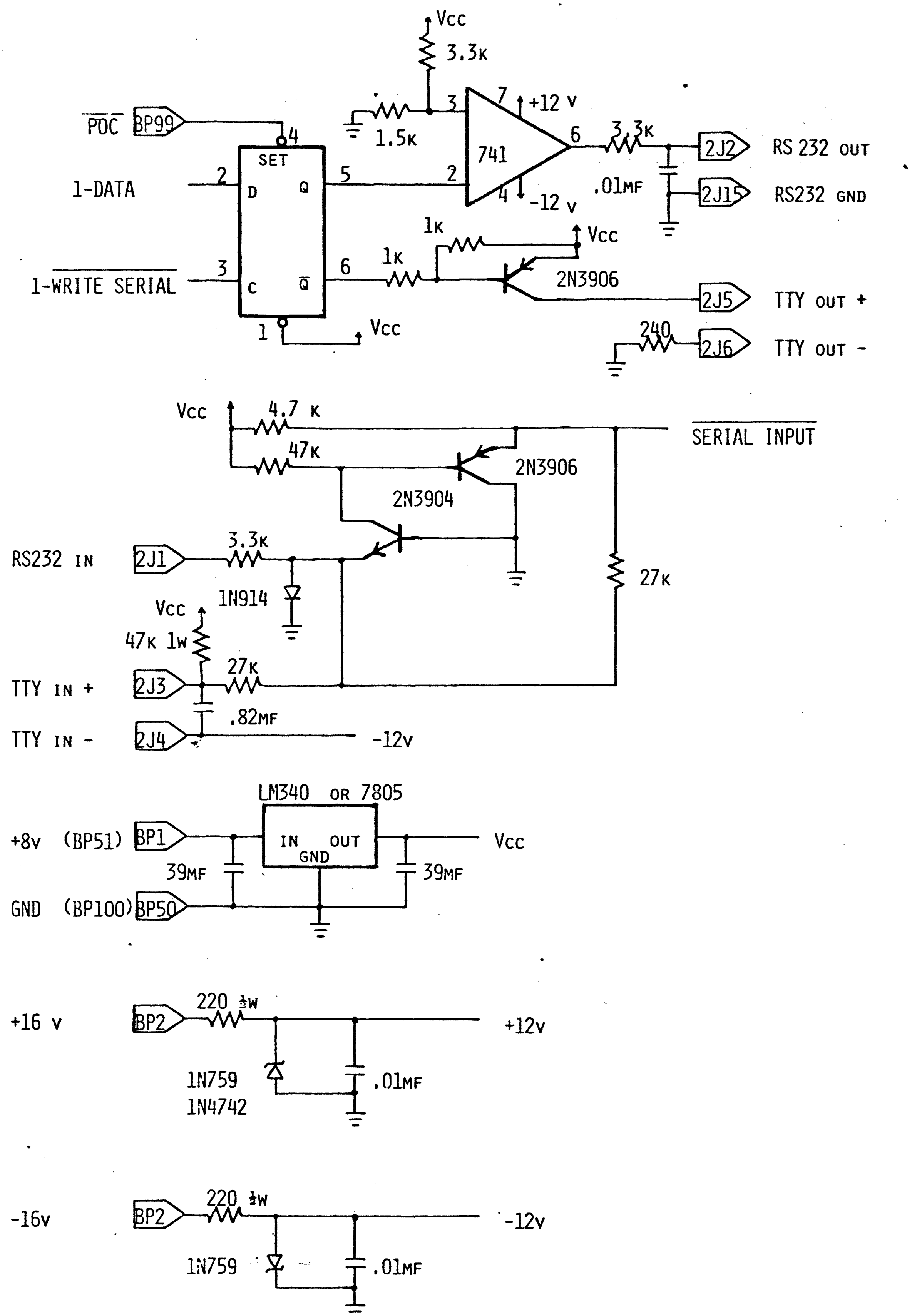
MICROMATION

UNIVERSAL FLOPPY DISK CONTROLLER

STATUS AND DISK WRITE LOGIC

© G. MORROW

PAGE 3 OF 4



MICROMATION

UNIVERSAL FLOPPY DISK CONTROLLER

SERIAL INTERFACE & POWER SUPPLIES

© G. MORROW

PAGE 4 OF 4

```
         aaTYPE MEMOREX.PRN


                        ;       UPDATED 1/2/78 TO TO RUN IN PROM AT F800
                        ;
                                ;
                                ;COMBINED ROUTINES TO BE INCLUDED IN DISK CONTRO
                                ;
                                ;
    F800                  ORG     0F800H
              ;

                      ;
FA00 =                SCRATCH:        EQU 0F800-H+200H
                      ;
                      ;
              ;
              ;       DRIVERS FOR MEMOREX DRIVE
              ;
              ;
              ;
              ;
              ;
              ;       MASK EQUATES
00FE =        ADDRESSMARK     EQU     0FEH
0040 =        SEEKDONEMASK    EQU     40H
0002 =        INMASK          EQU     02H
00FD =        OUTMASK         EQU     0FDH
0080 =        READYMASK       EQU     80H
0004 =        HOMEMASK        EQU     04H
0020 =        HEADMASK        EQU     20H
0010 =        AMASK           EQU     10H
0008 =        BMASK           EQU     08H
                      ;
0023 =        HEADSETTLE      EQU     35D
000A =        STEPSETTLE      EQU     10D
0006 =        STEPDELAY       EQU     6D
              ;
              ;
                      ;
                      ;
                      ;
              BOOTSTRAP:
                      ;THIS ROUTINE READS TRACK 0,SECTOR 1 INTO MEMORY
                      ;AT LOCATION ZERO AND THEN JUMPS TO ZERO
                      ;
FA70 =                STACK           EQU    SCRATCH+70H
0000 =                COLDBOOT        EQU    0
                      ;
                      ;
F800 3170FA           LXI     SP,STACK       ;SET STACK POINTER TO BUFFER
F803 0E00             MVI     C,0            ;SELECT DRIVE A
F805 CD8AF8           CALL    SELDSK
F808 CD22F8           CALL    HOME
F80B C200F8           JNZ     BOOTSTRAP      ;LOOP IF DRIVE NOT READY
F80E 0E01             MVI     C,1
F810 CD56F9           CALL    SETSEC         ;SET SECTOR ONE
F813 010000           LXI     B,COLDBOOT     ;SET DMA ADDRESS
F816 CD64F9           CALL    SETDMA         ;AT ZERO
F819 CDC1F8           CALL    DISKREAD
F81C C200F8           JNZ     BOOTSTRAP      ;LOOP IF ERROR
F81F C30000           JMP     COLDBOOT       ;JUMP TO BOOT
                      ;
```

```
                        ;
               HOME:
F822- CD81F8                    CALL     DISKREADY       ;IS DEVICE READY?
F825 D0                         RNC                     ;IF CARRY SET, THEN DIS
F826 2177FA                     LXI     H,TRACK ;POINT H-L TO TRACKBUFFER
               ATHOME:

F829 CD6AF8                     CALL STEPIN     ;STEP AWAY FROM HOME
F82C 1A                         LDAX    D       ;READ STATUS
F82D 1F                         RAR     ;CHECK TRACK ZERO BIT
F82E DA29F8                     JC      ATHOME  ;CONTINUE STEPPING IN TILL NOT
               GOHOME:
F831 CD61F8                     CALL    STEPOUT ;GO TOWARDS HOME
F834 1A                         LDAX    D       ;CHECK STATUS
F835 1F                         RAR             ;CHECK TRACK ZERO BIT
F836 D231F8                     JNC     GOHOME  ;LOOP UNTIL AT HOME
               INITIALIZE:
F839 2176FA                     LXI     H,ADDRPTR
F83C 36FE                       MVI     M,ADDRESSMARK
               FILLLOOP:
F83E 2C                         INR     L               ;BUMP ADDRESS BUFFER PT
F83F 3600                       MVI     M,0             ;FILL BUFFER WITH ZERO
F841 C23EF8                     JNZ     FILLLOOP
F844 C9                         RET
               ;
               SETTRK:
F845 AF                         XRA     A               ;GET A ZERO
F846 B1                         ORA     C               ;IS TRACK-0?
F847 F8                         RM                      ;IF YES, RETURN
F848 3E4C                       MVI     A,76D           ;COMPARE TO LAST TRACK
F84A 91                         SUB     C
F84B D8                         RC                      ;CHECK IF TRACK GREATER
F84C CD81F8                     CALL    DISKREADY       ;CHECK READY
F84F D0                         RNC             ;RETURNIF NOT
               STEPLOOP:
F850 2177FA                     LXI     H,TRACK         ;POINT TO PRESENT TRACK
F853 7E                         MOV     A,M             ;GET PRESENT TRACK
F854 B9                         CMP     C               ;COMPARE TO DESIRED TRA
F855 CA7BF8                     JZ      DONESTEP        ;ON CORRECT TRACK
F858 CD5EF8                     CALL    STEPHEAD        ;CARRY SET TO INDICATE
F85B C350F8                     JMP     STEPLOOP        ;GO AROUND AGAIN
               ;
               STEPHEAD:
F85E DA6AF8                     JC      STEPIN          ;GO INWARDS IF CARRY SE

               STEPOUT:
F861 3A70FA                     LDA     CONTROLBYTE     ;CHECK DRIVE SELECT
F864 35                         DCR     M               ;DECREMENT TRACK BUFFER
F865 E6FD                       ANI     OUTMASK         ;SET D1 FOR DIRECTION
F867 C370F8                     JMP     DOSTEP          ;EXECUTE STEP
               STEPIN:
F86A 3A70FA                     LDA     CONTROLBYTE
F86D 34                         INR     M               ;INCREMENT TRACK REGIS
F86E F602                       ORI     INMASK          ;SET IN CODE
               DOSTEP:
F870 12                         STAX    D       ;OUT PUT DIRECTION
F871 3C                         INR     A               ;SET STEP BIT
F872 12                         STAX    D               ;OUTPUT STEP
F873 3D                         DCR     A               ;CLEAR STEP BIT
F874 12                         STAX    D               ;CLEAR STEP BIT ON POR
F875 0606                       MVI     B,      STEPDELAY       ;SET UP DELAY
F877 CDA3F8                     CALL    DELAY           ;DELAY FOR STEP TIME
