Persephone Mini/MicroFloppy Disk Controller

The document itself begins on the next page.

Document source:

Original backup tapes owned by Dutchman2000, obtained by Atarimania.
Documentary research and PDF layout by Laurent Delsarte.
Note that these backup tapes contain A LOT of information spread out in many folders, meaning it will take time to process the important bits.

Document identification:

Original file name:	PERSEPH.PRN
Title of document:	Persephone Mini/MicroFloppy Disk Controller
Author(s):	Mike Barall (Michael Barall) & Lane Winner
Original file date:	1984-11-15
Type of document:	Software, commented source code
Target audience:	Internal
Status:	Ready
Reference (Atari):	(unknown)
Reference (Laurent Delsarte):	For any discussion, this PDF has been given the reference **BKUP-1984-11-15-SOFT-0007A-A** which should be quoted in any communication.
Tags:	#Atari #8bit #6502 #400 #800 #1200XL #600XL #800XL #1400XL #1450XLD #810 #815 #1050 #1050CR #1055 #SIO #Intel8050 #MCS-48 #WD1770

Comments:

This is the printout of the original source code, fully commented.

What makes this document so interesting are all these valuable comments, which outline all the additional supported features compared to the existing Atari 810, 815 and 1050 5¼-inch floppy disk drives. It concerns a universal driver that can operate with the unreleased Atari 1050CR ("Cost Reduced") 5¼-inch floppy disk drive, as well as with the also unreleased Atari 1055 3½-inch floppy disk drive.

Support is provided for single- or double-sided drives; for SD, 2D (1050 dual density), and DD density formats; and sector sizes of either 128 or 256 bytes. There's also an option to double the communication speed of the SIO port to 38,400 baud. Not all combinations are supported, but everything is clearly explained.

Additionally, I'm offering a companion .ZIP file to download, containing all the contents of the directory – around thirty files in total.

Produced with ATARI CAMAC Assembler.

This page intentionally left blank

Persephone Mini/MicroFloppy Disk Controller

```
               ATARI CAMAC Assembler Ver  1.0A  Page   1
PERSEPHONE MINI/MICROFLOPPY DISK CONTROLLER          D1:PERSEPH.ASM
DOCUMENTATION

0000
                        LIST    -I,-M,-G,R

                ;**      FUNCTION
                ;*
                ;*      This program is the controller for the Atari 1050 CR (cost
                ;*      reduced) mini floppy disk drive and for a possible future
                ;*      micro floppy disk drive.  It resides in the ROM of an
                ;*      Intel 8050 microcontroller.  Its function is to accept
                ;*      commands from the Atari serial bus and to execute them by
                ;*      driving a Western Digital 1770 FDC chip.


                ;**      MODS
                ;*
                ;*      Original Author Mike Barall 1984-09-11
                ;*      Winner                      1984-11-15


                ;**      HARDWARE CONFIGURATION

                ;*      The clock rate is 8 MHz, which gives a time of 1.875 us per
                ;*      machine cycle.

                ;*      P10 and P11 select a register in the 1770 chip:
                ;*              P11 P10         Register
                ;*               0   0          Status or Command
                ;*               0   1          Track
                ;*               1   0          Sector
                ;*               1   1          Data

                ;*      P12 is connected to the INTRQ line of the 1770

                ;*      P13 selects FM (1) or MFM (0)

                ;*      P14 is connected to the 1770 Master Reset pin (0 = reset).
                ;*      The controller holds P14 low for 500 us.

                ;*      P15 is connected to the 1770 R/W pin (1 = read, 0 = write).

                ;*      P16 is the side select output (0 = side 0, 1 = side 1).
                ;*      The controller delays 500 us after changing sides.  P16 is
                ;*      externally tied to ground if the drive has only one head.

                ;*      P17 is connected to the serial bus data in line (drive to
                ;*      computer).  This line is normally high.  Note: P17 is the
                ;*      inverse of the SID line.

                ;*      P20 through P24 have no external connections.  They are used by
                ;*      the controller as 5 bits of RAM.

                ;*      P25 has no external connection on a mini floppy and is externally
                ;*      grounded on a microfloppy.  In the former case, P25 is used as
                ;*      one bit of RAM.

                ;*      P26 and P27 are connected to the device select switches.  They
                ;*      select this drive's unit number, as follows:
```

```
*               P27 P26         Unit Number
*                0   0          1
*                0   1          2
*                1   1          3
*                1   0          4

*       INT is connected to the serial bus command line.  This line is
*       normally high, and is low during the transmission of a
*       command frame.

*       T0 is connected to the serial bus data out line (computer to
*       drive).  Note: T0 is the inverse of the SOD line.

*       T1 is connected to the DRQ line of the 1770.  A high on this
*       line indicates that the 1770 data register is full (on a
*       read) or empty (on a write).

*       DB0-DB7 (the external data bus) are connected to the 1770
*       data bus.


**      MEMORY ALLOCATION

*       F0 = SIO fast mode enable/disable (1 = enabled).

*       F1 = Retry flag (1 = retry in progress).

*       P20, P21, P22 = Current activity:
*
*       P22 P21 P20
*        0   0   0      Idle
*        0   0   1      Get sector
*        0   1   0      Put sector without verify
*        0   1   1      Put sector with verify
*        1   0   0      Write option table
*        1   0   1      Miscellaneous disk operation
*        1   1   0      Not used
*        1   1   1      Other

*       P23 = Head position known flag (1 = position known).

*       P24, P25 = Current configuration:
*
*       P25 P24
*        1   0          SS/SD (FM, 128 bytes per sector)
*        1   1          SS/2D (MFM, 128 bytes per sector)
*        0   0          SS/DD (MFM, 256 bytes per sector)
*        0   1          DS/DD (MFM, 256 bytes per sector)
*
*       For micro floppies, only the SS/DD and DS/DD modes are possible,
*       and there are sixteen sectors per track.


**      COMMANDS

* 52    GET SECTOR
*
```

```
                *       Read the specified logical sector.

                * 50    PUT SECTOR WITHOUT VERIFY
                *
                *       Write the specified logical sector.

                * 57    PUT SECTOR WITH VERIFY
                *
                *       Write the specified logical sector and then re-read it to
                *       verify that it was written correctly.

                * 53    STATUS
                *
                *       Return a 4-byte status block.

                * 21    FORMAT
                *
                *       Format the diskette in the current configuration.

                * 22    DUAL DENSITY FORMAT
                *
                *       Format the diskette in SS/2D mode.  Valid only when the drive
                *       is configured to SS/SD or SS/2D.

                * 4E    GET OPTION TABLE
                *
                *       Return a 12-byte option block.

                * 4F    PUT OPTION TABLE
                *
                *       Accept a 12-byte option block and change to the indicated
                *       configuration.

                * 48    SET SIO SPEED
                *
                *       Change SIO baud rate to the rate specified by auxiliary byte
                *       #1 (00 = 19200 baud, 01 = 38400 baud).

                * 72    GET COPY PROTECTION SECTOR
                *
                *       Step to the track number (0-39) specified by auxiliary byte #1
                *       on the first side of the disk and read sector F7.


                **      STATUS BLOCK FORMAT
                *
                *       Byte 1: Controller status
                *               Bit 0 = 1 Invalid command frame
                *               Bit 1 = 1 Invalid data frame
                *               Bit 2 = 1 Operation Unsuccessful
                *               Bit 3 = 1 Write protected (valid only after a write)
                *               Bit 4 = 1 Motor on
                *               Bits 5-7 = Device configuration
                *                       7 6 5
                *                       0 0 0  SS/SD
                *                       0 0 1  SS/DD
                *                       0 1 0  Reserved
```

```
*                       0 1 1  Reserved
*                       1 0 0  SS/2D
*                       1 0 1  DS/DD
*                       1 1 0  Reserved
*                       1 1 1  Reserved
*
*       Byte 2 = 1770 status register (inverted).
*
*       Byte 3 = FORMAT timeout (= $78).
*
*       Byte 4 = Zero.


**      OPTION TABLE FORMAT
*
*       Byte 1 = Number of tracks (= 28).
*
*       Byte 2 = Step rate (= 00).
*                00 = 6 ms
*                01 = 12 ms
*                02 = 20 ms
*                03 = 30 ms
*
*       Byte 3 = Number of sectors per track (high).
*       Byte 4 = Number of sectors per track (low).
*
*       Byte 5 = Number of sides:
*                00 = single sided
*                01 = double sided
*
*       Byte 6 = Recording mode:
*                00 = FM
*                04 = MFM
*
*       Byte 7 = Number of bytes per sector (high).
*       Byte 8 = Number of bytes per sector (low).
*
*       Byte 9 = Drive present:
*                00 = not on line
*                01 = on line
*
*       Byte 10 = SIO bus speed:
*                41 = 19200 baud
*                48 = 38400 baud
*
*       Byte 11 = Reserved.
*       Byte 12 = Reserved.
*
*
*       NOTE: Bytes 1, 2, 3, 9, 10, 11, and 12 are ignored by the PUT
*       OPTION TABLE command.  The remaining bytes are used to set
*       the drive configuration, as follows:
*
*       IF bytes 5-6 = 00 and 00 AND the drive is a mini floppy,
*       THEN set SS/SD configuration;
*       ELSE IF bytes 4-8 = 1A, 00, 04, 00, and 80 respectively,
*            AND the drive is a mini floppy,
```

```
                *       THEN set SS/2D configuration;
                *       ELSE IF bytes 5-6 = 00 and 04 respectively,
                *       THEN set SS/DD configuration;
                *       ELSE IF bytes 5-6 = 01 and 04 AND drive has two heads,
                *       THEN set DS/DD configuration;
                *       ELSE return error and leave configuration unchanged.


                **      SIO TIMING

                *       At 19200 baud, one bit time is 27.78 CPU cycles (which we
                *       take as 28 CPU cycles).  At 38400 baud, one bit time is
                *       13.89 CPU cycles (which we take as 14 CPU cycles).

                *       The controller is responsible for generating the following
                *       delays:
                *
                *       t2 = delay between raising of COMMAND and transmission of ACK
                *               t2 min = 0 us
                *               t2 max = 16 ms
                *
                *       t4 = delay between transmission of the last bit of the data
                *            frame by the computer and transmission of first bit of
                *            ACK by the drive
                *               t4 min = 850 us
                *               t4 max = 16 ms
                *
                *       t5 = delay between transmission of last bit of ACK and
                *            transmission of first bit of CMPLT
                *               t5 min = 250 us (19200 baud), 750 us (38400 baud)
                *               t5 max = 255 sec (handler dependent)

                *       The computer generates the following delays:
                *
                *       t0 = delay between lowering of COMMAND and transmission of
                *            first bit of command frame
                *               t0 min = 750 us
                *               t0 max = 1600 us
                *
                *       t1 = delay between transmission of last bit of command frame
                *            and raising of COMMAND
                *               t1 min = 650 us
                *               t1 max = 950 us
                *
                *       t3 = delay between receipt of last bit of ACK and transmission
                *            of first bit of data frame by the computer
                *               t3 min = 1000 us
                *               t4 max = 1800 us
```

EQUATES

```
0000

                **      TIMING EQUATES

      = 0177    CLOCK   EQU     375             ;CPU CYCLE TIME IN UNITS OF 5 NS

      = 0108    US500   EQU     [[50000/CLOCK]-1]*2     ;LOOP COUNT FOR 500 US DELAY
      = 0068    US200   EQU     [[20000/CLOCK]-1]*2     ;LOOP COUNT FOR 200 US DELAY
      = 013E    US600   EQU     [[60000/CLOCK]-1]*2     ;LOOP COUNT FOR 600 US DELAY

      = 009A    TRIP    EQU     154             ;TIMER OVERFLOW INTERVAL IN UNITS OF
                                                ;100 US (= CLOCK*32*256*5/100000)

      = 0040    SEC1    EQU     10000/TRIP      ;TRIP COUNT FOR 1 SECOND DELAY
      = 00C2    SEC3    EQU     30000/TRIP      ;TRIP COUNT FOR 3 SECOND DELAY
      = 0010    MS240   EQU     [2400/TRIP]+1   ;TRIP COUNT FOR 240 MS DELAY
      = 0095    MS2300  EQU     23000/TRIP      ;TRIP COUNT FOR 2.3 SECOND DELAY


                **      WD1770 REGISTER SELECT EQUATES

      = 00DC    WDCLR   EQU     $DC             ;MASK TO CLEAR REGISTER SELECTION

      = 0020    WDRSTA  EQU     $20             ;READ STATUS REGISTER
      = 0000    WDWCMD  EQU     $00             ;WRITE COMMAND REGISTER
      = 0021    WDRTRK  EQU     $21             ;READ TRACK REGISTER
      = 0001    WDWTRK  EQU     $01             ;WRITE TRACK REGISTER
      = 0022    WDRSEC  EQU     $22             ;READ SECTOR REGISTER
      = 0002    WDWSEC  EQU     $02             ;WRITE SECTOR REGISTER
      = 0023    WDRDAT  EQU     $23             ;READ DATA REGISTER
      = 0003    WDWDAT  EQU     $03             ;WRITE DATA REGISTER


                **      WD1770 COMMANDS

      = 0002    SRATE   EQU     $02             ;STEP RATE CODE

      = 0080    RDSEC   EQU     $80             ;READ SECTOR
      = 00A2    WRSEC   EQU     $A2             ;WRITE SECTOR WITHOUT PRECOMP
      = 00A0    WRSECP  EQU     $A0             ;WRITE SECTOR WITH PRECOMP
      = 00C4    RDADR   EQU     $C4             ;READ ADDRESS WITH 30 MS DELAY
      = 00F6    WRTRK   EQU     $F6             ;WRITE TRACK WITHOUT PRECOMP
      = 00F4    WRTRKP  EQU     $F4             ;WRITE TRACK WITH PRECOMP
      = 00D0    FRCINT  EQU     $D0             ;FORCE INTERRUPT
      = 0002    REST    EQU     $00 OR SRATE    ;RESTORE
      = 000A    RESTF   EQU     $08 OR SRATE    ;RESTORE W/O MOTOR SPIN-UP
      = 0012    SEEK    EQU     $10 OR SRATE    ;SEEK

      = 00F7    IMPSEC  EQU     $F7             ;'IMPOSSIBLE' SECTOR NUMBER


                **      CONTROL LINES FOR PORT 1

      = 0004    INTRQ   EQU     $04             ;1770 INTERRUPT REQUEST
      = 0008    DDEN    EQU     $08             ;1770 DOUBLE DENSITY ENABLE
      = 0010    MR      EQU     $10             ;1770 MASTER RESET
      = 0040    SSO     EQU     $40             ;SIDE SELECT OUTPUT
```

EQUATES

```
= 0080      SID     EQU     $80             ;SERIAL INPUT DATA LINE
= 0004      INP1    EQU     $04             ;INPUT LINES ON PORT 1

                    **      CONTROL LINES FOR PORT 2

= 00C0      UNITMK  EQU     $C0             ;MASK FOR UNIT NUMBER SWITCHES
= 0000      UNIT1   EQU     $00             ;UNIT 1 SELECT
= 0040      UNIT2   EQU     $40             ;UNIT 2 SELECT
= 00C0      UNIT3   EQU     $C0             ;UNIT 3 SELECT
= 0080      UNIT4   EQU     $80             ;UNIT 4 SELECT
= 00C0      INP2    EQU     $C0             ;INPUT LINES FOR PORT 2

                    **      COMMUNICATION VALUES

= 0041      ACK     EQU     $41             ;ACKNOWLEDGE
= 0043      COMPLT  EQU     $43             ;SUCCESSFUL COMPLETION
= 0045      ERROR   EQU     $45             ;COMPLETION WITH ERROR
= 004E      NAK     EQU     $4E             ;NOT ACKNOWLEDGE
= 0048      HSACK   EQU     $48             ;FAST MODE ACKNOWLEDGE

                    **      ACTIVITY CODES

= 00F8      ACLR    EQU     $F8             ;MASK TO CLEAR ACTIVITY CODE

= 0000      AIDLE   EQU     $00             ;IDLE
= 0001      AGET    EQU     $01             ;GET SECTOR
= 0002      APUT    EQU     $02             ;PUT SECTOR
= 0003      APUTV   EQU     $03             ;PUT SECTOR WITH VERIFY
= 0004      APOPT   EQU     $04             ;PUT OPTION TABLE
= 0005      AMISC   EQU     $05             ;MISCELLANEOUS DISK OPERATION
= 0007      AOTHER  EQU     $07             ;OTHER

                    ASSERT  AIDLE=0         ;LEGAL ASSUMPTION

                    **      HEAD POSITION KNOWN BIT

= 0008      HKNOWN  EQU     $08             ;HEAD POSITION KNOWN

                    **      CONFIGURATION CODES

= 00CF      CFCLR   EQU     $CF             ;MASK TO CLEAR CONFIGURATION CODE
= 0020      SECTSZ  EQU     $20             ;MASK TO READ SECTOR SIZE BIT

= 0020      CFSSSD  EQU     $20             ;SS/SD
= 0030      CFSS2D  EQU     $30             ;SS/2D
= 0000      CFSSDD  EQU     $00             ;SS/DD
= 0010      CFDSDD  EQU     $10             ;DS/DD

                    **      INTERNAL RAM ADDRESSES

= 00FF      BSL256  EQU     $FF             ;BAD SECTOR LIST FOR 256-BYTE SECTORS
```

