```
      TITLE 'PERSEPHONE MINI/MICROFLOPPY DISK CONTROLLER'
      SUBTTL     'DOCUMENTATION'

      LIST  -I,-M,-G,R

**    FUNCTION
*
*     This program is the controller for the Atari 1050 CR (cost
*     reduced) minifloppy disk drive and for a possible future
*     microfloppy disk drive.  It resides in the ROM of an
*     Intel 8050 microcontroller.  It's 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 09/11/84


**    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 minifloppy 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 microfloppies, 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 minifloppy,
*     THEN set SS/SD configuration;
*     ELSE IF bytes 4-8 = 1A, 00, 04, 00, and 80 respectively,
*          AND the drive is a minifloppy,
*     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
```

```
      SUBTTL     'EQUATES'


**    TIMING EQUATES

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

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

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

SEC1  EQU   10000/TRIP  ;TRIP COUNT FOR 1 SECOND DELAY
SEC3  EQU   30000/TRIP  ;TRIP COUNT FOR 3 SECOND DELAY
MS240 EQU   [2400/TRIP]+1    ;TRIP COUNT FOR 240 MS DELAY


**    WD1770 REGISTER SELECT EQUATES

WDCLR EQU   $DC         ;MASK TO CLEAR REGISTER SELECTION

WDRSTA      EQU   $20         ;READ STATUS REGISTER
WDWCMD      EQU   $00         ;WRITE COMMAND REGISTER
WDRTRK      EQU   $21         ;READ TRACK REGISTER
WDWTRK      EQU   $01         ;WRITE TRACK REGISTER
WDRSEC      EQU   $22         ;READ SECTOR REGISTER
WDWSEC      EQU   $02         ;WRITE SECTOR REGISTER
WDRDAT      EQU   $23         ;READ DATA REGISTER
WDWDAT      EQU   $03         ;WRITE DATA REGISTER


**    WD1770 COMMANDS

SRATE EQU   $02         ;STEP RATE CODE

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

IMPSEC      EQU   $F7         ;'IMPOSSIBLE' SECTOR NUMBER


**    CONTROL LINES FOR PORT 1

INTRQ EQU   $04         ;1770 INTERRUPT REQUEST
```

```
DDEN  EQU   $08         ;1770 DOUBLE DENSITY ENABLE
MR    EQU   $10         ;1770 MASTER RESET
SSO   EQU   $40         ;SIDE SELECT OUTPUT
SID   EQU   $80         ;SERIAL INPUT DATA LINE
INP1  EQU   $04         ;INPUT LINES ON PORT 1


**    CONTROL LINES FOR PORT 2

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


**    COMMUNICATION VALUES

ACK   EQU   $41         ;ACKNOWLEDGE
COMPLT      EQU   $43         ;SUCCESSFUL COMPLETION
ERROR EQU   $45         ;COMPLETION WITH ERROR
NAK   EQU   $4E         ;NOT ACKNOWLEDGE
HSACK EQU   $48         ;FAST MODE ACKNOWLEDGE


**    ACTIVITY CODES

ACLR  EQU   $F8         ;MASK TO CLEAR ACTIVITY CODE

AIDLE EQU   $00         ;IDLE
AGET  EQU   $01         ;GET SECTOR
APUT  EQU   $02         ;PUT SECTOR
APUTV EQU   $03         ;PUT SECTOR WITH VERIFY
APOPT EQU   $04         ;PUT OPTION TABLE
AMISC EQU   $05         ;MISCELLANEOUS DISK OPERATION
AOTHER      EQU   $07         ;OTHER

      ASSERT      AIDLE=0           ;LEGAL ASSUMPTION


**    HEAD POSITION KNOWN BIT

HKNOWN      EQU   $08         ;HEAD POSITION KNOWN


**    CONFIGURATION CODES

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

CFSSSD      EQU   $20         ;SS/SD
CFSS2D      EQU   $30         ;SS/2D
CFSSDD      EQU   $00         ;SS/DD
```

```
CFDSDD      EQU   $10         ;DS/DD


**    INTERNAL RAM ADDRESSES

BSL256      EQU   $FF         ;BAD SECTOR LIST FOR 256-BYTE SECTORS
CSTAT EQU   BSL256-4    ;CONTROLLER STATUS
HSTAT EQU   CSTAT-1           ;HARDWARE STATUS
CDEVIC      EQU   HSTAT-1           ;COMMAND FRAME DEVICE NUMBER
CCOMND      EQU   CDEVIC-1    ;COMMAND FRAME COMMAND
CAUX1 EQU   CCOMND-1    ;COMMAND FRAME AUXILIARY #1
CAUX2 EQU   CAUX1-1           ;COMMAND FRAME AUXILIARY #2
R0SAV EQU   CAUX2-1           ;SAVE AREA FOR R0
R1SAV EQU   R0SAV-1           ;SAVE AREA FOR R1
R2SAV EQU   R1SAV-1           ;SAVE AREA FOR R2
R3SAV EQU   R2SAV-1           ;SAVE AREA FOR R3
R4SAV EQU   R3SAV-1           ;SAVE AREA FOR R4
R5SAV EQU   R4SAV-1           ;SAVE AREA FOR R5

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

```
      SUBTTL      'MACROS'


**    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
```

```
      SUBTTL     'COMMAND FRAME PROCESSOR'

      ORG   $1000

**    RESET COMES HERE

      LNGJMP     RESET      ;GO DO RESET ROUTINE


**    EXTERNAL INTERRUPT COMES HERE

      ASSERT     *=$1003

      SEL   RB0
      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.

      SEL   MB0
      DIS   TCNTI
      STOP  TCNT
      CALL  ENABLE           ;RESET INTERRUPT-IN-PROGRESS FLIP-FLOP

;     JUMP  INVDF       ;INDICATE INVALID DATA FRAME


**    INVDF - PROCESS INVALID DATA FRAME

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

;     JUMP  IDLE        ;GO TO IDLE STATE


**    IDLE - ENTER IDLE STATE

IDLE
      MOVI  R7,SEC3          ;3 SECOND TRIP COUNT
      STRT  T
      ANLI  P2,ACLR          ;INDICATE IDLE STATE
      EN    I
IDLE1
      JTF   IDLE2       ;IF TIMER TRIPPED
      JUMP  IDLE1       ;WAIT FOR TRIP
IDLE2
      DJNZ  R7,IDLE1    ;IF 3 SECONDS NOT ELAPSED
IDLE4
      SETREG     RSTA
```

