FLOPPY.MAC

```
VER     DEFL 4                  ;Version 4 module

        DSEG

        SUBTTL Skew tables
        PAGE

;       ***************************
;       *       SKEW TABLES       *
;       ***************************

;Make skew table using floppy number to establish format
FN      DEFL 1                  ;Initialise floppy number
DN      DEFL NDRVW

        REPT NDRVF
        MLAB1 FM.,FORM,%FN
        MLAB2 SKEW.,SKW,%FM.,SECS.,MXS,%FM.
        MLAB1 CGS.,CGS,%FM.

        IF FM. EQ NULLF
        ASGSKW %DN              ;;Assignable skew

        ELSE

        IF SKEW. EQ SPSK
        SPCSKW %FM.             ;;Special skew

        ELSE

        IF CGS. EQ SECCYL
SECS.   DEFL SECS./2            ;;Skew on one side only
        ENDIF

        IF SKEW. GT 1
        MULTI S,%SECS.,%SKEW.   ;;Test for multi expansion

        IF FIRST
        SKEW %SECS.,%SKEW.
        ENDIF

        ENDIF
        ENDIF
        ENDIF

FN      DEFL FN+1
DN      DEFL DN+1
        ENDM

        SUBTTL Floppy Data Base
        PAGE

;       *********************************
;       *       FLOPPY DATA BASE        *
;       *********************************

;       * Drive parameters *

LDRIV:  DEFB 0FFH               ;~ Requested logical drive
FMTLOG: DEFB 0FFH               ;~ Logical drive log
PDRIV:  DEFB 0FFH               ;~ Requested physical drive
DRVLOG: DEFB 0FFH               ;~ Physical drive log
```

```
;       * Drive parameters *

LDRIV:  DEFB 0FFH               ;^ Requested logical drive
FMTLOG: DEFB 0FFH               ;^ Logical drive log
PDRIV:  DEFB 0FFH               ;^ Requested physical drive
DRVLOG: DEFB 0FFH               ;^ Physical drive log

TRKA:   REPT NDRVF              ;^ Floppy track record
        DEFB 0FFH
        ENDM

DBASE:                          ;Data Base start

;       * Format parameters *

;YES/NO options
;Bit 0 set = double step
;Bit 1 set = deleted data mark
;Bit 2 set = sectors start at 1
;Bit 3 set = no side 2 flag
;Bit 4 set = tracks are reversed on side 2
;Bit 5 set = tracks are cylindrical
;Bit 6 set = physical sector numbers on side 2 carry on from side 1
;Bit 7 set = assignable drive
FN      DEFL 1
YNOPT:  REPT NDRVF
OPT_    DEFL 0

;;Get Floppy and format type
        MLAB2 FG.,FASG,%FN,FM.,FORM,%FN
        MLAB1 FT.,FPPY,%FG.
;;Test for assignable drive
        IF FM.  EQ NULLF
OPT_    DEFL OPT_ OR 80H
        ENDIF

;;Get Drive and Format track density
        MLAB2 LBL1.,DTY,%FT.,LBL2.,FTY,%FM.
        IF (LBL1. EQ TP96) AND (LBL2. EQ TP48)
;;Double step flag when using 48tpi format on 96tpi drives
OPT_    DEFL OPT_ OR 01
        ENDIF

;;Get data mark info
        MLAB1 LBL1.,DDM,%FM.
        IF LBL1. NE 0
;;Data mark is deleted
OPT_    DEFL OPT_ OR 02
        ENDIF

;;Get sector offset
        MLAB1 LBL1.,SOF,%FM.
        IF LBL1.
;;Sectors start at 1
OPT_    DEFL OPT_ OR 04
        ENDIF

;;Get side 2 flag info
        MLAB1 LBL1.,BSF,%FM.
        IF LBL1.
;;ID header not set for side 2
OPT_    DEFL OPT_ OR 08
        ENDIF

;;Get side handling
        MLAB1 LBL1.,CGS,%FM.