EQUATES

```
      = 00FB      CSTAT   EQU     BSL256-4        ;CONTROLLER STATUS
      = 00FA      HSTAT   EQU     CSTAT-1         ;HARDWARE STATUS
      = 00F9      CDEVIC  EQU     HSTAT-1         ;COMMAND FRAME DEVICE NUMBER
      = 00F8      CCOMND  EQU     CDEVIC-1        ;COMMAND FRAME COMMAND
      = 00F7      CAUX1   EQU     CCOMND-1        ;COMMAND FRAME AUXILIARY #1
      = 00F6      CAUX2   EQU     CAUX1-1         ;COMMAND FRAME AUXILIARY #2
      = 00F5      R0SAV   EQU     CAUX2-1         ;SAVE AREA FOR R0
      = 00F4      R1SAV   EQU     R0SAV-1         ;SAVE AREA FOR R1
      = 00F3      R2SAV   EQU     R1SAV-1         ;SAVE AREA FOR R2
      = 00F2      R3SAV   EQU     R2SAV-1         ;SAVE AREA FOR R3
      = 00F1      R4SAV   EQU     R3SAV-1         ;SAVE AREA FOR R4
      = 00F0      R5SAV   EQU     R4SAV-1         ;SAVE AREA FOR R5

      = 007F      BSL128  EQU     $7F             ;BAD SECTOR LIST FOR 128-BYTE SECTORS
```