```
      MOVX  A,XR0       ;READ THE STATUS REGISTER
      CPL   A
      JB7   IDLE5       ;IF MOTOR OFF

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

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

;     ENTRY POINT FOR RETURN FROM INTERRUPT

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


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

ENABLE
      RETR


**    COMPRO - COMMAND FRAME PROCESSOR

COMPRO

;     GET OFF THE SERIAL BUS

      SETSID

;     RESET THE INTERRUPT-IN-PROGRESS FLIP-FLOP

      DIS   TCNTI
      DIS   I
      CALL  ENABLE

;     CHECK TO SEE IF WE WERE IN MID-OPERATION

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

;     CONSTRUCT PSEUDO-STATUS IN CSTAT

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

```
;     SET UP MOTOR TIMEOUT

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

;     INDICATE IDLE STATE

      ANLI  P2,ACLR

;     INITIATE A FORCED INTERRUPT TO THE 1770 IF NECESSARY

      XRLI  A,APOPT
      JZ    COM1       ;IF PUT OPTION TABLE
      XRLI  A,APOPT XOR AOTHER
      JZ    COM1       ;IF OTHER NON-DISK OPERATION
      IN    A,P1
      ANLI  A,MR
      JZ    COM1       ;IF RESET IS LOW

      SETREG     WCMD
      MOVI  A,FRCINT
      MOVX  XR0,A

;     INITIALIZE FOR COMMAND FRAME RECEPTION

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

;     CHECK COMMAND LINE

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

;     STORE BYTE OF COMMAND FRAME INTO RAM AND UPDATE CHECKSUM

COM2
      MOV   XR0,A
      DECR  R0
      ADD   A,R1
      ADDCI A,0
      MOV   R1,A
      PAUSE R3,12
      JNSOD COM12       ;IF STOP BIT IS BAD

;     WAIT FOR START BIT

COM5
      JSOD  COM3        ;IF START BIT NOT DETECTED
```

```
;     INITIALIZE FOR DATA BYTE RECEPTION

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

;     LOOP TO READ IN BYTE BIT-BY-BIT

COM6
      PAUSE R3,19       ;DELAY 19 CPU CYCLES
      CLR   C
      JNSOD COM7        ;IF BIT IS A ZERO
      CPL   C
      JUMP  COM8
COM7
      PAUSE R3,3        ;EQUALIZE TIME OF THE TWO PATHS
COM8
      RRC   A           ;SHIFT IN THE BIT
      JNC   COM6        ;IF NOT DONE WITH 8 BITS

;     CHECK CHECKSUM

      DJNZ  R2,COM2           ;IF CHECKSUM NOT RECEIVED YET
      XRL   A,R1
      JNZ   COM12       ;IF CHECKSUM DOESN'T MATCH
      PAUSE R3,15
      JNSOD COM12       ;IF STOP BIT IS WRONG
      JNI   COM11       ;IF COMMAND LINE IS STILL LOW
      JUMP  INTRET

;     CHECK TO SEE IF COMMAND IS FOR US

COM11
      MOVI  R1,$31            ;DEVICE NUMBER FOR UNIT #1
      IN    A,P2        ;READ DEVICE SELECT SWITCHES
      ANLI  A,UNITMK    ;SELECT UNIT # BITS
      XRLI  A,UNIT1
      JZ    COM10       ;IF UNIT 1
      INCR  R1
      XRLI  A,UNIT2 XOR UNIT1
      JZ    COM10       ;IF UNIT 2
      INCR  R1
      XRLI  A,UNIT3 XOR UNIT2
      JZ    COM10       ;IF UNIT 3
      INCR  R1          ;ELSE, MUST BE UNIT 4
COM10
      MOVI  R0,CDEVIC
      MOV   A,XR0       ;BRING IN DEVICE NUMBER
      XRL   A,R1
      JZ    COM13       ;IF COMMAND IS FOR US
COM12
      JNI   COM12       ;WAIT FOR COMMAND TO GO HIGH
      JUMP  INTRET

;     BRANCH TO VARIOUS COMMAND ROUTINES
```

```
COM13
      JUMP  COMLK
      ASSERT      *<=$1100
      ORG   $1100
COMLK

      DECR  R0
      MOV   A,XR0       ;BRING IN COMMAND
      XRLI  A,$52
      JZ    COM28
      XRLI  A,$72 XOR $52
      JNZ   COM20
COM28
      ORLI  P2,AGET           ;SET ACTIVITY CODE FOR GET
      LNGJMP      GET         ;GET SECTOR
COM20
      XRLI  A,$50 XOR $72
      JNZ   COM21
      ORLI  P2,APUT           ;SET ACTIVITY CODE FOR PUT
      LNGJMP      PUT         ;PUT SECTOR
COM21
      XRLI  A,$57 XOR $50
      JNZ   COM22
      ORLI  P2,APUTV    ;SET ACTIVITY CODE FOR PUT/VERIFY
      LNGJMP      PUTV        ;PUT/VERIFY SECTOR
COM22
      XRLI  A,$53 XOR $57
      JNZ   COM23
      ORLI  P2,AOTHER   ;ACTIVITY CODE FOR 'OTHER'
      LNGJMP      STAT        ;STATUS
COM23
      XRLI  A,$21 XOR $53
      JNZ   COM24
      ORLI  P2,AMISC    ;ACTIVITY CODE FOR MISCELLANEOUS
      LNGJMP      FMT         ;FORMAT
COM24
      XRLI  A,$4E XOR $21
      JNZ   COM25
      ORLI  P2,AOTHER   ;ACTIVITY CODE FOR 'OTHER'
      LNGJMP      GOPT        ;GET OPTION TABLE
COM25
      XRLI  A,$4F XOR $4E
      JNZ   COM26
      ORLI  P2,APOPT    ;ACTIVITY CODE FOR PUT OPTION TABLE
      LNGJMP      POPT        ;PUT OPTION TABLE
COM26
      XRLI  A,$48 XOR $4F
      JNZ   COM27
      ORLI  P2,AOTHER   ;ACTIVITY CODE FOR 'OTHER'
      LNGJMP      SPEED       ;SET SIO SPEED
COM27
      XRLI  A,$22 XOR $48
      JNZ   INVCF
```

```
      IN    A,P2
      ANLI  A,SECTSZ
      JZ    INVCF       ;IF SECTOR SIZE IS 256
      ORLI  P2,AMISC    ;ACTIVITY CODE FOR MISCELLANEOUS
      LNGJMP      FMTD        ;FORMAT DUAL DENSITY


**    INVCF - PROCESS INVALID COMMAND FRAME

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

      MOVI  A,NAK
      LNGCAL      XBYTE       ;SEND NAK
      JUMP  INTRET            ;RETURN
```

```
      SUBTTL     'POWER-ON INTITIALIZATION'

**    RESET - POWER ON INITIALIZATION

RESET
      DIS   I
      DIS   TCNTI
      MOVI  A,INP1+WDRSTA+DDEN
      OUTL  P1,A
      MOVI  A,INP2+CFSSSD+AIDLE
      OUTL  P2,A
      SEL   RB0
      CLR   F0          ;INIT TO SLOW MODE
      CLR   F1

      STOP  TCNT
      CLR   A
      MOV   T,A
      MOVI  R0,MS240
      STRT  T
      JTF   RESET1           ;CLEAR TIMER OVERFLOW FLAG
RESET1
      JTF   RESET2
      JUMP  RESET1
RESET2
      DJNZ  R0,RESET1   ;IF NOT FINISHED WITH 1/4 SECOND DELAY
      STOP  TCNT

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

```
      SUBTTL     'SERIAL ACK/NAK TRANSMITTER'


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

SENDAK
      STOP  TCNT
      CLR   A
      MOV   T,A
      JTF   SNDAK1            ;CLEAR TIMER OVERFLOW FLAG
SNDAK1
      MOVI  A,HSACK           ;HIGH SPEED ACK
      JF0   SNDAK2            ;IF HIGH SPEED ENABLED
      MOVI  A,ACK       ;NORMAL ACK
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

XBYTE
      CLR   F1          ;CLEAR RETRY FLAG

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

      CLRSID                  ;SEND START BIT
      CLR   C
      CPL   C
      PAUSE R6,3        ;DELAY 3 CPU CYCLES
XBYTE1
      PAUSE R6,18       ;DELAY 18 CPU CYCLES
      RRC   A           ;SHIFT OUT NEXT BIT
      JC    XBYTE2            ;IF BIT IS A ONE
      CLRSID                  ;SEND A ZERO
      JUMP  XBYTE3
XBYTE2
      SETSID                  ;SEND A ONE
      JUMP  XBYTE3            ;EQUALIZE THE TIME AROUND BOTH PATHS
XBYTE3
      CLR   C
      JNZ   XBYTE1            ;IF NOT DONE WITH 8 BITS
      RET
```

```
      SUBTTL     'SERIAL DATA FRAME RECEIVER'


**    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
      ORG   $1200

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

;     READ IN THE FIRST [R0] BYTES

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

      NOPP
      NOPP
      NOPP
      NOPP
      JF0   RECV6       ;DELAY 6 CPU CYCLES IN FAST MODE
      NOPP
      NOPP              ;DELAY 8 CYCLES IN SLOW MODE

RECV2
      NOPP
```

```
      NOPP
      NOPP
      JF0   RECV6       ;DELAY 5 CPU CYCLES IN FAST MODE
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP              ;DELAY 19 CYCLES IN SLOW MODE

RECV6
      CLR   C
      JNSOD RECV3       ;IF BIT IS A ZERO
      CPL   C
      JUMP  RECV4
RECV3
      NOPP
      NOPP
      NOPP              ;EQUALIZE TIME AROUND BOTH PATHS
RECV4
      RRC   A           ;SHIFT IN THE BIT
      JNC   RECV2       ;IF NOT DONE WITH 8 BITS
      JF0   RECV7       ;DELAY 2 CYCLES IN FAST MODE
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP              ;DELAY 16 CYCLES IN SLOW MODE

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

;     BRING IN THE LAST DATA BYTE

RECV8
```

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

;     COMPARE CALCULATED CHECKSUM TO RECEIVED CHECKSUM

RECV13
      CLR   F1          ;CLEAR RETRY FLAG
      NOPP              ;DELAY 1 CPU CYCLE
      JNSOD RECV19            ;IF STOP BIT IS BAD
RECV14
      JSOD  RECV14            ;WAIT FOR START BIT
      STOP  TCNT
      DIS   TCNTI
      MOV   T,A         ;SAVE LAST BYTE IN TIMER
      ADD   A,R0
      ADDCI A,0         ;FINISH CHECKSUM CALCULATION
      CLR   C
      CPL   C

      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      JF0   RECV18            ;DELAY 7 CYCLES IN FAST MODE
      NOPP              ;DELAY 8 CYCLES IN SLOW MODE

RECV15
      NOPP
      NOPP
      NOPP
      NOPP
      JF0   RECV18            ;DELAY 6 CPU CYCLES IN FAST MODE
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP              ;DELAY 20 CYCLES IN SLOW MODE

RECV18
```

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

;     ACKNOWLEDGE THE DATA FRAME

      PAUSE R0,US500
      PAUSE R0,US500    ;DELAY 1000 US

      CLRSID                  ;SEND START BIT
      MOVI  A,ACK       ;NORMAL ACK
      CLR   C
      CPL   C
      PAUSE R0,1
RECV23
      PAUSE R0,2
      JF0   RECV26            ;IF FAST MODE
      PAUSE R0,14
RECV26
      RRC   A           ;SHIFT OUT NEXT BIT
      JC    RECV24            ;IF BIT IS A ONE
      CLRSID                  ;SEND OUT A ZERO
      JUMP  RECV25
RECV24
      SETSID                  ;SEND OUT A ONE
      JUMP  RECV25            ;EQUALIZE TIME AROUND BOTH PATHS
RECV25
      CLR   C
      JNZ   RECV23            ;IF DONE WITH 8 BITS

;     RETURN TO CALLER

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

```
      SUBTTL     'SERIAL DATA FRAME TRANSMITTER'


**    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
      ORG   $1300

SEND

;     SEND COMPLT/ERROR

      MOV   A,PSW
      ORLI  A,1         ;INDICATE NO NEED TO UPDATE CSTAT
      MOV   PSW,A
      MOVI  A,ERROR
      JF1   SEND0       ;IF ERROR
      MOV   A,PSW
      ANLI  A,$FE
      MOV   PSW,A       ;INDICATE CSTAT UPDATE DESIRED
      MOVI  A,COMPLT
SEND0

;     LOOP TO SEND ONE BYTE

SEND1
      CLRSID                ;SEND START BIT
      CLR   C
      CPL   C
      NOPP
      NOPP
      NOPP              ;DELAY 3 CYCLES
SEND2
      NOPP
      NOPP
      JF0   SEND3       ;DELAY 4 CYCLES IN FAST MODE
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
```

```
      NOPP
      NOPP
      NOPP
      NOPP              ;DELAY 18 CYCLES IN SLOW MODE
SEND3
      RRC   A           ;SHIFT OUT NEXT BIT
      JC    SEND4       ;IF BIT IS A ONE
      CLRSID                ;SEND OUT A ZERO
      JUMP  SEND5
SEND4
      SETSID                ;SEND OUT A ONE
      JUMP  SEND5       ;EQUALIZE TIME AROUND BOTH PATHS
SEND5
      CLR   C
      JNZ   SEND2       ;IF NOT DONE WITH 8 BITS

;     CHECK TO SEE IF DATA FOLLOWS

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

;     SEND THE DATA FRAME

SEND11
      CLRSID                ;SEND START BIT
      ADD   A,XR0
      ADDCI A,0         ;UPDATE CHECKSUM
      XCH   A,XR0       ;SAVE CHECKSUM, BRING DATA BYTE TO A
      CLR   C
      CPL   C
      JF0   SEND16          ;IF FAST MODE
      JUMP  SEND10
SEND12
      NOPP
      NOPP
      JF0   SEND13          ;DELAY 4 CYCLES IN FAST MODE
      NOPP
SEND10
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
```

```
SEND16
      NOPP              ;DELAY 18 CYCLES IN SLOW MODE
SEND13
      RRC   A           ;SHIFT OUT NEXT BIT
      JC    SEND14            ;IF BIT IS A ONE
      CLRSID                  ;SEND OUT A ZERO
      JUMP  SEND15
SEND14
      SETSID                  ;SEND OUT A ONE
      JUMP  SEND15            ;EQUALIZE TIME AROUND BOTH PATHS
SEND15
      CLR   C
      JNZ   SEND12            ;IF NOT DONE WITH 8 BITS

;     GET NEXT BYTE TO SEND

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

SEND6
      INCR  R0
      MOV   A,XR0       ;BRING CHECKSUM TO A
      DECR  R0
      JF0   SEND11            ;IF FAST MODE
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
SEND18
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP              ;DELAY 12 CYCLES
      JUMP  SEND11            ;GO SEND THE BYTE

SEND17
      MOV   A,R1        ;MOVE CHECKSUM TO A
      JF0   SEND1       ;IF FAST MODE
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
```

```
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP
      NOPP              ;DELAY 12 CYCLES
      JUMP  SEND1       ;GO SEND THE CHECKSUM

;     UPDATE CSTAT IF NECESSARY, GO TO IDLE

SEND7
      MOV   A,PSW
      JB0   SEND9       ;IF NO NEED TO UPDATE CSTAT
      MOVI  R0,CSTAT
      MOVI  XR0,0       ;UPDATE STATUS TO SUCCESS
SEND9
      LNGJMP      IDLE        ;GO TO IDLE STATE
```

```
      SUBTTL     'STATUS'


**    STAT - GET STATUS

STAT
      LNGCAL     SENDAK           ;SEND ACKNOWLEDGE
      PAUSE R0,US500
      PAUSE R0,US500    ;1000 US COMPLT DELAY
      MOVI  R3,0        ;FOR SS/SD
      IN    A,P1
      ANLI  A,DDEN
      JNZ   STAT5       ;IF FM
      MOVI  R3,$80           ;FOR SS/2D
      IN    A,P2
      JB5   STAT5       ;IF 128 BYTES/SECTOR
      MOVI  R3,$A0           ;FOR DS/DD
      JB4   STAT5       ;IF DS/DD
      MOVI  R3,$20
STAT5
      MOVI  R0,CSTAT
      MOV   A,XR0       ;GET CONTROLLER STATUS
      ORL   A,R3
      MOV   R3,A

      MOVI  R2,$FF           ;HARDWARE STATUS FOR NO ERROR
      IN    A,P1
      ANLI  A,MR
      JZ    STAT1       ;IF RESET IS LOW
      SETREG     RSTA
      MOVX  A,XR0       ;READ 1770 STATUS REGISTER
      CPL   A           ;INVERT IT
      MOV   R2,A        ;SAVE IT
      JB7   STAT1       ;IF MOTOR OFF
      MOV   A,R3
      ORLI  A,$10       ;TURN ON MOTOR BIT
      MOV   R3,A

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

