# FLOPPY DISK SYSTEM

# CONTROLLER

# FIB
# FIRMWARE

## FUNCTIONAL DESCRIPTION

The controller for the Floppy System is designed to make the operation as easy as possible for the end user. All power on initialization sequences and error recovery procedures are contained within the firmware in the floppy disk controller. Hence, if a hardware error is indicated by the floppy disk controller, it is an unrecoverable error and the user need not have error recovery procedures in his/her software. Similarly, the floppy controller is designed to monitor drives going NOT READY and becoming READY again, and to insure that proper head position is performed on these drives.

The communication between the master MPU and the IFM uses an output instruction (to a port address which has been selected on the floppy disk controller board) for passing single byte commands to the controller. The floppy disk controller has a DMA access to the master microprocessor's memory for retrieving string commands and transferring data and status back to the MPU. The two types of commands, (i.e., the single byte command, and the data string command) are described in more detail below.

The FIF board has provisions for using interrupts to assist the MPU program in determining when a command has been completed. The interrupt request line can be attached to any of the eight request lines for the PIB via a wired jumper. An interrupt request is generated whenever the processing of a command string is complete (i.e., the status byte is set non-zero). It is cleared whenever a command is accepted from the MPU via the output instruction. It should be noted that undefined commands are treated as NOT's and can be used to clear the interrupt request.

The master microprocessor is put into a wait state whenever it does an output instruction to the port of the floppy disk controller. It is held in this state until the floppy disk controller has taken the command from the master microprocessor and started processing it. The completion of an operation for single byte commands is signified by the IFM's ability to accept another command. The completion of a command string obtained from the MPU's memory is signified by the passing of a proper status word to the master microprocessor. Note that these command strings must always be in RAM memory since part of the command string is a location for the IFM to transmit back the status of the last operation.

## THEORY OF OPERATION

### FIF/IFM Firmware Description

This description will be divided into three separate sections. The first section will describe the major storage areas and the use of each area during the routine. The second section will give an individual description of frequently called subroutines. The third section will consist of a straight-line description of the code in the order that it is normally executed.

### Part I Description of the Major Storage Areas

This description will consist of given the label used for the storage area in the program and then describing the use of this storage area.

PBAS

32 bytes in length and is used to contain the pointers to the address in main memory for retrieving command strings. These pointers are originally set to dummy values contained within the firmware and can be modified by the main program as described in the Users Guide.

PTRK

1 byte. Contains the physical track number that we are presently working on.

LTRK

1 byte. Contains the logical track number we are presently working on. Note that for normal operation the logical track number and physical track number will coincide. However, for strict IBM compatibility in the media readable format the logical track and physical track may differ if there are defective tracks on the disk.

SECT

1 byte. This byte contains the present sector we are working on, and has a value from 1 to 26. During the Read All routine it is used to contain the delay time before we start reading.

BUFA

2 bytes. This is the address pointer to the location in main memory where the data to be written on the floppy is contained or where the data read from the floppy is to be stored.

FUNC

1 byte. This byte in the four MSBs contains the function we are presently executing. This function is a number in the range is from 0 to 5. The fact that the logical track differs from the physical track has been removed prior to storing the general function in this byte.

INTF

1 byte. This is the interupt flag and controls whether the interupt routine returns a fatal error signifies that a drive is not ready. In all cases, except when we are testing for drive ready, the interupt flag has a value of 0 and this indicates that the drive was ready and has become not ready during an operation. This is a fatal error. In this case no retries are attempted.

RTRC

1 byte. This is the retry counter and is used for counting the number of attempts to do an operation when an error occurs. Ten attempts to retry an operation are made prior to telling the main program that an error has occurred.

STPT

2 bytes. This holds the address in main memory of the status byte. This byte is the second byte in all command strings.

HDFL

This byte contains the head flag to indicate to the firmware whether or not the head on the selected disk drive is lowered. If it contains a value of 0 the head is raised. If it contains a value of 1 the head has been lowered. During the time after an operation has been completed and prior to lifting of the head a value of 2 is put in this location and then after one revolution the head is raised.

DRVO

1 byte. This byte contains the present location of the head for drive 0. The most significant bit is the software write protect flag for this drive. If it is a 1, the drive is protected. A value is 7FH for the track location indicates that the drive should be restored over track 0 prior to any operation being performed.

This value is put into this location upon power up to indicate that all drives should be restored. It is also inserted whenever a drive is detected to be off line and should be restored over track 0 for alignment when it comes on line. It is also set when a track address error has occurred and we must restore to track 0 to attempt to relocate the desired track.

DRV1 1 byte. Refer to DRV0

DRV2 1 byte. Refer to DRV0.

DRV3 1 byte. Refer to DRV0.

TDRV

1 byte. This byte contains, in the four least significant bits, a select bit for the drive we are about to operate on. It is loaded by the drive address routine, DRAD and is references in the drive ready routine, DRDY when we are going to see if a selected drive is ready. The least significant bit corresponds to selecting drive 0 while bit 3 corresponds to selecting drive 3 and so forth.

SDRV

1 byte. This byte contains in the 4 least significant bits the select bit for the present drive.

DATB

133 bytes. This area is the data buffer location for the operating program. Note that the label is at the bottom of the stack so that data is pushed into this using the PUSH instruction during operation and popped off it during a write operation.

INTE

1 byte. This byte contains the present status of the interupt bit.

STAK

26 bytes. This is the stack area used for the program execution.

## Description of the Frequently Called Subroutines

The format for this section is to give the four-letter name for the subroutine followed by a description of the operations performed within the subroutine.

DRAD

This is the drive address select routine. It operates by looking at the four least significant bits of the B register to determine what drive, if any, is selected. For the drive that is selected it sets that bit in TDRV and puts the address of DRV0 through DRV3, whichever is appropriate, in register pair H. Note that the routine can be called repetitively to perform operations on different drives.

Each time it sets up the address for a selected drive it clears the corresponding bit in the B register. Upon entry, if no drive is selected it returns with the 0 flag (Z) set. If any drive is selected it returns with Z = 0.

DRDY

This is the drive ready test routine. It tests the drive whose address is contained in register pair H to see if it is ready. Upon entry, it compares the value of TDRV and SDRV to see if the same drive is being reselected. If the same drive is being selected and the head on that drive is already lowered it does not reselect the head since this would be a duplicate operation. If either it is a different drive or the head has already been raised, the selected drive bit is set in control register 1. The interupts are then enabled follow by a SUB A register instruction. If, on the following instruction the A register has a value of 1 this indicates that the selected drive was not ready and the restore flag is set in its status location. The routine exits with the Z = 0 if the selected drive is not ready.

If the selected drive is ready, the routine checks to see if the restore flag for this drive is set. If it is, it checks to see if this drive is over track 0. If the drive is over track 0 it executes a step out to insure that the drive goes off track 0 and then steps back in to realign the track 0 position. If the drive is already off track 0 it attempts to move the head out until a track 0 indication occurs. If this indication does not occur within a maximum of 77 steps the drive is determined to be inoperable and a failure exit is taken. If the drive does go back over track 0 then the track address set to 0 and a successful exit is used.

LDHD

The load head routine is called when the formatter is going to perform the final step to reach the desired track. This routine loads the head and sets the greater than 43 bit if it is required.

CRCC Routine

This routine will compute the cyclic redundancy check bytes for data stored in a buffer location pointed to by register pair D, and for the number of bytes contained in register C. A complete description of the operation of this routine can be found in other INTEL publications.

HDRC

This routine computes the CRC for a address sector in the disk if calls the CRC routine with each byte as it would appear in the header sector and returns with the value of the CRC for this header in register pair H.

## SYNC

This is the major routine used to find the desired sector on a track. Upon entry it calls HDRC to compute the expected CRC for this track. It then sets up in the registers the values of the track, sector, clock and address mark patterns so it can compare all of these within the time limit for each byte.

After the initial set up, the PLO is sunk and the routine goes into a wait pattern waiting for a clock byte which has bit 5 missing. Once this byte has been found the clock pattern and data pattern are read into register pair H. The clock pattern is compared first to see if it is appropriate. If it is not, the routine recycles and tries sinking the PLO again to obtain a new clock pattern. Each time the routine attempts to resink it checks to see if an index pulse has occurred. If two successive index pulses occur, it determines that it cannot sink properly on this track and indicates an error.

If the clock byte is correct, it does an increment on the data byte. (Note that the data byte should initially have a value of FE Hex and therefore 2 increments will cause it to be 0.) Next, it reads and compares the track address byte. If it is not equal a branch is made to determine whether or not the data byte was correct. If the data byte was not correct then it was a false sink and we attempt to resink. If the data byte was correct, then the track address we read was incorrect and we have a track address error which will cause us to restore the drive and reattempt the operation.