MACROS

0000

```
                ;**      JDRQ - JUMP IF DRQ

                JDRQ    MACRO   ADDR
                %L      JT1     %1
                        ENDM


                ;**      JNDRQ - JUMP IF NOT DRQ

                JNDRQ   MACRO   ADDR
                %L      JNT1    %1
                        ENDM


                ;**      SETREG - SET WD1770 REGISTER

                SETREG  MACRO   REG
                %L      ANLI    P1,WDCLR
                        ORLI    P1,WD%1
                        ENDM


                ;**      JSOD - JUMP IF SERIAL OUTPUT DATA HIGH

                JSOD    MACRO   ADDR
                %L      JNT0    %1
                        ENDM


                ;**      JNSOD - JUMP IF SERIAL OUTPUT DATA LOW

                JNSOD   MACRO   ADDR
                %L      JT0     %1
                        ENDM


                ;**      CLRSID - CLEAR SERIAL INPUT DATA

                CLRSID  MACRO
                %L      ORLI    P1,SID
                        ENDM


                ;**      SETSID - SET SERIAL INPUT DATA

                SETSID  MACRO
                %L      ANLI    P1,$FF-SID
                        ENDM


                ;**      WINT - WAIT FOR INTERRUPT

                WINT    MACRO
                %L
                MM%K    JUMP    MM%K
```

```
                ENDM


**      LNGJMP - LONG JUMP

LNGJMP  MACRO   ADDR
%L      DB      $E5 OR [[[%1]&$0800]/$80]
        JUMP    %1
        ENDM


**      PAUSE - DELAY FOR A FEW CPU CYCLES

PAUSE   MACRO   REGISTER,CYCLES
%L
        IF      [%2]&1
        NOPP
        ENDIF
        IF      [[%2]&$FFFE]=2
        NOPP
        NOPP
        ENDIF
        IF      [%2]>3
        MOVI    %1,[[%2]/2]-1
MM%K    DJNZ    %1,MM%K
        ENDIF
        ENDM


**      LNGCAL - LONG CALL

LNGCAL  MACRO   ADDR
%L      DB      $E5 OR [[[%1]&$0800]/$80]
        CALL    %1
        DB      $E5 OR [[[*]&$0800]/$80]
        ENDM
```

```
0000
0000  = 1000              ORG     $1000

                  **      RESET COMES HERE

1000                      LNGJMP  RESET           ;GO DO RESET ROUTINE


                  **      EXTERNAL INTERRUPT COMES HERE

                          ASSERT  *=$1003

1003                      SEL     RB0
1004                      LNGJMP  COMPRO          ;GO TO COMMAND FRAME PROCESSOR

                  **      TIMER INTERRUPT COMES HERE

                          ASSERT  *=$1007

                  ;       THIS CODE HANDLES TIMEOUTS WHICH OCCUR DURING RECEPTION OF
                  ;       A DATA FRAME FROM THE SERIAL BUS.

1007                      SEL     MB0
1008                      DIS     TCNTI
1009                      STOP    TCNT
100A                      CALL    ENABLE          ;RESET INTERRUPT-IN-PROGRESS FLIP-FLOP

                  ;       JUMP    INVDF           ;INDICATE INVALID DATA FRAME


                  **      INVDF - PROCESS INVALID DATA FRAME

100C              INVDF
100C                      MOVI    R0,CSTAT
100E                      MOVI    XR0,$02         ;INVALID DATA FRAME

                  ;       JUMP    IDLE            ;GO TO IDLE STATE


                  **      IDLE - ENTER IDLE STATE

1010              IDLE
1010                      MOVI    R7,SEC3         ;3 SECOND TRIP COUNT
1012                      STRT    T
1013                      ANLI    P2,ACLR         ;INDICATE IDLE STATE
1015                      EN      I
1016              IDLE1
1016                      JTF     IDLE2           ;IF TIMER TRIPPED
1018                      JUMP    IDLE1           ;WAIT FOR TRIP
101A              IDLE2
101A                      DJNZ    R7,IDLE1        ;IF 3 SECONDS NOT ELAPSED
101C              IDLE4
101C                      SETREG  RSTA
1020                      MOVX    A,XR0           ;READ THE STATUS REGISTER
1021                      CPL     A
1022                      JB7     IDLE5           ;IF MOTOR OFF
```

```
                ;       MOTOR TIMED OUT, SO RESET 1770 TO FORCE MOTOR OFF

1024            IDLE3
1024                    ANLI    P2,$FF-HKNOWN   ;INDICATE HEAD POSITION UNKNOWN
1026                    ANLI    P1,$FF-MR       ;SEND RESET SIGNAL
1028            IDLE5
1028                    WINT                    ;WAIT FOR INTERRUPT

                ;       ENTRY POINT FOR RETURN FROM INTERRUPT

102A            INTRET
102A                    EN      I
102B                    IN      A,P1
102C                    ANLI    A,MR
102E                    JZ      IDLE5           ;IF RESET IS LOW
1030                    MOV     A,R7
1031                    JNZ     IDLE1           ;IF TIMING IN PROGRESS
1033                    JUMP    IDLE4           ;MAKE SURE MOTOR IS OFF


                **      ENABLE - RESET THE INTERRUPT-IN-PROGRESS FLIP-FLOP

1035            ENABLE
1035                    RETR


                **      COMPRO - COMMAND FRAME PROCESSOR

1036            COMPRO

                ;       GET OFF THE SERIAL BUS

1036                    SETSID

                ;       RESET THE INTERRUPT-IN-PROGRESS FLIP-FLOP

1038                    DIS     TCNTI
1039                    DIS     I
103A                    CALL    ENABLE

                ;       CHECK TO SEE IF WE WERE IN MID-OPERATION

103C                    IN      A,P2
103D                    ANLI    A,$FF-ACLR
103F                    JZ      COM1            ;IF NOT IN MID-OPERATION

                ;       CONSTRUCT PSEUDO-STATUS IN CSTAT

1041                    MOVI    R0,CSTAT
1043                    MOVI    XR0,$04         ;STATUS FOR UNSUCCESSFUL OPERATION

                ;       SET UP MOTOR TIMEOUT

1045                    MOVI    R7,SEC3         ;3 SECOND TRIP COUNT
1047                    STRT    T               ;START THE TIMER GOING

                ;       INDICATE IDLE STATE
```

```
1048                    ANLI    P2,ACLR

                ;       INITIATE A FORCED INTERRUPT TO THE 1770 IF NECESSARY

104A                    XRLI    A,APOPT
104C                    JZ      COM1            ;IF PUT OPTION TABLE
104E                    XRLI    A,APOPT XOR AOTHER
1050                    JZ      COM1            ;IF OTHER NON-DISK OPERATION
1052                    IN      A,P1
1053                    ANLI    A,MR
1055                    JZ      COM1            ;IF RESET IS LOW

1057                    SETREG  WCMD
105B                    MOVI    A,FRCINT
105D                    MOVX    XR0,A

                ;       INITIALIZE FOR COMMAND FRAME RECEPTION

105E            COM1
105E                    MOVI    R0,CDEVIC       ;STARTING ADDRESS
1060                    MOVI    R1,0            ;INIT CHECKSUM
1062                    MOVI    R2,5            ;NUMBER OF BYTES

                ;       CHECK COMMAND LINE

1064            COM3
1064                    JNI     COM5            ;IF COMMAND LINE LOW
1066            COM4
1066                    JUMP    INTRET          ;COMMAND FRAME SCREWED UP, RETURN

                ;       STORE BYTE OF COMMAND FRAME INTO RAM AND UPDATE CHECKSUM

1068            COM2
1068                    MOV     XR0,A
1069                    DECR    R0
106A                    ADD     A,R1
106B                    ADDCI   A,0
106D                    MOV     R1,A
106E                    PAUSE   R3,12
1072                    JNSOD   COM12           ;IF STOP BIT IS BAD

                ;       WAIT FOR START BIT

1074            COM5
1074                    JSOD    COM3            ;IF START BIT NOT DETECTED

                ;       INITIALIZE FOR DATA BYTE RECEPTION

1076                    MOVI    A,$80
1078                    PAUSE   R3,16           ;DELAY 16 CPU CYCLES

                ;       LOOP TO READ IN BYTE BIT-BY-BIT

107C            COM6
107C                    PAUSE   R3,19           ;DELAY 19 CPU CYCLES
1081                    CLR     C
```

```
1082                    JNSOD   COM7           ;IF BIT IS A ZERO
1084                    CPL     C
1085                    JUMP    COM8
1087            COM7
1087                    PAUSE   R3,3           ;EQUALIZE TIME OF THE TWO PATHS
108A            COM8
108A                    RRC     A              ;SHIFT IN THE BIT
108B                    JNC     COM6           ;IF NOT DONE WITH 8 BITS

                ;       CHECK CHECKSUM

108D                    DJNZ    R2,COM2        ;IF CHECKSUM NOT RECEIVED YET
108F                    XRL     A,R1
1090                    JNZ     COM12          ;IF CHECKSUM DOESN'T MATCH
1092                    PAUSE   R3,15
1097                    JNSOD   COM12          ;IF STOP BIT IS WRONG
1099                    JNI     COM11          ;IF COMMAND LINE IS STILL LOW
109B                    JUMP    INTRET

                ;       CHECK TO SEE IF COMMAND IS FOR US

109D            COM11
109D                    MOVI    R1,$31         ;DEVICE NUMBER FOR UNIT #1
109F                    IN      A,P2           ;READ DEVICE SELECT SWITCHES
10A0                    ANLI    A,UNITMK       ;SELECT UNIT # BITS
10A2                    XRLI    A,UNIT1
10A4                    JZ      COM10          ;IF UNIT 1
10A6                    INCR    R1
10A7                    XRLI    A,UNIT2 XOR UNIT1
10A9                    JZ      COM10          ;IF UNIT 2
10AB                    INCR    R1
10AC                    XRLI    A,UNIT3 XOR UNIT2
10AE                    JZ      COM10          ;IF UNIT 3
10B0                    INCR    R1             ;ELSE, MUST BE UNIT 4
10B1            COM10
10B1                    MOVI    R0,CDEVIC
10B3                    MOV     A,XR0          ;BRING IN DEVICE NUMBER
10B4                    XRL     A,R1
10B5                    JZ      COM13          ;IF COMMAND IS FOR US
10B7            COM12
10B7                    JNI     COM12          ;WAIT FOR COMMAND TO GO HIGH
10B9                    JUMP    INTRET

                ;       BRANCH TO VARIOUS COMMAND ROUTINES

10BB            COM13
10BB                    JUMP    COMLK
                        ASSERT  *<=$1100
10BD  = 1100            ORG     $1100
1100            COMLK

1100                    DECR    R0
1101                    MOV     A,XR0          ;BRING IN COMMAND
1102                    XRLI    A,$52
1104                    JZ      COM28
1106                    XRLI    A,$72 XOR $52
1108                    JNZ     COM20
```

```
110A              COM28
110A                      ORLI    P2,AGET         ;SET ACTIVITY CODE FOR GET
110C                      LNGJMP  GET             ;GET SECTOR
110F              COM20
110F                      XRLI    A,$50 XOR $72
1111                      JNZ     COM21
1113                      ORLI    P2,APUT         ;SET ACTIVITY CODE FOR PUT
1115                      LNGJMP  PUT             ;PUT SECTOR
1118              COM21
1118                      XRLI    A,$57 XOR $50
111A                      JNZ     COM22
111C                      ORLI    P2,APUTV        ;SET ACTIVITY CODE FOR PUT/VERIFY
111E                      LNGJMP  PUTV            ;PUT/VERIFY SECTOR
1121              COM22
1121                      XRLI    A,$53 XOR $57
1123                      JNZ     COM23
1125                      ORLI    P2,AOTHER       ;ACTIVITY CODE FOR 'OTHER'
1127                      LNGJMP  STAT            ;STATUS
112A              COM23
112A                      XRLI    A,$21 XOR $53
112C                      JNZ     COM24
112E                      ORLI    P2,AMISC        ;ACTIVITY CODE FOR MISCELLANEOUS
1130                      LNGJMP  FMT             ;FORMAT
1133              COM24
1133                      XRLI    A,$4E XOR $21
1135                      JNZ     COM25
1137                      ORLI    P2,AOTHER       ;ACTIVITY CODE FOR 'OTHER'
1139                      LNGJMP  GOPT            ;GET OPTION TABLE
113C              COM25
113C                      XRLI    A,$4F XOR $4E
113E                      JNZ     COM26
1140                      ORLI    P2,APOPT        ;ACTIVITY CODE FOR PUT OPTION TABLE
1142                      LNGJMP  POPT            ;PUT OPTION TABLE
1145              COM26
1145                      XRLI    A,$48 XOR $4F
1147                      JNZ     COM27
1149                      ORLI    P2,AOTHER       ;ACTIVITY CODE FOR 'OTHER'
114B                      LNGJMP  SPEED           ;SET SIO SPEED
114E              COM27
114E                      XRLI    A,$22 XOR $48
1150                      JNZ     INVCF
1152                      IN      A,P2
1153                      ANLI    A,SECTSZ
1155                      JZ      INVCF           ;IF SECTOR SIZE IS 256
1157                      ORLI    P2,AMISC        ;ACTIVITY CODE FOR MISCELLANEOUS
1159                      LNGJMP  FMTD            ;FORMAT DUAL DENSITY


                  **      INVCF - PROCESS INVALID COMMAND FRAME

115C              INVCF
115C                      MOVI    R0,CSTAT
115E                      MOVI    A,$01           ;STATUS FOR INVALID COMMAND FRAME
1160                      MOV     XR0,A           ;SAVE IT

1161                      MOVI    A,NAK
1163                      LNGCAL  XBYTE           ;SEND NAK
```

```
1167                    JUMP    INTRET          ;RETURN
```

```
1169
                **      RESET - POWER ON INITIALIZATION

1169            RESET
1169                    DIS     I
116A                    DIS     TCNTI
116B                    MOVI    A,INP1+WDRSTA+DDEN
116D                    OUTL    P1,A
116E                    MOVI    A,INP2+CFSSSD+AIDLE
1170                    OUTL    P2,A
1171                    SEL     RB0
1172                    CLR     F0              ;INIT TO SLOW MODE
1173                    CLR     F1

1174                    STOP    TCNT
1175                    CLR     A
1176                    MOV     T,A
1177                    MOVI    R0,MS240
1179                    STRT    T
117A                    JTF     RESET1          ;CLEAR TIMER OVERFLOW FLAG
117C            RESET1
117C                    JTF     RESET2
117E                    JUMP    RESET1
1180            RESET2
1180                    DJNZ    R0,RESET1       ;IF NOT FINISHED WITH 1/4 SECOND DELAY
1182                    STOP    TCNT

1183                    MOVI    R0,CSTAT
1185                    MOVI    XR0,0           ;INIT CONTROLLER STATUS
1187                    EN      I
1188                    WINT                    ;WAIT FOR INTERRUPT
```

```
118A

                ;**     SENDAK - SEND ACKNOWLEDGE
                ;*
                ;*      EXIT CONDITIONS: A & R6 MODIFIED, INTERRUPTS ENABLED,
                ;*              TIMER STOPPED AND CLEARED, TIMER OVERFLOW CLEARED,
                ;*              RETRY FLAG CLEARED

118A            SENDAK
118A                    STOP    TCNT
118B                    CLR     A
118C                    MOV     T,A
118D                    JTF     SNDAK1          ;CLEAR TIMER OVERFLOW FLAG
118F            SNDAK1
118F                    MOVI    A,HSACK         ;HIGH SPEED ACK
1191                    JF0     SNDAK2          ;IF HIGH SPEED ENABLED
1193                    MOVI    A,ACK           ;NORMAL ACK
1195            SNDAK2
                ;       JUMP    XBYTE           ;SEND BYTE, RETURN


                ;**     XBYTE - SEND A BYTE OVER SERIAL BUS AT 19200 BAUD
                ;*
                ;*      ENTRY CONDITION: BYTE TO BE SENT IN A
                ;*
                ;*      EXIT CONDITION: R6 & A ARE MODIFIED, INTERRUPTS ARE ENABLED,
                ;*              RETRY FLAG CLEARED

1195            XBYTE
1195                    CLR     F1              ;CLEAR RETRY FLAG

1196            XBYTE4
1196                    JNI     XBYTE4          ;WAIT FOR COMMAND TO GO HIGH
1198                    JNI     XBYTE4          ;JUST IN CASE
119A                    EN      I               ;RE-ENABLE INTERRUPTS

119B                    CLRSID                  ;SEND START BIT
119D                    CLR     C
119E                    CPL     C
119F                    PAUSE   R6,3            ;DELAY 3 CPU CYCLES
11A2            XBYTE1
11A2                    PAUSE   R6,18           ;DELAY 18 CPU CYCLES
11A6                    RRC     A               ;SHIFT OUT NEXT BIT
11A7                    JC      XBYTE2          ;IF BIT IS A ONE
11A9                    CLRSID                  ;SEND A ZERO
11AB                    JUMP    XBYTE3
11AD            XBYTE2
11AD                    SETSID                  ;SEND A ONE
11AF                    JUMP    XBYTE3          ;EQUALIZE THE TIME AROUND BOTH PATHS
11B1            XBYTE3
11B1                    CLR     C
11B2                    JNZ     XBYTE1          ;IF NOT DONE WITH 8 BITS
11B4                    RET
```

```
11B5

                **      RECV - RECEIVE DATA FRAME
                *
                *       ENTRY CONDITIONS: (NUMBER OF BYTES TO RECEIVE)-1 IN R0,
                *               TIMER STOPPED AND CLEARED, TIMER OVERFLOW CLEARED
                *
                *       EXIT CONDITIONS: DATA FRAME IN INTERNAL RAM,
                *               RETRY FLAG CLEARED,
                *               EXIT THRU INVDF IF CHECKSUM ERROR OR TIMEOUT.

                        ASSERT  *<=$1200
11B5  = 1200            ORG     $1200

1200            RECV
1200                    EN      TCNTI
1201                    STRT    T               ;BEGIN TIMEOUT FOR FIRST BYTE
1202            RECV22
1202                    JSOD    RECV22          ;WAIT FOR START BIT
1204                    CLR     A               ;INIT CHECKSUM
1205                    CLR     F1              ;INDICATE LAST BYTE NOT RECEIVED
1206                    NOPP
1207                    NOPP                    ;DELAY 2 CPU CYCLES
1208                    JUMP    RECV21

                ;       READ IN THE FIRST [R0] BYTES

120A            RECV5
120A                    JNSOD   RECV19          ;IF STOP BIT BAD
120C            RECV1
120C                    JSOD    RECV1           ;IF START BIT NOT DETECTED
120E                    INCR    R0
120F                    MOV     XR0,A           ;PLACE BYTE INTO RAM
1210                    DECR    R0
1211                    ADD     A,R1
1212                    ADDCI   A,0
1214            RECV21
1214                    MOV     R1,A            ;UPDATE CHECKSUM
1215            RECV10
1215                    CLR     A
1216                    MOV     T,A             ;RESET THE TIMER
1217                    MOVI    A,$80           ;INIT FOR DATA BYTE RECEPTION

1219                    NOPP
121A                    NOPP
121B                    NOPP
121C                    NOPP
121D                    JF0     RECV6           ;DELAY 6 CPU CYCLES IN FAST MODE
121F                    NOPP
1220                    NOPP                    ;DELAY 8 CYCLES IN SLOW MODE

1221            RECV2
1221                    NOPP
1222                    NOPP
1223                    NOPP
1224                    JF0     RECV6           ;DELAY 5 CPU CYCLES IN FAST MODE
1226                    NOPP
```

```
1227                    NOPP
1228                    NOPP
1229                    NOPP
122A                    NOPP
122B                    NOPP
122C                    NOPP
122D                    NOPP
122E                    NOPP
122F                    NOPP
1230                    NOPP
1231                    NOPP
1232                    NOPP
1233                    NOPP                    ;DELAY 19 CYCLES IN SLOW MODE

1234            RECV6
1234                    CLR     C
1235                    JNSOD   RECV3           ;IF BIT IS A ZERO
1237                    CPL     C
1238                    JUMP    RECV4
123A            RECV3
123A                    NOPP
123B                    NOPP
123C                    NOPP                    ;EQUALIZE TIME AROUND BOTH PATHS
123D            RECV4
123D                    RRC     A               ;SHIFT IN THE BIT
123E                    JNC     RECV2           ;IF NOT DONE WITH 8 BITS
1240                    JF0     RECV7           ;DELAY 2 CYCLES IN FAST MODE
1242                    NOPP
1243                    NOPP
1244                    NOPP
1245                    NOPP
1246                    NOPP
1247                    NOPP
1248                    NOPP
1249                    NOPP
124A                    NOPP
124B                    NOPP
124C                    NOPP
124D                    NOPP
124E                    NOPP
124F                    NOPP                    ;DELAY 16 CYCLES IN SLOW MODE

1250            RECV7
1250                    JF1     RECV13          ;IF LAST BYTE RECEIVED
1252                    DJNZ    R0,RECV5        ;IF MORE BYTES TO DO
1254                    JNSOD   RECV19          ;IF STOP BIT BAD

                ;       BRING IN THE LAST DATA BYTE

1256            RECV8
1256                    JSOD    RECV8           ;IF START BIT NOT DETECTED
1258                    XCH     A,R1            ;PLACE BYTE INTO RAM
1259                    ADD     A,R1
125A                    ADDC    A,R0            ;R0 CONTAINS 0
125B                    MOV     R0,A            ;UPDATE CHECKSUM
125C                    CPL     F1              ;INDICATE LAST BYTE RECEIVED
125D                    JUMP    RECV10          ;GO RECEIVE LAST BYTE
```

```
                ;       COMPARE CALCULATED CHECKSUM TO RECEIVED CHECKSUM

125F            RECV13
125F                    CLR     F1              ;CLEAR RETRY FLAG
1260                    NOPP                    ;DELAY 1 CPU CYCLE
1261                    JNSOD   RECV19          ;IF STOP BIT IS BAD
1263            RECV14
1263                    JSOD    RECV14          ;WAIT FOR START BIT
1265                    STOP    TCNT
1266                    DIS     TCNTI
1267                    MOV     T,A             ;SAVE LAST BYTE IN TIMER
1268                    ADD     A,R0
1269                    ADDCI   A,0             ;FINISH CHECKSUM CALCULATION
126B                    CLR     C
126C                    CPL     C

126D                    NOPP
126E                    NOPP
126F                    NOPP
1270                    NOPP
1271                    NOPP
1272                    JF0     RECV18          ;DELAY 7 CYCLES IN FAST MODE
1274                    NOPP                    ;DELAY 8 CYCLES IN SLOW MODE

1275            RECV15
1275                    NOPP
1276                    NOPP
1277                    NOPP
1278                    NOPP
1279                    JF0     RECV18          ;DELAY 6 CPU CYCLES IN FAST MODE
127B                    NOPP
127C                    NOPP
127D                    NOPP
127E                    NOPP
127F                    NOPP
1280                    NOPP
1281                    NOPP
1282                    NOPP
1283                    NOPP
1284                    NOPP
1285                    NOPP
1286                    NOPP
1287                    NOPP
1288                    NOPP                    ;DELAY 20 CYCLES IN SLOW MODE

1289            RECV18
1289                    RRC     A               ;SHIFT OUT NEXT BIT TO COMPARE
128A                    JC      RECV16          ;IF BIT IS A ONE
128C                    JNSOD   RECV17          ;IF RECEIVED BIT IS ZERO
128E            RECV19
128E                    LNGJMP  INVDF           ;GO PROCESS INVALID DATA FRAME
1291            RECV16
1291                    JNSOD   RECV19          ;IF RECEIVED BIT IS ZERO, ERROR
1293            RECV17
1293                    CLR     C
1294                    JNZ     RECV15          ;IF DONE ALL BITS IN CHECKSUM
```

```
                ;       ACKNOWLEDGE THE DATA FRAME

1296                    PAUSE   R0,US500
129A                    PAUSE   R0,US500        ;DELAY 1000 US

129E                    CLRSID                  ;SEND START BIT
12A0                    MOVI    A,ACK           ;NORMAL ACK
12A2                    CLR     C
12A3                    CPL     C
12A4                    PAUSE   R0,1
12A5            RECV23
12A5                    PAUSE   R0,2
12A7                    JF0     RECV26          ;IF FAST MODE
12A9                    PAUSE   R0,14
12AD            RECV26
12AD                    RRC     A               ;SHIFT OUT NEXT BIT
12AE                    JC      RECV24          ;IF BIT IS A ONE
12B0                    CLRSID                  ;SEND OUT A ZERO
12B2                    JUMP    RECV25
12B4            RECV24
12B4                    SETSID                  ;SEND OUT A ONE
12B6                    JUMP    RECV25          ;EQUALIZE TIME AROUND BOTH PATHS
12B8            RECV25
12B8                    CLR     C
12B9                    JNZ     RECV23          ;IF DONE WITH 8 BITS

                ;       RETURN TO CALLER

12BB                    IN      A,P2
12BC                    ANLI    A,$FF-ACLR      ;GET ACTIVITY CODE
12BE                    XRLI    A,APOPT
12C0                    JZ      RECV20          ;IF PUT OPTION TABLE
12C2                    LNGJMP  PUTRR           ;GO TO PUT RECEIVE RETURN
12C5            RECV20
12C5                    LNGJMP  POPTRR          ;GO TO PUT OPTION TABLE RECIEVE RETURN
```

```
12C8

                ;**     SEND - SEND DATA FRAME
                ;*
                ;*      ENTRY CONDITIONS:
                ;*              (NUMBER OF BYTES TO SEND)-1 IN R0 (0 IF NO DATA FRAME)
                ;*              F1 SET IF ERROR, CLEAR IF SUCCESS
                ;*
                ;*      EXIT CONDITIONS:
                ;*              ENTERS IDLE STATE
                ;*

                        ASSERT  *<=$1300
12C8  = 1300            ORG     $1300

1300            SEND

                ;       SEND COMPLT/ERROR

1300                    MOV     A,PSW
1301                    ORLI    A,1             ;INDICATE NO NEED TO UPDATE CSTAT
1303                    MOV     PSW,A
1304                    MOVI    A,ERROR
1306                    JF1     SEND0           ;IF ERROR
1308                    MOV     A,PSW
1309                    ANLI    A,$FE
130B                    MOV     PSW,A           ;INDICATE CSTAT UPDATE DESIRED
130C                    MOVI    A,COMPLT
130E            SEND0

                ;       LOOP TO SEND ONE BYTE

130E            SEND1
130E                    CLRSID                  ;SEND START BIT
1310                    CLR     C
1311                    CPL     C
1312                    NOPP
1313                    NOPP
1314                    NOPP                    ;DELAY 3 CYCLES
1315            SEND2
1315                    NOPP
1316                    NOPP
1317                    JF0     SEND3           ;DELAY 4 CYCLES IN FAST MODE
1319                    NOPP
131A                    NOPP
131B                    NOPP
131C                    NOPP
131D                    NOPP
131E                    NOPP
131F                    NOPP
1320                    NOPP
1321                    NOPP
1322                    NOPP
1323                    NOPP
1324                    NOPP
1325                    NOPP
1326                    NOPP                    ;DELAY 18 CYCLES IN SLOW MODE
```

```
1327              SEND3
1327                      RRC     A               ;SHIFT OUT NEXT BIT
1328                      JC      SEND4           ;IF BIT IS A ONE
132A                      CLRSID                  ;SEND OUT A ZERO
132C                      JUMP    SEND5
132E              SEND4
132E                      SETSID                  ;SEND OUT A ONE
1330                      JUMP    SEND5           ;EQUALIZE TIME AROUND BOTH PATHS
1332              SEND5
1332                      CLR     C
1333                      JNZ     SEND2           ;IF NOT DONE WITH 8 BITS

                  ;       CHECK TO SEE IF DATA FOLLOWS

1335                      MOV     A,R0
1336                      JZ      SEND7           ;IF NO DATA FOLLOWS
1338                      PAUSE   R0,US600
133C                      PAUSE   R0,US600        ;DELAY 1200 US, LEAVING R0 = 0
1340                      XCH     A,R0            ;RESTORE R0, INIT CHECKSUM (IN A) TO 0
1341                      CLR     F1              ;INDICATE LAST BYTE NOT SENT

                  ;       SEND THE DATA FRAME

1342              SEND11
1342                      CLRSID                  ;SEND START BIT
1344                      ADD     A,XR0
1345                      ADDCI   A,0             ;UPDATE CHECKSUM
1347                      XCH     A,XR0           ;SAVE CHECKSUM, BRING DATA BYTE TO A
1348                      CLR     C
1349                      CPL     C
134A                      JF0     SEND16          ;IF FAST MODE
134C                      JUMP    SEND10
134E              SEND12
134E                      NOPP
134F                      NOPP
1350                      JF0     SEND13          ;DELAY 4 CYCLES IN FAST MODE
1352                      NOPP
1353              SEND10
1353                      NOPP
1354                      NOPP
1355                      NOPP
1356                      NOPP
1357                      NOPP
1358                      NOPP
1359                      NOPP
135A                      NOPP
135B                      NOPP
135C                      NOPP
135D                      NOPP
135E                      NOPP
135F              SEND16
135F                      NOPP                    ;DELAY 18 CYCLES IN SLOW MODE
1360              SEND13
1360                      RRC     A               ;SHIFT OUT NEXT BIT
1361                      JC      SEND14          ;IF BIT IS A ONE
1363                      CLRSID                  ;SEND OUT A ZERO
1365                      JUMP    SEND15
```

```
1367              SEND14
1367                      SETSID                  ;SEND OUT A ONE
1369                      JUMP    SEND15          ;EQUALIZE TIME AROUND BOTH PATHS
136B              SEND15
136B                      CLR     C
136C                      JNZ     SEND12          ;IF NOT DONE WITH 8 BITS

                  ;       GET NEXT BYTE TO SEND

136E                      DJNZ    R0,SEND6        ;ADVANCE POINTER
1370                      JF1     SEND17          ;IF LAST BYTE HAS BEEN SENT
1372                      MOV     A,T             ;GET LAST BYTE OF FRAME
1373                      XCH     A,R1            ;BYTE TO RAM, CHECKSUM TO A
1374                      INCR    R0              ;SO DJNZ WILL FAIL AGAIN
1375                      CPL     F1              ;INDICATE LAST BYTE SENT
1376                      JF0     SEND11          ;IF FAST MODE
1378                      JUMP    SEND18

137A              SEND6
137A                      INCR    R0
137B                      MOV     A,XR0           ;BRING CHECKSUM TO A
137C                      DECR    R0
137D                      JF0     SEND11          ;IF FAST MODE
137F                      NOPP
1380                      NOPP
1381                      NOPP
1382                      NOPP
1383                      NOPP
1384              SEND18
1384                      NOPP
1385                      NOPP
1386                      NOPP
1387                      NOPP
1388                      NOPP
1389                      NOPP
138A                      NOPP                    ;DELAY 12 CYCLES
138B                      JUMP    SEND11          ;GO SEND THE BYTE

138D              SEND17
138D                      MOV     A,R1            ;MOVE CHECKSUM TO A
138E                      JF0     SEND1           ;IF FAST MODE
1390                      NOPP
1391                      NOPP
1392                      NOPP
1393                      NOPP
1394                      NOPP
1395                      NOPP
1396                      NOPP
1397                      NOPP
1398                      NOPP
1399                      NOPP
139A                      NOPP
139B                      NOPP                    ;DELAY 12 CYCLES
139C                      JUMP    SEND1           ;GO SEND THE CHECKSUM

                  ;       UPDATE CSTAT IF NECESSARY, GO TO IDLE
```

```
139E              SEND7
139E                      MOV     A,PSW
139F                      JB0     SEND9           ;IF NO NEED TO UPDATE CSTAT
13A1                      MOVI    R0,CSTAT
13A3                      MOVI    XR0,0           ;UPDATE STATUS TO SUCCESS
13A5              SEND9
13A5                      LNGJMP  IDLE            ;GO TO IDLE STATE
```