```
      SUBTTL     'SET SIO SPEED'


**    SPEED - SET SIO SPEED

      ASSERT     *<=$1400
      ORG   $1400

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

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

      CLR   F0
      CPL   F0
      JUMP  SPEED3

;     SET SLOW SPEED

SPEED1
      CLR   F0
SPEED3
      LNGCAL     SENDAK           ;ACKNOWLEDGE
      PAUSE R0,US500
      PAUSE R0,US500    ;DELAY, LEAVING R0 = 0
      CLR   F1          ;INDICATE NO ERROR
      LNGJMP     SEND        ;SEND COMPLT, GO TO IDLE

;     PROCESS INVALID CODE

SPEED2
      LNGJMP     INVCF       ;INVALID COMMAND FRAME
```

```
      SUBTTL     'GET OPTION TABLE'


**    GOPT - GET OPTION TABLE

GOPT
      LNGCAL     SENDAK           ;ACKNOWLEDGE
      PAUSE R0,US500
      PAUSE R0,US500    ;GENERATE COMPLT DELAY
      MOVI  R0,11
      MOVI  XR0,40           ;NUMBER OF TRACKS
      DECR  R0
      MOVI  XR0,SRATE   ;STEP RATE
      DECR  R0
      MOVI  XR0,0       ;SECTORS PER TRACK (MSB)
      DECR  R0
      IN    A,P2
      ANLI  A,$FF-CFCLR ;GET CURRENT CONFIG
      XRLI  A,CFSS2D
      MOVI  XR0,26           ;26 SECTORS PER TRACK FOR SS/2D
      JZ    GOPT1       ;IF SS/2D
      MOVI  XR0,18           ;18 SECTORS PER TRACK FOR OTHERS
      IN    A,P2
      JB5   GOPT1       ;IF SS/SD
      DIS   I
      ORLI  P2,SECTSZ
      IN    A,P2
      JB5   GOPT2       ;IF DRIVE IS A MINIFLOPPY
      MOVI  XR0,16           ;16 SECTORS PER TRACK FOR MICROFLOPPY
GOPT2
      ANLI  P2,$FF-SECTSZ    ;FIX UP SECTOR SIZE BIT
      EN    I
GOPT1
      DECR  R0
      MOVI  XR0,0       ;SINGLE SIDED
      IN    A,P2
      ANLI  A,$FF-CFCLR ;GET CURRENT CONFIG
      XRLI  A,CFDSDD
      JNZ   GOPT3       ;IF NOT DS/DD
      MOVI  XR0,1       ;DOUBLE SIDED
GOPT3
      DECR  R0
      MOVI  XR0,0       ;FM
      IN    A,P2
      ANLI  A,$FF-CFCLR ;GET CURRENT CONFIG
      XRLI  A,CFSSSD
      JNZ   GOPT4       ;IF NOT SS/SD
      MOVI  XR0,4       ;MFM
GOPT4
      DECR  R0
      MOVI  XR0,0
      DECR  R0
      MOVI  XR0,$80          ;128 BYTES PER SECTOR
      IN    A,P2
```

```
      JB5   GOPT5       ;IF 128 BYTES PER SECTOR
      INCR  R0
      MOVI  XR0,1
      DECR  R0
      MOVI  XR0,0       ;256 BYTES PER SECTOR
GOPT5
      DECR  R0
      MOVI  XR0,1       ;DRIVE PRESENT
      DECR  R0
      MOVI  XR0,HSACK   ;FAST MODE
      JF0   GOPT6       ;IF FAST MODE
      MOVI  XR0,ACK           ;SLOW MODE
GOPT6
      CLR   A
      MOV   T,A
      MOV   R1,A
      MOVI  R0,11       ;FRAME SIZE
      CLR   F1          ;NO ERROR
      LNGJMP      SEND        ;SEND FRAME, GO TO IDLE
```

```
      SUBTTL     'PUT OPTION TABLE'


**    POPT - PUT OPTION TABLE

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

;     CHECK FOR SS/SD MODE

      MOVI  R2,CFSSSD OR CFCLR
      MOV   A,R6        ;RECORDING MODE
      JNZ   POPT1       ;IF NOT FM
      MOV   A,R7        ;NUMBER OF SIDES
      JNZ   POPTER            ;IF NOT SINGLE SIDED, ERROR
      IN    A,P2
      JB5   POPTSU            ;IF MINIFOLPPY, SUCCESS
      JUMP  POPTER            ;ELSE, ERROR

;     CHECK RECORDING MODE FOR MFM

POPT1
      XRLI  A,4
      JNZ   POPTER            ;NOT MFM, ERROR

;     CHECK FOR DS/DD

      MOVI  R2,CFDSDD OR CFCLR
      MOV   A,R7        ;NUMBER OF SIDES
      JZ    POPT2       ;IF SINGLE SIDE
      XRLI  A,1
      JNZ   POPTER            ;NOT DOUBLE SIDED, ERROR
      ORLI  P1,SSO            ;TRY TO SELECT SECOND SIDE
      IN    A,P1
      ANLI  P1,$FF-SSO  ;RE-SELECT FIRST SIDE
      ANLI  A,SSO
      JNZ   POPTSU            ;SECOND SIDE OK
      JUMP  POPTER            ;ELSE ERROR

;     CHECK FOR SS/2D

POPT2
      MOVI  R2,CFSS2D OR CFCLR
      MOV   A,XR0       ;# OF SECTORS PER TRACK
      XRLI  A,26
```

```
      JNZ   POPT3       ;IF NOT SS/2D
      MOV   A,R5        ;# BYTES PER SECTOR MSB
      JNZ   POPT3       ;IF NOT SS/2D
      MOV   A,R4        ;# BYTES PER SECTOR LSB
      XRLI  A,$80
      JNZ   POPT3       ;IF NOT SS/2D
      IN    A,P2
      JB5   POPTSU            ;IN DRIVE IS A MINIFLOPPY
      JUMP  POPTER            ;IF MICROFLOPPY, ERROR

;     COME HERE FOR SS/DD

POPT3
      MOVI  R2,CFSSDD OR CFCLR

;     COME HERE FOR SUCCESSFUL CONFIGURE

POPTSU
      MOV   A,R2        ;NEW CONFIGURATION
      MOV   R1,A
      CLR   F1          ;NO ERROR
      JUMP  POPT4

;     COME HERE IF ERROR

POPTER
      CLR   F1
      CPL   F1          ;ERROR FLAG
      MOVI  R0,CSTAT
      MOVI  XR0,$04           ;STATUS FOR UNSUCCESSFUL OPERATION