If the track address compares another increment is done on the data byte. This should set the 0 flag if the data byte was correct. Following this, the next byte is read from the disk. This byte should contain 0. After this is read the jump on whether or not the data byte of the ID mark was correct is performed. If it is correct, the CRC value is popped after the stack and the sector number is read and compared from the disc.

If it compares correctly, the second zero byte is read and ORed with the first 0 byte to ensure that they are both zero.

Then the first CRC byte is read and compared, and finally, the second CRC byte is read and compared. If both of these bytes compare, 10 more bytes are read to locate the head one byte from where the write enable should be turned on, or 2 bytes prior to where the PLO should be resunk for reading the data section of this sector. Following the reading of these 10 bytes the routine is exited.

## RTRY

This routine is used to increment the retry counter. If 10 attempts have already been made to perform this operation, then an error is taken. If not, the routine exits to its calling location where another attempt is made to perform the operation.

## Straight Line Description of the Program

Prior to starting the main description, the interrupt processing routine will be described. The interrupt instruction used in the FIF is a RST 7 which causes an interrupt to location 38H. The interrupt routine looks at the intertupt flag. If it is 0, a selected drive has become not ready, and a fatal error is determined. No retries are are attempted because the drive or controller is obviously malfunctioning. A return is made to the main program through the error exit.

If the interrupt flag is a 1, it indicates that we are testing to see if the selected drive is ready, and the value of 1 is returned in register A to signify to the drive ready test routine that the selected drive is not ready.

On system power-up or RESET, the program sets the stack pointer and checks to see if drive Ø is ready. If it is ready, the program executes the bootstrap sequence as follows:

1. A JMP Ø instruction is DMA'd into system memory locations Ø, 1 and 2, so that the front panel operations RESET, EXAMINE Ø, and RUN cause the system processor to enter on infinite wait loop.

2. A small program to read sector 1 of track Ø from drive Ø is DMA'd into system memory starting at location 8Ø.

3. The low byte of the address in the JMP instruction is changed to cause the system processor to jump into the bootstrap code above location 8Ø.

If the drive is not ready, or when the bootstrap sequence is complete, the default pointers are set up. Once this is done, the software protect flag is turned on for all drives. No operation need be performed on the individual drives since this will be done by the scan loop while waiting for commands from the main processor. The scan loop actually has 2 separate sections. The first section is used when the head is down on a selected drive. This inhibits selecting of other drives since the head would be raised. We stay in this loop until two revolutions have occurred or another command request is received from the main processor. If two index pulses have occurred, the head on the selected drive is raised and the flags are set appropriately.

If the head was not down, we go into the section of the scan loop where we look at each drive. If a drive has become ready and a restore flag is set on it, then the restore is done on that drive. If a drive is ready and has been ready, no action is performed. If a drive becomes not ready, the restore flag is set on it. These operations are performed by calling the DRAD and DRDY subroutines. In either of these loops a check is made to see if the main program is requesting an action. If it is, a jump is immediately taken to location ACTN where the type of action requested is determined. An action code or 0 calls for a command string to be read from main memory and processed. This operation will be described in the next section.

An action code of 1 is the setting of a new address in main memory for a pointer. The address of the pointer is formed in subroutine PADO and then RBYT called to read the next two bytes from main processor and to store them into the proper pointer location.

An action code of 2 calls for a restore to be executed on all drives whose bit is set in the four least significant bits. Note that the subroutine DRAD is used to set the addresses for each of the bits and determine when all drives have been processed. For each selected drive, the flag is set in its respective status byte. The restore actually will be done in the scan loop at a later time.

An action code of 3 calls for the software write protected to be set in for each drive whose bit is set in the four least significant bits. Again, the same routine DRAD is used to cycle through the four least significant bits.

An action code of 5 calls for the software write protect bit to be cleared for each drive which is selected in the four least significant bits. If the action code is greater than 5, it is considered to be a NOP and the routine returns to the scan loop.

For an action code of 0, the address of the string in main memory is determined. Then the string is analyzed and as a function of the operation code the parameters are checked to ensure they have the proper value. This is done in a straight line fashion and the routine ends up at ACT6 with all of the parameters except for the separate logical track if it is a special IBM function. At this time, if it is a special IBM function, the logical track number is read and stored in LTRK after being parameter checked.

The drive is then checked to ensure that it is ready. If the drive is ready, and the write protect status is OK, the drive is then positioned. Prior to positioning the drive, the retry counter and interrupt flags are set to 0 since any interrupt from now on will be a fatal error on the part of the drive. The difference between present track and the desired track is computed. If it is 0, we go out to test if the head is loaded. If the head is already loaded, then we may continue without any time delay. If the head has not been loaded, we then come back to the section of the routine where the track step delay is performed. Just prior to the last step for the correct track, the head is loaded. The 16 millisecond delay is used.

If this operation requires steps in or out, register E is set with the direction bit and the head is revised.

The program then issues a track to track step each 6 milliseconds until there is 1 track to go. At this time, the load head routine is called and after the 6 millisecond step delay, we go through a 10 millisecond head settling delay. At the completion of the drive positioning, we are now ready to accomplish whatever operation was called for in the action code. This occurs at ACTB where the function in four MSBs of FUNC is interpreted and we jump to the proper routine. The separate routines for processing will be described in each of the following paragraphs.

The read all routine, RALL, starts by setting for 64 bytes to be read. Note that the clock and data values are saved for each byte and this is 128 bytes to be transferred. After the set up, the routine waits until a index pulse has occurred and then delays for the proper number of milliseconds (0 to 255). After the delay, the PLO is sunk and the routine reads the clock and data pattern from 64 successive bytes storing them in the data buffer area DATB. Then the data buffer area is moved to the proper location in main memory and the stack pointer is reset to it's proper value. The routine then goes to the scan loop to wait for another operation.

WRIT

The write routine is used to write a sector of data onto the disk. The 128 bytes are first retrieved from the main memory. After this, the CRC is computed. Note that 129 bytes are used to compute the CRC since the value of the data in the Data Address Mark is used as the first byte for the CRC computation. This is stored in memory so it is readily available using pop instructions during the execution of the actual write on the disk. The sector location routine SYNC is then called to get in the proper position on the disk for writing the data in this sector.

SYNC exits after reading 10 trailing 0 bytes following the address header. The setup is then done for the write routine and the write enable is turned on at approximately the end of the 11th byte. At this time, the data loop starts by writing 6 bytes of zeros followed by the Data Address Mark, 128 bytes of data, the two CRC bytes and 1 trailing 0 byte. At this time, a successful write has been completed and the write enable is cleared. A normal return is taken.

## WDSC

The next routine WDSC writes the Deleted Data Address Mark on an individual sector. It again starts by calling SYNC to find the proper sector on this track. Once SYNC exits, the set up is done in the registers to perform the write. A delay is executed prior to setting the write enable. This ensures that the 11th byte after the header will pass prior to the write enable being set. Once write enable is set, 6 bytes of zeros are written followed by the Deleted Data Address Mark and a trailing zero byte. At the completion of this byte, the write enable is cleared and a successful return is taken.

## Read Routine

This routine is used both for the check read function and the read function. The difference being that after the data is read and the CRC is checked, for a Check Read the data is not moved back to the main memory.

The routine starts by calling the SYNC to position to the proper sector. After returning from SYNC, the parameters for checking the data address mark are put into the registers. A delay is then executed to ensure that the head has passed the area where the write current is turned on during a write operation before syncing the PLO. Then the PLO is sunk and the read routine goes into the wait mode waiting for the next byte with a missing clock pulse to occur. When this byte is received, both the clock and data from the byte are read into register pair H. First the clock byte is compared to ensure that it has the proper value (Hex C7). If this value is proper, the data byte is compared. If the data byte does not compare with the value for a normal sector (FEH), a branch is made to check to see if this is a Deleted Data sector (FBH). If a Deleted Data Address Mark has been found, there is a Deleted data read error and no retries are attempted, and an error exit is immediately taken. If it is neither a Data nor a Deleted Data Address Mark, it is an error and a retry is attempted.

After the data address mark has been successfully read, the routine reads 130 bytes. The first 128 of these are data and the last two are the CRC bytes. The interrupt is disabled after he CRC bytes. The CRC is then checked, a byte at a time. Note that the data pattern from the Data Address Mark is the first byte processed by the CRC routine. The 130 bytes are then processed and the result should be a 0 CRC value, if a proper read has been performed. If not, the CRC error is set and a retry attempt is made. If all the data is correct, the value in FUNC is checked to see if it is a check read or a read. If the function is a read, a jump is made to return the data to the main memory. If it is a check read, a normal return is made.