STATUS

```
13A8

                **      STAT - GET STATUS

13A8            STAT
13A8                    LNGCAL  SENDAK          ;SEND ACKNOWLEDGE
13AC                    PAUSE   R0,US500
13B0                    PAUSE   R0,US500        ;1000 US COMPLT DELAY
13B4                    MOVI    R3,0            ;FOR SS/SD
13B6                    IN      A,P1
13B7                    ANLI    A,DDEN
13B9                    JNZ     STAT5           ;IF FM
13BB                    MOVI    R3,$80          ;FOR SS/2D
13BD                    IN      A,P2
13BE                    JB5     STAT5           ;IF 128 BYTES/SECTOR
13C0                    MOVI    R3,$A0          ;FOR DS/DD
13C2                    JB4     STAT5           ;IF DS/DD
13C4                    MOVI    R3,$20
13C6            STAT5
13C6                    MOVI    R0,CSTAT
13C8                    MOV     A,XR0           ;GET CONTROLLER STATUS
13C9                    ORL     A,R3
13CA                    MOV     R3,A

13CB                    MOVI    R2,$FF          ;HARDWARE STATUS FOR NO ERROR
13CD                    IN      A,P1
13CE                    ANLI    A,MR
13D0                    JZ      STAT1           ;IF RESET IS LOW
13D2                    SETREG  RSTA
13D6                    MOVX    A,XR0           ;READ 1770 STATUS REGISTER
13D7                    CPL     A               ;INVERT IT
13D8                    MOV     R2,A            ;SAVE IT
13D9                    JB7     STAT1           ;IF MOTOR OFF
13DB                    MOV     A,R3
13DC                    ORLI    A,$10           ;TURN ON MOTOR BIT
13DE                    MOV     R3,A

13DF            STAT1
13DF                    MOVI    R1,$78          ;TIMEOUT
13E1                    MOVI    A,0
13E3                    MOV     T,A             ;LAST BYTE = 0
13E4                    MOVI    R0,3            ;LENGTH OF FRAME -1
13E6                    CLR     F1              ;SIGNAL NO ERROR
13E7                    LNGJMP  SEND            ;SEND DATA FRAME
```

SET SIO SPEED

```
13EA

                **      SPEED - SET SIO SPEED

                        ASSERT  *<=$1400
13EA  = 1400            ORG     $1400

1400            SPEED
1400                    MOVI    R0,CAUX1
1402                    MOV     A,XR0
1403                    JZ      SPEED1          ;IF SLOW SPEED DESIRED
1405                    XRLI    A,1
1407                    JNZ     SPEED2          ;IF INVALID SPEED CODE

                ;       SET FAST SPEED
                ;       NOTE: INTERRUPTS MUST BE DISABLED AT THIS POINT

1409                    CLR     F0
140A                    CPL     F0
140B                    JUMP    SPEED3

                ;       SET SLOW SPEED

140D            SPEED1
140D                    CLR     F0
140E            SPEED3
140E                    LNGCAL  SENDAK          ;ACKNOWLEDGE
1412                    PAUSE   R0,US500
1416                    PAUSE   R0,US500        ;DELAY, LEAVING R0 = 0
141A                    CLR     F1              ;INDICATE NO ERROR
141B                    LNGJMP  SEND            ;SEND COMPLT, GO TO IDLE

                ;       PROCESS INVALID CODE

141E            SPEED2
141E                    LNGJMP  INVCF           ;INVALID COMMAND FRAME
```

```
1421

                **      GOPT - GET OPTION TABLE

1421            GOPT
1421                    LNGCAL  SENDAK          ;ACKNOWLEDGE
1425                    PAUSE   R0,US500
1429                    PAUSE   R0,US500        ;GENERATE COMPLT DELAY
142D                    MOVI    R0,11
142F                    MOVI    XR0,40          ;NUMBER OF TRACKS
1431                    DECR    R0
1432                    MOVI    XR0,SRATE       ;STEP RATE
1434                    DECR    R0
1435                    MOVI    XR0,0           ;SECTORS PER TRACK (MSB)
1437                    DECR    R0
1438                    IN      A,P2
1439                    ANLI    A,$FF-CFCLR     ;GET CURRENT CONFIG
143B                    XRLI    A,CFSS2D
143D                    MOVI    XR0,26          ;26 SECTORS PER TRACK FOR SS/2D
143F                    JZ      GOPT1           ;IF SS/2D
1441                    MOVI    XR0,18          ;18 SECTORS PER TRACK FOR OTHERS
1443                    IN      A,P2
1444                    JB5     GOPT1           ;IF SS/SD
1446                    DIS     I
1447                    ORLI    P2,SECTSZ
1449                    IN      A,P2
144A                    JB5     GOPT2           ;IF DRIVE IS A MINIFLOPPY
144C                    MOVI    XR0,16          ;16 SECTORS PER TRACK FOR MICROFLOPPY
144E            GOPT2
144E                    ANLI    P2,$FF-SECTSZ   ;FIX UP SECTOR SIZE BIT
1450                    EN      I
1451            GOPT1
1451                    DECR    R0
1452                    MOVI    XR0,0           ;SINGLE SIDED
1454                    IN      A,P2
1455                    ANLI    A,$FF-CFCLR     ;GET CURRENT CONFIG
1457                    XRLI    A,CFDSDD
1459                    JNZ     GOPT3           ;IF NOT DS/DD
145B                    MOVI    XR0,1           ;DOUBLE SIDED
145D            GOPT3
145D                    DECR    R0
145E                    MOVI    XR0,0           ;FM
1460                    IN      A,P2
1461                    ANLI    A,$FF-CFCLR     ;GET CURRENT CONFIG
1463                    XRLI    A,CFSSSD
1465                    JNZ     GOPT4           ;IF NOT SS/SD
1467                    MOVI    XR0,4           ;MFM
1469            GOPT4
1469                    DECR    R0
146A                    MOVI    XR0,0
146C                    DECR    R0
146D                    MOVI    XR0,$80         ;128 BYTES PER SECTOR
146F                    IN      A,P2
1470                    JB5     GOPT5           ;IF 128 BYTES PER SECTOR
1472                    INCR    R0
1473                    MOVI    XR0,1
1475                    DECR    R0
```

```
1476                    MOVI    XR0,0           ;256 BYTES PER SECTOR
1478            GOPT5
1478                    DECR    R0
1479                    MOVI    XR0,1           ;DRIVE PRESENT
147B                    DECR    R0
147C                    MOVI    XR0,HSACK       ;FAST MODE
147E                    JF0     GOPT6           ;IF FAST MODE
1480                    MOVI    XR0,ACK         ;SLOW MODE
1482            GOPT6
1482                    CLR     A
1483                    MOV     T,A
1484                    MOV     R1,A
1485                    MOVI    R0,11           ;FRAME SIZE
1487                    CLR     F1              ;NO ERROR
1488                    LNGJMP  SEND            ;SEND FRAME, GO TO IDLE
```

PUT OPTION TABLE

```
148B

                **      POPT - PUT OPTION TABLE

148B            POPT
148B                    LNGCAL  SENDAK          ;ACKNOWLEDGE
148F                    MOVI    R0,11           ;FRAME SIZE
1491                    LNGJMP  RECV            ;GO RECEIVE THE FRAME
1494            POPTRR
1494                    MOVI    R0,8            ;OFFSET TO # OF SECTORS PER TRACK
1496                    DIS     I
1497                    IN      A,P2
1498                    ORLI    A,CFCLR         ;GET CURRENT CONFIG
149A                    MOV     R1,A            ;SAVE IT
149B                    ORLI    P2,$FF-CFCLR    ;SET CONFIG TO ALL 1'S

                ;       CHECK FOR SS/SD MODE

149D                    MOVI    R2,CFSSSD OR CFCLR
149F                    MOV     A,R6            ;RECORDING MODE
14A0                    JNZ     POPT1           ;IF NOT FM
14A2                    MOV     A,R7            ;NUMBER OF SIDES
14A3                    JNZ     POPTER          ;IF NOT SINGLE SIDED, ERROR
14A5                    IN      A,P2
14A6                    JB5     POPTSU          ;IF MINIFOLPPY, SUCCESS
14A8                    JUMP    POPTER          ;ELSE, ERROR

                ;       CHECK RECORDING MODE FOR MFM

14AA            POPT1
14AA                    XRLI    A,4
14AC                    JNZ     POPTER          ;NOT MFM, ERROR

                ;       CHECK FOR DS/DD

14AE                    MOVI    R2,CFDSDD OR CFCLR
14B0                    MOV     A,R7            ;NUMBER OF SIDES
14B1                    JZ      POPT2           ;IF SINGLE SIDE
14B3                    XRLI    A,1
14B5                    JNZ     POPTER          ;NOT DOUBLE SIDED, ERROR
14B7                    ORLI    P1,SSO          ;TRY TO SELECT SECOND SIDE
14B9                    IN      A,P1
14BA                    ANLI    P1,$FF-SSO      ;RE-SELECT FIRST SIDE
14BC                    ANLI    A,SSO
14BE                    JNZ     POPTSU          ;SECOND SIDE OK
14C0                    JUMP    POPTER          ;ELSE ERROR

                ;       CHECK FOR SS/2D

14C2            POPT2
14C2                    MOVI    R2,CFSS2D OR CFCLR
14C4                    MOV     A,XR0           ;# OF SECTORS PER TRACK
14C5                    XRLI    A,26
14C7                    JNZ     POPT3           ;IF NOT SS/2D
14C9                    MOV     A,R5            ;# BYTES PER SECTOR MSB
14CA                    JNZ     POPT3           ;IF NOT SS/2D
14CC                    MOV     A,R4            ;# BYTES PER SECTOR LSB
```