;     SEND COMPLT/ERROR AND GO TO IDLE

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

```
      SUBTTL     'FORMAT ROUTINES'

**    FMT - FORMAT IN CURRENT DENSITY

      ASSERT     *<=$1500
      ORG   $1500

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

**    FMTD - FORMAT IN DUAL DENSITY

FMTD
      ANLI  P1,$FF-DDEN ;SELECT MFM


;     SEND ACKNOWLEDGE

FMT1
      LNGCAL     SENDAK

;     DO RESET SEQUENCE

      ANLI  P2,$FF-HKNOWN    ;INDICATE HEAD POSITION UNKNOWN
      ANLI  P1,$FF-MR   ;SEND RESET TO 1770
      PAUSE R0,US500
      ORLI  P1,MR       ;REMOVE RESET
      PAUSE R0,US500
      SETREG     WCMD
      MOVI  A,FRCINT
      MOVX  XR0,A       ;INITIATE FORCED INTERRUPT
      PAUSE R0,US200

;     INITIALIZE DATA FRAMES FOR ERROR

      IN    A,P2
      RL    A
      RL    A
      ANLI  A,$80
      DECR  A           ;OFFSET TO FIRST BYTE OF DATA FRAME
      MOV   R3,A        ;SAVE IT
      MOV   R0,A
      CLR   A
      MOV   XR0,A
      DECR  R0
      MOV   XR0,A
      CPL   A
      DECR  R0
      MOV   XR0,A
      DECR  R0
```

```
      MOV   XR0,A
      MOVI  R0,CSTAT
      MOVI  XR0,$04          ;STATUS FOR UNSUCCESSFUL OPERATION

;     SET FLAG TO INDICATE FORMAT

      CLR   F1

;     RESTORE TO TRACK 0

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

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

FMT3
      JTF   FMT4
FMT5
      IN    A,P1
      ANLI  A,INTRQ
      JZ    FMT3        ;IF NOT DONE

      STOP  TCNT
      SETREG     RSTA
      MOVX  A,XR0       ;READ STATUS TO CLEAR INTRQ
      SETREG     WTRK
      CLR   A
      MOVX  XR0,A       ;UPDATE TRACK REG TO 0
      MOV   R2,A        ;INIT R2 = TRACK
      ANLI  P1,$FF-SSO  ;SELECT SIDE 0
      PAUSE R0,US500
      JUMP  FMT6        ;GO TO TRACK LOOP

FMT4
      DJNZ  R0,FMT5          ;IF NOT TIMED OUT

;     PROCESS ERROR

FMT7
      STOP  TCNT
      SETREG     WCMD
      MOVI  A,FRCINT
      MOVX  XR0,A       ;DO FORCED INTERRUPT
      PAUSE R0,US200
      CLR   F1
      CPL   F1          ;INDICATE ERROR
      MOV   A,R3
      MOV   R0,A        ;DATA FRAME LENGTH
```

```
      LNGJMP     SEND       ;SEND ERROR, GO TO IDLE

;     MAIN TRACK LOOP
;     SEEK DESIRED TRACK

FMT6
      SETREG     WDAT
      MOV   A,R2        ;GET DESIRED TRACK
      MOVX  XR0,A       ;WRITE IT TO DATA REG
      ANLI  P2,$FF-HKNOWN    ;INDICATE HEAD POSITION UNKNOWN
      SETREG     WCMD
      MOVI  A,SEEK
      MOVX  XR0,A       ;ISSUE SEEK COMMAND
      CLR   A
      MOV   T,A
      JTF   FMT8        ;CLEAR TIMER OVERFLOW FLAG
FMT8
      MOVI  R0,SEC1          ;TRIP COUNT FOR 1-SECOND TIMEOUT

FMT9
      JTF   FMT10
FMT11
      IN    A,P1
      ANLI  A,INTRQ
      JZ    FMT9        ;IF NOT DONE

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

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

;     CONSTRUCT SECTOR INTERLEAVE TABLE IN INTERNAL RAM

ILV

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

      MOVI  R4,18       ;18 SECTORS PER TRACK
      MOVI  R5,$100-9   ;SECTOR INTERLEAVE FACTOR OF 9
      IN    A,P2
      JB5   ILV1        ;IF 128 BYTES/SECTOR
      JF0   ILV2        ;IF FAST MODE
      MOVI  R5,$100-15  ;SECTOR INTERLEAVE OF 15
ILV2
      DIS   I
      ORLI  P2,SECTSZ
      IN    A,P2
      JB5   ILV3        ;IF MINIFLOPPY
```

```
      MOVI  R4,16       ;16 SECTORS/TRACK FOR MICROFLOPPY
ILV3
      ANLI  P2,$FF-SECTSZ
      EN    I
      JUMP  ILV4
ILV1
      IN    A,P1
      JB3   ILV4        ;IF FM
      MOVI  R4,26       ;26 SECTORS/TRACK
      MOVI  R5,$100-13  ;SECTOR INTERLEAVE OF 13

;     CLEAR THE INTERLEAVE TABLE

ILV4
      MOV   A,R4
      MOV   R0,A        ;INIT R0 = NUMBER OF SECTORS
      ADDI  A,$7F
      MOV   R6,A        ;SAVE INIT INDEX INTO TABLE
      MOV   R1,A
ILV5
      MOVI  XR1,0       ;STORE 0 IN TABLE
      DECR  R1          ;ADVANCE POINTER
      MOV   A,R1
      JB7   ILV5        ;IF NOT DONE WITH TABLE

;     FILL THE TABLE

      MOV   A,R6
      JUMP  ILV7
ILV6
      MOV   A,R1        ;TABLE INDEX
      ADD   A,R5        ;ADD INTERLEAVE FACTOR
ILV8
      JB7   ILV7        ;IF STILL WITHIN RANGE
      ADD   A,R4        ;WRAP POINTER
ILV7
      MOV   R1,A
      MOV   A,XR1
      JZ    ILV9        ;IF TABLE ENTRY NOT YET USED
      DECR  R1
      MOV   A,R1
      JUMP  ILV8        ;USE NEXT ENTRY
ILV9
      IN    A,P1
      ANLI  A,SSO
      JNZ   ILV10       ;IF SECOND SIDE
      MOV   A,R0
      CPL   A
      ADD   A,R4
      ADDI  A,2         ;REFLECT SECTOR NUMBER
      JUMP  ILV11
ILV10
      MOV   A,R0
ILV11
```

```
      MOV   XR1,A       ;STORE SECTOR NUMBER IN TABLE
      DJNZ  R0,ILV6           ;IF MORE SECTORS TO DO
      MOV   A,R6
      MOV   R1,A        ;R1 = INIT INDEX INTO TABLE

;     INITIALIZE FOR WRITE TRACK

      JUMP  WTR
      ASSERT      *<=$1600
      ORG   $1600

WTR
      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

      MOVI  R4,$00
      MOVI  R5,6
      MOVI  R6,11
      MOVI  R7,12
      MOVI  R0,39
      IN    A,P1
      JB3   WTR1        ;IF FM
      MOVI  R4,$4E
      MOVI  R5,12
      MOVI  R6,22
      MOVI  R7,24
      MOVI  R0,59

;     ISSUE WRITE TRACK COMMAND

WTR1
      SETREG      WCMD
      MOVI  A,WRTRK
      MOVX  XR0,A
      SETREG      WDAT
      CLR   A
      MOV   T,A         ;RESET THE TIMER
      JTF   WTR2        ;CLEAR TIMER OVERFLOW FLAG
WTR2
      MOVI  A,SEC1            ;TRIP COUNT FOR 1 SECOND TIMEOUT
      XCH   A,R4
      STRT  T
      JUMP  WTR5        ;GO WRITE THE TRACK

**    VERIFY TRACK

VFY

;     ISSUE READ SECTOR COMMAND
```

```
VFY1
      STOP  TCNT
      CLR   A
      MOV   T,A
      MOV   A,R3
      INCR  A
      MOV   R0,A        ;SECTOR SIZE
      MOVI  R4,SEC1           ;TRIP COUNT FOR 1 SECOND TIMEOUT
      SETREG     WTRK
      MOV   A,R2        ;TRACK NUMBER
      MOVX  XR0,A
      SETREG     WSEC
      MOV   A,XR1       ;SECTOR NUMBER FROM INTERLEAVE TABLE
      DECR  R1          ;ADVANCE INTERLEAVE POINTER
      MOVX  XR0,A
      SETREG     WCMD
      MOVI  A,RDSEC
      MOVX  XR0,A
      SETREG     RDAT
      STRT  T

;     READ THE DATA BYTES

VFY2
      JTF   VFY3
VFY4
      JNDRQ VFY2
      JNDRQ VFY2
VFY5
      MOVX  A,XR0
      DJNZ  R0,VFY4           ;IF MORE BYTE TO READ

;     WAIT FOR OPERATION COMPLETE

      CLR   A
      MOV   T,A         ;RESET THE TIMER
      JTF   VFY6        ;RESET TIMER OVERFLOW FLAG
VFY6
      JTF   VFY7        ;IF TIMED OUT
      IN    A,P1
      ANLI  A,INTRQ
      JZ    VFY6        ;IF NOT DONE

;     READ STATUS

      STOP  TCNT
      SETREG     RSTA
      MOVX  A,XR0       ;READ STATUS REG
      ANLI  A,$3C       ;RELEVANT STATUS BITS
      JNZ   VFY7        ;IF ERROR

;     SEE IF MORE SECTORS TO DO
```

```
      MOV   A,R1
      JB7   VFY1        ;IF MORE SECTORS
      JUMP  FMT12       ;GO DO NEXT TRACK

;     HANDLE VERIFY TIMEOUTS

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

**    GO TO NEXT TRACK

FMT12
      IN    A,P1
      JB6   FMT13       ;IF SECOND SIDE
      INCR  R2          ;NEXT TRACK
      MOV   A,R2
      XRLI  A,40
      JNZ   FMT14       ;IF NOT DONE WITH 40 TRACKS
      IN    A,P2
      ANLI  A,$FF-CFCLR
      XRLI  A,CFDSDD
      JNZ   FMT15       ;IF NOT DOUBLE SIDED
      ORLI  P1,SSO            ;SELECT SECOND SIDE
      PAUSE R0,US500

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

;     COME HERE IF COMPLETE PASS IS FINISHED

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

;     COME HERE FOR SUCCESSFUL FORMAT/VERIFY

FMT16
      STOP  TCNT
      CLR   F1          ;INDICATE SUCCESS
      MOV   A,R3
      MOV   R0,A        ;DATA FRAME LENGTH
      MOVI  XR0,$FF
      DECR  R0
      MOVI  XR0,$FF           ;INDICATE NO BAD SECTORS
```

```
      INCR  R0
      LNGJMP      SEND        ;SEND COMPLT AND FRAME, GO TO IDLE

**    WRITE TRACK

      ASSERT      *<=$1700