FRMT

This is the format routine which writes all the gaps and index marks on a track for later use by the read and write routines. Due to the time constraints during writing, the set up must be made for all variable parameters prior to initiating the formatting of a track. Hence, upon entry the format routine calculates the CRC value for each header for each sector to be written. This is done using the routine HDRC and the resulting values are stored in the data buffer area. Note that the CRC value for the data sectors is identical for all 26 sectors since they all have the same address mark, followed by 128 bytes of 0. Hence, this CRC is used as constant. Once the header CRC's have been computed, the routine does the set up for starting a write. Then it waits for the index mark at which time it is to start writing.

At this time, it sets the write enable and writes the last 46 bytes of GAP IV. Note that whenever a gap is written in this routine, the counter is always set to value one less than the size of the gap and then the final byte is written. The reason for this is to permit the maximum set up time for writing the next byte which is always a special clock pattern byte.

A sector consists of first writing GAP I which is 32 bytes in length. Note that in this case the counter is set to 30 since a single trailing 0 byte is written after the CRC value at the end of each sector. In the case of the first GAP I (prior to sector 1), the gap is only 31 bytes in length.

After writing GAP I, the Index Address Mark is written followed by the track number, a zero byte, the sector (contained in register B throughout the routine), another zero byte, followed by the 2 CRC bytes (which have previously been stored in a data area). This is followed by GAP II which is 17 bytes in length. Hence, the counter is set to a value of 16. The end of GAP II is followed by the Data Address Mark and 128 bytes of zeros (the data field). The two CRC bytes, which are common for all data sectors, are then written followed by a trailing zero byte.