PUT OPTION TABLE

```
14CD                    XRLI    A,$80
14CF                    JNZ     POPT3           ;IF NOT SS/2D
14D1                    IN      A,P2
14D2                    JB5     POPTSU          ;IN DRIVE IS A MINIFLOPPY
14D4                    JUMP    POPTER          ;IF MICROFLOPPY, ERROR

                ;       COME HERE FOR SS/DD

14D6            POPT3
14D6                    MOVI    R2,CFSSDD OR CFCLR

                ;       COME HERE FOR SUCCESSFUL CONFIGURE

14D8            POPTSU
14D8                    MOV     A,R2            ;NEW CONFIGURATION
14D9                    MOV     R1,A
14DA                    CLR     F1              ;NO ERROR
14DB                    JUMP    POPT4

                ;       COME HERE IF ERROR

14DD            POPTER
14DD                    CLR     F1
14DE                    CPL     F1              ;ERROR FLAG
14DF                    MOVI    R0,CSTAT
14E1                    MOVI    XR0,$04         ;STATUS FOR UNSUCCESSFUL OPERATION

                ;       SEND COMPLT/ERROR AND GO TO IDLE

14E3            POPT4
14E3                    IN      A,P2
14E4                    ANL     A,R1            ;SET NEW CONFIG
14E5                    ORLI    A,INP2          ;SET INPUT BITS
14E7                    OUTL    P2,A            ;WRITE TO PORT
14E8                    EN      I
14E9                    PAUSE   R0,US500        ;DELAY, LEAVING R0 = 0 (NO DATA FRAME)
14ED                    LNGJMP  SEND            ;SEND COMPLT/ERROR, GO TO IDLE
```

```
14F0
                **      FMT - FORMAT IN CURRENT DENSITY

                        ASSERT  *<=$1500
14F0  = 1500            ORG     $1500

1500            FMT
1500                    ORLI    P1,DDEN         ;SELECT FM
1502                    IN      A,P2
1503                    ANLI    A,$FF-CFCLR     ;GET CURRENT CONFIG
1505                    XRLI    A,CFSSSD
1507                    JZ      FMT1            ;IF SS/SD

                **      FMTD - FORMAT IN DUAL DENSITY

1509            FMTD
1509                    ANLI    P1,$FF-DDEN     ;SELECT MFM


                ;       SEND ACKNOWLEDGE

150B            FMT1
150B                    LNGCAL  SENDAK

                ;       DO RESET SEQUENCE

150F                    ANLI    P2,$FF-HKNOWN   ;INDICATE HEAD POSITION UNKNOWN
1511                    ANLI    P1,$FF-MR       ;SEND RESET TO 1770
1513                    PAUSE   R0,US500
1517                    ORLI    P1,MR           ;REMOVE RESET
1519                    PAUSE   R0,US500
151D                    SETREG  WCMD
1521                    MOVI    A,FRCINT
1523                    MOVX    XR0,A           ;INITIATE FORCED INTERRUPT
1524                    PAUSE   R0,US200

                ;       INITIALIZE DATA FRAMES FOR ERROR

1528                    IN      A,P2
1529                    RL      A
152A                    RL      A
152B                    ANLI    A,$80
152D                    DECR    A               ;OFFSET TO FIRST BYTE OF DATA FRAME
152E                    MOV     R3,A            ;SAVE IT
152F                    MOV     R0,A
1530                    CLR     A
1531                    MOV     XR0,A
1532                    DECR    R0
1533                    MOV     XR0,A
1534                    CPL     A
1535                    DECR    R0
1536                    MOV     XR0,A
1537                    DECR    R0
1538                    MOV     XR0,A
1539                    MOVI    R0,CSTAT
153B                    MOVI    XR0,$04         ;STATUS FOR UNSUCCESSFUL OPERATION
```

```
                ;       SET FLAG TO INDICATE FORMAT

153D                    CLR     F1

                ;       RESTORE TO TRACK 0

153E            FMT2
153E                    ANLI    P2,$FF-HKNOWN   ;INDICATE HEAD POSITON UNKNOWN
1540                    SETREG  WCMD
1544                    MOVI    A,REST
1546                    MOVX    XR0,A           ;SEND RESTORE COMMAND TO 1770

1547                    MOVI    R0,SEC3         ;TRIP COUNT FOR 3-SECOND DELAY
1549                    CLR     A
154A                    MOV     T,A             ;RESET TIMER
154B                    STRT    T

154C            FMT3
154C                    JTF     FMT4
154E            FMT5
154E                    IN      A,P1
154F                    ANLI    A,INTRQ
1551                    JZ      FMT3            ;IF NOT DONE

1553                    STOP    TCNT
1554                    SETREG  RSTA
1558                    MOVX    A,XR0           ;READ STATUS TO CLEAR INTRQ
1559                    SETREG  WTRK
155D                    CLR     A
155E                    MOVX    XR0,A           ;UPDATE TRACK REG TO 0
155F                    MOV     R2,A            ;INIT R2 = TRACK
1560                    ANLI    P1,$FF-SSO      ;SELECT SIDE 0
1562                    PAUSE   R0,US500
1566                    JUMP    FMT6            ;GO TO TRACK LOOP

1568            FMT4
1568                    DJNZ    R0,FMT5         ;IF NOT TIMED OUT

                ;       PROCESS ERROR

156A            FMT7
156A                    STOP    TCNT
156B                    SETREG  WCMD
156F                    MOVI    A,FRCINT
1571                    MOVX    XR0,A           ;DO FORCED INTERRUPT
1572                    PAUSE   R0,US200
1576                    CLR     F1
1577                    CPL     F1              ;INDICATE ERROR
1578                    MOV     A,R3
1579                    MOV     R0,A            ;DATA FRAME LENGTH
157A                    LNGJMP  SEND            ;SEND ERROR, GO TO IDLE

                ;       MAIN TRACK LOOP
                ;       SEEK DESIRED TRACK

157D            FMT6
157D                    SETREG  WDAT
```

```
1581                    MOV     A,R2            ;GET DESIRED TRACK
1582                    MOVX    XR0,A           ;WRITE IT TO DATA REG
1583                    ANLI    P2,$FF-HKNOWN   ;INDICATE HEAD POSITION UNKNOWN
1585                    SETREG  WCMD
1589                    MOVI    A,SEEK
158B                    MOVX    XR0,A           ;ISSUE SEEK COMMAND
158C                    CLR     A
158D                    MOV     T,A
158E                    JTF     FMT8            ;CLEAR TIMER OVERFLOW FLAG
1590            FMT8
1590                    MOVI    R0,SEC1         ;TRIP COUNT FOR 1-SECOND TIMEOUT

1592            FMT9
1592                    JTF     FMT10
1594            FMT11
1594                    IN      A,P1
1595                    ANLI    A,INTRQ
1597                    JZ      FMT9            ;IF NOT DONE

1599                    STOP    TCNT
159A                    SETREG  RSTA
159E                    MOVX    XR0,A           ;READ STATUS REG TO CLEAR INTRQ
159F                    JUMP    ILV             ;GO CONSTRUCT INTERLEAVE TABLE

15A1            FMT10
15A1                    DJNZ    R0,FMT11        ;IF NOT TIMED OUT
15A3                    JUMP    FMT7            ;GO PROCESS ERROR

                ;       CONSTRUCT SECTOR INTERLEAVE TABLE IN INTERNAL RAM

15A5            ILV

                ;       SET R4 = NUMBER OF SECTORS PER TRACK
                ;           R5 = -(SECTOR INTERLEAVE FACTOR)
                ;           R1 = INIT INDEX INTO INTERLEAVE TABLE

15A5                    MOVI    R4,18           ;18 SECTORS PER TRACK
15A7                    MOVI    R5,$100-9       ;SECTOR INTERLEAVE FACTOR OF 9
15A9                    IN      A,P2
15AA                    JB5     ILV1            ;IF 128 BYTES/SECTOR
15AC                    JF0     ILV2            ;IF FAST MODE
15AE                    MOVI    R5,$100-15      ;SECTOR INTERLEAVE OF 15
15B0            ILV2
15B0                    DIS     I
15B1                    ORLI    P2,SECTSZ
15B3                    IN      A,P2
15B4                    JB5     ILV3            ;IF MINIFLOPPY
15B6                    MOVI    R4,16           ;16 SECTORS/TRACK FOR MICROFLOPPY
15B8            ILV3
15B8                    ANLI    P2,$FF-SECTSZ
15BA                    EN      I
15BB                    JUMP    ILV4
15BD            ILV1
15BD                    IN      A,P1
15BE                    JB3     ILV4            ;IF FM
15C0                    MOVI    R4,26           ;26 SECTORS/TRACK
15C2                    MOVI    R5,$100-13      ;SECTOR INTERLEAVE OF 13
```

```
                ;       CLEAR THE INTERLEAVE TABLE

15C4            ILV4
15C4                    MOV     A,R4
15C5                    MOV     R0,A            ;INIT R0 = NUMBER OF SECTORS
15C6                    ADDI    A,$7F
15C8                    MOV     R6,A            ;SAVE INIT INDEX INTO TABLE
15C9                    MOV     R1,A
15CA            ILV5
15CA                    MOVI    XR1,0           ;STORE 0 IN TABLE
15CC                    DECR    R1              ;ADVANCE POINTER
15CD                    MOV     A,R1
15CE                    JB7     ILV5            ;IF NOT DONE WITH TABLE

                ;       FILL THE TABLE

15D0                    MOV     A,R6
15D1                    JUMP    ILV7
15D3            ILV6
15D3                    MOV     A,R1            ;TABLE INDEX
15D4                    ADD     A,R5            ;ADD INTERLEAVE FACTOR
15D5            ILV8
15D5                    JB7     ILV7            ;IF STILL WITHIN RANGE
15D7                    ADD     A,R4            ;WRAP POINTER
15D8            ILV7
15D8                    MOV     R1,A
15D9                    MOV     A,XR1
15DA                    JZ      ILV9            ;IF TABLE ENTRY NOT YET USED
15DC                    DECR    R1
15DD                    MOV     A,R1
15DE                    JUMP    ILV8            ;USE NEXT ENTRY
15E0            ILV9
15E0                    IN      A,P1
15E1                    ANLI    A,SSO
15E3                    JNZ     ILV10           ;IF SECOND SIDE
15E5                    MOV     A,R0
15E6                    CPL     A
15E7                    ADD     A,R4
15E8                    ADDI    A,2             ;REFLECT SECTOR NUMBER
15EA                    JUMP    ILV11
15EC            ILV10
15EC                    MOV     A,R0
15ED            ILV11
15ED                    MOV     XR1,A           ;STORE SECTOR NUMBER IN TABLE
15EE                    DJNZ    R0,ILV6         ;IF MORE SECTORS TO DO
15F0                    MOV     A,R6
15F1                    MOV     R1,A            ;R1 = INIT INDEX INTO TABLE

                ;       INITIALIZE FOR WRITE TRACK

15F2                    JUMP    WTR
                        ASSERT  *<=$1600
15F4  = 1600            ORG     $1600

1600            WTR
1600                    JF1     VFY             ;IF VERIFY
```

```
                ;       SET R4 = GAP FILLER
                ;           R5 = PRE-MARK GAP LENGTH
                ;           R6 = POST-ID GAP LENGTH
                ;           R7 = POST-DATA GAP LENGTH
                ;           R0 = POST-INDEX GAP LENGTH - 1

1602                    MOVI    R4,$00
1604                    MOVI    R5,6
1606                    MOVI    R6,11
1608                    MOVI    R7,12
160A                    MOVI    R0,39
160C                    IN      A,P1
160D                    JB3     WTR1            ;IF FM
160F                    MOVI    R4,$4E
1611                    MOVI    R5,12
1613                    MOVI    R6,22
1615                    MOVI    R7,24
1617                    MOVI    R0,59

                ;       ISSUE WRITE TRACK COMMAND

1619            WTR1
1619                    SETREG  WCMD
161D                    MOVI    A,WRTRK
161F                    MOVX    XR0,A
1620                    SETREG  WDAT
1624                    CLR     A
1625                    MOV     T,A             ;RESET THE TIMER
1626                    JTF     WTR2            ;CLEAR TIMER OVERFLOW FLAG
1628            WTR2
1628                    MOVI    A,SEC1          ;TRIP COUNT FOR 1 SECOND TIMEOUT
162A                    XCH     A,R4
162B                    STRT    T
162C                    JUMP    WTR5            ;GO WRITE THE TRACK

                **      VERIFY TRACK

162E            VFY

                ;       ISSUE READ SECTOR COMMAND

162E            VFY1
162E                    STOP    TCNT
162F                    CLR     A
1630                    MOV     T,A
1631                    MOV     A,R3
1632                    INCR    A
1633                    MOV     R0,A            ;SECTOR SIZE
1634                    MOVI    R4,SEC1         ;TRIP COUNT FOR 1 SECOND TIMEOUT
1636                    SETREG  WTRK
163A                    MOV     A,R2            ;TRACK NUMBER
163B                    MOVX    XR0,A
163C                    SETREG  WSEC
1640                    MOV     A,XR1           ;SECTOR NUMBER FROM INTERLEAVE TABLE
1641                    DECR    R1              ;ADVANCE INTERLEAVE POINTER
1642                    MOVX    XR0,A
```

```
1643                    SETREG  WCMD
1647                    MOVI    A,RDSEC
1649                    MOVX    XR0,A
164A                    SETREG  RDAT
164E                    STRT    T

                ;       READ THE DATA BYTES

164F            VFY2
164F                    JTF     VFY3
1651            VFY4
1651                    JNDRQ   VFY2
1653                    JNDRQ   VFY2
1655            VFY5
1655                    MOVX    A,XR0
1656                    DJNZ    R0,VFY4         ;IF MORE BYTE TO READ

                ;       WAIT FOR OPERATION COMPLETE

1658                    CLR     A
1659                    MOV     T,A             ;RESET THE TIMER
165A                    JTF     VFY6            ;RESET TIMER OVERFLOW FLAG
165C            VFY6
165C                    JTF     VFY7            ;IF TIMED OUT
165E                    IN      A,P1
165F                    ANLI    A,INTRQ
1661                    JZ      VFY6            ;IF NOT DONE

                ;       READ STATUS

1663                    STOP    TCNT
1664                    SETREG  RSTA
1668                    MOVX    A,XR0           ;READ STATUS REG
1669                    ANLI    A,$3C           ;RELEVANT STATUS BITS
166B                    JNZ     VFY7            ;IF ERROR

                ;       SEE IF MORE SECTORS TO DO

166D                    MOV     A,R1
166E                    JB7     VFY1            ;IF MORE SECTORS
1670                    JUMP    FMT12           ;GO DO NEXT TRACK

                ;       HANDLE VERIFY TIMEOUTS

1672            VFY3
1672                    JNDRQ   VFY8
1674                    JDRQ    VFY5
1676            VFY8
1676                    DJNZ    R4,VFY4         ;IF NOT TIMED OUT
1678            VFY7
1678                    JUMP    FMT7            ;GO HANDLE ERROR

                **      GO TO NEXT TRACK

167A            FMT12
167A                    IN      A,P1
167B                    JB6     FMT13           ;IF SECOND SIDE
```

```
167D                    INCR    R2              ;NEXT TRACK
167E                    MOV     A,R2
167F                    XRLI    A,40
1681                    JNZ     FMT14           ;IF NOT DONE WITH 40 TRACKS
1683                    IN      A,P2
1684                    ANLI    A,$FF-CFCLR
1686                    XRLI    A,CFDSDD
1688                    JNZ     FMT15           ;IF NOT DOUBLE SIDED
168A                    ORLI    P1,SSO          ;SELECT SECOND SIDE
168C                    PAUSE   R0,US500

1690            FMT13
1690                    MOV     A,R2
1691                    JZ      FMT15           ;IF DONE WITH 40 TRACKS
1693                    DECR    R2              ;NEXT TRACK
1694            FMT14
1694                    JUMP    FMT6            ;GO DO NEW TRACK

                ;       COME HERE IF COMPLETE PASS IS FINISHED

1696            FMT15
1696                    JF1     FMT16           ;IF WE JUST FINISHED VERIFY
1698                    CPL     F1              ;INDICATE VERIFY IN PROGRESS
1699                    JUMP    FMT2            ;GO DO SECOND PASS

                ;       COME HERE FOR SUCCESSFUL FORMAT/VERIFY

169B            FMT16
169B                    STOP    TCNT
169C                    CLR     F1              ;INDICATE SUCCESS
169D                    MOV     A,R3
169E                    MOV     R0,A            ;DATA FRAME LENGTH
169F                    MOVI    XR0,$FF
16A1                    DECR    R0
16A2                    MOVI    XR0,$FF         ;INDICATE NO BAD SECTORS
16A4                    INCR    R0
16A5                    LNGJMP  SEND            ;SEND COMPLT AND FRAME, GO TO IDLE

                **      WRITE TRACK

                        ASSERT  *<=$1700
16A8  = 1700            ORG     $1700

                ;       WRITE POST-INDEX GAP (GAP I)

1700            WTR3
1700                    JTF     WTR4
1702            WTR5
1702                    JNDRQ   WTR3
1704                    JNDRQ   WTR3            ;FILTER DRQ GLITCHES
1706            WTR50
1706                    MOVX    XR0,A           ;WRITE A GAP BYTE
1707                    DJNZ    R0,WTR5         ;IF NOT DONE WITH GAP
1709                    MOV     T,A             ;RESET THE TIMER
170A                    JTF     WTR9            ;RESET TIMER OVERFLOW FLAG
170C            WTR9
170C                    JDRQ    WTR10
```

```
170E            WTR11
170E                    JTF     WTR12
1710                    JNDRQ   WTR11
1712            WTR10
1712                    MOVX    XR0,A           ;WRITE LAST BYTE OF GAP
1713                    MOV     R4,A

                ;       SECTOR LOOP

1714            WTR6

                ;       WRITE PRE-IDAM GAP (END OF GAP III)

1714                    MOV     A,R5
1715                    MOV     R0,A
1716                    CLR     A
1717                    JDRQ    WTR44
1719            WTR8
1719                    JTF     WTR12
171B            WTR13
171B                    JNDRQ   WTR8
171D            WTR44
171D                    MOVX    XR0,A
171E                    DJNZ    R0,WTR13

                ;       WRITE MFM SYNC MARKS

1720                    IN      A,P1
1721                    JB3     WTR14           ;IF FM
1723                    MOVI    A,$F5           ;F5 WRITES MFM SYNC MARK
1725                    JDRQ    WTR15
1727            WTR16
1727                    JTF     WTR12
1729                    JNDRQ   WTR16
172B            WTR15
172B                    MOVX    XR0,A
172C                    MOVI    R0,2            ;WRITE 2 MORE SYNC MARKS
172E            WTR17
172E                    JTF     WTR12
1730            WTR18
1730                    JNDRQ   WTR17
1732                    MOVX    XR0,A
1733                    DJNZ    R0,WTR18

                ;       WRITE ID FIELD

1735            WTR14
1735                    MOVI    A,$FE           ;ID ADDRESS MARK (IDAM)