      ORG   $1700

;     WRITE POST-INDEX GAP (GAP I)

WTR3
      JTF   WTR4
WTR5
      JNDRQ WTR3
      JNDRQ WTR3        ;FILTER DRQ GLITCHES
WTR50
      MOVX  XR0,A       ;WRITE A GAP BYTE
      DJNZ  R0,WTR5           ;IF NOT DONE WITH GAP
      MOV   T,A         ;RESET THE TIMER
      JTF   WTR9        ;RESET TIMER OVERFLOW FLAG
WTR9
      JDRQ  WTR10
WTR11
      JTF   WTR12
      JNDRQ WTR11
WTR10
      MOVX  XR0,A       ;WRITE LAST BYTE OF GAP
      MOV   R4,A

;     SECTOR LOOP

WTR6

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

      MOV   A,R5
      MOV   R0,A
      CLR   A
      JDRQ  WTR44
WTR8
      JTF   WTR12
WTR13
      JNDRQ WTR8
WTR44
      MOVX  XR0,A
      DJNZ  R0,WTR13

;     WRITE MFM SYNC MARKS

      IN    A,P1
      JB3   WTR14       ;IF FM
      MOVI  A,$F5       ;F5 WRITES MFM SYNC MARK
      JDRQ  WTR15
WTR16
```

```
      JTF   WTR12
      JNDRQ WTR16
WTR15
      MOVX  XR0,A
      MOVI  R0,2        ;WRITE 2 MORE SYNC MARKS
WTR17
      JTF   WTR12
WTR18
      JNDRQ WTR17
      MOVX  XR0,A
      DJNZ  R0,WTR18

;     WRITE ID FIELD

WTR14
      MOVI  A,$FE       ;ID ADDRESS MARK (IDAM)
      JDRQ  WTR19
WTR20
      JTF   WTR12
      JNDRQ WTR20
WTR19
      MOVX  XR0,A       ;WRITE IDAM

      MOV   A,R2        ;TRACK NUMBER
WTR21
      JTF   WTR12
      JNDRQ WTR21
      MOVX  XR0,A       ;WRITE TRACK NUMBER

      IN    A,P1
      RL    A
      RL    A
      ANLI  A,1         ;SIDE NUMBER
WTR22
      JTF   WTR12
      JNDRQ WTR22
      MOVX  XR0,A       ;WRITE SIDE NUMBER

      MOV   A,XR1       ;SECTOR NUMBER FROM INTERLEAVE TABLE
      DECR  R1          ;ADVANCE INTERLEAVE POINTER
WTR23
      JTF   WTR12
      JNDRQ WTR23
      MOVX  XR0,A       ;WRITE SECTOR NUMBER

      IN    A,P2
      JB5   WTR24       ;IF SECTOR SIZE IS 128
      INCR  R0
WTR24
      MOV   A,R0        ;SECTOR SIZE CODE
WTR25
      JTF   WTR12
      JNDRQ WTR25
      MOVX  XR0,A       ;WRITE SECTOR SIZE CODE
```

```
      MOVI  A,$F7       ;F7 WRITES CRC
WTR26
      JTF   WTR12
      JNDRQ WTR26
      MOVX  XR0,A       ;WRITE CRC

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

      MOV   A,R6
      MOV   R0,A        ;LENGTH OF POST-ID GAP
      MOV   A,R4        ;GAP FILLER
WTR27
      JTF   WTR12
WTR28
      JNDRQ WTR27
      MOVX  XR0,A
      DJNZ  R0,WTR28

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

      MOV   A,R5
      MOV   R0,A
      CLR   A
      MOV   T,A         ;RESET THE TIMER
WTR29
      JTF   WTR12
WTR30
      JNDRQ WTR29
      MOVX  XR0,A
      DJNZ  R0,WTR30

;     WRITE MFM SYNC MARKS

      IN    A,P1
      JB3   WTR35       ;IF FM
      MOVI  A,$F5       ;F5 WRITES MFM SYNC MARK
      JDRQ  WTR32
WTR31
      JTF   WTR12
      JNDRQ WTR31
WTR32
      MOVX  XR0,A
      MOVI  R0,2        ;WRITE 2 MORE SYNC MARKS
WTR33
      JTF   WTR12
WTR34
      JNDRQ WTR33
      MOVX  XR0,A
      DJNZ  R0,WTR34

;     WRITE DATA FIELD

WTR35
```

```
      MOVI  A,$FB       ;FB IS DATA ADDRESS MARK (DAM)
      JDRQ  WTR37
WTR36
      JTF   WTR12
      JNDRQ WTR36
WTR37
      MOVX  XR0,A       ;WRITE DAM

      MOV   A,R3
      MOV   R0,A
      INCR  R0          ;SECTOR SIZE
      MOVI  A,$FF       ;DATA BYTE = FF
WTR39
      JTF   WTR12
WTR40
      JNDRQ WTR39
      MOVX  XR0,A       ;WRITE DATA BYTE
      DJNZ  R0,WTR40    ;IF MORE DATA

      MOVI  A,$F7       ;F7 WRITES CRC
WTR41
      JTF   WTR12
      JNDRQ WTR41
      MOVX  XR0,A       ;WRITE CRC

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

      MOV   A,R7        ;LENGTH OF POST-DATA GAP
      MOV   R0,A
      MOV   A,R4        ;GAP FILLER BYTE
WTR42
      JTF   WTR12
WTR43
      JNDRQ WTR42
      MOVX  XR0,A       ;WRITE A GAP BYTE
      DJNZ  R0,WTR43

;     SEE IF MORE SECTORS TO WRITE

      MOV   A,R1
      JB7   WTR6        ;IF MORE SECTORS

;     WRITE GAP IV

      MOVI  R0,SEC1           ;TRIP COUNT FOR 1 SECOND TIMEOUT
WTR45
      MOV   A,R4        ;GAP FILLER BYTE
      JNDRQ WTR46
      MOVX  XR0,A       ;WRITE GAP BYTE
WTR46
      IN    A,P1
      JB2   WTR47       ;IF DONE
      MOV   A,R4
      JNDRQ WTR48
```

```
      MOVX  XR0,A       ;WRITE GAP BYTE
WTR48
      JTF   WTR49
      JUMP  WTR45
WTR49
      DJNZ  R0,WTR45    ;IF NOT TIMED OUT

;     HANDLE TIMEOUT

WTR12
      JUMP  FMT7

WTR4
      JNDRQ WTR51
      JDRQ  WTR50       ;FILTER DRQ GLITCHES
WTR51
      DJNZ  R4,WTR5           ;IF NOT TIMED OUT
      IN    A,P1
      CPL   A
      JB2   WTR52       ;IF OPERATION NOT COMPLETE
      SETREG     RSTA
      MOVX  A,XR0       ;READ STATUS
      CPL   A
      JB6   WTR52       ;IF NOT WRITE PROTECT
      MOVI  R0,CSTAT
      MOVI  XR0,$0C           ;STATUS FOR WRITE PROTECT
WTR52
      JUMP  FMT7

;     HANDLE WRITE TRACK OPERATION COMPLETE

WTR47
      SETREG     RSTA
      MOVX  A,XR0       ;READ STATUS
      ANLI  A,$44       ;RELEVANT STATUS BITS
      JNZ   WTR52       ;IF ERROR
      JUMP  FMT12       ;GO TO NEXT TRACK
```

```
      SUBTTL     'TRACK/SECTOR CALCULATION'


**    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
      ORG   $1800

CTS

;     SAVE REGISTERS WE WILL USE

      MOV   A,R0
      MOVI  R0,R0SAV
      MOV   XR0,A
      DECR  R0
      MOV   A,R1
      MOV   XR0,A
      DECR  R0
      MOV   A,R2
      MOV   XR0,A
      DECR  R0
      MOV   A,R3
      MOV   XR0,A
      DECR  R0
      MOV   A,R4
      MOV   XR0,A
      DECR  R0
      MOV   A,R5
      MOV   XR0,A

;     CHECK FOR COPY PROTECT COMMAND

      MOVI  R0,CCOMND
      MOV   A,XR0
      DECR  R0
      XRLI  A,$72
      JNZ   CTS10       ;IF NOT COPY PROTECT COMMAND
      MOV   A,XR0       ;BRING IN TRACK #
      MOV   R0,A
      ADDI  A,$FF-39
      JC    CTSERR            ;IF TRACK NUMBER TOO BIG
```

```
      ANLI  P1,$FF-SSO  ;SELECT FIRST SIDE
      MOVI  A,IMPSEC    ;'IMPOSSIBLE' SECTOR NUMBER
      JUMP  CTS11

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

CTS10
      MOV   A,XR0
      ADDI  A,$FF
      MOV   R2,A
      DECR  R0
      MOV   A,XR0
      ADDCI A,$FF
      MOV   R3,A

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

      MOVI  R1,$100-18
      MOVI  R4,LOW [0-720]
      MOVI  R5,HIGH [0-720]
      IN    A,P2
      JB5   CTS1        ;IF 128 BYTES PER SECTOR
      DIS   I
      ORLI  P2,SECTSZ   ;TRY TO SET 128 BYTES PER SECTOR
      IN    A,P2
      JB5   CTS2        ;IF DRIVE IS A MINIFLOPPY
      MOVI  R1,$100-16  ;16 SECTORS PER TRACK FOR MICROFLOPPY
      MOVI  R4,LOW [0-640]
      MOVI  R5,LOW [0-640]
CTS2
      ANLI  P2,$FF-SECTSZ
      EN    I
      JUMP  CTS3
CTS1
      IN    A,P1
      ANLI  A,DDEN
      JNZ   CTS3        ;IF SINGLE DENSITY
      MOVI  R1,$100-26  ;26 SECTORS PER TRACK FOR DUAL DENSITY
      MOVI  R4,LOW [0-1040]
      MOVI  R5,HIGH [0-1040]

;     DO BOUNDS CHECK, UPDATE SIDE SELECT OUTPUT

CTS3
      MOV   A,R2
      ADD   A,R4
      MOV   R0,A
      MOV   A,R3
      ADDC  A,R5
      JNC   CTS4        ;IF IN BOUNDS ON FIRST SIDE
      XCH   A,R0
      ADD   A,R4
      CPL   A
```

```
      MOV   R2,A
      MOV   A,R0
      ADDC  A,R5
      CPL   A
      MOV   R3,A
      JC    CTSERR          ;IF OUT OF BOUNDS ON SECOND SIDE

      IN    A,P2
      ANLI  A,$FF-CFCLR
      XRLI  A,CFDSDD
      CLR   C
      CPL   C           ;INDICATE ERROR
      JNZ   CTSERR            ;IF SINGLE SIDED

      ORLI  P1,SSO            ;SELECT SECOND SIDE
      JUMP  CTS5

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

;     CALCULATE TRACK AND SECTOR

CTS5
      MOVI  R0,$FF            ;INIT TO TRACK 0
      MOV   A,R2        ;SECTOR NUMBER LSB
      INCR  R3
CTS6
      INCR  R0          ;COUNT THE TRACK
      ADD   A,R1        ;SUBTRACT NUMBER OF SECTORS PER TRACK
      JC    CTS6        ;IF NO BORROW
      DJNZ  R3,CTS6           ;DECREMENT MSB
      XCH   A,R1
      CPL   A
      INCR  A
      ADD   A,R1        ;ADD BACK NUMBER OF SECTORS