The sector number is checked to see if the format for 26 sectors has been written If not, a return is made to write the next sector. If the 26 sectors have been written, then the first section of GAP IV is written. This consists fo writing 0's until the index pulse occurs. At this time the write enable is cleared and a successful return to the main routine is taken.

```
                        ;26 OCT 76. BRH. ADDED BOOTSTRAP LOADER.
                        ;21 OCT 76. BRH. MERGED PROGRAM INTO ONE FILE.
                        ;
                        ; ******** FLOPPY INTERFACE FIRMWARE ********
                        ;
                        ; IFM RAM ADDRESSES
                        ;
0800 =                  PBAS    EQU     800H     ;RAM STORAGE FOR COMMAND POINTERS
0820 =                  PTRK    EQU     PBAS+32  ;PHYSICAL TRACK NUMBER
0821 =                  LTRK    EQU     PTRK+1   ;LOGICAL TRACK NUMBER
0822 =                  SECT    EQU     LTRK+1   ;SECTOR NUMBER
0823 =                  BUFA    EQU     SECT+1   ;ADDRESS OF BUFFER IN MAIN MEMORY
0825 =                  FUNC    EQU     BUFA+2   ;PRESENT FUNCTION IN 4 MSB
0826 =                  INTE    EQU     FUNC+1   ;INTERRUPT FLAG (LATCH CONTENTS)
0827 =                  INTF    EQU     INTE+1   ;INTERRUPTS: 1 FOR TESTING DRIVE,
                                                 ;  0 IF INTERRUPT IS FATAL ERROR
0828 =                  RTRC    EQU     INTF+1   ;RETRY COUNTER FOR ERROR PROCESSING
0829 =                  STPT    EQU     RTRC+1   ;HOLDS STATUS BYTE ADDRESS
082B =                  HDFL    EQU     STPT+2   ;0 IF HEAD UP, 1 IF DOWN,
                                                 ;  2 IF WAITING TO BE RAISED
082C =                  DRV0    EQU     HDFL+1   ;DRIVE TRACK LOCATION IN 7 LSB.
082D =                  DRV1    EQU     DRV0+1   ;  MSB IS 1 IF DRIVE IS SOFTWARE
082E =                  DRV2    EQU     DRV1+1   ;  WRITE PROTECTED.  7 LSB = 7F
082F =                  DRV3    EQU     DRV2+1   ;  IF DRIVE MUST BE RESTORED.
0830 =                  TDRV    EQU     DRV3+1   ;DRIVE SELECT BIT FOR NEXT DRIVE
0831 =                  SDRV    EQU     TDRV+1   ;DRIVE SELECT BIT FOR THIS DRIVE
08B6 =                  DATB    EQU     SDRV+133 ;TOP OF 133 BYTE DATA BUFFER
0900 =                  STAK    EQU     900H     ;TOP OF STACK
0900 =                  RAM     EQU     900H     ;START OF DEBUG RAM
                        ;
                        ; POWER UP ROUTINE
                        ;
0000                            ORG     0
0000 310009                     LXI     SP,STAK ;SET STACK POINTER
0003 97                         SUB     A       ;SET PARAMS FOR DRDY...
0004 322C08                     STA     DRV0
0007 323108                     STA     SDRV
000A 0601                       MVI     B,1     ;FORM DRIVE 0 ADDRESS...
000C CDBD02                     CALL    DRAD
000F CDE302                     CALL    DRDY    ;CHECK TO SEE IF DRIVE 0 IS READY
0012 C28100                     JNZ     SCLPH   ;SKIP BOOTSTRAP IF NOT READY
0015 97                         SUB     A       ;RESET SYSTEM MEMORY BIT...
0016 320640                     STA     4006H
0019 67                         MOV     H,A     ;ZERO HL...
001A 6F                         MOV     L,A
001B 3EC3                       MVI     A,0C3H  ;GET JUMP OPCODE
001D 320080                     STA     8000H   ;STORE AT 0 IN SYSTEM MEMORY
0020 220180                     SHLD    8001H   ;STORE ADDRESS TO JUMP TO
0023 218080                     LXI     H,8080H ;START OF BOOT CODE IN SYSTEM
0026 114200                     LXI     D,SYSCM ;POINTER TO BOOT CODE IN PROM
0029 0E1F                       MVI     C,SYSLN ;LENGTH OF BOOT CODE
002B CD9103                     CALL    DMOV    ;TRANSFER CODE TO SYSTEM MEMORY
002E 3E87                       MVI     A,SYSTR ;POP IN LO BYTE OF JUMP ADDRESS
0030 320180                     STA     8001H   ;  SO SMPU JUMPS TO BOOT CODE
0033 C38100                     JMP     SCLPH   ;CONTINUE AT SCAN LOOP HEADER
                        ;
```

```
0038                  ORG   38H
0038 3A2708   INTR:   LDA   INTF
003B B7               ORA      A
003C C0               RNZ
003D 0E91             MVI   C,91H
003F C38302           JMP   AERX
              ;
              ; SYSTEM BOOT CODE
              ;
0042 21       SYSCM:  DB        21H       ;READ SECTOR COMMAND
0043 00       STATS:  DB        00H       ;STATUS BYTE
0044 00               DB        00H       ;HI BYTE OF TRACK #
0045 00               DB        00H       ;LO BYTE OF TRACK #
0046 01               DB        01H       ;SECTOR #
0047 0000             DW        0000H     ;ADDRESS OF BUFFER
              ;
0049 97       START:  SUB       A         ;MAKE ZERO
004A 328100           STA       SYSTA     ;RESET STATUS BYTE
004D D3FD             OUT       0FDH      ;EXECUTE POINTER ZERO
004F 3A8100   STAR0:  LDA       SYSTA     ;GET STATUS BYTE
0052 B7               ORA       A         ;IS IT 0?
0053 CA8D00           JZ        SYST0     ;LOOP AS LONG AS IT IS
0056 FE01             CPI       1         ;STATUS OKAY?
0058 CA0000           JZ        0         ;JUMP INTO BOOTSTRAP IF SO
005B 2F               CMA                 ;PUT ERROR CODE IN LIGHTS...
005C D3FF             OUT       0FFH
005E C38700           JMP       SYSTR     ;TRY TO BOOT AGAIN
              ;
001F =        SYSLN   EQU       $-SYSCM
0081 =        SYSTA   EQU       STATS-SYSCM+80H
0087 =        SYSTR   EQU       START-SYSCM+80H
008D =        SYST0   EQU       STAR0-SYSCM+80H
              ;
              ; DEFAULT POINTERS
              ;
0061 8000     DEFP:   DW        00080H
0063 0010             DW        01000H
0065 0020             DW        02000H
0067 0030             DW        03000H
0069 0040             DW        04000H
006B 0050             DW        05000H
006D 0060             DW        06000H
006F 0070             DW        07000H
0071 0080             DW        08000H
0073 0090             DW        09000H
0075 00A0             DW        0A000H
0077 00B0             DW        0B000H
0079 00C0             DW        0C000H
007B 00D0             DW        0D000H
007D 00E0             DW        0E000H
007F 00F0             DW        0F000H
              ;
              ; SCAN LOOP HEADER
              ;
0081 210008   SCLPH:  LXI       H,PBAS    ;SET DEFAULT POINTERS...
0084 116100           LXI       D,DEFP
0087 0E20             MVI       C,32
0089 CD9103           CALL      DMOV
```

```
008C 97                 SUB     A          ;INITIALIZE VARIABLES...
008D 322B08             STA     HDFL
0090 322608             STA     INTE
0093 323108             STA     SDRV
0096 3E7F               MVI     A,7FH
0098 212C08             LXI     H,DRV0
009B 77                 MOV     M,A
009C 23                 INX     H
009D 77                 MOV     M,A
009E 23                 INX     H
009F 77                 MOV     M,A
00A0 23                 INX     H
00A1 77                 MOV     M,A
                ;
                ; MAIN SCAN LOOP
                ;
00A2 3A2B08     SCLP:   LDA     HDFL       ;GET HEAD FLAG
00A5 B7                 ORA     A
00A6 CAD300             JZ   SCL1          ;JUMP IF HEAD NOT DOWN
00A9 3A0018     SCL2:   LDA  1800H         ;DOWN, CHECK COMMAND
00AC 1F                 RAR
00AD DAE800             JC   ACTN
00B0 3A0440             LDA  4004H         ;NO, CHECK FOR INDEX
00B3 E602               ANI  2
00B5 CAA900             JZ   SCL2
00B8 3A2608             LDA  INTE          ;YES, CLEAR IT AND
00BB F680               ORI  80H           ;MAINTAIN INTERRUPT FLAG
00BD 320640             STA  4006H
00C0 3A2B08             LDA  HDFL          ;SEE IF TIME TO RAISE
00C3 3C                 INR  A
00C4 322B08             STA  HDFL
00C7 FE03               CPI  3
00C9 C2A900             JNZ  SCL2
00CC 97                 SUB  A             ;YES
00CD 320540             STA  4005H
00D0 322B08             STA  HDFL
00D3 060F       SCL1:   MVI  B,0FH         ;SET TO TEST ALL DRIVES
00D5 3A0018             LDA  1800H
00D8 1F                 RAR
00D9 DAE800             JC   ACTN
00DC CDBD02             CALL DRAD          ;GET DRIVE ADDRESS
00DF CAD300             JZ   SCL1          ;JMP IF ALL DONE
00E2 CDE302             CALL DRDY          ;TEST THIS DRIVE
00E5 C3D500             JMP  SCL1+2
                ; ACTION JUMP MODULE - PERFORMS COMMON OPERATIONS
                ; VERIFIES COMMAND STRING, THEN JUMPS TO PROPER ROUTINE
00E8 97         ACTN:   SUB  A             ;CLEAR INTERUPT FLAG
00E9 320640             STA  4006H
00EC 322608             STA  INTE