1737                    JDRQ    WTR19
1739            WTR20
1739                    JTF     WTR12
173B                    JNDRQ   WTR20
173D            WTR19
173D                    MOVX    XR0,A           ;WRITE IDAM

173E                    MOV     A,R2            ;TRACK NUMBER
173F            WTR21
```

```
173F                    JTF     WTR12
1741                    JNDRQ   WTR21
1743                    MOVX    XR0,A           ;WRITE TRACK NUMBER

1744                    IN      A,P1
1745                    RL      A
1746                    RL      A
1747                    ANLI    A,1             ;SIDE NUMBER
1749            WTR22
1749                    JTF     WTR12
174B                    JNDRQ   WTR22
174D                    MOVX    XR0,A           ;WRITE SIDE NUMBER

174E                    MOV     A,XR1           ;SECTOR NUMBER FROM INTERLEAVE TABLE
174F                    DECR    R1              ;ADVANCE INTERLEAVE POINTER
1750            WTR23
1750                    JTF     WTR12
1752                    JNDRQ   WTR23
1754                    MOVX    XR0,A           ;WRITE SECTOR NUMBER

1755                    IN      A,P2
1756                    JB5     WTR24           ;IF SECTOR SIZE IS 128
1758                    INCR    R0
1759            WTR24
1759                    MOV     A,R0            ;SECTOR SIZE CODE
175A            WTR25
175A                    JTF     WTR12
175C                    JNDRQ   WTR25
175E                    MOVX    XR0,A           ;WRITE SECTOR SIZE CODE

175F                    MOVI    A,$F7           ;F7 WRITES CRC
1761            WTR26
1761                    JTF     WTR12
1763                    JNDRQ   WTR26
1765                    MOVX    XR0,A           ;WRITE CRC

                ;       WRITE POST-ID GAP (START OF GAP II)

1766                    MOV     A,R6
1767                    MOV     R0,A            ;LENGTH OF POST-ID GAP
1768                    MOV     A,R4            ;GAP FILLER
1769            WTR27
1769                    JTF     WTR12
176B            WTR28
176B                    JNDRQ   WTR27
176D                    MOVX    XR0,A
176E                    DJNZ    R0,WTR28

                ;       WRITE PRE-DAM GAP (END OF GAP II)

1770                    MOV     A,R5
1771                    MOV     R0,A
1772                    CLR     A
1773                    MOV     T,A             ;RESET THE TIMER
1774            WTR29
1774                    JTF     WTR12
1776            WTR30
```

```
1776                    JNDRQ   WTR29
1778                    MOVX    XR0,A
1779                    DJNZ    R0,WTR30

                ;       WRITE MFM SYNC MARKS

177B                    IN      A,P1
177C                    JB3     WTR35           ;IF FM
177E                    MOVI    A,$F5           ;F5 WRITES MFM SYNC MARK
1780                    JDRQ    WTR32
1782            WTR31
1782                    JTF     WTR12
1784                    JNDRQ   WTR31
1786            WTR32
1786                    MOVX    XR0,A
1787                    MOVI    R0,2            ;WRITE 2 MORE SYNC MARKS
1789            WTR33
1789                    JTF     WTR12
178B            WTR34
178B                    JNDRQ   WTR33
178D                    MOVX    XR0,A
178E                    DJNZ    R0,WTR34

                ;       WRITE DATA FIELD

1790            WTR35
1790                    MOVI    A,$FB           ;FB IS DATA ADDRESS MARK (DAM)
1792                    JDRQ    WTR37
1794            WTR36
1794                    JTF     WTR12
1796                    JNDRQ   WTR36
1798            WTR37
1798                    MOVX    XR0,A           ;WRITE DAM

1799                    MOV     A,R3
179A                    MOV     R0,A
179B                    INCR    R0              ;SECTOR SIZE
179C                    MOVI    A,$FF           ;DATA BYTE = FF
179E            WTR39
179E                    JTF     WTR12
17A0            WTR40
17A0                    JNDRQ   WTR39
17A2                    MOVX    XR0,A           ;WRITE DATA BYTE
17A3                    DJNZ    R0,WTR40        ;IF MORE DATA

17A5                    MOVI    A,$F7           ;F7 WRITES CRC
17A7            WTR41
17A7                    JTF     WTR12
17A9                    JNDRQ   WTR41
17AB                    MOVX    XR0,A           ;WRITE CRC

                ;       WRITE POST-DATA GAP (START OF GAP III)

17AC                    MOV     A,R7            ;LENGTH OF POST-DATA GAP
17AD                    MOV     R0,A
17AE                    MOV     A,R4            ;GAP FILLER BYTE
17AF            WTR42
```

```
17AF                    JTF     WTR12
17B1            WTR43
17B1                    JNDRQ   WTR42
17B3                    MOVX    XR0,A           ;WRITE A GAP BYTE
17B4                    DJNZ    R0,WTR43

                ;       SEE IF MORE SECTORS TO WRITE

17B6                    MOV     A,R1
17B7                    JB7     WTR6            ;IF MORE SECTORS

                ;       WRITE GAP IV

17B9                    MOVI    R0,SEC1         ;TRIP COUNT FOR 1 SECOND TIMEOUT
17BB            WTR45
17BB                    MOV     A,R4            ;GAP FILLER BYTE
17BC                    JNDRQ   WTR46
17BE                    MOVX    XR0,A           ;WRITE GAP BYTE
17BF            WTR46
17BF                    IN      A,P1
17C0                    JB2     WTR47           ;IF DONE
17C2                    MOV     A,R4
17C3                    JNDRQ   WTR48
17C5                    MOVX    XR0,A           ;WRITE GAP BYTE
17C6            WTR48
17C6                    JTF     WTR49
17C8                    JUMP    WTR45
17CA            WTR49
17CA                    DJNZ    R0,WTR45        ;IF NOT TIMED OUT

                ;       HANDLE TIMEOUT

17CC            WTR12
17CC                    JUMP    FMT7

17CE            WTR4
17CE                    JNDRQ   WTR51
17D0                    JDRQ    WTR50           ;FILTER DRQ GLITCHES
17D2            WTR51
17D2                    DJNZ    R4,WTR5         ;IF NOT TIMED OUT
17D4                    IN      A,P1
17D5                    CPL     A
17D6                    JB2     WTR52           ;IF OPERATION NOT COMPLETE
17D8                    SETREG  RSTA
17DC                    MOVX    A,XR0           ;READ STATUS
17DD                    CPL     A
17DE                    JB6     WTR52           ;IF NOT WRITE PROTECT
17E0                    MOVI    R0,CSTAT
17E2                    MOVI    XR0,$0C         ;STATUS FOR WRITE PROTECT
17E4            WTR52
17E4                    JUMP    FMT7

                ;       HANDLE WRITE TRACK OPERATION COMPLETE

17E6            WTR47
17E6                    SETREG  RSTA
17EA                    MOVX    A,XR0           ;READ STATUS
```

```
17EB                    ANLI    A,$44           ;RELEVANT STATUS BITS
17ED                    JNZ     WTR52           ;IF ERROR
17EF                    JUMP    FMT12           ;GO TO NEXT TRACK
```

```
17F1

                **      CTS - CALCULATE TRACK AND SECTOR
                *
                *       ENTRY CONDITIONS:
                *               LOGICAL SECTOR NUMBER IN CAUX1 AND CAUX2
                *               PSW[1:0] = RETURN TARGET (CTSRT0 - CTSRT2)
                *
                *       EXIT CONDITIONS:
                *               PHYSICAL TRACK NUMBER IN 1770 DATA REG
                *               PHYSICAL SECTOR NUMBER IN 1770 SECTOR REG
                *               A MODIFIED
                *               R0-R7 UNMODIFIED
                *               C SET IF ERROR
                *               C CLEAR IF SUCCESS

                        ASSERT  *<=$17FD
17F1  = 1800            ORG     $1800

1800            CTS

                ;       SAVE REGISTERS WE WILL USE

1800                    MOV     A,R0
1801                    MOVI    R0,R0SAV
1803                    MOV     XR0,A
1804                    DECR    R0
1805                    MOV     A,R1
1806                    MOV     XR0,A
1807                    DECR    R0
1808                    MOV     A,R2
1809                    MOV     XR0,A
180A                    DECR    R0
180B                    MOV     A,R3
180C                    MOV     XR0,A
180D                    DECR    R0
180E                    MOV     A,R4
180F                    MOV     XR0,A
1810                    DECR    R0
1811                    MOV     A,R5
1812                    MOV     XR0,A

                ;       CHECK FOR COPY PROTECT COMMAND

1813                    MOVI    R0,CCOMND
1815                    MOV     A,XR0
1816                    DECR    R0
1817                    XRLI    A,$72
1819                    JNZ     CTS10           ;IF NOT COPY PROTECT COMMAND
181B                    MOV     A,XR0           ;BRING IN TRACK #
181C                    MOV     R0,A
181D                    ADDI    A,$FF-39
181F                    JC      CTSERR          ;IF TRACK NUMBER TOO BIG
1821                    ANLI    P1,$FF-SSO      ;SELECT FIRST SIDE
1823                    MOVI    A,IMPSEC        ;'IMPOSSIBLE' SECTOR NUMBER
1825                    JUMP    CTS11
```

```
                ;       BRING REDUCED LOGICAL SECTOR NUMBER INTO R2 (LSB) AND R3 (MSB)

1827            CTS10
1827                    MOV     A,XR0
1828                    ADDI    A,$FF
182A                    MOV     R2,A
182B                    DECR    R0
182C                    MOV     A,XR0
182D                    ADDCI   A,$FF
182F                    MOV     R3,A

                ;       SET R1 = NEGATIVE OF NUMBER OF SECTORS PER TRACK
                ;       R4,R5 = NEGATIVE OF NUMBER OF SECTORS PER SIDE (LSB,MSB)

1830                    MOVI    R1,$100-18
1832                    MOVI    R4,LOW [0-720]
1834                    MOVI    R5,HIGH [0-720]
1836                    IN      A,P2
1837                    JB5     CTS1            ;IF 128 BYTES PER SECTOR
1839                    DIS     I
183A                    ORLI    P2,SECTSZ       ;TRY TO SET 128 BYTES PER SECTOR
183C                    IN      A,P2
183D                    JB5     CTS2            ;IF DRIVE IS A MINIFLOPPY
183F                    MOVI    R1,$100-16      ;16 SECTORS PER TRACK FOR MICROFLOPPY
1841                    MOVI    R4,LOW [0-640]
1843                    MOVI    R5,LOW [0-640]
1845            CTS2
1845                    ANLI    P2,$FF-SECTSZ
1847                    EN      I
1848                    JUMP    CTS3
184A            CTS1
184A                    IN      A,P1
184B                    ANLI    A,DDEN
184D                    JNZ     CTS3            ;IF SINGLE DENSITY
184F                    MOVI    R1,$100-26      ;26 SECTORS PER TRACK FOR DUAL DENSITY
1851                    MOVI    R4,LOW [0-1040]
1853                    MOVI    R5,HIGH [0-1040]

                ;       DO BOUNDS CHECK, UPDATE SIDE SELECT OUTPUT

1855            CTS3
1855                    MOV     A,R2
1856                    ADD     A,R4
1857                    MOV     R0,A
1858                    MOV     A,R3
1859                    ADDC    A,R5
185A                    JNC     CTS4            ;IF IN BOUNDS ON FIRST SIDE
185C                    XCH     A,R0
185D                    ADD     A,R4
185E                    CPL     A
185F                    MOV     R2,A
1860                    MOV     A,R0
1861                    ADDC    A,R5
1862                    CPL     A
1863                    MOV     R3,A
1864                    JC      CTSERR          ;IF OUT OF BOUNDS ON SECOND SIDE
```

```
1866                    IN      A,P2
1867                    ANLI    A,$FF-CFCLR
1869                    XRLI    A,CFDSDD
186B                    CLR     C
186C                    CPL     C               ;INDICATE ERROR
186D                    JNZ     CTSERR          ;IF SINGLE SIDED

186F                    ORLI    P1,SSO          ;SELECT SECOND SIDE
1871                    JUMP    CTS5

1873            CTS4
1873                    ANLI    P1,$FF-SSO      ;SELECT FIRST SIDE

                ;       CALCULATE TRACK AND SECTOR

1875            CTS5
1875                    MOVI    R0,$FF          ;INIT TO TRACK 0
1877                    MOV     A,R2            ;SECTOR NUMBER LSB
1878                    INCR    R3
1879            CTS6
1879                    INCR    R0              ;COUNT THE TRACK
187A                    ADD     A,R1            ;SUBTRACT NUMBER OF SECTORS PER TRACK
187B                    JC      CTS6            ;IF NO BORROW
187D                    DJNZ    R3,CTS6         ;DECREMENT MSB
187F                    XCH     A,R1
1880                    CPL     A
1881                    INCR    A
1882                    ADD     A,R1            ;ADD BACK NUMBER OF SECTORS
1883                    INCR    A               ;ADD 1 FOR PHYSICAL SECTOR #

                ;       STORE TRACK AND SECTOR INTO 1770

1884            CTS11
1884                    SETREG  WSEC
1888                    MOVX    XR0,A           ;WRITE SECTOR # INTO SECTOR REG
1889                    SETREG  WDAT
188D                    MOV     A,R0
188E                    MOVX    XR0,A           ;WRITE TRACK # INTO DATA REG
188F                    CLR     C               ;INDICATE SUCCESS

                ;       RESTORE SAVED REGISTERS

1890            CTSERR
1890                    MOVI    R0,R5SAV
1892                    MOV     A,XR0
1893                    MOV     R5,A
1894                    INCR    R0
1895                    MOV     A,XR0
1896                    MOV     R4,A
1897                    INCR    R0
1898                    MOV     A,XR0
1899                    MOV     R3,A
189A                    INCR    R0
189B                    MOV     A,XR0
189C                    MOV     R2,A
189D                    INCR    R0
189E                    MOV     A,XR0
```

```
189F                    MOV     R1,A
18A0                    INCR    R0
18A1                    MOV     A,XR0
18A2                    MOV     R0,A

                ;       RETURN TO CALLER

18A3                    MOV     A,PSW
18A4                    JB0     CTS8
18A6                    JB1     CTS7
18A8                    LNGJMP  CTSRT0
18AB            CTS7
18AB                    LNGJMP  CTSRT2
18AE            CTS8
18AE                    LNGJMP  CTSRT1
```

```
18B1

                **      GET - GET SECTOR
                **      PUT - PUT SECTOR
                **      PUTV - PUT SECTOR WITH VERIFY

                        ASSERT  *<=$1900
18B1  = 1900            ORG     $1900

1900            GET
1900            PUT
1900            PUTV

                ;       RAISE 1770 RESET IF IT IS LOW

1900                    IN      A,P1
1901                    ANLI    A,MR
1903                    JNZ     SECT0           ;IF RESET IS HIGH
1905                    ORLI    P1,MR           ;RAISE RESET
1907                    PAUSE   R6,US500        ;DELAY 500 US
190B            SECT0
190B                    SETREG  WCMD
190F                    MOVI    A,FRCINT
1911                    MOVX    XR0,A           ;INITIATE FORCED INTERRUPT
1912                    PAUSE   R6,US200        ;DELAY 200 US

                ;       CALCULATE TRACK AND SECTOR NUMBERS

1916                    MOV     A,PSW
1917                    ANLI    A,$FC
1919                    MOV     PSW,A           ;SELECT RETURN TARGET = CTSRT0
191A                    LNGJMP  CTS
191D            CTSRT0
191D                    JNC     SECT1           ;IF SUCCESS
191F                    IN      A,P2
1920                    CPL     A
1921                    JB5     SECT2           ;IF SECTOR SIZE = 256
1923                    IN      A,P1
1924                    ANLI    A,DDEN
1926                    JZ      SECT2           ;IF MFM
1928                    ANLI    P1,$FF-DDEN     ;SET TO MFM
192A                    MOV     A,PSW
192B                    ANLI    A,$FC
192D                    ORLI    A,1
192F                    MOV     PSW,A           ;SELECT RETURN TARGET = CTSRT1
1930                    LNGJMP  CTS             ;TRY AGAIN TO CALCULATE
1933            CTSRT1
1933                    JNC     SECT1           ;IF SUCCESS
1935                    ORLI    P1,DDEN         ;RETURN SETTING TO FM
1937            SECT2
1937                    LNGJMP  INVCF           ;INVALID COMMAND FRAME

                ;       SEND ACKNOWLEDGE

193A            SECT1
193A                    LNGCAL  SENDAK
```

```
                ;       IN CASE OF PUT, RECEIVE DATA FRAME

193E                    IN      A,P2
193F                    ANLI    A,$FF-ACLR
1941                    XRLI    A,AGET
1943                    JZ      SECT3           ;IF GET

1945                    MOVI    R0,$7F          ;FOR 128-BYTE SECTOR
1947                    MOV     A,PSW
1948                    ORLI    A,1
194A                    MOV     PSW,A           ;SIGNAL NO BUFFER EXPANSION
194B                    IN      A,P2
194C                    JB5     SECT4           ;IF SECTOR SIZE IS 128 BYTES
194E                    IN      A,P1
194F                    ANLI    A,SSO
1951                    JNZ     SEC5            ;IF SECOND SIDE
1953                    SETREG  RDAT
1957                    MOVX    A,XR0
1958                    JNZ     SEC5            ;IF NOT TRACK 0
195A                    SETREG  RSEC
195E                    MOVX    A,XR0
195F                    ADDI    A,$FC
1961                    JC      SEC5            ;IF NOT SECTOR 1, 2, OR 3
1963                    MOV     A,PSW
1964                    ANLI    A,$FE           ;INDICATE BUFFER EXPANSION
1966                    MOV     PSW,A
1967                    JUMP    SECT4
1969            SEC5
1969                    MOVI    R0,$FF          ;FOR 256-BYTE SECTORS
196B            SECT4
196B                    LNGJMP  RECV            ;GO RECEIVE DATA FRAME
196E            PUTRR
196E                    MOV     A,PSW
196F                    JB0     SECT3           ;IF NO BUFFER EXPANSION
1971                    MOVI    R0,$80
1973                    MOV     A,T             ;LAST BYTE OF FRAME
1974                    MOV     XR0,A           ;STORE IT IN HIGH RAM
1975                    DECR    R0
1976            SEC6
1976                    MOV     A,XR0           ;GET A BYTE OF THE DATA FRAME
1977                    XCH     A,R0
1978                    ORLI    A,$80
197A                    XCH     A,R0
197B                    MOV     XR0,A           ;STORE IT INTO HIGH RAM
197C                    XCH     A,R0
197D                    ANLI    A,$7F
197F                    XCH     A,R0
1980                    DJNZ    R0,SEC6         ;IF DONE WITH 128 BYTES