      INCR  A           ;ADD 1 FOR PHYSICAL SECTOR #

;     STORE TRACK AND SECTOR INTO 1770

CTS11
      SETREG     WSEC
      MOVX  XR0,A       ;WRITE SECTOR # INTO SECTOR REG
      SETREG     WDAT
      MOV   A,R0
      MOVX  XR0,A       ;WRITE TRACK # INTO DATA REG
      CLR   C           ;INDICATE SUCCESS

;     RESTORE SAVED REGISTERS

CTSERR
      MOVI  R0,R5SAV
      MOV   A,XR0
      MOV   R5,A
      INCR  R0
```

```
      MOV   A,XR0
      MOV   R4,A
      INCR  R0
      MOV   A,XR0
      MOV   R3,A
      INCR  R0
      MOV   A,XR0
      MOV   R2,A
      INCR  R0
      MOV   A,XR0
      MOV   R1,A
      INCR  R0
      MOV   A,XR0
      MOV   R0,A

;     RETURN TO CALLER

      MOV   A,PSW
      JB0   CTS8
      JB1   CTS7
      LNGJMP      CTSRT0
CTS7
      LNGJMP      CTSRT2
CTS8
      LNGJMP      CTSRT1
```

```
      SUBTTL     'SECTOR I/O'


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

      ASSERT     *<=$1900
      ORG   $1900

GET
PUT
PUTV

;     RAISE 1770 RESET IF IT IS LOW

      IN    A,P1
      ANLI  A,MR
      JNZ   SECT0       ;IF RESET IS HIGH
      ORLI  P1,MR       ;RAISE RESET
      PAUSE R6,US500    ;DELAY 500 US
      SETREG     WCMD
      MOVI  A,FRCINT
      MOVX  XR0,A       ;INITIATE FORCED INTERRUPT
      PAUSE R6,US200    ;DELAY 200 US

;     CALCULATE TRACK AND SECTOR NUMBERS

SECT0
      MOV   A,PSW
      ANLI  A,$FC
      MOV   PSW,A       ;SELECT RETURN TARGET = CTSRT0
      LNGJMP     CTS
CTSRT0
      JNC   SECT1       ;IF SUCCESS
      IN    A,P2
      CPL   A
      JB5   SECT2       ;IF SECTOR SIZE = 256
      IN    A,P1
      ANLI  A,DDEN
      JZ    SECT2       ;IF MFM
      ANLI  P1,$FF-DDEN ;SET TO MFM
      MOV   A,PSW
      ANLI  A,$FC
      ORLI  A,1
      MOV   PSW,A       ;SELECT RETURN TARGET = CTSRT1
      LNGJMP     CTS         ;TRY AGAIN TO CALCULATE
CTSRT1
      JNC   SECT1       ;IF SUCCESS
      ORLI  P1,DDEN           ;RETURN SETTING TO FM
SECT2
      LNGJMP     INVCF       ;INVALID COMMAND FRAME

;     SEND ACKNOWLEDGE
```

```
SECT1
      LNGCAL     SENDAK

;     IN CASE OF PUT, RECEIVE DATA FRAME

      IN    A,P2
      ANLI  A,$FF-ACLR
      XRLI  A,AGET
      JZ    SECT3       ;IF GET

      MOVI  R0,$7F            ;FOR 128-BYTE SECTOR
      MOV   A,PSW
      ORLI  A,1
      MOV   PSW,A       ;SIGNAL NO BUFFER EXPANSION
      IN    A,P2
      JB5   SECT4       ;IF SECTOR SIZE IS 128 BYTES
      IN    A,P1
      ANLI  A,SSO
      JNZ   SEC5        ;IF SECOND SIDE
      SETREG     RDAT
      MOVX  A,XR0
      JNZ   SEC5        ;IF NOT TRACK 0
      SETREG     RSEC
      MOVX  A,XR0
      ADDI  A,$FC
      JC    SEC5        ;IF NOT SECTOR 1, 2, OR 3
      MOV   A,PSW
      ANLI  A,$FE       ;INDICATE BUFFER EXPANSION
      MOV   PSW,A
      JUMP  SECT4
SEC5
      MOVI  R0,$FF            ;FOR 256-BYTE SECTORS
SECT4
      LNGJMP     RECV        ;GO RECEIVE DATA FRAME
PUTRR
      MOV   A,PSW
      JB0   SECT3       ;IF NO BUFFER EXPANSION
      MOVI  R0,$80
      MOV   A,T         ;LAST BYTE OF FRAME
      MOV   XR0,A       ;STORE IT IN HIGH RAM
      DECR  R0
SEC6
      MOV   A,XR0       ;GET A BYTE OF THE DATA FRAME
      XCH   A,R0
      ORLI  A,$80
      XCH   A,R0
      MOV   XR0,A       ;STORE IT INTO HIGH RAM
      XCH   A,R0
      ANLI  A,$7F
      XCH   A,R0
      DJNZ  R0,SEC6           ;IF DONE WITH 128 BYTES

;     CHECK TO SEE IF HEAD POSITION IS KNOWN
```

```
SECT3
      CLR   F1          ;INIT HARD RETRY FLAG
      IN    A,P2
      ANLI  A,HKNOWN
      JZ    SEC19       ;IF HEAD UNKNOWN, DO RESTORE SEQUENCE
      JUMP  SEC20       ;DO SEEK
SEC19
      SETREG     RDAT


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

RST

;     READ IN DESIRED TRACK AND SECTOR

      MOVX  A,XR0       ;GET DESIRED TRACK
      MOV   R0,A
      MOV   A,PSW
      XRL   A,R0
      ANLI  A,$F8
      XRL   A,R0
      MOV   PSW,A       ;STORE BITS 2-0 OF TRACK IN PSW
      MOV   A,R0
      RL    A
      RL    A
      MOV   R0,A
      SETREG     RSEC
      MOVX  A,XR0       ;GET DESIRED SECTOR
      XRLI  A,IMPSEC
      JZ    RST2        ;IF 'IMPOSSIBLE' SECTOR
      XRLI  A,IMPSEC
RST2
      XRL   A,R0
      ANLI  A,$1F
      XRL   A,R0        ;COMBINE TRACK[5:3] WITH SECTOR[4:0]

;     DO HARD RESET TO THE 1770

      ANLI  P2,$FF-HKNOWN    ;INDICATE HEAD POSITION UNKNOWN
      ANLI  P1,$FF-MR   ;LOWER RESET
      PAUSE R0,US500    ;500 US DELAY
      ORLI  P1,MR       ;RAISE RESET
      PAUSE R0,US500    ;500 US DELAY
      MOV   R0,A
      MOVI  A,FRCINT
      MOVX  XR0,A       ;DO FORCED INTERRUPT
      MOV   A,R0
      PAUSE R0,US200    ;200 US DELAY
```

```
;     WRITE SECTOR NUMBER INTO SECTOR REG

      MOV   R0,A
      ANLI  A,$1F
      JNZ   RST1
      MOVI  A,IMPSEC    ;'IMPOSSIBLE' SECTOR
RST1
      SETREG     WSEC
      MOVX  XR0,A

;     ISSUE RESTORE COMMAND

      MOVI  A,REST
      SETREG     WCMD
      MOVX  XR0,A

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

      MOV   A,T         ;LAST DATA BYTE
      XCH   A,R0        ;PUT DATA BYTE IN R0
      RR    A
      RR    A
      RR    A
      RR    A
      RR    A           ;MOVE TRACK[5:3] INTO BITS 2-0
      ANLI  A,$07       ;INIT 5-BIT COUNTER

;     DELAY APPROXIMATELY 100 MS

      JUMP  RST3
      ASSERT     *<=$1A00
      ORG   $1A00

RST3
      MOV   T,A         ;INIT TIMER
      STRT  T
RST4
      JTF   RST5
      JUMP  RST4
RST5
      JTF   RST6
      JUMP  RST5
RST6
      JTF   RST7
      JUMP  RST6
RST7
      JTF   RST8
      JUMP  RST7
RST8
      JTF   RST9
      JUMP  RST8
RST9
      JTF   RST10
```

```
      JUMP  RST9
RST10
      JTF   RST11
      JUMP  RST10
RST11
      STOP  TCNT

;     SEE IF RESTORE COMMAND IS DONE

      MOV   T,A
      IN    A,P1
      JB2   RST12       ;IF RESTORE IS DONE
      MOV   A,T
      ADDI  A,$08
      JNC   RST3        ;IF TIME NOT EXPIRED
      RL    A
      RL    A
      RL    A
      ANLI  A,$38
      XCH   A,R0
      MOV   T,A         ;PUT LAST DATA BYTE IN TIMER
      MOVI  A,FRCINT
      SETREG     WCMD
      MOVX  XR0,A       ;INITIATE FORCED INTERRUPT
      MOV   A,PSW
      XRL   A,R0
      ANLI  A,$07
      XRL   A,R0        ;RECONSTRUCT TRACK NUMBER
      PAUSE R0,US200    ;DELAY 200 US
      SETREG     WTRK
      MOVX  XR0,A       ;WRITE DESIRED TRACK INTO TRACK REG
      JUMP  HRT         ;GO DO HARD RETRY

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

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

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

      ANLI  P1,$FF-DDEN ;SELECT MFM
      MOV   A,R0        ;LAST BYTE TO A
      PAUSE R0,US500    ;DELAY 500 US
      MOV   R0,A        ;LAST BYTE TO R0
      MOVI  A,RDADR
      SETREG     WCMD
      MOVX  XR0,A       ;ISSUE READ ADDRESS COMMAND
      CLR   A
      MOV   T,A         ;INIT TIMER
```

```
      MOVI  A,MS240          ;TRIP COUNT FOR 240 MS DELAY
      JTF   RST16       ;CLEAR TIMER OVERFLOW FLAG
RST16
      STRT  T
      SETREG     RDAT        ;PREPARE TO READ DATA REG

RST17
      JTF   RST18
RST34
      JNDRQ RST17
      JNDRQ RST17       ;FILTER DRQ GLITCHES

;     READ SIX BYTES FROM THE 1770

RST19
      MOVX  A,XR0       ;READ FIRST BYTE
      CLR   A
      MOV   T,A         ;INIT TIMER
      JTF   RST20       ;CLEAR TIMER OVERFLOW FLAG
RST20
      JDRQ  RST21
RST22
      JTF   RST25
      JNDRQ RST22
RST21
      MOVX  A,XR0       ;READ SECOND BYTE
RST23
      JTF   RST25
      JNDRQ RST23
      MOVX  A,XR0       ;READ THIRD BYTE
RST24
      JTF   RST25
      JNDRQ RST24
      MOVX  A,XR0       ;READ FOURTH BYTE
RST29
      JTF   RST25
      JNDRQ RST29
      MOVX  A,XR0       ;READ FIFTH BYTE
RST30
      JTF   RST25
      JNDRQ RST30
      MOVX  A,XR0       ;READ SIXTH BYTE

RST31
      JTF   RST25
      IN    A,P1
      CPL   A
      JB2   RST31       ;IF 1770 STILL BUSY

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