00EF 3A0014             LDA  1400H         ;GET COMMAND WORD
00F2 47                 MOV  B,A
00F3 E6F0               ANI  0F0H          ;GET TYPE
00F5 CA5401             JZ   ACT1          ;OF COMMAND STRING
00F8 C6F0               ADI  0F0H
00FA CA0F01             JZ   ACT2          ;POINTER SET
00FD C6F0               ADI  0F0H
00FF CA3401             JZ   ACT3          ;RESTORE DRIVE
0102 C6F0               ADI  0F0H
```

```
0104 CA4201              JZ    ACT4        ;SOFTWARE WRITE PROTECT
0107 C6F0                ADI   0F0H
0109 CA4701              JZ    ACT5        ;SOFTWARE WRITE ENABLE
010C C3A200              JMP   SCLP        ;NOP FUNCTION
010F CD2801      ACT2:   CALL  PADD        ;POINTER ADDRESS
0112 CD1C01              CALL  RBYT        ;LSB OF POINTER ADDRESS
0115 23                  INX   H
0116 CD1C01              CALL  RBYT        ;MSB
0119 C3A200              JMP   SCLP
011C 3A0018      RBYT:   LDA   1800H       ;WAIT
011F 1F                  RAR
0120 D21C01              JNC   RBYT
0123 3A0014              LDA   1400H       ;DATA
0126 77                  MOV   M,A         ;SAVE
0127 C9                  RET
0128 210008      PADD:   LXI   H,PBAS      ;BASE ADDRESS
012B 78                  MOV   A,B
012C E60F                ANI   0FH
012E 07                  RLC
012F 4F                  MOV   C,A
0130 0600                MVI   B,0
0132 09                  DAD   B
0133 C9                  RET
0134 0E7F        ACT3:   MVI   C,7FH       ;RESTORE FLAG
0136 CD8D02              CALL  DRAD        ;GET DRIVE ADDRESS
0139 CAA200              JZ    SCLP
013C 7E                  MOV   A,M
013D B1                  ORA   C           ;SET NEW STATUS
013E 77                  MOV   M,A
013F C33601              JMP   ACT3+2
0142 0E80        ACT4:   MVI   C,80H       ;PROTECT INDICATOR
0144 C33601              JMP   ACT3+2
0147 CD8D02      ACT5:   CALL  DRAD
014A CAA200              JZ    SCLP
014D 7E                  MOV   A,M
014E E67F                ANI   7FH         ;CLEAR INDICATOR
0150 77                  MOV   M,A
0151 C34701              JMP   ACT5
0154 CD2801      ACT1:   CALL  PADD
0157 4E                  MOV   C,M         ;CONVERT TO TRUE ADDRESS
0158 23                  INX   H
0159 66                  MOV   H,M
015A 69                  MOV   L,C         ;SECOND BYTE
015B 23                  INX       H       ;SAVE STATUS BYTE ADDRESS...
015C 222908              SHLD      STPT
015F 2B                  DCX       H
0160 CD9A03              CALL  FADD        ;TAKE CARE OF MSB
0163 0EC1                MVI   C,0C1H      ;SET FOR COMMAND STRING ERROR
0165 46                  MOV   B,M         ;COMMAND
0166 23                  INX   H
0167 3E00                MVI   A,0
0169 86                  ADD   M
016A C28302              JNZ   AERX        ;STATUS BYTE NOT ZERO
016D 23                  INX   H           ;TRACK MSB MUST BE ZERO
016E 86                  ADD   M
016F C27F02              JNZ   AERX-4
0172 23                  INX   H
0173 EB                  XCHG
```

```
0174 CDBD02             CALL DRAD
0177 CA8202             JZ   AERX-1        ;NO DRIVE SELECTED
017A CDBD02             CALL DRAD
017D C28102             JNZ  AERX-2        ;MORE THAN ONE
0180 1A                 LDAX D             ;TRACK NUMBER
0181 13                 INX  D
0182 FE4D               CPI  77
0184 D27F02             JNC  AERX-4
0187 322008             STA  PTRK          ;SAVE
018A 322108             STA  LTRK
018D 78                 MOV  A,B           ;COMMAND
018E E6F0               ANI  0F0H
0190 CAA002             JZ   RDAL          ;READ ALL
0193 FEA1               CPI  0A1H
0195 D28002             JNC  AERX-3        ;ILLEGAL TOO LARGE
0198 FE60               CPI  60H
019A DAA201             JC   $+8
019D C6A0               ADI  0A0H
019F CA8002             JZ   AERX-3
01A2 322508             STA  FUNC          ;SAVE COMMAND
01A5 FE30               CPI  30H
01A7 CACA01             JZ   ACT6          ;JMP IF SECTOR NOT REQUIRED
01AA 1A                 LDAX D             ;GET IT
01AB 13                 INX  D
01AC B7                 ORA     A
01AD CA7E02             JZ   AERX-5
01B0 FE1B               CPI  27
01B2 D27E02             JNC  AERX-5
01B5 322208             STA  SECT          ;SAVE GOOD SECTOR VALUE
01B8 3A2508             LDA  FUNC          ;SEE IF NEED BUFFER LOC
01BB FE21               CPI  21H
01BD D2CA01             JNC  ACT6
01C0 1A          ACT7:  LDAX D             ;YES, GET IT
01C1 322308             STA  BUFA
01C4 13                 INX  D
01C5 1A                 LDAX D
01C6 13                 INX  D
01C7 322408             STA  BUFA+1
01CA 78          ACT6:  MOV  A,B           ;ALL PROCESSED, SEE IF IBM SPECIAL
01CB FE6F               CPI  6FH
01CD DADF01             JC   ACTQ
01D0 1A                 LDAX D             ;YES, GET LOGICAL TRACK
01D1 B7                 ORA     A
01D2 C27C02             JNZ     AERX-7
01D5 13                 INX  D
01D6 1A                 LDAX D
01D7 FE4D               CPI  77
01D9 D27C02             JNC  AERX-7
01DC 322108             STA  LTRK          ;SAVE IT
01DF 0EA1        ACTQ:  MVI  C,0A1H        ;SET DRIVE TEST ERROR
01E1 CDE302             CALL DRDY          ;SEE IF DRIVE READY
01E4 C28302             JNZ  AERX
01E7 78                 MOV  A,B           ;OKAY, SEE IF WRITE
01E8 E610               ANI  10H
01EA CAF901             JZ   ACT8
01ED 3A0440             LDA  4004H         ;YES, CHECK HARDWARE PROTECT
01F0 1F                 RAR
01F1 D28202             JNC  AERX-1
```

```
01F4 7E                 MOV   A,M
01F5 17                 RAL                   ;TEST SOFTWARE PROTECT
01F6 DA8102             JC    AERX-2          ;YES
01F9 97         ACT8:   SUB   A               ;SET PROCESSING FLAGS
01FA 322808             STA   RTRC            ;CLEAR RETRY COUNTER
01FD 322708             STA   INTF            ;SET INTERRUPT FLAG
0200 7E         ACTD:   MOV   A,M             ;NOW POSITION DRIVE
0201 E67F               ANI   7FH             ;GET PRESENT TRACK
0203 57                 MOV   D,A             ;SAVE STATUS OF PROTECT
0204 7E                 MOV   A,M
0205 E680               ANI   80H
0207 77                 MOV   M,A             ;SEEK NEW TRACK
0208 3A2008             LDA   PTRK
020B 86                 ADD   M
020C 77                 MOV   M,A
020D 3A2008             LDA   PTRK
0210 92                 SUB   D               ;GET DIFFERENCE
0211 1E00               MVI   E,0             ;SET FOR STEP IN
0213 57                 MOV   D,A
0214 CAAB02             JZ    TSTH            ;ALREADY ON TRACK
0217 F22302             JP    ACT9-4
021A 1E08               MVI   E,8             ;STEP OUT
021C 97                 SUB   A
021D 92                 SUB   D
021E 57                 MOV   D,A
021F 3A3108             LDA       SDRV        ;RAISE HEAD...
0222 320540             STA       4005H
0225 00                 NOP                   ;DELAY...
0226 00                 NOP
0227 15         ACT9:   DCR   D               ;TEST FOR TIME TO LOAD HEAD
0228 CC6A03             CZ    LDHD
022B 7B                 MOV   A,E             ;LOAD STEP REGISTER
022C 320640             STA   4006H
022F C604               ADI   4               ;NOW SET STEP BIT
0231 320640             STA   4006H
0234 E608               ANI   8               ;CLEAR STEP BIT
0236 C600               ADI   0
0238 320640             STA   4006H
023B FB         ACTP:   EI                    ;INTERRUPT IF DRIVE GOES NOT READY
023C CD8303             CALL  TMLD            ;WAIT 6 MILLISECONDS
023F CD8303             CALL  TMLD
0242 CD8303             CALL  TMLD
0245 15                 DCR   D               ;SEE IF MORE REQUIRED
0246 F22802             JP    ACT9+1
0249 1605               MVI   D,5             ;WAIT 10 MILLISECONDS FOR HEAD
024B CD8303             CALL  TMLD
024E 15                 DCR   D
024F C24B02             JNZ   $-4
                ;HEAD IS NOW LOADED, WE'RE ON RIGHT TRACK
                ;AND WE'RE READY TO TAKE ACTION
                ; RECALL THAT POINTER TO STATUS LOCATION IS SAVED
                ; IN STPT, FUNCTION IS IN 4 MSB OF FUNC
0252 F3         ACTB:   DI                    ;DISABLE INTERRUPT UNTIL REQUIRED
0253 3A2508             LDA   FUNC
0256 216A02             LXI   H,JMPT          ;BASE OF TABLE
0259 0F                 RRC
025A 0F                 RRC
025B 0F                 RRC
```

```
025C 0F                 RRC
025D E607               ANI  7
025F C26302    ACTA:    JNZ  $+4
0262 E9                 PCHL            ;JUMP TO ROUTINE
0263 23                 INX  H