                ;       CHECK TO SEE IF HEAD POSITION IS KNOWN

1982            SECT3
1982                    CLR     F1              ;INIT HARD RETRY FLAG
1983                    IN      A,P2
1984                    ANLI    A,HKNOWN
1986                    JZ      SEC19           ;IF HEAD UNKNOWN, DO RESTORE SEQUENCE
1988                    JUMP    SEC20           ;DO SEEK
```

```
198A              SEC19
198A                      SETREG  RDAT


                  **      RST - RESTORE SEQUENCE
                  *
                  *       ENTRY CONDITIONS:
                  *               P1 SET UP TO READ REG CONTAINING DESIRED TRACK
                  *               LAST BYTE OF DATA FRAME IN TIMER

198E              RST

                  ;       READ IN DESIRED TRACK AND SECTOR

198E                      MOVX    A,XR0           ;GET DESIRED TRACK
198F                      MOV     R0,A
1990                      MOV     A,PSW
1991                      XRL     A,R0
1992                      ANLI    A,$F8
1994                      XRL     A,R0
1995                      MOV     PSW,A           ;STORE BITS 2-0 OF TRACK IN PSW
1996                      MOV     A,R0
1997                      RL      A
1998                      RL      A
1999                      MOV     R0,A
199A                      SETREG  RSEC
199E                      MOVX    A,XR0           ;GET DESIRED SECTOR
199F                      XRLI    A,IMPSEC
19A1                      JZ      RST2            ;IF 'IMPOSSIBLE' SECTOR
19A3                      XRLI    A,IMPSEC
19A5              RST2
19A5                      XRL     A,R0
19A6                      ANLI    A,$1F
19A8                      XRL     A,R0            ;COMBINE TRACK[5:3] WITH SECTOR[4:0]

                  ;       DO HARD RESET TO THE 1770

19A9                      ANLI    P2,$FF-HKNOWN   ;INDICATE HEAD POSITION UNKNOWN
19AB                      ANLI    P1,$FF-MR       ;LOWER RESET
19AD                      PAUSE   R0,US500        ;500 US DELAY
19B1                      ORLI    P1,MR           ;RAISE RESET
19B3                      PAUSE   R0,US500        ;500 US DELAY
19B7                      MOV     R0,A
19B8                      MOVI    A,FRCINT
19BA                      MOVX    XR0,A           ;DO FORCED INTERRUPT
19BB                      MOV     A,R0
19BC                      PAUSE   R0,US200        ;200 US DELAY

                  ;       WRITE SECTOR NUMBER INTO SECTOR REG

19C0                      MOV     R0,A
19C1                      ANLI    A,$1F
19C3                      JNZ     RST1
19C5                      MOVI    A,IMPSEC        ;'IMPOSSIBLE' SECTOR
19C7              RST1
19C7                      SETREG  WSEC
19CB                      MOVX    XR0,A
```

```
                ;       ISSUE RESTORE COMMAND

19CC                    MOVI    A,REST
19CE                    SETREG  WCMD
19D2                    MOVX    XR0,A

                ;       SET UP COUNTER AND TIMER FOR RESTORE 3-SECOND TIMEOUT

19D3                    MOV     A,T             ;LAST DATA BYTE
19D4                    XCH     A,R0            ;PUT DATA BYTE IN R0
19D5                    RR      A
19D6                    RR      A
19D7                    RR      A
19D8                    RR      A
19D9                    RR      A               ;MOVE TRACK[5:3] INTO BITS 2-0
19DA                    ANLI    A,$07           ;INIT 5-BIT COUNTER

                ;       DELAY APPROXIMATELY 100 MS

19DC                    JUMP    RST3
                        ASSERT  *<=$1A00
19DE  = 1A00            ORG     $1A00

1A00            RST3
1A00                    MOV     T,A             ;INIT TIMER
1A01                    STRT    T
1A02            RST4
1A02                    JTF     RST5
1A04                    JUMP    RST4
1A06            RST5
1A06                    JTF     RST6
1A08                    JUMP    RST5
1A0A            RST6
1A0A                    JTF     RST7
1A0C                    JUMP    RST6
1A0E            RST7
1A0E                    JTF     RST8
1A10                    JUMP    RST7
1A12            RST8
1A12                    JTF     RST9
1A14                    JUMP    RST8
1A16            RST9
1A16                    JTF     RST10
1A18                    JUMP    RST9
1A1A            RST10
1A1A                    JTF     RST11
1A1C                    JUMP    RST10
1A1E            RST11
1A1E                    STOP    TCNT

                ;       SEE IF RESTORE COMMAND IS DONE

1A1F                    MOV     T,A
1A20                    IN      A,P1
1A21                    JB2     RST12           ;IF RESTORE IS DONE
1A23                    MOV     A,T
```

```
1A24                    ADDI    A,$08
1A26                    JNC     RST3            ;IF TIME NOT EXPIRED
1A28                    RL      A
1A29                    RL      A
1A2A                    RL      A
1A2B                    ANLI    A,$38
1A2D                    XCH     A,R0
1A2E                    MOV     T,A             ;PUT LAST DATA BYTE IN TIMER
1A2F                    MOVI    A,FRCINT
1A31                    SETREG  WCMD
1A35                    MOVX    XR0,A           ;INITIATE FORCED INTERRUPT
1A36                    MOV     A,PSW
1A37                    XRL     A,R0
1A38                    ANLI    A,$07
1A3A                    XRL     A,R0            ;RECONSTRUCT TRACK NUMBER
1A3B                    PAUSE   R0,US200        ;DELAY 200 US
1A3F                    SETREG  WTRK
1A43                    MOVX    XR0,A           ;WRITE DESIRED TRACK INTO TRACK REG
1A44                    JUMP    HRT             ;GO DO HARD RETRY

                ;       COME HERE IF RESTORE COMPLETE
                ;       AT THIS POINT, LAST DATA BYTE IS IN R0

1A46            RST12
1A46                    SETREG  RSTA
1A4A                    MOVX    A,XR0           ;READ STATUS REG TO CLEAR INTRQ
1A4B                    IN      A,P2
1A4C                    CPL     A
1A4D                    JB5     RST13           ;IF SS/DD OR DS/DD

                ;       FOR SS/SD AND SS/2D, DO READ ADDRESS TO DETERMINE DENSITY

1A4F                    ANLI    P1,$FF-DDEN     ;SELECT MFM
1A51                    MOV     A,R0            ;LAST BYTE TO A
1A52                    PAUSE   R0,US500        ;DELAY 500 US
1A56                    MOV     R0,A            ;LAST BYTE TO R0
1A57                    MOVI    A,RDADR
1A59                    SETREG  WCMD
1A5D                    MOVX    XR0,A           ;ISSUE READ ADDRESS COMMAND
1A5E                    CLR     A
1A5F                    MOV     T,A             ;INIT TIMER
1A60                    MOVI    A,MS240         ;TRIP COUNT FOR 240 MS DELAY
1A62                    JTF     RST16           ;CLEAR TIMER OVERFLOW FLAG
1A64            RST16
1A64                    STRT    T
1A65                    SETREG  RDAT            ;PREPARE TO READ DATA REG

1A69            RST17
1A69                    JTF     RST18
1A6B            RST34
1A6B                    JNDRQ   RST17
1A6D                    JNDRQ   RST17           ;FILTER DRQ GLITCHES

                ;       READ SIX BYTES FROM THE 1770

1A6F            RST19
1A6F                    MOVX    A,XR0           ;READ FIRST BYTE
```

```
1A70                    CLR     A
1A71                    MOV     T,A             ;INIT TIMER
1A72                    JTF     RST20           ;CLEAR TIMER OVERFLOW FLAG
1A74            RST20
1A74                    JDRQ    RST21
1A76            RST22
1A76                    JTF     RST25
1A78                    JNDRQ   RST22
1A7A            RST21
1A7A                    MOVX    A,XR0           ;READ SECOND BYTE
1A7B            RST23
1A7B                    JTF     RST25
1A7D                    JNDRQ   RST23
1A7F                    MOVX    A,XR0           ;READ THIRD BYTE
1A80            RST24
1A80                    JTF     RST25
1A82                    JNDRQ   RST24
1A84                    MOVX    A,XR0           ;READ FOURTH BYTE
1A85            RST29
1A85                    JTF     RST25
1A87                    JNDRQ   RST29
1A89                    MOVX    A,XR0           ;READ FIFTH BYTE
1A8A            RST30
1A8A                    JTF     RST25
1A8C                    JNDRQ   RST30
1A8E                    MOVX    A,XR0           ;READ SIXTH BYTE

1A8F            RST31
1A8F                    JTF     RST25
1A91                    IN      A,P1
1A92                    CPL     A
1A93                    JB2     RST31           ;IF 1770 STILL BUSY

1A95                    SETREG  RSTA
1A99                    MOVX    A,XR0           ;READ STATUS REG
1A9A                    ANLI    A,$1C           ;RELEVANT STATUS BITS FOR READ ADDRESS
1A9C                    JNZ     RST25           ;IF READ ADDRESS FAILED
1A9E                    JUMP    RST26           ;IF READ ADDRESS IS SUCCESSFUL

                ;       HANDLE TIMER TRIPS WHILE WAITING FOR FIRST BYTE

1AA0            RST18
1AA0                    JNDRQ   RST32
1AA2                    JDRQ    RST19           ;FILTER DRQ GLITCHES
1AA4            RST32
1AA4                    DECR    A
1AA5                    JNDRQ   RST33
1AA7                    JDRQ    RST19           ;FILTER DRQ GLITCHES
1AA9            RST33
1AA9                    JNZ     RST34           ;IF NOT TIMED OUT

                ;       COME HERE IF READ ADDRESS FAILS
                ;       AT THIS POINT, LAST DATA BYTE IS IN R0

1AAB            RST25
1AAB                    MOVI    A,FRCINT
1AAD                    SETREG  WCMD
```

```
1AB1                    MOVX    XR0,A           ;DO FORCED INTERRUPT
1AB2                    MOV     A,R0
1AB3                    PAUSE   R0,US200        ;DELAY 200 US
1AB7                    ORLI    P1,DDEN         ;SELECT FM
1AB9                    PAUSE   R0,US500        ;DELAY 500 US
1ABD                    MOV     R0,A

                ;       FINISH UP READ ADDRESS
                ;       AT THIS POINT, LAST DATA BYTE IN R0

1ABE            RST26
1ABE                    STOP    TCNT
1ABF                    MOV     A,R0
1AC0                    MOV     T,A             ;LAST DATA BYTE TO TIMER
1AC1                    MOV     A,PSW
1AC2                    ANLI    A,$FC
1AC4                    ORLI    A,2
1AC6                    MOV     PSW,A           ;SELECT RETRUN TARGET = CTSRT2
1AC7                    LNGJMP  CTS             ;CALCULATE TRACK AND SECTOR
1ACA            CTSRT2
1ACA                    JNC     RST35           ;IF SUCCESS
1ACC                    CLR     F1
1ACD                    CPL     F1              ;FORCE HARD RETRY TO FAIL
1ACE                    JUMP    HRT             ;GO DO HARD RETRY

                ;       RECONSTRUCT TRACK NUMBER

1AD0            RST13
1AD0                    MOV     A,T
1AD1                    RL      A
1AD2                    RL      A
1AD3                    RL      A
1AD4                    ANLI    A,$38
1AD6                    XCH     A,R0
1AD7                    MOV     T,A             ;PUT LAST DATA BYTE INTO TIMER
1AD8                    MOV     A,PSW
1AD9                    XRL     A,R0
1ADA                    ANLI    A,$07
1ADC                    XRL     A,R0            ;RECONSTRUCT TRACK NUMBER
1ADD                    SETREG  WDAT
1AE1                    MOVX    XR0,A           ;WRITE DESIRED TRACK INTO DATA REG
1AE2            RST35
1AE2                    CLR     A
1AE3                    SETREG  WTRK
1AE7                    MOVX    XR0,A           ;UPDATE TRACK REG TO ZERO

                ;       END OF RESTORE SEQUENCE

                ;       ISSUE SEEK COMMAND
                ;       AT THIS POINT, LAST DATA BYTE IS IN TIMER

1AE8            SEC7
1AE8                    JUMP    SEC20
                        ASSERT  *<=$1B00
1AEA  = 1B00            ORG     $1B00

                ;       CHECK TO SEE IF ALREADY AT CORRECT TRACK
```

```
1B00              SEC20
1B00                      SETREG  RTRK
1B04                      MOVX    A,XR0           ;READ CURRENT TRACK
1B05                      MOV     R0,A
1B06                      SETREG  RDAT
1B0A                      MOVX    A,XR0           ;READ DESIRED TRACK
1B0B                      XRL     A,R0            ;COMPARE
1B0C                      JNZ     SEC21           ;IF NOT AT CORRECT TRACK
1B0E                      MOV     A,T
1B0F                      MOV     R0,A            ;PUT LAST DATA BYTE IN R0
1B10                      JUMP    SEC22           ;GO FINISH SEEK
1B12              SEC21
1B12                      ANLI    P2,$FF-HKNOWN   ;INDICATE HEAD POSITION UNKNOWN
1B14                      MOVI    A,SEEK
1B16                      SETREG  WCMD
1B1A                      MOVX    XR0,A           ;ISSUE SEEK COMMAND
1B1B                      MOV     A,T             ;FETCH LAST DATA BYTE
1B1C                      MOV     R0,A            ;STORE IT
1B1D                      CLR     A
1B1E                      MOV     T,A             ;INIT TIMER
1B1F                      MOV     A,PSW
1B20                      ANLI    A,$3F
1B22                      ORLI    A,$07           ;INIT 5-BIT COUNTER WITHIN PSW
1B24                      STRT    T

                  ;       WAIT FOR SEEK COMMAND TO BE COMPLETE

1B25              SEC8
1B25                      MOV     PSW,A
1B26              SEC10
1B26                      JTF     SEC11
1B28                      IN      A,P1
1B29                      JB2     SEC9            ;IF SEEK DONE
1B2B                      JUMP    SEC10
1B2D              SEC11
1B2D                      JTF     SEC12
1B2F                      IN      A,P1
1B30                      JB2     SEC9            ;IF SEEK DONE
1B32                      JUMP    SEC11
1B34              SEC12
1B34                      JTF     SEC13
1B36                      IN      A,P1
1B37                      JB2     SEC9            ;IF SEEK DONE
1B39                      JUMP    SEC12
1B3B              SEC13
1B3B                      JTF     SEC14
1B3D                      IN      A,P1
1B3E                      JB2     SEC9            ;IF SEEK DONE
1B40                      JUMP    SEC13
1B42              SEC14
1B42                      JTF     SEC15
1B44                      IN      A,P1
1B45                      JB2     SEC9            ;IF SEEK DONE
1B47                      JUMP    SEC14
1B49              SEC15
1B49                      JTF     SEC16
```

```
1B4B                    IN      A,P1
1B4C                    JB2     SEC9            ;IF SEEK DONE
1B4E                    JUMP    SEC15

                ;       HANDLE POSSIBLE SEEK TIMEOUT

1B50            SEC16
1B50                    MOV     A,PSW
1B51                    DECR    A
1B52                    JB3     SEC8            ;IF NOT TIMED OUT
1B54                    ADDI    A,$48
1B56                    JNC     SEC8            ;IF NOT TIMED OUT
1B58                    STOP    TCNT
1B59                    MOV     A,R0
1B5A                    MOV     T,A             ;PUT LAST DATA BYTE IN TIMER
1B5B                    MOVI    A,FRCINT
1B5D                    SETREG  WCMD
1B61                    MOVX    XR0,A           ;DO FORCED INTERRUPT
1B62                    PAUSE   R0,US200        ;DELAY 200 US
1B66                    SETREG  RDAT
1B6A                    MOVX    A,XR0           ;READ IN TARGET TRACK
1B6B                    SETREG  WTRK
1B6F                    MOVX    XR0,A           ;WRITE IT INTO TRACK REG
1B70                    JUMP    HRT             ;GO DO HARD RETRY

                ;       COME HERE FOR SUCCESSFUL SEEK
                ;       AT THIS POINT, LAST DATA BYTE IN R0

1B72            SEC9
1B72                    SETREG  RSTA
1B76                    MOVX    A,XR0           ;READ STATUS REG TO CLEAR INTRQ
1B77            SEC22
1B77                    ORLI    P2,HKNOWN       ;INDICATE HEAD POSITION KNOWN
1B79                    MOV     A,PSW
1B7A                    ANLI    A,$F8
1B7C                    ORLI    A,2

                ;       COME HERE TO EXECUTE SOFT RETRY

1B7E                    JUMP    SEC18
                        ASSERT  *<=$1C00
1B80  = 1C00            ORG     $1C00

1C00            SEC18
1C00                    MOV     PSW,A           ;INIT SOFT RETRY COUNTER
1C01                    STOP    TCNT
1C02                    JTF     SEC17           ;CLEAR TIMER OVERFLOW FLAG
1C04            SEC17
1C04                    CLR     A
1C05                    MOV     T,A             ;INIT TIMER FOR ID FIELD TIMEOUT


                **      HANDLE GET COMMAND

1C06                    IN      A,P2
1C07                    ANLI    A,$FF-ACLR
1C09                    XRLI    A,AGET
```

```
1C0B                    JZ      GET1            ;IF A GET COMMAND
1C0D                    JUMP    PUT1

                ;       INITIALIZATION FOR GET

1C0F            GET1
1C0F                    IN      A,P2
1C10                    MOVI    R1,$7F          ;INIT INDEX FOR 128-BYTE SECTORS
1C12                    JB5     GET2            ;IF SECTOR SIZE IS 128
1C14                    MOVI    R1,$FF          ;INIT INDEX FOR 256-BYTE SECTORS
1C16            GET2
1C16                    MOV     A,R1
1C17                    MOV     R0,A
1C18                    DECR    R0              ;INDEX FOR SECOND BYTE
1C19                    MOVI    R2,MS2300       ;TRIP COUNT FOR 2.3 SECOND DELAY

                ;       ISSUE READ SECTOR COMMAND

1C1B                    SETREG  WCMD
1C1F                    MOVI    A,RDSEC
1C21                    MOVX    XR0,A

                ;       READ FIRST DATA BYTE

1C22                    SETREG  RDAT
1C26            GET3
1C26                    JTF     GET4
1C28            GET6
1C28                    JNDRQ   GET3            ;WAIT FOR DRQ
1C2A                    JNDRQ   GET3            ;FILTER DRQ GLITCHES
1C2C            GET5
1C2C                    MOVX    A,XR0           ;BRING IN FIRST BYTE
1C2D                    CPL     A               ;COMPLEMENT IT
1C2E                    MOV     XR1,A           ;STORE IT IN RAM
1C2F                    CLR     A
1C30                    MOV     T,A             ;RESET THE TIMER
1C31                    JTF     GET18           ;RESET TIMER OVERFLOW FLAG
1C33            GET18
1C33                    JDRQ    GET19

                ;       READ IN THE MIDDLE BYTES

1C35            GET8
1C35                    JTF     GET9
1C37            GET7
1C37                    JNDRQ   GET8
1C39            GET19
1C39                    MOVX    A,XR0           ;READ DATA BYTE
1C3A                    CPL     A               ;COMPLEMENT IT
1C3B                    MOV     XR0,A           ;STORE IT IN RAM