F87A C9                         RET             ;
```

```
               DONESTEP:
F87B 060A                      MVI     B,STEPSETTLE    ;SET UP HEAD SETTLEING
F87D CDA3F8                    CALL    DELAY
F880 C9                        RET
               ;
               ;
               ;
               ;
               DISKREADY:
F881 CDB1F8                    CALL    HEADLOAD
               ; CAUTION  IT IS ASSUMED THAT HEADLOAD SETS D,E TO DISKFUNCTIC
F884 AF                        XRA     A               ;CLEAR THE ZERO FLAG
F885 1A                        LDAX    D               ;GET FUNCTION BYTE FRC
F886 07                        RLC             ;READY BIT SHIFTED INTO CARRY
F887 D8                        RC                      ;CARRY ,ZERO SET
F888 3C                        INR     A               ;CLEAR ZERO FLAG
F889 C9                        RET                     ;DRIVE NOT READY
               ;
               ;
               SELDSK:
F88A CDB1F8                    CALL    HEADLOAD
F88D AF                        XRA     A               ;GET ZEROS
F88E 81                        ADD     C               ;ZERO=1 IF DRIVE A, Z
F88F CA97F8                    JZ      SELECTA
               ;       DO A SELECTB
F892 3E08                      MVI     A,BMASK         ;GET SELECT MASK FOR
F894 C399F8                    JMP     DOSELECT
               SELECTA:
F897 3E10                      MVI     A,AMASK         ;GET SELECT MASK FOR
               DOSELECT:
F899 3270FA                    STA     CONTROLBYTE     ;SET UP DRIVE STATUS
F89C 3202FB                    STA     DISKFUNCTION    ;SEND TO CONTROLLER
F89F AF                        XRA     A               ;SET ZERO FLAG
F8A0 C3BAF8                    JMP     CALLDELAY       ;CALL DELAY FOR HEADL
               ;
               ;
               ;
               DELAY:
F8A3 2E1F                      MVI     L,31            ;# OF MILLISECS DELAY
               DELAYLP:
F8A5 3A00FB                    LDA     DATAPORT        ;THIS INSTRUCTION CAU
                                                       ;A 32 MICOR-SECOND DE
                                                       ;IF THE HEAD IS LOADE
F8A8 2D                        DCR     L
F8A9 C2A5F8                    JNZ     DELAYLP
F8AC 05-                       DCR     B
F8AD C2A3F8                    JNZ     DELAY
F8B0 C9                        RET
               ;
               ;
               HEADLOAD:
F8B1 1102FB                    LXI     D,DISKFUNCTION
F8B4 1A                        LDAX    D       ;READ DISK STATUS
F8B5 E620                      ANI     HEADMASK        ;CHECK FOR HEAD LOADI
F8B7 13                        INX     D       ;POINT TO HEADLOAD
F8B8 1A                        LDAX    D       ;STROBE HEADLOAD COUNTER
F8B9 1B                        DCX     D       ;SET D,E TO DISKFUNCTION, FO
               CALLDELAY:
F8BA 0623                      MVI     B,HEADSETTLE    ;SET UP HEADSETTLING
F8BC CCA3F8                    CZ      DELAY   ;LET HEAD SETTLE
F8BF AF                        XRA     A       ;SET ZERO FLAG
F8C0 C9                        RET             ;FOR RETURN
               ;
               ;
```

```
                       ;
                       ;
                       ;       MAP OF SCRATCH AREA
FA6F =                 BOOTSTACK       EQU     SCRATCH+6FH
FA70 =                 CONTROLBYTE     EQU     SCRATCH+70H
FA71 =                 SPEED           EQU     SCRATCH+71H
FA73 =                 RETRYCOUNT      EQU     SCRATCH+73H
FA74 =                 DMAADDR         EQU     SCRATCH+74H
FA76 =                 ADDRPTR         EQU     SCRATCH+76H
FA7D =                 DATAPTR         EQU     SCRATCH+7DH
FAFD =                 LASTDATA        EQU     SCRATCH+0FDH
007D =                 DATABYTE        EQU     7DH
FA79 =                 SECTOR          EQU     SCRATCH+79H
FA77 =                 TRACK           EQU     SCRATCH+77H
FB00 =                 DATAPORT        EQU     SCRATCH+100H
FB01 =                 MARKPORT        EQU     SCRATCH+101H
FB02 =                 DISKFUNCTION    EQU     SCRATCH+102H
FB03 =                 LOADPORT        EQU     SCRATCH+103H
FB03 =                 SERIALOUTPORT   EQU     SCRATCH+103H
FAFE =                 CRCBUFFER       EQU     SCRATCH+0FEH
84BF =                 RESIDUE         EQU     84BFH
                       ;
                       ;
                       ;
                       ;
               DISKREAD:
F8C1 0E00              MVI     C,0     ;SET READ FLAG
F8C3 CDFEF8            CALL ENTRY      ;EXECUTE READ
F8C6 C0                RNZ             ; RETURN IF ERROR
F8C7 2D                DCR     L       ;POINT TO CRC
F8C8 EB                XCHG            ;MOVE LAST BYTE ADDRESS TO DE
F8C9 217DFA            LXI     H,DATAPTR       ;POINT TO ADDR OF DATA MARK
F8CC CD70F9            CALL    CREECH  ;COMPUTE CRC
F8CF 78                MOV     A,B     ;MOVE HIGH RESIDUE TO ACC
F8D0 B1                ORA     C       ;COMPARE TO C
F8D1 C0                RNZ             ;CRC ERROR IF B,C NOT ZERO
               DATAXFER:
F8D2 0680              MVI     B,128   ;SET BYTE COUNTER
F8D4 117DFA            LXI     D,DATAPTR       ;POINT TO DATA MARK
F8D7 2A74FA            LHLD    DMAADDR ;POINT H,L TO DESTINATION
               XFERLOOP:
F8DA 13                INX     D       ;POINT TO NEXT BYTE IN BUFFER
F8DB 1A                LDAX    D       ;GET BYTE FROM BUFFER
F8DC 77                MOV     M,A     ;STORE BYTE IN MAIN MEMORY
F8DD 23                INX     H       ;PONINT TO NEXT BYTE IN MEMORY
F8DE 05                DCR     B       ;HIT BYTE COUNTER
F8DF C2DAF8            JNZ     XFERLOOP        ;GO AROUND FOR MORE
F8E2 C9                RET             ;ZERO SET TO INDICATE NO ERROR
                       ;
                       ;
               DISKWRITE:
F8E3 2A74FA            LHLD    DMAADDR ;POINT TO DATA IN MAIN MEMORY
F8E6 EB                XCHG            ;MOVE ADDRESS TO DE
F8E7 217EFA            LXI     H,DATAPTR+1     ;POINT TO DATA BUFFER
               LOADLOOP:
F8EA 1A                LDAX    D       ;GET BYTE FROM MENORY
F8EB 77                MOV     M,A     ;MOVE INTO BUFFER
F8EC 13                INX     D       ;NEXT BYTE IN MEMORY
F8ED 2C                INR     L       ;NEXT BUFFER BYTE
F8EE C2EAF8            JNZ     LOADLOOP        ;END OF BUFFER?
F8F1 11FDFA            LXI     D,LASTDATA      ;POINT TO LAST DAT BYTE
F8F4 2E7D              MVI     L,DATABYTE      ;LOAD LOW ORDER ADDRER OF D
```

```
F8F6 CD70F9            CALL    CREECH  ;COMPUTE CRC
F8F9 71                MOV     M,C     ;STORE CRC IN CRC BUFFER
F8FA 23                INX     H       ;NEXT BYTE
F8FB 70                MOV     M,B     ;STORE LAST CRC BYTE
F8FC 0E07              MVI     C,7     ;SET UP WRITE FLAG
                       ;
                       ;
                       ;
F8FE CD81F8     ENTRY: CALL DISKREADY  ;CHECK FOR HEAD LOADED
F901 C0                RNZ             ;DISK NOT READY
F902 F3                DI              ;DISABLE INTERRUPTS TO PROTECT REA
F903 CD0FF9            CALL READWRITE  ;EXECUTE READ OR WRITE
F906 FB                EI              ;ENABLE INTERRUPTS FOLLOWING READ/
F907 C8                RZ      ;RETURN IF NO ERROR
F908 3E04              MVI     A,4     ;WRONG SECTOR HEADER READ?
F90A B8                CMP     B       ;B CONTAINS POINTER WHERE ERROR O
F90B CAFEF8            JZ      ENTRY   ;RETRY IF WRONG SECTOR
F90E C9                RET             ;RETURN WITH NO ZERO TO INDICATE
                       ;
                       ;
                READWRITE:
F90F 2176FA            LXI     H,      ADDRPTR ;POINT TO ADDR MRK
F912 1101FB            LXI     D,      MARKPORT        ; POINT TO PORT F
F915 0606              MVI     B,6     ;SET BYTE COUNTER
                ADDRMARKLOOP:
F917 1A                LDAX    D       ;READ MARK
F918 BE                CMP     M       ;ADDRESS MARK?
F919 C217F9            JNZ     ADDRMARKLOOP    ;IF NOT TRY AGAIN
F91C 1B                DCX     D       ;POINT TO DATA PORT
                ADDRESSHEADER:
F91D 23                INX     H       ;LOOK AT NEXT BYTE IN HEADER
F91E 1A                LDAX    D       ;READ NEXT BYTE FROM DISK
F91F BE                CMP     M       ;RIGHT DATA READ?
F920 C0                RNZ             ;RETURN IF ERROR
F921 05                DCR     B       ;HIT BYTE COUNTER
F922- C21DF9            JNZ     ADDRESSHEADER   ;TRY AGAIN IF NOT DONE
F925 060A              MVI     B,10    ;SET BYTE COUNTER FOR CRAP IN GAP
                GAPLOOP:
F927 1A                LDAX    D       ;READ BYTE OF GAP
F928-05                DCR     B       ;HIT BYTE COUNTER
F929 C227F9            JNZ     GAPLOOP ;RETURN IF NOT LAST GAP BYTE
F92C 79                MOV     A,C     ;CHECK READ/WRITE FLAG
F92D B7                ORA     A       ;FLAG = 0 ?
F92E CA46F9            JZ      READSECTOR      ;GET OUT FOR READ
F931 AF                XRA     A               ;SET UP TO WRITE ZEROS I
                ZEROWRITE:
F932 12                STAX    D               ;WRITE A ZERO DATA BYTE
F933 0D                DCR     C               ;LAST BYTE
F934 C232F9            JNZ     ZEROWRITE               ;GO AROUND FOR M
F937 13                INX     D               ;POINT TO MARKPORT
F938 23                INX     H       ;POINT TO DATA MARK
F939 7E                MOV     A,M     ;GET DATA MARK
F93A 12                STAX    D       ;WRITE DATA MARK
F93B 1B                DCX     D               ;POINT TO DATAPOINT
F93C 23                INX     H       ;POINT TO DATA
                WRITEDATALOOP:
F93D 7E                MOV     A,M             ;GET DATA BYTE
F93E 12                STAX    D               ;WRITE DATA TO DISK
F93F 2C                INR     L               ;POINT TO NEXT BYTE
F940 C23DF9            JNZ     WRITEDATALOOP   ;LOOP IF NOT LAST BYTE
F943 AF                XRA     A               ;CLEAR ACC,SET ZERO
F944 12                STAX    D               ;WRITE ZERO
F945 C9                RET                     ;FINISHED
```

```
               ;
               READSECTOR:
F946 1A                LDAX    D       ;READ PAST CRAP IN GAP
F947 1A                LDAX    D               ;DITTO
F948 1A                LDAX    D
F949 13                INX     D       ;POINT TO MARKPORT
F94A 23                INX     H       ;POINT TO DATA MARK
F94B 1A                LDAX    D       ;READ DATA MARK
F94C BE                CMP     M       ;COMPARE
F94D C0                RNZ             ;RETURN WITH ERROR IF NOT
F94E 1B                DCX     D       ;POINT TO DATAPORT
               READDATALOOP:
F94F 2C                INR     L       ;POINT TO NEXT BYTE IN BUFFER
F950 C8                RZ              ;GET OUT IF LAST BYTE
F951 1A                LDAX    D       ;READ DATA BYTE
F952- 77               MOV     M,A     ;STORE BYTE IN MEMORY
F953 C34FF9            JMP     READDATALOOP    ;GO AROUND FOR MORE;
               ;
               ;
               ;
               ;
               ;
               ;
               SETSEC:
F956 2179FA            LXI     H,SECTOR        ;POINT TO SECTOR BUFFER
F959 71                MOV     M,C     ;STORE REGISTER NUMBER FROM C R
F95A CD6AF9            CALL    SETADDRCRC      ;COMPUTE CRC OF HEADER
F95D 71                MOV     M,C     ;STORE FIRST CRC BYTE
F95E 23                INX     H       ;POINT TO NEXT BUFFER BYTE
F95F 70                MOV     M,B     ;STORE SECOND CRC BYTE
F960 23                INX     H       ;POINT TO NEXT BYTE
F961 36FB              MVI     M,0FBH  ;STORE DATA MARK
F963 C9                RET             ;DONE
                       ;
                       ;
                       ;
               SETDMA:
F964 60                MOV     H,B     ;MOVE B,C PAIR TO H,L
F965 69                MOV     L,C
F966 2274FA            SHLD DMAADDR    ;STORE ADDRESS IN BUFFER
F969 C9                RET
                       ;
                       ;
```

```
                                   ;
                       SETADDRCRC:
F96A 2176FA                        LXI      H,ADDRPTR         ;STARTING ADDRESS IN H