        IF LBL1. EQ TRKREV
;;Tracks are reversed on side 2
OPT_    DEFL OPT_ OR 10H
        ENDIF
        IF LBL1. EQ TRKCYL
;;Tracks are cylindrical
OPT_    DEFL OPT_ OR 20H
        ENDIF

;;Get sectors on 2
        MLAB1 LBL1.,SCY,%FM.
        IF LBL1. NE 0
;;Physical sectors on side 2 carry on from side 1
OPT_    DEFL OPT_ OR 40H
        ENDIF

        DEFB OPT
```

```
        MLAB1 LBL1.,CGS,%FM.
        IF LBL1. EQ TRKREV
;;Tracks are reversed on side 2
OPT_    DEFL OPT_ OR 10H
        ENDIF
        IF LBL1. EQ TRKCYL
;;Tracks are cylindrical
OPT_    DEFL OPT_ OR 20H
        ENDIF

;;Get sectors on 2
        MLAB1 LBL1.,SCY,%FM.
        IF LBL1. NE 0
;;Physical sectors on side 2 carry on from side 1
OPT_    DEFL OPT_ OR 40H
        ENDIF

        DEFB OPT_
FN      DEFL FN+1
        ENDM

;Physical sectors per logical track
FN      DEFL 1                  ;Initialise floppy number
MAXSEC: REPT NDRVF
        MLAB1 FM.,FORM,%FN      ;;Get Format type
        MLAB1 LBL1.,MXS,%FM.    ;;Get sectors per track
        DEFB LBL1.
FN      DEFL FN+1
        ENDM

;Sector to change sides
FN      DEFL 1                  ;Initialise floppy number
SIDSEC: REPT NDRVF
;;Get Format type
        MLAB1 FM.,FORM,%FN
;;Get sectors per track, and side handling
        MLAB2 LBL1.,MXS,%FM.,LBL2.,CGS,%FM.

        IF LBL2. EQ SECCYL
        DEFB LBL1./2            ;;Split track in 2 by sectors
        ELSE
        DEFB LBL1.              ;;Don't change by sector
        ENDIF

FN      DEFL FN+1
        ENDM

;Track to change sides
FN      DEFL 1                  ;Initialise floppy number
SIDTRK: REPT NDRVF
;;Get Format type
        MLAB1 FM.,FORM,%FN
;;Get tracks per format, side handling, and change side track
        MLAB2 LBL1.,MXT,%FM.,LBL2.,CGS,%FM.
        MLAB1 LBL3.,CGT,%FM.

        IF (LBL2. EQ TRKCRY) OR (LBL2. EQ TRKREV)
        IF LBL3.
        DEFB LBL3. AND 7FH      ;;Use parsed track for change
        ELSE
        DEFB LBL1./2            ;;Split total tracks in 2
        ENDIF
        ELSE
        DEFB LBL1.              ;;Don't switch by track
        ENDIF

FN      DEFL FN+1
        ENDM

IXPTR:                          ;Driver data base pointer

;Format assignment
FN      DEFL 1                  ;Initialise floppy number
FORMS:  REPT NDRVF
```

```
;;Get Format type
        MLAB1 FM.,FORM,%FN
        DEFB FM.
FN      DEFL FN+1
        ENDM

;Sector size 1=128, 2=256, 4=512, 8=1024
;Bit 7 set indicates data is inverse
FN      DEFL 1                  ;Initialise floppy number
SCSZ:   REPT NDRVF
;;Get Physical sector size
        MLAB1 FM.,FORM,%FN
        MLAB2 SEC.,SEC,%FM.,INV.,INV,%FM.
SZE_    DEFL HIGH (SEC.*2)
        IF INV. NE 0
SZE_    DEFL SZE_ OR 80H
        ENDIF
        DEFB SZE_
FN      DEFL FN+1
        ENDM

;Track skew
;FN     DEFL 1                  ;Initialise floppy number
;TKSKEW:         REPT NDRVF
;;Get Track skew
;       MLAB1 FM.,FORM,%FN
;       MLAB1 TSK.,TSK,%FM.
;       DEFB TSK.