```
;     HANDLE TIMER TRIPS WHILE WAITING FOR FIRST BYTE

RST18
      JNDRQ RST32
      JDRQ  RST19       ;FILTER DRQ GLITCHES
RST32
      DECR  A
      JNDRQ RST33
      JDRQ  RST19       ;FILTER DRQ GLITCHES
RST33
      JNZ   RST34       ;IF NOT TIMED OUT

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

RST25
      MOVI  A,FRCINT
      SETREG     WCMD
      MOVX  XR0,A       ;DO FORCED INTERRUPT
      MOV   A,R0
      PAUSE R0,US200    ;DELAY 200 US
      ORLI  P1,DDEN           ;SELECT FM
      PAUSE R0,US500    ;DELAY 500 US
      MOV   R0,A

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

RST26
      STOP  TCNT
      MOV   A,R0
      MOV   T,A         ;LAST DATA BYTE TO TIMER
      MOV   A,PSW
      ANLI  A,$FC
      ORLI  A,2
      MOV   PSW,A       ;SELECT RETRUN TARGET = CTSRT2
      LNGJMP     CTS         ;CALCULATE TRACK AND SECTOR
CTSRT2
      JNC   RST35       ;IF SUCCESS
      CLR   F1
      CPL   F1          ;FORCE HARD RETRY TO FAIL
      JUMP  HRT         ;GO DO HARD RETRY

;     RECONSTRUCT TRACK NUMBER

RST13
      MOV   A,T
      RL    A
      RL    A
      RL    A
      ANLI  A,$38
      XCH   A,R0
      MOV   T,A         ;PUT LAST DATA BYTE INTO TIMER
```

```
      MOV   A,PSW
      XRL   A,R0
      ANLI  A,$07
      XRL   A,R0        ;RECONSTRUCT TRACK NUMBER
      SETREG     WDAT
      MOVX  XR0,A       ;WRITE DESIRED TRACK INTO DATA REG
RST35
      CLR   A
      SETREG     WTRK
      MOVX  XR0,A       ;UPDATE TRACK REG TO ZERO

;     END OF RESTORE SEQUENCE

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

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

SEC20
      ANLI  P2,$FF-HKNOWN    ;INDICATE HEAD POSITION UNKNOWN
      MOVI  A,SEEK
      SETREG     WCMD
      MOVX  XR0,A       ;ISSUE SEEK COMMAND
      MOV   A,T         ;FETCH LAST DATA BYTE
      MOV   R0,A        ;STORE IT
      CLR   A
      MOV   T,A         ;INIT TIMER
      MOV   A,PSW
      ANLI  A,$3F
      ORLI  A,$07       ;INIT 5-BIT COUNTER WITHIN PSW
      STRT  T

;     WAIT FOR SEEK COMMAND TO BE COMPLETE

SEC8
      MOV   PSW,A
SEC10
      JTF   SEC11
      IN    A,P1
      JB2   SEC9        ;IF SEEK DONE
      JUMP  SEC10
SEC11
      JTF   SEC12
      IN    A,P1
      JB2   SEC9        ;IF SEEK DONE
      JUMP  SEC11
SEC12
      JTF   SEC13
      IN    A,P1
      JB2   SEC9        ;IF SEEK DONE
      JUMP  SEC12
```

```
SEC13
      JTF   SEC14
      IN    A,P1
      JB2   SEC9        ;IF SEEK DONE
      JUMP  SEC13
SEC14
      JTF   SEC15
      IN    A,P1
      JB2   SEC9        ;IF SEEK DONE
      JUMP  SEC14
SEC15
      JTF   SEC16
      IN    A,P1
      JB2   SEC9        ;IF SEEK DONE
      JUMP  SEC15

;     HANDLE POSSIBLE SEEK TIMEOUT

SEC16
      MOV   A,PSW
      DECR  A
      JB3   SEC8        ;IF NOT TIMED OUT
      ADDI  A,$48
      JNC   SEC8        ;IF NOT TIMED OUT
      STOP  TCNT
      MOV   A,R0
      MOV   T,A         ;PUT LAST DATA BYTE IN TIMER
      MOVI  A,FRCINT
      SETREG     WCMD
      MOVX  XR0,A       ;DO FORCED INTERRUPT
      PAUSE R0,US200    ;DELAY 200 US
      SETREG     RDAT
      MOVX  A,XR0       ;READ IN TARGET TRACK
      SETREG     WTRK
      MOVX  XR0,A       ;WRITE IT INTO TRACK REG
      JUMP  HRT         ;GO DO HARD RETRY

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

SEC9
      ORLI  P2,HKNOWN   ;INDICATE HEAD POSITION KNOWN
      SETREG     RSTA
      MOVX  A,XR0       ;READ STATUS REG TO CLEAR INTRQ
      MOV   A,PSW
      ANLI  A,$F8
      ORLI  A,2

;     COME HERE TO EXECUTE SOFT RETRY

      JUMP  SEC18
      ASSERT     *<=$1C00
      ORG   $1C00
```

```
SEC18
      MOV   PSW,A       ;INIT SOFT RETRY COUNTER
      STOP  TCNT
      JTF   SEC17       ;CLEAR TIMER OVERFLOW FLAG
SEC17
      CLR   A
      MOV   T,A         ;INIT TIMER FOR ID FIELD TIMEOUT


**    HANDLE GET COMMAND

      IN    A,P2
      ANLI  A,$FF-ACLR
      XRLI  A,AGET
      JZ    GET1        ;IF A GET COMMAND
      JUMP  PUT1

;     INITIALIZATION FOR GET

GET1
      IN    A,P2
      MOVI  R1,$7F            ;INIT INDEX FOR 128-BYTE SECTORS
      JB5   GET2        ;IF SECTOR SIZE IS 128
      MOVI  R1,$FF            ;INIT INDEX FOR 256-BYTE SECTORS
GET2
      MOV   A,R1
      MOV   R0,A
      DECR  R0          ;INDEX FOR SECOND BYTE
      MOVI  R2,SEC1           ;TRIP COUNT FOR 1 SECOND DELAY

;     ISSUE READ SECTOR COMMAND

      SETREG      WCMD
      MOVI  A,RDSEC
      MOVX  XR0,A

;     READ FIRST DATA BYTE

      SETREG      RDAT
GET3
      JTF   GET4
GET6
      JNDRQ GET3        ;WAIT FOR DRQ
      JNDRQ GET3        ;FILTER DRQ GLITCHES
GET5
      MOVX  A,XR0       ;BRING IN FIRST BYTE
      CPL   A           ;COMPLEMENT IT
      MOV   XR1,A       ;STORE IT IN RAM
      CLR   A
      MOV   T,A         ;RESET THE TIMER
      JTF   GET18       ;RESET TIMER OVERFLOW FLAG
GET18
      JDRQ  GET19
```

```
;     READ IN THE MIDDLE BYTES

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

;     READ IN THE LAST BYTE

GET10
      JTF   GET9
      JNDRQ GET10
      MOVX  A,XR0       ;BRING IN LAST BYTE
      CPL   A           ;COMPLEMENT IT
      MOV   R0,A        ;STORE IT IN R0

;     WAIT FOR OPERATION COMPLETE

GET11
      JTF   GET9
      IN    A,P1
      ANLI  A,INTRQ
      JZ    GET11       ;IF NOT COMPLETE
      STOP  TCNT
      MOV   A,R0
      MOV   T,A         ;PUT LAST BYTE IN TIMER
      SETREG     RSTA
      MOVX  A,XR0
      MOV   R0,A        ;STORE OPERATION STATUS IN R0

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

      IN    A,P1
      ANLI  A,SSO OR DDEN
      JNZ   GET12       ;IF FM OR SIDE 1
      SETREG     RTRK
      MOVX  A,XR0
      JNZ   GET12       ;IF NOT TRACK 0
      SETREG     RSEC
      MOVX  A,XR0
      ADDI  A,$FC
      JC    GET12       ;IF NOT SECTOR 1, 2, OR 3

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

      MOV   A,R0
      MOV   T,A         ;STORE STATUS IN TIMER
      MOVI  R0,$7F
```

```
GET14
      MOV   A,R0
      ORLI  A,$80
      MOV   R0,A        ;POINT TO HIGH RAM
      MOV   A,XR0       ;FETCH A BYTE FROM HIGH RAM
      XCH   A,R0
      ANLI  A,$7F
      XCH   A,R0
      MOV   XR0,A       ;WRITE IT TO LOW RAM
      DJNZ  R0,GET14    ;IF MORE BYTES TO COPY
      MOVI  R0,$80
      MOV   A,XR0       ;NEW LAST BYTE
      MOV   R0,A
      MOV   A,T
      XCH   A,R0        ;PUT STATUS IN R0
      MOV   T,A         ;PUT LAST BYTE IN TIMER
      MOVI  A,$7F       ;TRANSMIT POINTER
      JUMP  GET15

GET13
      MOV   A,R0
      ANLI  A,$FB       ;TURN OFF LOST DATA BIT
      MOV   R0,A

;     INITIALIZE R0 FOR TRANSMISSION

GET12
      IN    A,P2
      RL    A
      RL    A
      ANLI  A,$80
      DECR  A
GET15
      XCH   A,R0        ;STATUS TO ACCUMULATOR

;     CHECK STATUS

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

;     SUCCESSFUL COMPLETION OF GET

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

;     HANDLE GET TIMEOUTS

GET4
      JNDRQ GET16
      JDRQ  GET5        ;FILTER DRQ GLITCHES
GET16
      DJNZ  R2,GET6           ;IF TIME NOT EXHAUSTED
      STOP  TCNT
```

```
      SETREG     WCMD
      MOVI  A,FRCINT
      MOVX  XR0,A       ;DO FORCED INTERRUPT
      PAUSE R0,US200    ;DELAY 200 US
      JUMP  HRT         ;DO HARD RETRY

GET9
      STOP  TCNT
      SETREG     WCMD
      MOVI  A,FRCINT
      MOVX  XR0,A       ;DO FORCED INTERRUPT
      PAUSE R0,US200


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

SRT
      MOV   A,T
      MOV   R0,A        ;COPY LAST DATA BYTE INTO R0
      MOV   A,PSW
      DECR  A
      JB3   SEC18       ;IF RETRIES NOT EXHAUSTED