F96D 117AFA                        LXI      D,SECTOR+1        ;ENDING ADDRESS IN D,E
                       CREECH:              ;ROUTINE TO COMPUTE CRC
F970 01FFFF                        LXI      B,-1
F973 D5                            PUSH     D
F974 7E                            MOV      A,M
F975 A9                            XRA      C
F976 57                            MOV      D,A
F977 0F                            RRC
F978 0F                            RRC
F979 0F                            RRC
F97A 0F                            RRC
F97B E60F                          ANI      0FH
F97D AA                            XRA      D
F97E 5F                            MOV      E,A
F97F 0F                            RRC
F980 0F                            RRC
F981 0F                            RRC
F982 57                            MOV      D,A
F983 E61F                          ANI      1FH
F985 A8                            XRA      B
F986 4F                            MOV      C,A
F987 7A                            MOV      A,D
F988 E6E0                          ANI      0E0H
F98A AB                            XRA      E
F98B 47                            MOV      B,A
F98C 7A                            MOV      A,D
F98D 0F                            RRC
F98E E6F0                          ANI      0F0H
F990 A9                            XRA      C
F991 4F                            MOV      C,A
F992 23                            INX      H
F993 D1                            POP      D
F994 7A                            MOV      A,D
F995 BC                            CMP      H
F996 D8                            RC
F997 C273F9                        JNZ      CREECH+3
F99A 7B                            MOV      A,E
F99B BD                            CMP      L
F99C D8                            RC
F99D C373F9                        JMP      CREECH+3
                                   ;
                                   ;
```

```
                          ;
             ;             SOFTWARE UART ROUTINES
             ;
             SERIALIN:
             ;
F9A0 0601                           MVI B,1 ;SET TO SUPRESS OUTPUT IN DF
F9A2 2A71FA                         LHLD SPEED       ;GET SPEED CONSTANT

             SWAIT:
F9A5 E5                             PUSH H  ;SAVE ON STACK
F9A6 1EFF                           MVI E,0FFH       ;INITILIZE 1/2 THE S
F9A8 3A02FB  SLOOK:                 LDA     DISKFUNCTION    ;LOOK FOR SE
F9AB 1F                             RAR     ;ROTATE INTO CARRY
F9AC 1F                             RAR
F9AD DAA8F9                         JC      SLOOK    ;IS SERIAL INPUT BIT
F9B0 CDEAF9                         CALL SERIALDELAY        ;WAIT HALF A
F9B3 E1                             POP H   ;RESET SPEED CONSTANT
F9B4 3A02FB                         LDA     DISKFUNCTION    ;VERIFY THAT
F9B7 1F                             RAR              ;IS STILL PRESENT
F9B8 1F                             RAR
F9B9 DAA5F9                         JC      SWAIT
F9BC 16FF                           MVI     D,0FFH   ;INITIALIZE OTHER HA
             GTBIT:
F9BE E5                             PUSH H  ;UPDATE THE STACK
F9BF 29                             DAD     H        ;CALCULATE THE SPEED
F9C0 2B                             DCX     H        ;CONSTANT FOR A FULL
F9C1 3A02FB                         LDA     DISKFUNCTION    ;GET THE I N

F9C4 1F                             RAR     ;ROTATE TO BIT ZERO
F9C5 5F                             MOV     E,A      ;UPDATE THE SHIFT RE
F9C6 CDEAF9                         CALL    SERIALDELAY     ;DELAY ONE B
F9C9 E1                             POP H   ;GET THE SPEED CONSTANT
F9CA DABEF9                         JC      GTBIT    ;HAS THE START BIT S
F9CD 7A                             MOV     A,D      ;MOVE BYTE TO ACC
F9CE E67F                           ANI     7FH      ;CLEAR HIGH BIT
F9D0 C9                             RET
             ;
             ;
             SERIALOUT:
F9D1 79                             MOV     A,C      ;MOVE CHARACTER TO A
F9D2 87                             ADD     A        ;ADD A START BIT
F9D3 47                             MOV     B,A      ;MAKE BIT 0 OF B A Z
F9D4 5F                             MOV     E,A      ;SHIFTED DATA TO E
F9D5 3E0B                           MVI     A,11     ;THIS IS THE BIT COU
F9D7 4F                             MOV     C,A      ;COUNT TO REG C
F9D8 17                             RAL              ;LOAD D WITH THE RES
F9D9 57                             MOV     D,A      ;BITS AND HIGH ORDER
             OLOOP:
F9DA 2A71FA                         LHLD    SPEED    ;GET THE SPEED CONST
F9DD 2A71FA                         LHLD    SPEED    ;PADDING
F9E0 29                             DAD     H        ;ADJUST FOR OUTPUT
F9E1 2B                             DCX     H        ;LOOP
F9E2 CDEAF9                         CALL    SERIALDELAY     ;OUTPUT DATA
F9E5 0D                             DCR     C        ;DECREMENT BIT COUNT
F9E6 C2DAF9                         JNZ     OLOOP
F9E9 C9                             RET
```

```
                         ;
                         ;
                         SERIALDELAY:
F9EA 7B                                 MOV     A,E
F9EB B0                                 ORA     B
F9EC 0F                                 RRC             ;TO THE SERIAL PORT
F9ED 0F                                 RRC
F9EE 0F                                 RRC             ;AT PROPER BIT
F9EF 3203FB                             STA     SERIALOUTPORT
F9F2 2D                                 DCR     L       ;DECREMENT SPEED
F9F3 00                                 NOP             ;PADDING
F9F4 C2EAF9                             JNZ     SERIALDELAY     ;LOOP UNTIL T
F9F7 7A                                 MOV     A,D     ;ROTATE
F9F8 1F                                 RAR             ;ONE
F9F9 7B                                 MOV     A,E     ;BIT
F9FA 1F                                 RAR             ;POSITION
F9FB 5F                                 MOV     E,A     ;TO THE
F9FC 7A                                 MOV     A,D     ;RIGHT