1C3C                    DJNZ    R0,GET7         ;IF MORE BYTES TO DO

                ;       READ IN THE LAST BYTE

1C3E            GET10
1C3E                    JTF     GET9
1C40                    JNDRQ   GET10
```

```
1C42                    MOVX    A,XR0           ;BRING IN LAST BYTE
1C43                    CPL     A               ;COMPLEMENT IT
1C44                    MOV     R0,A            ;STORE IT IN R0

                ;       WAIT FOR OPERATION COMPLETE

1C45            GET11
1C45                    JTF     GET9
1C47                    IN      A,P1
1C48                    ANLI    A,INTRQ
1C4A                    JZ      GET11           ;IF NOT COMPLETE
1C4C                    STOP    TCNT
1C4D                    MOV     A,R0
1C4E                    MOV     T,A             ;PUT LAST BYTE IN TIMER
1C4F                    SETREG  RSTA
1C53                    MOVX    A,XR0
1C54                    MOV     R0,A            ;STORE OPERATION STATUS IN R0

                ;       RECOGNIZE MFM, TRACK 0, SIDE 0, SECTOR 1-3

1C55                    IN      A,P1
1C56                    ANLI    A,SSO OR DDEN
1C58                    JNZ     GET12           ;IF FM OR SIDE 1
1C5A                    SETREG  RTRK
1C5E                    MOVX    A,XR0
1C5F                    JNZ     GET12           ;IF NOT TRACK 0
1C61                    SETREG  RSEC
1C65                    MOVX    A,XR0
1C66                    ADDI    A,$FC
1C68                    JC      GET12           ;IF NOT SECTOR 1, 2, OR 3

1C6A                    IN      A,P2
1C6B                    JB5     GET13           ;IF SS/SD OR SS/2D

1C6D                    MOV     A,R0
1C6E                    MOV     T,A             ;STORE STATUS IN TIMER
1C6F                    MOVI    R0,$7F
1C71            GET14
1C71                    MOV     A,R0
1C72                    ORLI    A,$80
1C74                    MOV     R0,A            ;POINT TO HIGH RAM
1C75                    MOV     A,XR0           ;FETCH A BYTE FROM HIGH RAM
1C76                    XCH     A,R0
1C77                    ANLI    A,$7F
1C79                    XCH     A,R0
1C7A                    MOV     XR0,A           ;WRITE IT TO LOW RAM
1C7B                    DJNZ    R0,GET14        ;IF MORE BYTES TO COPY
1C7D                    MOVI    R0,$80
1C7F                    MOV     A,XR0           ;NEW LAST BYTE
1C80                    MOV     R0,A
1C81                    MOV     A,T
1C82                    XCH     A,R0            ;PUT STATUS IN R0
1C83                    MOV     T,A             ;PUT LAST BYTE IN TIMER
1C84                    MOVI    A,$7F           ;TRANSMIT POINTER
1C86                    JUMP    GET15

1C88            GET13
```

```
1C88                    MOV     A,R0
1C89                    ANLI    A,$FB           ;TURN OFF LOST DATA BIT
1C8B                    MOV     R0,A

                ;       INITIALIZE R0 FOR TRANSMISSION

1C8C            GET12
1C8C                    IN      A,P2
1C8D                    RL      A
1C8E                    RL      A
1C8F                    ANLI    A,$80
1C91                    DECR    A
1C92            GET15
1C92                    XCH     A,R0            ;STATUS TO ACCUMULATOR

                ;       CHECK STATUS

1C93                    ANLI    A,$3C           ;MASK RELEVANT BITS
1C95                    JNZ     SRT             ;IF ERROR, GO DO SOFT RETRY

                ;       SUCCESSFUL COMPLETION OF GET

1C97            GET17
1C97                    CLR     F1              ;INDICATE SUCCESS
1C98                    LNGJMP  SEND            ;SEND DATA FRAME, GO TO IDLE

                ;       HANDLE GET TIMEOUTS

1C9B            GET4
1C9B                    JNDRQ   GET16
1C9D                    JDRQ    GET5            ;FILTER DRQ GLITCHES
1C9F            GET16
1C9F                    DJNZ    R2,GET6         ;IF TIME NOT EXHAUSTED
1CA1                    STOP    TCNT
1CA2                    SETREG  WCMD
1CA6                    MOVI    A,FRCINT
1CA8                    MOVX    XR0,A           ;DO FORCED INTERRUPT
1CA9                    PAUSE   R0,US200        ;DELAY 200 US
1CAD                    JUMP    HRT             ;DO HARD RETRY

1CAF            GET9
1CAF                    STOP    TCNT
1CB0                    SETREG  WCMD
1CB4                    MOVI    A,FRCINT
1CB6                    MOVX    XR0,A           ;DO FORCED INTERRUPT
1CB7                    PAUSE   R0,US200


                **      SRT - SOFT RETRY
                *
                *       ENTRY CONDITIONS:
                *               LAST DATA BYTE IN TIMER

1CBB            SRT
1CBB                    MOV     A,T
1CBC                    MOV     R0,A            ;COPY LAST DATA BYTE INTO R0
1CBD                    MOV     A,PSW
```

```
1CBE                    DECR    A
1CBF                    JB3     SEC18           ;IF RETRIES NOT EXHAUSTED


                ;**     HRT - HARD RETRY
                ;*
                ;*      ENTRY CONDITIONS:
                ;*              LAST DATA BYTE IN TIMER

1CC1            HRT
1CC1                    JF1     HRT1            ;IF RETRIES EXHAUSTED
1CC3                    CPL     F1              ;INDICATE RETRY IN PROGRESS
1CC4                    SETREG  RTRK
1CC8                    JUMP    RST             ;GO DO RESTORE SEQUENCE

1CCA            HRT1
1CCA                    MOVI    R0,CSTAT
1CCC                    MOVI    XR0,4           ;STATUS CODE FOR UNSUCCESSFUL OPERATION
1CCE                    MOVI    R0,0            ;NO DATA FRAME
1CD0                    IN      A,P2
1CD1                    ANLI    A,$FF-ACLR
1CD3                    XRLI    A,AGET
1CD5                    JNZ     HRT2            ;IF PUT
1CD7                    IN      A,P2
1CD8                    RL      A
1CD9                    RL      A
1CDA                    ANLI    A,$80
1CDC                    DECR    A
1CDD                    MOV     R0,A            ;DATA FRAME LENGTH
1CDE            HRT2
1CDE                    LNGJMP  SEND            ;SEND ERROR, GO TO IDLE


                ;**     HANDLE PUT COMMAND
                ;*
                ;*      ENTRY CONDITIONS:
                ;*              LAST DATA BYTE IN R0

                        ASSERT  *<=$1D00
1CE1  = 1D00            ORG     $1D00

1D00            PUT1

                ;       INITIALIZE FOR PUT

1D00                    IN      A,P2
1D01                    RL      A
1D02                    RL      A
1D03                    ANLI    A,$80
1D05                    DECR    A
1D06                    XCH     A,R0
1D07                    XCH     A,XR0
1D08                    CPL     A               ;COMPLEMENT FIRST BYTE
1D09                    MOV     R0,A

                ;       ISSUE WRITE SECTOR COMMAND
```

```
1D0A                    SETREG  WCMD
1D0E                    MOVI    A,WRSEC
1D10                    MOVX    XR0,A
1D11                    SETREG  WDAT

                ;       SEND FIRST DATA BYTE

1D15                    MOV     A,R0
1D16                    MOVI    R0,MS2300       ;TRIP COUNT FOR 2.3 SECOND TIMEOUT
1D18                    STRT    T

1D19            PUT2
1D19                    JTF     PUT3
1D1B            PUT4
1D1B                    JNDRQ   PUT2
1D1D                    JNDRQ   PUT2            ;FILTER DRQ GLITCHES
1D1F            PUT5
1D1F                    MOVX    XR0,A           ;WRITE FIRST BYTE
1D20                    CPL     A               ;UN-COMPLEMENT
1D21                    XCH     A,R0
1D22                    CLR     A
1D23                    MOV     T,A             ;RESET THE TIMER
1D24                    JTF     PUT6            ;CLEAR TIMER OVERFLOW FLAG
1D26            PUT6
1D26                    IN      A,P2
1D27                    RL      A
1D28                    RL      A
1D29                    ANLI    A,$80
1D2B                    DECR    A
1D2C                    XCH     A,R0
1D2D                    XCH     A,XR0           ;LAST BYTE TO ACCUMULATOR
1D2E                    DECR    R0

                ;       WRITE THE MIDDLE BYTES

1D2F            PUT7
1D2F                    XCH     A,XR0           ;GET A BYTE FROM RAM
1D30                    CPL     A               ;COMPLEMENT IT
1D31                    JDRQ    PUT13
1D33            PUT8
1D33                    JTF     PUT9            ;IF TIMED OUT
1D35                    JNDRQ   PUT8
1D37            PUT13
1D37                    MOVX    XR0,A           ;WRITE THE BYTE
1D38                    CPL     A               ;UN-COMPLEMENT IT
1D39                    XCH     A,XR0           ;PUT IT BACK INTO RAM
1D3A                    DJNZ    R0,PUT7         ;ADVANCE POINTER

                ;       WRITE THE LAST BYTE

1D3C                    MOV     R0,A            ;SAVE LAST BYTE IN R0
1D3D                    CPL     A               ;COMPLEMENT IT
1D3E                    JDRQ    PUT14
1D40            PUT10
1D40                    JTF     PUT11
1D42                    JNDRQ   PUT10
1D44            PUT14
```

```
1D44                    MOVX    XR0,A           ;WRITE THE LAST BYTE

                ;       WAIT FOR OPERATION COMPLETE

1D45            PUT12
1D45                    JTF     PUT11
1D47                    IN      A,P1
1D48                    ANLI    A,INTRQ
1D4A                    JZ      PUT12           ;IF NOT COMPLETE
1D4C                    STOP    TCNT
1D4D                    MOV     A,R0
1D4E                    MOV     T,A             ;PUT LAST BYTE IN TIMER
1D4F                    SETREG  RSTA
1D53                    MOVX    A,XR0           ;READ STATUS REG
1D54                    JB6     PUT18           ;IF WRITE PROTECT
1D56                    ANLI    A,$14           ;RELEVANT STATUS BITS
1D58                    JZ      PTV             ;IF SUCCESS, GO DO VERIFY
1D5A                    JUMP    SRT             ;FAILURE, GO DO SOFT RETRY

                ;       HANDLE PUT TIMEOUTS

1D5C            PUT3
1D5C                    JNDRQ   PUT16
1D5E                    JDRQ    PUT5
1D60            PUT16
1D60                    DJNZ    R0,PUT4         ;IF TIME NOT EXHAUSTED
1D62                    STOP    TCNT
1D63                    CPL     A               ;UN-COMPLEMENT
1D64                    MOV     R0,A
1D65                    IN      A,P2
1D66                    RL      A
1D67                    RL      A
1D68                    ANLI    A,$80
1D6A                    DECR    A
1D6B                    XCH     A,R0
1D6C                    XCH     A,XR0           ;RESTORE FIRST BYTE TO RAM
1D6D                    MOV     T,A             ;LAST BYTE TO TIMER
1D6E                    IN      A,P1
1D6F                    ANLI    A,INTRQ
1D71                    JZ      PUT17           ;IF OPERATION NOT COMPLETE
1D73                    SETREG  RSTA
1D77                    MOVX    A,XR0           ;READ STATUS REG
1D78                    JB6     PUT18           ;IF WRITE PROTECT
1D7A            PUT17
1D7A                    SETREG  WCMD
1D7E                    MOVI    A,FRCINT
1D80                    MOVX    XR0,A           ;DO FORCED INTERRUPT
1D81                    PAUSE   R0,US200
1D85                    JUMP    HRT             ;GO DO HARD RETRY

1D87            PUT9
1D87                    CPL     A
1D88                    XCH     A,XR0           ;PUT BYTE BACK IN RAM
1D89                    MOV     R0,A            ;SAVE LAST BYTE IN R0
1D8A            PUT11
1D8A                    STOP    TCNT
1D8B                    MOV     A,R0
```

```
1D8C                    MOV     T,A             ;PUT LAST BYTE IN TIMER
1D8D                    IN      A,P1
1D8E                    ANLI    A,INTRQ
1D90                    JZ      PUT19           ;IF OPERATION NOT COMPLETE
1D92                    SETREG  RSTA
1D96                    MOVX    A,XR0           ;READ STATUS
1D97                    JB6     PUT18           ;IF WRITE PROTECT
1D99            PUT19
1D99                    SETREG  WCMD
1D9D                    MOVI    A,FRCINT
1D9F                    MOVX    XR0,A           ;DO FORCED INTERRUPT
1DA0                    PAUSE   R0,US200        ;DELAY 200 US
1DA4                    JUMP    SRT             ;GO DO SOFT RETRY

1DA6            PUT18
1DA6                    MOVI    R0,CSTAT
1DA8                    MOVI    XR0,$0C         ;STATUS CODE FOR WRITE PROTECT
1DAA                    MOVI    R0,0            ;NO DATA FRAME
1DAC                    CLR     F1
1DAD                    CPL     F1              ;INDICATE ERROR
1DAE                    LNGJMP  SEND            ;SEND ERROR, GO TO IDLE

                ;**      HANDLE VERIFICATION
                ;*
                ;*      ENTRY CONDITIONS:
                ;*              LAST DATA BYTE IN TIMER

1DB1            PTV
1DB1                    JUMP    PTV1
                        ASSERT  *<=$1E00
1DB3  = 1E00            ORG     $1E00

1E00            PTV1
1E00                    IN      A,P2
1E01                    ANLI    A,$FF-ACLR
1E03                    XRLI    A,APUTV
1E05                    JNZ     PTV10           ;IF VERIFY NOT REQUIRED

                ;       SHIFT AND COMPLEMENT INTERNAL RAM

1E07                    IN      A,P2
1E08                    RL      A
1E09                    RL      A
1E0A                    ANLI    A,$80
1E0C                    MOV     R0,A            ;INDEX TO FIRST DATA BYTE
1E0D                    DECR    R0
1E0E                    RLC     A               ;SET CARRY FOR LATER USE
1E0F                    MOV     A,T             ;LAST DATA BYTE TO ACCUMULATOR
1E10                    CPL     A               ;COMPLEMENT IT
1E11                    XCH     A,XR0           ;FIRST DATA BYTE TO ACCUMULATOR
1E12                    CPL     A               ;COMPLEMENT IT
1E13                    DECR    R0
1E14            PTV11
1E14                    XCH     A,XR0           ;BRING A DATA BYTE TO ACCUMULATOR
1E15                    CPL     A               ;COMPLEMENT IT
1E16                    INCR    R0
1E17                    XCH     A,XR0           ;SHIFT IT FORWARD ONE POSITION
```

```
1E18                    DECR    R0
1E19                    XCH     A,XR0           ;RESTORE FIRST DATA BYTE TO ACCUMULATOR
1E1A                    DJNZ    R0,PTV11        ;IF MORE BYTES TO DO
1E1C                    MOV     R0,A            ;PUT FIRST DATA BYTE IN R0

                ;       ISSUE READ SECTOR COMMAND

1E1D                    STOP    TCNT
1E1E                    JTF     PTV12           ;CLEAR TIMER OVERFLOW FLAG
1E20            PTV12
1E20                    CLR     A
1E21                    MOV     T,A             ;RESET THE TIMER
1E22                    SETREG  WCMD
1E26                    MOVI    A,RDSEC
1E28                    MOVX    XR0,A
1E29                    SETREG  RDAT
1E2D                    STRT    T
1E2E                    MOVI    A,SEC1          ;TRIP COUNT FOR 1 SECOND TIMEOUT

                ;       READ THE FIRST TWO BYTES

1E30            PTV2
1E30                    JTF     PTV3
1E32            PTV4
1E32                    JNDRQ   PTV2
1E34                    JNDRQ   PTV2            ;FILTER DRQ GLITCHES
1E36            PTV13
1E36                    MOVX    A,XR0           ;READ FIRST BYTE
1E37                    XRL     A,R0            ;COMPARE
1E38                    JNZ     PTV5            ;IF INCORRECT
1E3A                    MOV     R0,A            ;STORE ZERO IN R0

1E3B                    JDRQ    PTV6
1E3D            PTV7
1E3D                    INCR    A
1E3E                    JDRQ    PTV6
1E40                    JZ      PTV5            ;IF TIMED OUT
1E42                    JNDRQ   PTV7
1E44            PTV6
1E44                    MOVX    A,XR0           ;READ SECOND BYTE
1E45                    XCH     A,R0
1E46                    RRC     A
1E47                    DECR    A               ;INDEX TO SECOND BYTE
1E48                    XCH     A,R0
1E49                    XRL     A,XR0           ;COMPARE
1E4A                    JNZ     PTV5            ;IF INCORRECT
1E4C                    DECR    R0

                ;       READ AND COMPARE THE REMAINING BYTES

1E4D            PTV14
1E4D                    JDRQ    PTV15
1E4F            PTV16
1E4F                    INCR    A
1E50                    JDRQ    PTV15
1E52                    JZ      PTV5            ;IF TIMED OUT
1E54                    JNDRQ   PTV16
```

```
1E56              PTV15
1E56                      MOVX    A,XR0           ;READ BYTE
1E57                      XRL     A,XR0           ;COMPARE
1E58                      JNZ     PTV5            ;IF INCORRECT
1E5A                      DJNZ    R0,PTV14        ;IF MORE BYTES TO DO

                  ;       WAIT FOR OPERATION COMPLETE

1E5C                      CLR     A
1E5D                      MOV     T,A             ;RESET TIMER
1E5E                      JTF     PTV17           ;CLEAR TIMER OVERFLOW FLAG
1E60              PTV17
1E60                      JTF     PTV5            ;IF TIMED OUT
1E62                      IN      A,P1
1E63                      ANLI    A,INTRQ
1E65                      JZ      PTV17           ;IF NOT COMPLETE
1E67                      STOP    TCNT
1E68                      SETREG  RSTA
1E6C                      MOVX    A,XR0           ;READ STATUS
1E6D                      ANLI    A,$3C           ;SELECT RELEVANT STATUS BITS
1E6F                      JNZ     PTV5            ;IF ERROR

                  ;       COME HERE FOR SUCCESSFUL VERIFY

1E71              PTV10
1E71                      CLR     F1              ;INDICATE NO ERROR
1E72                      MOVI    R0,0            ;NO DATA FRAME TO TRANSMIT
1E74                      LNGJMP  SEND            ;SEND COMPLETE, GO TO IDLE

                  ;       HANDLE VERIFY TIMEOUTS AND ERRORS

1E77              PTV3
1E77                      JNDRQ   PTV18
1E79                      JDRQ    PTV13
1E7B              PTV18
1E7B                      DECR    A
1E7C                      JNDRQ   PTV19
1E7E                      JDRQ    PTV13
1E80              PTV19
1E80                      JNZ     PTV4            ;IF 1 SECOND TIME NOT EXHAUSTED

1E82              PTV5
1E82                      STOP    TCNT
1E83                      MOVI    A,FRCINT
1E85                      SETREG  WCMD
1E89                      MOVX    XR0,A           ;DO FORCED INTERRUPT
1E8A                      PAUSE   R0,US200
1E8E                      CLR     F1
1E8F                      CPL     F1              ;SO HARD RETRY WILL FAIL
1E90                      JUMP    HRT             ;SEND ERROR, GO TO IDLE

                          ASSERT  *<=$1F00


no ERRORs,  478 Labels, $6958 free.
```