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

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

HRT1
      MOVI  R0,CSTAT
      MOVI  XR0,4       ;STATUS CODE FOR UNSUCCESSFUL OPERATION
      MOVI  R0,0        ;NO DATA FRAME
      IN    A,P2
      ANLI  A,$FF-ACLR
      XRLI  A,AGET
      JNZ   HRT2        ;IF PUT
      IN    A,P2
      RL    A
      RL    A
      ANLI  A,$80
      DECR  A
      MOV   R0,A        ;DATA FRAME LENGTH
HRT2
      LNGJMP     SEND       ;SEND ERROR, GO TO IDLE
```

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

      ASSERT      *<=$1D00
      ORG   $1D00

PUT1

;     INITIALIZE FOR PUT

      IN    A,P2
      RL    A
      RL    A
      ANLI  A,$80
      DECR  A
      XCH   A,R0
      XCH   A,XR0
      CPL   A           ;COMPLEMENT FIRST BYTE
      MOV   R0,A

;     ISSUE WRITE SECTOR COMMAND

      SETREG      WCMD
      MOVI  A,WRSEC
      MOVX  XR0,A
      SETREG      WDAT

;     SEND FIRST DATA BYTE

      MOV   A,R0
      MOVI  R0,SEC1           ;TRIP COUNT FOR 1 SECOND TIMEOUT
      STRT  T

PUT2
      JTF   PUT3
PUT4
      JNDRQ PUT2
      JNDRQ PUT2        ;FILTER DRQ GLITCHES
PUT5
      MOVX  XR0,A       ;WRITE FIRST BYTE
      CPL   A           ;UN-COMPLEMENT
      XCH   A,R0
      CLR   A
      MOV   T,A         ;RESET THE TIMER
      JTF   PUT6        ;CLEAR TIMER OVERFLOW FLAG
PUT6
      IN    A,P2
      RL    A
      RL    A
      ANLI  A,$80
```

```
      DECR  A
      XCH   A,R0
      XCH   A,XR0       ;LAST BYTE TO ACCUMULATOR
      DECR  R0

;     WRITE THE MIDDLE BYTES

PUT7
      XCH   A,XR0       ;GET A BYTE FROM RAM
      CPL   A           ;COMPLEMENT IT
      JDRQ  PUT13
PUT8
      JTF   PUT9        ;IF TIMED OUT
      JNDRQ PUT8
PUT13
      MOVX  XR0,A       ;WRITE THE BYTE
      CPL   A           ;UN-COMPLEMENT IT
      XCH   A,XR0       ;PUT IT BACK INTO RAM
      DJNZ  R0,PUT7           ;ADVANCE POINTER

;     WRITE THE LAST BYTE

      MOV   R0,A        ;SAVE LAST BYTE IN R0
      CPL   A           ;COMPLEMENT IT
      JDRQ  PUT14
PUT10
      JTF   PUT11
      JNDRQ PUT10
PUT14
      MOVX  XR0,A       ;WRITE THE LAST BYTE

;     WAIT FOR OPERATION COMPLETE

PUT12
      JTF   PUT11
      IN    A,P1
      ANLI  A,INTRQ
      JZ    PUT12       ;IF NOT COMPLETE
      STOP  TCNT
      MOV   A,R0
      MOV   T,A         ;PUT LAST BYTE IN TIMER
      SETREG     RSTA
      MOVX  A,XR0       ;READ STATUS REG
      JB6   PUT18       ;IF WRITE PROTECT
      ANLI  A,$14       ;RELEVANT STATUS BITS
      JZ    PTV         ;IF SUCCESS, GO DO VERIFY
      JUMP  SRT         ;FAILURE, GO DO SOFT RETRY

;     HANDLE PUT TIMEOUTS

PUT3
      JNDRQ PUT16
      JDRQ  PUT5
PUT16
```

```
      DJNZ  R0,PUT4          ;IF TIME NOT EXHAUSTED
      STOP  TCNT
      CPL   A           ;UN-COMPLEMENT
      MOV   R0,A
      IN    A,P2
      RL    A
      RL    A
      ANLI  A,$80
      DECR  A
      XCH   A,R0
      XCH   A,XR0       ;RESTORE FIRST BYTE TO RAM
      MOV   T,A         ;LAST BYTE TO TIMER
      IN    A,P1
      ANLI  A,INTRQ
      JZ    PUT17       ;IF OPERATION NOT COMPLETE
      SETREG     RSTA
      MOVX  A,XR0       ;READ STATUS REG
      JB6   PUT18       ;IF WRITE PROTECT
PUT17
      SETREG     WCMD
      MOVI  A,FRCINT
      MOVX  XR0,A       ;DO FORCED INTERRUPT
      PAUSE R0,US200
      JUMP  HRT         ;GO DO HARD RETRY

PUT9
      CPL   A
      XCH   A,XR0       ;PUT BYTE BACK IN RAM
      MOV   R0,A        ;SAVE LAST BYTE IN R0
PUT11
      STOP  TCNT
      MOV   A,R0
      MOV   T,A         ;PUT LAST BYTE IN TIMER
      IN    A,P1
      ANLI  A,INTRQ
      JZ    PUT19       ;IF OPERATION NOT COMPLETE
      SETREG     RSTA
      MOVX  A,XR0       ;READ STATUS
      JB6   PUT18       ;IF WRITE PROTECT
PUT19
      SETREG     WCMD
      MOVI  A,FRCINT
      MOVX  XR0,A       ;DO FORCED INTERRUPT
      PAUSE R0,US200    ;DELAY 200 US
      JUMP  SRT         ;GO DO SOFT RETRY

PUT18
      MOVI  R0,CSTAT
      MOVI  XR0,$0C          ;STATUS CODE FOR WRITE PROTECT
      MOVI  R0,0        ;NO DATA FRAME
      CLR   F1
      CPL   F1          ;INDICATE ERROR
      LNGJMP     SEND        ;SEND ERROR, GO TO IDLE
```

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

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

PTV1
      IN    A,P2
      ANLI  A,$FF-ACLR
      XRLI  A,APUTV
      JNZ   PTV10       ;IF VERIFY NOT REQUIRED

;     SHIFT AND COMPLEMENT INTERNAL RAM

      IN    A,P2
      RL    A
      RL    A
      ANLI  A,$80
      MOV   R0,A        ;INDEX TO FIRST DATA BYTE
      DECR  R0
      RLC   A           ;SET CARRY FOR LATER USE
      MOV   A,T         ;LAST DATA BYTE TO ACCUMULATOR
      CPL   A           ;COMPLEMENT IT
      XCH   A,XR0       ;FIRST DATA BYTE TO ACCUMULATOR
      CPL   A           ;COMPLEMENT IT
      DECR  R0
PTV11
      XCH   A,XR0       ;BRING A DATA BYTE TO ACCUMULATOR
      CPL   A           ;COMPLEMENT IT
      INCR  R0
      XCH   A,XR0       ;SHIFT IT FORWARD ONE POSITION
      DECR  R0
      XCH   A,XR0       ;RESTORE FIRST DATA BYTE TO ACCUMULATOR
      DJNZ  R0,PTV11    ;IF MORE BYTES TO DO
      MOV   R0,A        ;PUT FIRST DATA BYTE IN R0

;     ISSUE READ SECTOR COMMAND

      STOP  TCNT
      JTF   PTV12       ;CLEAR TIMER OVERFLOW FLAG
PTV12
      CLR   A
      MOV   T,A         ;RESET THE TIMER
      SETREG      WCMD
      MOVI  A,RDSEC
      MOVX  XR0,A
      SETREG      RDAT
      STRT  T
      MOVI  A,SEC1            ;TRIP COUNT FOR 1 SECOND TIMEOUT
```

```
;     READ THE FIRST TWO BYTES

PTV2
      JTF   PTV3
PTV4
      JNDRQ PTV2
      JNDRQ PTV2        ;FILTER DRQ GLITCHES
PTV13
      MOVX  A,XR0       ;READ FIRST BYTE
      XRL   A,R0        ;COMPARE
      JNZ   PTV5        ;IF INCORRECT
      MOV   R0,A        ;STORE ZERO IN R0

      JDRQ  PTV6
PTV7
      INCR  A
      JDRQ  PTV6
      JZ    PTV5        ;IF TIMED OUT
      JNDRQ PTV7
PTV6
      MOVX  A,XR0       ;READ SECOND BYTE
      XCH   A,R0
      RRC   A
      DECR  A           ;INDEX TO SECOND BYTE
      XCH   A,R0
      XRL   A,XR0       ;COMPARE
      JNZ   PTV5        ;IF INCORRECT
      DECR  R0

;     READ AND COMPARE THE REMAINING BYTES

PTV14
      JDRQ  PTV15
PTV16
      INCR  A
      JDRQ  PTV15
      JZ    PTV5        ;IF TIMED OUT
      JNDRQ PTV16
PTV15
      MOVX  A,XR0       ;READ BYTE
      XRL   A,XR0       ;COMPARE
      JNZ   PTV5        ;IF INCORRECT
      DJNZ  R0,PTV14    ;IF MORE BYTES TO DO

;     WAIT FOR OPERATION COMPLETE

      CLR   A
      MOV   T,A         ;RESET TIMER
      JTF   PTV17       ;CLEAR TIMER OVERFLOW FLAG
PTV17
      JTF   PTV5        ;IF TIMED OUT
      IN    A,P1
      ANLI  A,INTRQ
      JZ    PTV17       ;IF NOT COMPLETE
```

```
      STOP  TCNT
      SETREG     RSTA
      MOVX  A,XR0       ;READ STATUS
      ANLI  A,$3C       ;SELECT RELEVANT STATUS BITS
      JNZ   PTV5        ;IF ERROR

;     COME HERE FOR SUCCESSFUL VERIFY

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

;     HANDLE VERIFY TIMEOUTS AND ERRORS

PTV3
      JNDRQ PTV18
      JDRQ  PTV13
PTV18
      DECR  A
      JNDRQ PTV19
      JDRQ  PTV13
PTV19
      JNZ   PTV4        ;IF 1 SECOND TIME NOT EXHAUSTED

PTV5
      STOP  TCNT
      MOVI  A,FRCINT
      SETREG     WCMD
      MOVX  XR0,A       ;DO FORCED INTERRUPT
      PAUSE R0,US200
      CLR   F1
      CPL   F1          ;SO HARD RETRY WILL FAIL
      JUMP  HRT         ;SEND ERROR, GO TO IDLE

      ASSERT     *<=$1F00
```