F9FD 1F                                 RAR             ;WITH END AROUND
F9FE 57                                 MOV     D,A     ;BIT PRESERVED
F9FF C9                                 RET

A>
```

```
TYPE

A-TYPE MEMCBIOS

B-TYPE MEMCBIOS.PRN


                 ;       CBIOS FOR MICROMATION 16K VERSION OF CP/M VERSION 1.3
                 ;
                 ;       COPYRIGHT (C) 1977, MICROMATION AND DIGITAL RESEARCH
                 ;
                 ;
                 ;
                 ;       FEB 17,1978
                 ;       DRIVERS FOR MEMOREX DRIVE
                 ;
 0010 =          MSIZE           EQU     16         ;SIZE OF OPERATING SYSTEM IN K
                                                    ;(CURRENTLY 16K).  THIS NUMBER
                                                    ;CHANGED FOR LARGER SYSTEMS.
 3E00 =          LOCATION        EQU     MSIZE*1024-512  ;ORG LOCATION FOR THE
                 ;
 3E00                    ORG     LOCATION           ;BASE OF BIOS IN 16K SYSTEM
                 ;
                 ;
                 ;       MAP OF SCRATCH AREA
                 ;
 FA00 =          SCRATCH         EQU     0FA00H             ;BASE ADDR OF RAM SCRA
                 ;
 FA10 =          PRESDSK         EQU     SCRATCH+10H
 FA6F =          BOOTSTACK       EQU     SCRATCH+6FH
 FA70 =          CONTROLBYTE     EQU     SCRATCH+70H
 FA73 =          RETRYCOUNT      EQU     SCRATCH+73H
 FA74 =          DMAADDR         EQU     SCRATCH+74H
 FA76 =          ADDRPTR         EQU     SCRATCH+76H
 FA77 =          TRACK           EQU     SCRATCH+77H
 FA79 =          SECTOR          EQU     SCRATCH+79H
                 ;
                 ;       PSEUDO PORTS IN ROM
                 ;
 FB00 =          DATAPORT        EQU     SCRATCH+100H
 FB01 =          MARKPORT        EQU     SCRATCH+101H
 FB02 =          DISKFUNCTION    EQU     SCRATCH+102H
 FB03 =          LOADPORT        EQU     SCRATCH+103H
 FB03 =          SERIALOUTPORT   EQU     SCRATCH+103H
                 ;
                 ;
                 ;
 F822 =          HOME    EQU     0F822H
 F88A =          SELDSK  EQU     0F88AH
 F845 =          SETTRK  EQU     0F845H
 F956 =          SETSEC  EQU     0F956H  ;SET SECTOR NUMBER
 F964 =          SETDMA  EQU     0F964H
                 ;
                 ;
 F8E3 =          DISKWRITE       EQU     0F8E3H
 F8C1 =          DISKREAD        EQU     0F8C1H
 F8D2 =          DATAXFER        EQU     0F8D2H
                 ;
                 ;
 FA71 =          SPEED           EQU     SCRATCH+71H
 F9D1 =          SERIALOUT       EQU     0F9D1H
 F9A0 =          SERIALIN        EQU     0F9A0H
                 ;
```

```
               ;
0000 =         CBASE   EQU     (MSIZE-16)*1024 ;BIAS FOR SYSTEMS GREATER THAN
2900 =         CPMB    EQU     CBASE+2900H
3106 =         BDOS    EQU     CBASE+3106H
2880 =         CCPM    EQU     CPMB-128
1500 =         CPML    EQU     $-CPMB
002A =         NSECTS  EQU     2AH     ;CHANGE FOR LARGER MEMORY
               ;
3E00 C32D3E            JMP     COLDBOOT
3E03 C3553E    EBOOT:  JMP     WBOOT
3E06 C3023F            JMP     CONST
3E09 C31B3F            JMP     CONIN
3E0C C32F3F            JMP     CONOUT
3E0F C3333F            JMP     LIST
3E12 C3333F            JMP     PUNCH
3E15 C3333F            JMP     READER
3E18 C322F8            JMP     HOME            ;HOME
3E1B C3343E            JMP     IOSELDSK                ;SELDSK
3E1E C345F8            JMP     SETTRK          ;SETTRK
3E21 C356F9            JMP     SETSEC          ;SETSEC
3E24 C364F9            JMP     SETDMA          ;SETDMA
3E27 C3343F            JMP     READ            ;DISKREAD
3E2A C3503F            JMP     WRITE           ;DISKWRITE
               ;
               COLDBOOT:
3E2D AF                XRA     A
3E2E 3210FA            STA     PRESDSK         ;INITIALIZE PRESENT DISK
3E31 C3DA3E            JMP     BOOT
               ;
               ;
               IOSELDSK:
3E34 2100FA            LXI     H,SCRATCH       ;POINT TO BOTTOM OF SCRATCH
3E37 3A10FA            LDA     PRESDSK         ;GET PRESENT DRIVE #
3E3A FE10              CPI     10H             ;CHECK FOR VALID #
3E3C D24B3E            JNC     GOSELDSK        ;GET OUT IF INVALID
3E3F 6F                MOV     L,A             ;POINT TO TRACK OF PRESENT DRI
3E40 2C                INR     L       ;INCREMENT TO NEXT BYTE
3E41 3A77FA            LDA     TRACK   ;GET PRESENT TRACK
3E44 77                MOV     M,A             ;STORE IN BUFFER
3E45 69                MOV     L,C             ;POINT TO SELECTED DRIVE BUFFE
3E46 2C                INR     L               ;NEXT BYTE
3E47 7E                MOV     A,M             ;GET TRACK OF SELECTED DRIVE
3E48 3277FA            STA     TRACK   ;UPDATE CONTROLLER
               GOSELDSK:
3E4B 79                MOV     A,C             ;LOAD SELECTED DRIVE
3E4C 3210FA            STA     PRESDSK         ;UPDATE DISK BUFFER
3E4F C38AF8            JMP     SELDSK          ;GO TO CONTROLLER
               ;
               ;
3E52 C9        ERRORV: RET                     ;NOT CURRENTLY USED
3E53 00                NOP                     ;RESERVED FOR FUTURE ERROR
3E54 00                NOP                     ;REPORTING
               ;
3E55 318000    WBOOT:  LXI     SP,80H
3E58 3A10FA            LDA     PRESDSK         ;GET PRESENTLY SELECTED DRIVE
3E5B 32D93E            STA     CURRDRIVE       ;STORE IN BUFFER
               STARTBOOT:
3E5E 0E00              MVI     C,0
3E60 CD8AF8            CALL    SELDSK          ;SELECT DRIVE A TO REBOOT
3E63 CD22F8            CALL    HOME
```

```
                        ;
                        ;       READ DISKETTE FOR TWO TRACKS, STARTING AT BOOT LOADER
3E66 018028                             LXI     B,CCPM  ;ONE SECTOR BOOT
3E69 CD64F9                             CALL    SETDMA
                        RDTRK:  ;READ THE FIRST/NEXT TRACK
3E6C 110000                             LXI     D,0     ;SECTOR NUMBER = 0000
                        RDSEC:  ;READ THE FIRST/NEXT SECTOR
3E6F 7B                                 MOV     A,E     ;E IS SECTOR NUMBER