;FN     DEFL FN+1
;       ENDM

;First spare
SPARE1: REPT NDRVF
        DEFB 0
        ENDM

;Second spare
SPARE2: REPT NDRVF
        DEFB 0
        ENDM

;       * Drive parameters *

IYPTR:
;Drive type, not used internally
FN      DEFL 1                  ;Initialise floppy number
FLPTYP: REPT 4
        MLAB1 FP.,FPPY,%FN      ;;Get Floppy type
        IF FP. EQ 0             ;;No drive
        DEFB 0FFH
        ELSE
        MLAB1 LBL1.,DTY,%FP.    ;;Get step rate
        DEFB LBL1.
        ENDIF
FN      DEFL FN+1
        ENDM

;Step speed in millisecs
FN      DEFL 1                  ;Initialise floppy number
STPSPD: REPT 4
        MLAB1 FP.,FPPY,%FN      ;;Get Floppy type
        MLAB1 LBL1.,STP,%FP.    ;;Get step rate
        DEFB LBL1.
FN      DEFL FN+1
        ENDM

;Track settling time
FN      DEFL 1                  ;Initialise floppy number
SETTL:  REPT 4
        MLAB1 FP.,FPPY,%FN      ;;Get Floppy type
        MLAB1 LBL1.,STL,%FP.    ;;Get settling time
        DEFB LBL1.
FN      DEFL FN+1
        ENDM

;Headload delay in ms.
FN      DEFL 1                  ;Initialise floppy number
```

```
        ENDM

;Track settling time
FN      DEFL 1                  ;Initialise floppy number
SETTL:  REPT 4
        MLAB1 FP.,FPPY,%FN      ;;Get Floppy type
        MLAB1 LBL1.,STL,%FP.    ;;Get settling time
        DEFB LBL1.
FN      DEFL FN+1
        ENDM

;Headload delay in ms.
FN      DEFL 1                  ;Initialise floppy number
HDDEL:  REPT 4
        MLAB1 FP.,FPPY,%FN      ;;Get Floppy type
        MLAB1 LBL1.,HDL,%FP.    ;;Get head load delay
        DEFB LBL1.
FN      DEFL FN+1
        ENDM

LINK:   DEFB MPILNK             ;Say if MPILNK is made

        SUBTTL WD2797 Diskette Routines
        PAGE

;Read/Write to sector in LSECT
;At track in LTRAK
;On drive in LDRIV,PDRIV
;To/From address in DMAADR
RWFLPY: PUSH IX                 ;Save users IX
        PUSH IY                 ;Save users IY
        CALL SETFLP             ;Set up for floppy read/write
        JR NZ,FLPERR            ;Error during set up, take it back
        CALL SETCNT             ;Set multi IO counters
        LD B,A                  ;Sector count to B
        JR GETSEC

GETALL: LD (DMAADR),HL          ;Increment DMA address
        LD HL,LSECT
        INC (HL)                ;Bump logical sector (assume 8 bit)
        LD A,(IX+MAXSEC-IXPTR)  ;Check for sector overflow
TSTSEC: CP (HL)                 ;Test for sector overflow
        JR NZ,GETSEC            ;No overflow
        LD (HL),0               ;Reset sector
        DEC HL
        DEC HL                  ;Back to logical track
        INC (HL)                ;(Assume 8 bit)

GETSEC: PUSH BC                 ;Save sector count
        CALL FRDWR              ;Transfer a sector
        POP BC
        JR NZ,FLPERR            ;Error
        DJNZ GETALL             ;Transfer all sectors
XYOUT:  POP IY
        POP IX
        LD HL,0
        LD (SKTAB),HL           ;Cancel skew table
        RET

;Floppy error, reset counters
FLPERR: LD HL,0
        LD (MULCNT),HL          ;Clear count down
        JR XYOUT                ;Recover IX

;Set counters for multi IO
;Returns sector count in A
SETCNT: LD HL,MULCNT            ;Sectors to read
        LD A,(HL)
```

```
        LD (HL),0               ;Clear multi-count
        OR A
        JR NZ,MULTI
        INC A                   ;Multi count uninitialised so make it 1
MULTI:  INC HL                  ;Point to countdown
        LD (HL),A               ;Set up countdown
        DEC (HL)                ;Deduct first sector
        RET

;Error during set-up, display and prompt retry
SETERR: CALL SHWERR             ;Show the error
        RET NZ                  ;Don't retry
;Set up for floppy read/write
;Log in drive and start motor
;Initialise IX index to data base
;Load the heads
;Head load delay if new drive
;Set current track
;Initialise Sec size and command
;Called only at begining of a multi read/write
SETFLP: LD HL,LDRIV             ;Logical drive
        LD A,(HL)               ;Get logical drive
        LD B,A                  ;Keep it
        INC HL                  ;FMTLOG
        CP (HL)                 ;Compare against last call
        LD (HL),A               ;and update
        PUSH AF                 ;Save flag
        INC HL                  ;PDRIV
        LD A,(HL)               ;Physical drive code
        OUT (DRVPRT),A          ;Kick motor
        INC HL                  ;DRVLOG
        RES 7,(HL)              ;Resest side flag
        CP (HL)                 ;New drive ?