0264 23                 INX  H
0265 23                 INX  H
0266 3D                 DCR  A
0267 C35F02             JMP  ACTA
               ;FOLLOWING IS JUMP TABLE FOR ACTIONS 0 TO 5
               ; NOTE THAT 7 TO 11 ARE SAME AS 0 TO 5 WITH DIFF LTRK
026A C3D703    JMPT:    JMP  RALL
026D C3D904             JMP  WRIT
0270 C36B05             JMP  READ
0273 C3EF05             JMP  FRMT
0276 C36B05             JMP  READ       ;JUST IGNORE TRANSFER AT END
0279 C32905             JMP  WDSC
               ;ERROR RETURN ROUTINE BASE ERROR IS IN C
027C 0C                 INR  C
027D 0C                 INR  C
027E 0C                 INR  C
027F 0C                 INR  C
0280 0C                 INR  C
0281 0C                 INR  C
0282 0C                 INR  C
0283 2A2908    AERX:    LHLD STPT       ;GET STATUS BYTE ADDRESS
0286 CD9A03             CALL FADD
0289 71                 MOV  M,C
028A F3                 DI              ;INTERRUPTS OFF
028B 97                 SUB  A          ;CLEAR CONTROL REGISTERS
028C 320540             STA  4005H
028F 322B08             STA  HDFL
0292 3E10               MVI  A,10H
0294 320640             STA  4006H      ;SET INTERRUPT FLAG
0297 322608             STA  INTE
029A 310009             LXI  SP,STAK    ;RESET STACK POINTER
029D C3A200             JMP  SCLP
02A0 322508    RDAL:    STA  FUNC       ;SET FUNCTION
02A3 1A                 LDAX D          ;GET DEAAY
02A4 13                 INX  D
02A5 322208             STA  SECT       ;SAVE IT
02A8 C3C001             JMP  ACT7       ;GO GET BUFFER LOC
02AB 3A2B08    TSTH:    LDA  HDFL       ;SEE IF HEAD LOAD REQUIRED
02AE B7                 ORA     A
02AF 3E01               MVI  A,1        ;ALREADY LOADED
02B1 322B08             STA  HDFL       ;RESET FLAG
02B4 C25202             JNZ  ACTB       ;GO PROCESS FUNCTION
02B7 CD6A03             CALL LDHD
02BA C33B02             JMP  ACTP
               ; FORM DRIVE ADDRESS FOR DRIVE SELECTED IN 4 LSB OF B.
               ;    CLEAR BIT OF DRIVE SELECTED. RETURN NON-ZERO IF
               ;    THERE WAS A DRIVE AND ZERO IF NONE WAS SELECTED.
02BD 3E0F      DRAD:    MVI  A,0FH      ;SEE IF ANY
02BF A0                 ANA  B
02C0 C8                 RZ
02C1 3E01               MVI  A,1        ;0?
02C3 212C08             LXI  H,DRV0
02C6 A0                 ANA  B
```

```
02C7 C2DB02            JNZ   DRA1
02CA 3E02              MVI   A,2         ;1?
02CC 23                INX   H
02CD A0                ANA   B
02CE C2DB02            JNZ   DRA1
02D1 3E04              MVI   A,4         ;2?
02D3 23                INX   H
02D4 A0                ANA   B
02D5 C2DB02            JNZ   DRA1
02D8 3E08              MVI   A,8         ;MUST BE 3
02DA 23                INX   H
02DB 323008    DRA1:   STA   TDRV        ;SAVE SELECT BIT
02DE 2F                CMA               ;CLEAR THIS BIT
02DF A0                ANA   B
02E0 47                MOV   B,A
02E1 3C                INR   A           ;SET FLAG
02E2 C9                RET
               ; TEST DRIVE READY. RETURN NON-ZERO FLAG IF NOT READY.
               ; IF READY, PERFORM RESTORE IF FLAG SET (DRIVE TRACK = 7F).
02E3 3E01      DRDY:   MVI   A,1         ;SEE IF SELECTED
02E5 322708            STA   INTF        ;DRIVE IS READY
02E8 3A3008            LDA   TDRV        ;SELECT DRIVE AND
02EB 57                MOV   D,A         ;SEE IF SAME AS LAST
02EC 3A3108            LDA   SDRV
02EF 92                SUB   D
02F0 C2FA02            JNZ   DRD6
02F3 3A2B08            LDA   HDFL        ;SAME SEE IF HEAD DOWN
02F6 B7                ORA      A
02F7 C20403            JNZ   DRD5        ;JUMP IF DOWN
02FA 97        DRD6:   SUB   A           ;CLEAR HEAD FLAG
02FB 322B08            STA   HDFL
02FE 3A3008            LDA   TDRV
0301 320540            STA   4005H
0304 FB        DRD5:   EI
0305 97                SUB      A        ;IF DRIVE NOT READY, INTERRUPT
0306 F3                DI                ;  OCCURS AND RETURNS A 1 IN A
0307 B7                ORA      A
0308 C24003            JNZ   DRD1
030B 3A3008            LDA   TDRV        ;OKAY, SAVE DRIVE BIT
030E 323108            STA   SDRV
0311 7E                MOV   A,M         ;CHECK FOR RESTORE FLAG
0312 E67F              ANI   7FH
0314 FE7F              CPI   7FH
0316 C23E03            JNZ   DRD2
0319 3A0440            LDA   4004H       ;YES, SEE IF OVER TRK 0
031C E604              ANI   4
031E C44B03            CNZ   STIN        ;IF SO, STEP IN
0321 3A0440            LDA   4004H       ;STILL THERE?
0324 E604              ANI   4
0326 C0                RNZ               ;RETURN DRIVE NO GOOD
0327 164D              MVI   D,77        ;SET COUNT
0329 CD4F03    DRD3:   CALL  STOU        ;STEP OUT
032C 3A0440            LDA   4004H       ;THERE?
032F E604              ANI   4
0331 C23A03            JNZ   DRD4
0334 15                DCR   D           ;NO, MAX TRIES?
0335 C22903            JNZ   DRD3
0338 14                INR   D           ;SET NON-ZERO FLAG
```

```
0339 C9                 RET
033A 3E80       DRD4:   MVI  A,80H        ;THERE, SET TRACK 0
033C A6                 ANA  M
033D 77                 MOV  M,A
033E 97         DRD2:   SUB  A            ;SET ZERO FLAG
033F C9                 RET
0340 97        'DRD1:   SUB  A            ;NOT READY, CLEAR HEAD FLAG
0341 322B08             STA  HDFL
0344 3E80               MVI  A,80H        ;SET RESTORE FLAG
0346 A6                 ANA  M
0347 C67F               ADI  7FH
0349 77                 MOV  M,A
034A C9                 RET
                ;STEP HEAD IN
034B 97         STIN:   SUB  A
034C C35103             JMP  STOU+2
                ;STEP HEAD OUT
034F 3E08       STOU:   MVI  A,8
0351 320640             STA  4006H
0354 C604               ADI  4            ;ADD STEP BIT
0356 320640             STA  4006H
0359 E608               ANI  8
035B C600               ADI  0
035D 320640             STA  4006H
0360 1606               MVI  D,6          ;WAIT 12 MILLISECONDS
0362 CD8303     ST01:   CALL TMLD
0365 15                 DCR  D
0366 C26203             JNZ  ST01
0369 C9                 RET
                ; LOAD HEAD ROUTINE.  SET > 43 BIT IF REQUIRED
036A 7E         LDHD:   MOV  A,M          ;GET TRACK
036B E67F               ANI  7FH
036D FE2C               CPI  44
036F 3F                 CMC
0370 3E80               MVI  A,80H
0372 1F                 RAR
0373 0F                 RRC
0374 0F                 RRC
0375 4F                 MOV  C,A
0376 3A3108             LDA  SDRV         ;GET SELECT BIT
0379 81                 ADD  C
037A 320540             STA  4005H
037D 3E01               MVI  A,1          ;SET HEAD DOWN FLAG
037F 322B08             STA  HDFL
0382 C9                 RET
0383 CD8A03     TMLD:   CALL MLDL         ;TWO MILLISECONDS IS
0386 CD8A03             CALL MLDL         ;ONE PLUS ONE
0389 C9                 RET
038A 3E83       MLDL:   MVI  A,131
038C 3D                 DCR  A
038D C28C03             JNZ  $-1
0390 C9                 RET
                ;MOVE C BYTES FROM D TO H
0391 1A         DMOV:   LDAX D
0392 77                 MOV  M,A
0393 13                 INX  D
0394 23                 INX  H
0395 0D                 DCR  C
```

```
0396 C29103            JNZ   DMOV
0399 C9                RET
039A 7C        FADD:   MOV   A,H          ;SET MEMORY BIT FOR TRANSFER
039B E680              ANI   80H
039D 0F                RRC
039E 0F                RRC
039F 320640            STA   4006H
03A2 7C                MOV   A,H          ;SET BIT 15 FOR INTERNAL
03A3 F680              ORI   80H
03A5 67                MOV   H,A
03A6 C9                RET
               ;
0048 =         CRCU    EQU      48H
0029 =         CRCL    EQU      29H
               ;
               ; COMPUTE CRC FROM BUFFER POINTED TO BY PAIR D
               ; BUFFER LENGTH IS IN C
               ;
03A7 21FFFF    CRCC:   LXI   H,0FFFFH     ;PRESET CRC VALUE
03AA 1A                LDAX D             ;GET A BYTE
03AB CDB403            CALL CRC1          ;DO ONE BYTE
03AE 13                INX   D
03AF 0D                DCR   C
03B0 C2AA03            JNZ   CRCC+3
03B3 C9                RET
03B4 C5        CRC1:   PUSH B             ;COMPUTE FOR ONE BYTE
03B5 D5                PUSH D
03B6 AC                XRA   H
03B7 47                MOV   B,A
03B8 07                RLC
03B9 07                RLC
03BA 07                RLC
03BB 07                RLC
03BC A8                XRA   B
03BD 4F                MOV   C,A
03BE E6F0              ANI   0F0H
03C0 57                MOV   D,A
03C1 81                ADD   C
03C2 5F                MOV   E,A
03C3 7A                MOV   A,D
03C4 CE00              ACI   0