3E70 FE1A                               CPI     26      ;
3E72 CAAD3E                             JZ      NXTTRK  ;0...25 COUNTS SECTORS
                        ;       GET SKEWED SECTOR NUMBER
3E75 21BF3E                             LXI     H,TRAN
3E78 19                                 DAD     D       ;HL IS ADDRESS OF SKEWED SECTO
3E79 7E                                 MOV     A,M     ;1...26 IN REG A
3E7A 13                                 INX     D       ;TO NEXT SECTOR
3E7B D5                                 PUSH    D       ;SAVE SECTOR NUMBER
3E7C C5                                 PUSH    B       ;SAVE DMA ADDRESS
3E7D 4F                                 MOV     C,A     ;READY FOR SECTOR SET
3E7E F5                                 PUSH    PSW     ;SAVE SKEWED SECTOR NUMBER
3E7F CD56F9                             CALL    SETSEC
3E82 F1                                 POP     PSW     ;COUNT TO DMA POSITION
3E83 E1                                 POP     H       ;COPY OF DMA BASE ADDRESS
3E84 E5                                 PUSH    H       ;BACK TO STACK
3E85 118000                             LXI     D,128   ;SECTOR SIZE
                        MUL:
3E88 3D                                 DCR     A       ;REGA*128
3E89 CA903E                             JZ      MUL1
3E8C 19                                 DAD     D       ;+128
3E8D C3883E                             JMP     MUL
                        MUL1:   ;HL IS DMA ADDRESS FOR THIS SECTOR
3E90 2274FA                             SHLD    DMAADDR ;STORE IT DIRECTLY
3E93 3A77FA                             LDA     TRACK
3E96 B7                                 ORA     A
3E97 CAA23E                             JZ      RELP    ;IF TRACK 0, THEN CONTINUE
3E9A 3A79FA                             LDA     SECTOR  ;IF TRACK 1 AND SECTOR - 18
3E9D D612                               SUI     18      ;THEN SKIP THE READ
3E9F F2A83E                             JP      SKIPREAD
                        RELP:
3EA2 CD343F                             CALL    READ    ;READ THE DATA
3EA5 C25E3E                             JNZ     STARTBOOT               ;STAY HERE WHILE ERRO
                        SKIPREAD:
3EA8 C1                                 POP     B       ;RECALL BASE DMA ADDRESS
3EA9 D1                                 POP     D       ;RECALL SECTOR NUMBER
3EAA C36F3E                             JMP     RDSEC   ;FOR ANOTHER SECTOR
                        NXTTRK:
3EAD 3A77FA                             LDA     TRACK   ;0,1?
3EB0 B7                                 ORA     A
3EB1 C2DA3E                             JNZ     BOOT    ;STOP AT TRACK 1
3EB4 0E01                               MVI     C,1     ;SEEK 1 IF NOT
3EB6 CD45F8                             CALL    SETTRK
3EB9 018035                             LXI     B,CCPM+26*128   ;MOVE TO NEXT TRACK P
3EBC C36C3E                             JMP     RDTRK   ;TO READ THE ENTIRE TRACK
                        ;
                        ;
                        ;
                        ;
```

```
                TRAN:           ;TRANSLATION TABLE FOR SKEW FACTOR
                ;
3EBF 01                         DB      01H
3EC0 05                         DB      05H
3EC1 09                         DB      09H
3EC2 0D                         DB      0DH
3EC3 11                         DB      11H
3EC4 15                         DB      15H
3EC5 19                         DB      19H
3EC6 03                         DB      03H
3EC7 07                         DB      07H
3EC8 0B                         DB      0BH
3EC9 0F                         DB      0FH
3ECA 13                         DB      13H
3ECB 17                         DB      17H
3ECC 02                         DB      02H
3ECD 06                         DB      06H
3ECE 0A                         DB      0AH
3ECF 0E                         DB      0EH
3ED0 12                         DB      12H
3ED1 16                         DB      16H
3ED2 1A                         DB      1AH
3ED3 04                         DB      04H
3ED4 08                         DB      08H
3ED5 0C                         DB      0CH
3ED6 10                         DB      10H
3ED7 14                         DB      14H
3ED8 18                         DB      18H
                ;
3ED9 00         CURRDRIVE       DB      0
                ;
                BOOT:
                ;
                ; SET THE SOFTWARE UART SPEED
                ; IF THE CBIOS IS MODIFIED FOR AN I/O BOARD THE
                ;  CODE TO PROGRAM THE UART SHOULD BE
                ;  PUT HERE AND THE INSTRUCTION TO SET THE SOFTWARE
                ;  UART SPEED REMOVED.
3EDA 210700             LXI     H,0007H
3EDD 2271FA             SHLD    SPEED
                ;
                ;
3EE0 3EC3               MVI     A,0C3H
3EE2 320000             STA     0
3EE5 21033E             LXI     H,EBOOT
3EE8 220100             SHLD    1
3EEB 320500             STA     5
3EEE 210631             LXI     H,BDOS
3EF1 220600             SHLD    6
3EF4 018000             LXI     B,80H
3EF7 CD64F9             CALL    SETDMA
3EFA FB                 EI
3EFB 3AD93E             LDA     CURRDRIVE       ;ACTIVE DISK
3EFE 4F                 MOV     C,A
3EFF C30029             JMP     CPMB
                ;SOFTWARE UART CONSOLE ROUTINES
                CONST:
3F02 060A               MVI     B,10            ;EACH LOOP = 35 MICROSECONDS
                CONST1:
3F04 3A02FB             LDA     DISKFUNCTION    ;LOOK FOR BIT
3F07 1F                 RAR
3F08 1F                 RAR
3F09 D2123F             JNC     CONSTFD         ;FOUND BIT
3F0C 05                 DCR     B
3F0D C2043F             JNZ     CONST1          ;LOOP FOR SPECIFIED TIME
3F10 AF                 XRA     A               ;ZERO A - NO CHARACTER FOUND
```

```
               CONSTFD:
3F12 CDA0F9            CALL    SERIALIN        ;GET BYTE
3F15 322E3F            STA     OLDBYTE         ;SAVE BYTE
3F18 3EFF              MVI     A,0FFH
3F1A C9                RET
               CONIN:
3F1B 3A2E3F            LDA     OLDBYTE         ;WAS A BYTE THERE