03C6 AD                XRA   L
03C7 67                MOV   H,A
03C8 78                MOV   A,B
03C9 E6F0              ANI   0F0H
03CB 47                MOV   B,A
03CC AB                XRA   E
03CD 6F                MOV   L,A
03CE 78                MOV   A,B
03CF 0F                RRC
03D0 0F                RRC
03D1 0F                RRC
03D2 AC                XRA   H
03D3 67                MOV   H,A
03D4 D1                POP   D
03D5 C1                POP   B
03D6 C9                RET
               ;READ ALL LOOP...INTERLEAVE CLOCK AND DATA (64 BYTES WORTH)
```

```
03D7 0E40      RALL:    MVI  C,64          ;BYTE COUNTER
03D9 3E80               MVI  A,80H         ;INHIBIT INDEX INTERRUPT
03DB 320640             STA  4006H
03DE 31B708             LXI  SP,DATB+1     ;SET DATA AREA
03E1 FB                 EI
03E2 3A0440    RAL1:    LDA  4004H         ;WAIT FOR INDEX
03E5 E602               ANI  2
03E7 CAE203             JZ   RAL1
03EA 3A2208             LDA  SECT
03ED FE00               CPI  0
03EF CAFA03             JZ   RAL2
03F2 57                 MOV  D,A
03F3 CD8A03             CALL MLDL
03F6 15                 DCR  D
03F7 C2F303             JNZ  $-4
03FA 3E01      RAL2:    MVI  A,1           ;SYNC PLO
03FC 320640             STA  4006H
03FF 97                 SUB  A
0400 C600               ADI  0             ;DELAY 12 MICROSECONDS
0402 C600               ADI  0
0404 320640             STA  4006H
0407 2A0240    RAL4:    LHLD 4002H         ;GET 64 BYTES WORTH
040A E5                 PUSH H
040B 0D                 DCR  C
040C C20704             JNZ  RAL4
040F 2A2308    RAL7:    LHLD BUFA          ;MOVE TO MAIN MEMORY
0412 CD9A03             CALL FADD
0415 11B608             LXI  D,DATB
0418 0E80               MVI  C,128
041A 1A        RAL5:    LDAX D
041B 1B                 DCX  D
041C 77                 MOV  M,A
041D 23                 INX  H
041E 0D                 DCR  C
041F C21A04             JNZ  RAL5
0422 2A2908    RAL6:    LHLD STPT          ;SUCCESSFUL RETURN
0425 CD9A03             CALL FADD
0428 F3                 DI
0429 34                 INR  M
042A 3E10               MVI  A,10H         ;SET INTERRUPT FLAG
042C 320640             STA  4006H
042F 322608             STA  INTE          ;AND SAVE THE SET VALUE
0432 310009             LXI  SP,STAK       ;RESET STACK POINTER
0435 C3A200             JMP  SCLP          ;WAIT FOR MORE
               ;ROUTINE TO SYNC ON START OF SECTOR
               ;EXITS AFTER TEN BYTES OF ZEROES FOLLOWING HEADER
0438 CD4E05    SYNC:    CALL HDRC          ;COMPUTE CRC FOR HEADER
043B E5                 PUSH H             ;SAVE FOR LATER
043C 97        SYN2:    SUB  A             ;CLEAR INDEX COUNTER
043D 322608             STA  INTE
0440 110240             LXI  D,4002H       ;REGISTER ADDRESS
0443 3A2108             LDA  LTRK          ;SET FOR FAST CHECKING
0446 4F                 MOV  C,A
0447 3A2208             LDA  SECT
044A 47                 MOV  B,A
044B 31FC08             LXI  SP,STAK-4     ;WHERE CRC IS STORED
044E 3A0440             LDA  4004H         ;TEST FOR INDEX
0451 E602               ANI  2
```

```
0453 CA6F04              JZ    SYN1
0456 3E80                MVI   A,80H         ;CLEAR IT
0458 320640              STA   4006H
045B 3A2608              LDA   INTE          ;YES, SEE IF SECOND
045E 3C                  INR   A
045F 322608              STA   INTE
0462 FE03                CPI   3
0464 C26F04              JNZ   SYN1
0467 0E93                MVI   C,93H         ;SECOND SET ERROR
0469 CDE205     SYN4:    CALL  RTRY          ;SEE IF RETRY COUNT EXPIRED
046C C33C04              JMP   SYN2          ;RETURN SAYS TRY AGAIN
046F 3E01       SYN1:    MVI   A,1           ;SYNC PLO
0471 320640              STA   4006H
0474 97                  SUB   A
0475 E3                  XTHL
0476 E3                  XTHL
0477 C600                ADI   0
0479 320640              STA   4006H
047C FB                  EI
047D 3EC7                MVI   A,0C7H        ;WAIT FOR CHARACTER
047F 2A0040              LHLD  4000H
0482 BC                  CMP   H
0483 C24004              JNZ   SYN2+4        ;CLOCK PATTERN CHECK
0486 2C                  INR   L             ;DATA SHOULD BE FC
0487 1A                  LDAX  D             ;GET TRACK
0488 91                  SUB   C
0489 C2B404              JNZ   SYN3          ;NOT SAME - WRONG TRACK?
048C 2C                  INR   L             ;COMPLETE DATA BYTE CHECK
048D EB                  XCHG
048E 4E                  MOV   C,M           ;GET ZERO BYTE
048F C24004              JNZ   SYN2+4        ;JMP IF DATA BYTE NOT CORRECT
0492 D1                  POP   D             ;GET CRC BYTES
0493 7E                  MOV   A,M           ;GET SECTOR
0494 90                  SUB   B             ;CORRECT?
0495 C24004              JNZ   SYN2+4
0498 79                  MOV   A,C           ;YES
0499 B6                  ORA   M             ;LAST ZERO BYTE
049A 0E95                MVI   C,95H
049C C26904              JNZ   SYN4          ;JUMP ON FORMAT ERROR
049F 7E                  MOV   A,M           ;CRC BYTE 1
04A0 92                  SUB   D             ;CHECK IT
04A1 C2A604              JNZ   $+5
04A4 7B                  MOV   A,E
04A5 96                  SUB   M             ;CHECK CRC BYTE 2
04A6 0E94                MVI   C,94H
04A8 C26904              JNZ   SYN4          ;JUMP ON ERROR
04AB 7E                  MOV   A,M           ;OKAY, WAIT TEN BYTES
04AC 0E09                MVI   C,9
04AE 7E                  MOV   A,M
04AF 0D                  DCR   C
04B0 C2AE04              JNZ   $-2
04B3 C9                  RET                 ;RETURN OKAY
04B4 2C         SYN3:    INR   L             ;SEE IF DATA BYTE CORRECT
04B5 C24004              JNZ   SYN2+4
04B8 F3                  DI                  ;YES, THERE IS TRACK ADDRESS ERROR
04B9 3A3108              LDA   SDRV          ;SET DRIVE FOR RESTORE
04BC 47                  MOV   B,A
04BD 310009              LXI   SP,STAK       ;CORRECT SP
```

```
04C0 CDBD02            CALL DRAD
04C3 3E80              MVI  A,80H
04C5 A6                ANA  M
04C6 C67F              ADI  7FH
04C8 77                MOV  M,A
04C9 0E92              MVI  C,92H       ;SEE IF RETRY DONE
04CB CDE205            CALL RTRY
04CE CDE302            CALL DRDY        ;OKAY, POSITION DRIVE AGAIN
04D1 CA0002            JZ   ACTD        ;OVER 0 THEN GO AGAIN
04D4 0E91              MVI  C,91H       ;DRIVE INOPERABLE
04D6 C38302            JMP  AERX
                ;ROUTINE TO WRITE A DATA BLOCK
04D9 2A2308     WRIT:  LHLD BUFA        ;FIRST GET DATA
04DC CD9A03            CALL FADD
04DF 113208            LXI  D,DATB-132
04E2 D5                PUSH D
04E3 EB                XCHG
04E4 36FB              MVI  M,0FBH      ;DATA HEADER FOR CRC
04E6 23                INX  H
04E7 0E80              MVI  C,128
04E9 CD9103            CALL DMOV
04EC D1                POP  D           ;GET DATA ADDRESS AGAIN
04ED 0E81              MVI  C,129
04EF CDA703            CALL CRCC
04F2 7C                MOV  A,H         ;SAVE IT
04F3 12                STAX D
04F4 13                INX  D
04F5 7D                MOV  A,L
04F6 12                STAX D
04F7 13                INX  D
04F8 97                SUB  A           ;TRAILING ZERO
04F9 12                STAX D
04FA CD3804            CALL SYNC        ;FIND PROPER SECTOR
04FD 210740            LXI  H,4007H     ;SET WRITE ADDRESS
0500 014205            LXI  B,542H      ;SET COUNTERS
0503 313308            LXI  SP,DATB-131
0506 D1                POP  D           ;GET FIRST DATA BYTES
0507 3E02              MVI  A,2         ;SET WRITE ENABLE
0509 320640            STA  4006H
050C 97                SUB  A           ;WRITE ZERO BYTES
050D 05                DCR  B
050E 77                MOV  M,A
050F C20D05            JNZ  $-2
0512 3EFB              MVI  A,0FBH      ;WRITE DATA INDEX MARK
0514 321F40            STA  401FH
0517 C31C05            JMP  $+5         ;START DATA LOOP
051A 72         WRT2:  MOV  M,D
051B D1                POP  D           ;GET NEXT TWO BYTES OF DATA
051C 0D                DCR  C           ;SEE IF DONE
051D 73                MOV  M,E         ;WRITE DATA BYTE
051E C21A05            JNZ  WRT2
0521 73                MOV  M,E         ;WAIT FOR ZERO BYTE
0522 97                SUB  A           ;CLEAR WRITE ENABLE
0523 320640            STA  4006H
0526 C32204            JMP  RAL6        ;SUCCESSFUL RETURN
                ;ROUTINE TO WRITE DELETED DATA ADDRESS MARK
0529 CD3804     WDSC:  CALL SYNC        ;FIND SECTOR
052C 210740            LXI  H,4007H     ;DATA WRITE REGISTER
```

```