3F1E FE00              CPI     0
3F20 CA2A3F            JZ      CONIN1          ;NO
3F23 F5                PUSH    PSW             ;SAVE A
3F24 AF                XRA     A
3F25 322E3F            STA     OLDBYTE         ;ZERO OLDBYTE
3F28 F1                POP     PSW
3F29 C9                RET
               CONIN1:
3F2A CDA0F9            CALL    SERIALIN
3F2D C9                RET
3F2E 00        OLDBYTE DB      0
               CONOUT:
3F2F CDD1F9            CALL    SERIALOUT
3F32 C9                RET
               LIST:
               PUNCH:
3F33 C9        READER: RET
               ;
               ;
               ;
               ;       ERROR CHECKING READ AND WRITE RTNS FOR
               ;       MICROMATION CBIOS
               ;
               ;               AUGUST 24, 1977
               ;
               ;
0014 =         RETRYLIMIT              EQU     20      ;NUMBER OF RETRIES
               READ:
3F34 CDA83F                    CALL    READYNOW
3F37 AF                        XRA     A       ;GET A ZERO
3F38 3273FA                    STA     RETRYCOUNT
               RETRYREAD:
3F3B CDC1F8                    CALL    DISKREAD        ;CALL PROM RTN
3F3E 3E00                      MVI     A,0             ;ZERO ACCUM, LEAVE F
3F40 C8                        RZ                      ;IF NO ERROR THEN RE
3F41 CD523E                    CALL    ERRORV
3F44 CD7A3F                    CALL    ERRORCHECK
3F47 C23B3F                    JNZ     RETRYREAD       ;IF ERROR RETRY
3F4A CDD2F8                    CALL    DATAXFER        ;TRANSFERS DATE
3F4D 3E0F                      MVI     A,0FH           ;ERROR CODE
3F4F C9                        RET                     ;EITHER RETRY SUCCES
               ;
               ;
               WRITE:
3F50 CDA83F                    CALL    READYNOW
3F53 3A02FB                    LDA     DISKFUNCTION    ;READ STATUS
3F56 E604                      ANI     04H             ;CHECK WRITE PROTECT
3F58 CA643F                    JZ      NOTPROTECT
3F5B 11D33F                    LXI     D,WPMSG         ;SET UP ADDR OF MSG
3F5E CDBB3F                    CALL    PRINTMSG
3F61 C30000                    JMP     0               ;WARM BOOT
                       NOTPROTECT:
3F64 AF                        XRA     A               ;GET A ZERO
3F65 3273FA                    STA     RETRYCOUNT
               RETRYWRITE:
3F68 CDE3F8                    CALL    DISKWRITE       ;CALL PROM RTN
3F6B 3E00                      MVI     A,0             ;ZERO ACCUM, LEAVE F
3F6D C8                        RZ                      ;IF NO ERROR THEN RE
```

```
3F6E CD523E                    CALL    ERRORV
3F71 CD7A3F                    CALL    ERRORCHECK       ;
3F74 C2683F                    JNZ     RETRYWRITE       ;IF ERROR, RETRY
3F77 3E0F                      MVI     A,0FH            ;RETURN ERROR CODE
3F79 C9                        RET                      ; MORE THAN RETRYLIM
               ;
               ;
               ERRORCHECK:
3F7A 3A73FA                    LDA     RETRYCOUNT       ;GET NUMBER OF RETRI
3F7D 3C                        INR     A                ;ADD ONE
3F7E 3273FA                    STA     RETRYCOUNT
3F81 FE15                      CPI     RETRYLIMIT+1     ;HAVE WE RETRIED ENO
3F83 C8                        RZ                       ;IF YES, RETURN WZE
3F84 3E77                      MVI     A,77H            ;IS ERROR A TRACK ER
3F86 BD                        CMP     L                ;L HOLDS LOCATION OF
3F87 C0                        RNZ                      ;IF NOT AT 77 THEN N
               ;       TRACKERROR
3F88 3A73FA                    LDA     RETRYCOUNT
3F8B D60A                      SUI     10
3F8D F8                        RM
3F8E 3A77FA                    LDA     TRACK
3F91 4F                        MOV     C,A              ;GET TRACK IN C
3F92 3A79FA                    LDA     SECTOR
3F95 47                        MOV     B,A              ;GET SECTOR IN B
3F96 C5                        PUSH    B                ;SAVE TRACK AND SECT
3F97 CD22F8                    CALL    HOME             ;PROM RTN TO HOME HE
3F9A C1                        POP     B                ;RESTORE TRACK AND S
3F9B C5                        PUSH    B                ;SAVE TRACK AND SECT
3F9C CD45F8                    CALL    SETTRK           ;PROM RTN TO FIND TR
3F9F C1                        POP     B                ;GET TRACK AND SECTO
3FA0 48                        MOV     C,B              ;GET SECTOR IN REG C
3FA1 CD56F9                    CALL    SETSEC           ;PROM RTN TO FIND SE
3FA4 3EFF                      MVI     A,0FFH           ;TURN OFF THE
3FA6 3D                        DCR     A                ;     ZERO FLAG FOR
3FA7 C9                        RET
               ;
               ;
               READYNOW:
3FA8 3A02FB                    LDA     DISKFUNCTION     ;CHECK STATUS
3FAB 07                        RLC             ;CHECREADY LINE
3FAC D8                        RC              ;CARRY SET,DRIVE READY
3FAD 11C93F                    LXI     D,NOTRDYMSG      ;POINT TO MSG BUFFE
3FB0 CDBB3F                    CALL    PRINTMSG
               READYLOOP:
3FB3 3A02FB                    LDA     DISKFUNCTION
3FB6 07                        RLC
3FB7 D8                        RC
3FB8 C3B33F                    JMP     READYLOOP        ;LOOP TILL READY
               ;
               PRINTMSG:
3FBB 1A                        LDAX    D       ;GET FIRST CHARACTER
3FBC FE24                      CPI     '$'     ;END DELIMITER?
3FBE C8                        RZ              ;RET IF DONE
3FBF D5                        PUSH    D
3FC0 4F                        MOV     C,A
3FC1 CD2F3F                    CALL    CONOUT
3FC4 D1                        POP     D
3FC5 13                        INX     D
3FC6 C3BB3F                    JMP     PRINTMSG         ;LOOP UNTIL DONE
               ;
               NOTRDYMSG:
3FC9 4E4F542052        DB      'NOT READY$'
                       ;
               WPMSG:
3FD3 5752495445        DB      'WRITE PROTECTED$'
```

MOVCPM DOCUMENTATION

If the CBIOS is not changed, the following procedure may be used to generate new systems for any size memory.

Notes the file CPM.COM has been included with this diskette, which enables you to generate a CP/M system for any memory size, up to 64K bytes. the command

```
CPM <cr>
```

(where <cr> denotes the carriage-return key) loads the CPM.COM program and gives it control. This program then examines the current memory configuation, and produces a new CP/M system which is relocated to the top of the memory (actually, the highest contiguous RAM area is used). The newly constructed CP/M system then gets control, and the system starts with the normal sign-on message.

The command

```
CPM * *
```

constructs a new version of the CP/M system, but leaves it in memory, ready for a sysgen operation. The message

```
READY FOR "SYSGEN" OR
"SAVE 32 CPMxx.COM"
```

is printed at the console upon completion, where xx is the memory size in kilobytes. The operator can then type

```
        SYSGEN                   to start the system generation
with the response
        GET SYSTEM (Y/N)?n       user must respond with "n"
and the message
        PUT SYSTEM (Y/N)?y       user must respond with y

        DESTINATION ON B, THEN TYPE RETURN
```

Place the new diskette on drive B, and type a return when ready (note that if you answer with an "a" rather than a "y" to the prompt above, SYSGEN will place the CP/M system on drive A instead of drive B). Sysgen will then type

```
FUNCTION COMPLETE, REBOOTING
```

The user can then go through the reboot process with the old or new diskette.

The operator could also have typed

```
SAVE 32 CPMxx.COM
```

at the completion of CPM.COM, which would place the CP/M memory image on disk. In this case, the relocated memory image can be "patched" to include custom I/O drivers, as described in the CP/M Alteration Guide.

(over)

Note that the memory size can be given explicitly to the CPM.COM program when it is started in order to override the internal mechanisms which determine the amount of memory on the system. In this case, the operator must type

```
CPM xx
```

or

```
CPM xx *
```

where xx is the memory size in decimal kilobytes. The first form produces a CP/M system which operates in xx kilobytes, and starts the newly created system when the relocation is complete. The second form creates the new system, but leaves it in memory for a sysgen or save operation.

For example, the invocation

```
CPM 48 *
```

starts CPM.COM, and creates a 48K system in memory. Upon completion, the message

```
READY FOR "SYSGEN" OR
"SAVE 32.CPM48.COM"
```

is typed. The operator can then perform the sysgen or save operation as described above. Note that the newly created system is serialized with the number attached to your original diskette, and are subject to the conditions of the Software Licensing Agreement included in this package.