052F 010105              LXI  B,501H      ;SET COUNTERS
0532 3C                  INR  A           ;EQUALIZE TIME WITH WRITE
0533 3C                  INR  A
0534 3C                  INR  A
0535 3C                  INR  A
0536 3E02                MVI  A,2         ;SET WRITE ENABLE
0538 320640              STA  4006H
053B 97                  SUB  A
053C 05                  DCR  B           ;WRITE ZERO BYTES
053D 77                  MOV  M,A
053E C23C05-             JNZ  $-2
0541 3EF8                MVI  A,0F8H      ;WRITE DELETED DATA HEADER
0543 321F40              STA  401FH
0546 97                  SUB  A           ;TRAILING HEADER
0547 77                  MOV  M,A
0548 320640              STA  4006H       ;CLEAR WRITE ENABLE
054B C32204              JMP  RAL6        ;SUCCESSFUL RETURN
054E 21FFFF     HDRC:    LXI  H,0FFFFH    ;COMPUTE HEADER CRC
0551 3EFE                MVI  A,0FEH      ;ADDRESS INDEX MARK
0553 CDB403              CALL CRC1
0556 3A2108              LDA  LTRK        ;TRACK ADDRESS
0559 CDB403              CALL CRC1
055C 97                  SUB  A           ;ZERO BYTE
055D CDB403              CALL CRC1
0560 3A2208              LDA  SECT        ;SECTOR NUMBER
0563 CDB403              CALL CRC1
0566 97                  SUB  A           ;ZERO BYTE
0567 CDB403              CALL CRC1        ;IT IS ALL COMPUTED IN H,L NOW
056A C9                  RET
                ;ROUTINE TO READ A SECTOR OF DATA
056B CD3804     READ:    CALL SYNC        ;FIND SECTOR
056E 110240              LXI  D,4002H     ;DATA READ ADDRESS
0571 014106              LXI  B,641H      ;SET COUNTERS
0574 31B708              LXI  SP,DATB+1   ;SET DATA BUFFER READ POINTER
0577 05                  DCR  B           ;DELAY PAST HEAD TURN ON AREA
0578 C27705              JNZ  $-1
057B 3E01                MVI  A,1         ;SYNC PLO
057D 320640              STA  4006H
0580 97                  SUB  A
0581 E3                  XTHL
0582 E3                  XTHL
0583 E3                  XTHL
0584 E3                  XTHL
0585 320640              STA  4006H
0588 3EC7                MVI  A,0C7H      ;SET UP FOR TESTING DATA
058A 06FB                MVI  B,0FBH      ;ADDRESS MARK
058C 2A0040              LHLD 4000H       ;READ IT FROM DISK
058F 94                  SUB  H           ;CHECK CLOCK PATTERN
0590 C2D905              JNZ  RED1
0593 EB                  XCHG
0594 56                  MOV  D,M         ;GET FIRST DATA BYTE
0595 7B                  MOV  A,E         ;TEST DATA FROM INDEX MARK
0596 90                  SUB  B
0597 C2CF05              JNZ  RED4        ;JMP TO SEE IF DELETED DATA
059A 5E         RED2:    MOV  E,M         ;ALL OKAY, GET ANOTHER DATA BYTE
059B D5                  PUSH D           ;STORE TWO OF THEM
059C 0D                  DCR  C
059D 56                  MOV  D,M         ;GET NEXT BYTE
```

```
059E C29A05            JNZ   RED2         ;LOOP UNTIL DONE
05A1 5E                MOV   E,M          ;THIS IS SECOND OF CRC BYTES
05A2 D5                PUSH D             ;SAVE THEM
05A3 F3                DI                 ;ALL DONE
05A4 310009            LXI   SP,STAK
05A7 21FFFF            LXI   H,0FFFFH     ;NOW CHECK CRC
05AA 11B608            LXI   D,DATB
05AD 0E82              MVI   C,130
05AF 3EFB              MVI   A,0FBH       ;FIRST DO HEADER DATA
05B1 CDB403            CALL CRC1
05B4 1A        RED3:   LDAX D             ;GET DATA BYTE
05B5 CDB403            CALL CRC1
05B8 1B                DCX   D
05B9 0D                DCR   C            ;SEE IF DONE
05BA C2B405            JNZ   RED3
05BD 7C                MOV   A,H          ;SEE IF CRC CORRECT
05BE B5                ORA   L
05BF C2D905            JNZ   RED1
05C2 3A2508            LDA   FUNC         ;OKAY, SEE IF CHECK READ
05C5 E6F0              ANI   0F0H
05C7 FE20              CPI   20H
05C9 CA0F04            JZ    RAL7         ;GO MORE DATA IFF READ
05CC C32204            JMP   RAL6         ;SUCCESSFUL RETURN IF CHECK READ
05CF C603      RED4:   ADI   3            ;TEST FOR DELETED DATA
05D1 C2D905            JNZ   RED1
05D4 0E97              MVI   C,97H        ;YES, SET ERROR
05D6 C38302            JMP   AERX         ;  AND RETURN
05D9 0E96      RED1:   MVI   C,96H        ;CRC ERROR
05DB F3                DI
05DC CDE205            CALL RTRY          ;SEE IF TRY AGAIN
05DF C36B05            JMP   READ         ;YES
               ;TEST RETRY COUNTER. RETURN IF TIME TO TRY AGAIN
05E2 3A2808    RTRY:   LDA   RTRC         ;INCREMENT RETRY COUNTER
05E5 3C                INR   A
05E6 322808            STA   RTRC
05E9 FE0B              CPI   11           ;SEE IF ALL DONE
05EB C0                RNZ
05EC C38302            JMP   AERX         ;TRIED MAX TIMES, SO QUIT
               ;ROUTINE TO FORMAT A TRACK OF A DISKETTE
05EF 3E1A      FRMT:   MVI   A,26         ;COMPUTE AND SAVE HEADER
05F1 322208            STA   SECT         ;CRC VALUES FOR ALL 26 SECTORS
05F4 11B608            LXI   D,DATB
05F7 CD4E05    FRM1:   CALL HDRC
05FA 7D                MOV   A,L
05FB 12                STAX D
05FC 1B                DCX   D
05FD 7C                MOV   A,H
05FE 12                STAX D
05FF 1B                DCX   D            ;CRC SAVED
0600 3A2208            LDA   SECT         ;SET NEXT SECTOR
0603 3D                DCR   A
0604 322208            STA   SECT
0607 C2F705            JNZ   FRM1         ;LOOP UNTIL 26 DONE
060A 13                INX   D            ;SET D TO CRC OF FIRST SECTOR
060B 210740            LXI   H,4007H      ;WRITE REGISTER
060E 012C01            LXI   B,12CH       ;SET COUNTERS
0611 3E80              MVI   A,80H
0613 320640            STA   4006H        ;WAIT FOR INDEX
```

```
0616 FB                 EI
0617 3A0440             LDA   4004H
061A E602               ANI   2
061C CA1706             JZ    $-5
061F 3E82               MVI   A,82H         ;SET WRITE ENABLE
0621 320640             STA   4006H
0624 97                 SUB   A             ;SET ZERO BYTES
0625 77                 MOV   M,A
0626 0D                 DCR   C             ;WRITE GAP 4
0627 C22506             JNZ   $-2
062A 77                 MOV   M,A
062B 3EFC               MVI   A,0FCH        ;WRITE INDEX ADDRESS MARK
062D 321740             STA   4017H
0630 97         FRM2:   SUB   A             ;DO A SECTOR'S WORTH
0631 0E1E               MVI   C,30          ;FOR GAP 1
0633 77                 MOV   M,A
0634 0D                 DCR   C
0635 C23306             JNZ   $-2
0638 77                 MOV   M,A           ;LAST BYTE
0639 3EFE               MVI   A,0FEH        ;ID ADDRESS MARK
063B 321F40             STA   401FH
063E 3A2108             LDA   LTRK          ;TRACK ADDRESS
0641 77                 MOV   M,A
0642 97                 SUB   A             ;ZERO BYTE
0643 77                 MOV   M,A
0644 0E10               MVI   C,16          ;SET COUNT FOR GAP 2
0646 70                 MOV   M,B           ;SECTOR NUMBER
0647 77                 MOV   M,A           ;ZERO BYTE
0648 1A                 LDAX  D             ;CRC BYTE 1
0649 13                 INX   D
064A 77                 MOV   M,A
064B 1A                 LDAX  D             ;CRC BYTE 2
064C 13                 INX   D
064D 77                 MOV   M,A
064E 97                 SUB   A             ;ZERO BYTES FOR GAP 2
064F 0D                 DCR   C
0650 C24D06             JNZ   $-3
0653 77                 MOV   M,A           ;LAST BYTE
0654 3EFB               MVI   A,0FBH        ;DATA ADDRESS MARK
0656 321F40             STA   401FH
0659 97                 SUB   A             ;DATA BYTES FOR SECTOR ARE ALL ZERO
065A 0E7F               MVI   C,127         ;NUMBER BYTES LESS ONE
065C 77                 MOV   M,A           ;WRITE ONE
065D 0D                 DCR   C
065E C25C06             JNZ   $-2
0661 77                 MOV   M,A           ;LAST BYTE
0662 3E48               MVI   A,CRCU
0664 77                 MOV   M,A
0665 3E29               MVI   A,CRCL
0667 77                 MOV   M,A
0668 04                 INR   B             ;ADVANCE SECTOR NUMBER
0669 3E1B               MVI   A,27
066B 90                 SUB   B
066C 71                 MOV   M,C           ;WRITE ZERO BYTE
066D C23006             JNZ   FRM2          ;LOOP TILL DONE
0670 3A0440     FRM4:   LDA   4004H         ;WRITE ZEROES TO INDEX
0673 71                 MOV   M,C
0674 E602               ANI   2
```

```
0676 C27D06               JNZ  FRM3
0679 71                   MOV  M,C
067A C37006               JMP  FRM4
067D 97          FRM3:    SUB  A          ;CLEAR WRITE ENABLE
067E 320640               STA  4006H
0681 C32204               JMP  RAL6
0684                      END
```