        LD (HL),A               ;Save physical code
        PUSH AF                 ;Save flag
        LD IY,IYPTR-1           ;Point IY at drive data
GOIY:   INC IY
        RRCA
        JR NC,GOIY
        LD A,B                  ;Recover logical drive
        SUB NDRVW               ;Take off Winis
        LD B,A
        INC B
        LD IX,IXPTR-1           ;Point to track map
GOTRAK: INC IX
        DJNZ GOTRAK             ;Get into map
        LD A,(IX+FORMS-IXPTR)
        CP NULLF                ;Test for assignable format
        JR Z,QUITSU             ;Quit set up if it is
        CALL LOADH              ;Load heads
        POP AF                  ;Recover drive log flag
        JR Z,FNDTRK             ;Same physical drive
        LD A,(IY+HDDEL-IYPTR)   ;Head load delay on changing drives
        CALL DELAY
;Find out where head is on first using a drive
FNDTRK: POP AF                  ;Recover format log flag
        LD A,(IX+TRKA-IXPTR)    ;Get current track
        JR NZ,NEWFMT            ;Logical drive change
        CP 0FFH                 ;Test track
        JR NZ,ONTRAK            ;Valid track so use it
NEWFMT: CALL WAITON             ;Wait til drive ready
        JR C,SETERR             ;Time out
        LD A,READID
        OUT (FDCCOM),A          ;Read address header
RDHEAD: IN A,(STAPRT)
        AND 00000011B           ;Wait for IRQ or READY off
        JR Z,RDHEAD
        IN A,(FDCDAT)           ;Clear DRQ and BUSY
        IN A,(FDCSTA)
        AND 11111011B           ;We know we've lost data
        JR Z,HDROK
        AND 10000000B           ;Ready?
        JR NZ,SETERR            ;No
```

```
        JR C,SETERR             ;Time out
        LD A,READID
        OUT (FDCCOM),A          ;Read address header
RDHEAD: IN A,(STAPRT)
        AND 00000011B           ;Wait for IRQ or READY off
        JR Z,RDHEAD
        IN A,(FDCDAT)           ;Clear DRQ and BUSY
        IN A,(FDCSTA)
        AND 11111011B           ;We know we've lost data
        JR Z,HDROK
        AND 10000000B           ;Ready?
        JR NZ,SETERR            ;No
        CALL TRACK0             ;Drive is at least turning
        XOR A
        JR ONTRAK
HDROK:  IN A,(FDCSEC)           ;Get track
        LD (IX+TRKA-IXPTR),A    ;Update track map
ONTRAK: OUT (FDCTRK),A          ;Update track register
        LD A,(RWFLAG)
        OR A                    ;0 if write
        LD A,RDSEC              ;Read sec command, with side 0
        JR NZ,GOTCOM
        LD A,WRSEC              ;Write sec command, with side 0, true data mark
        BIT 1,(IX+YNOPT-IXPTR)  ;Test for deleted data mark
        JR Z,GOTCOM
        SET 0,A                 ;Deleted data mark
GOTCOM: LD (COMM),A
        LD A,(IX+SCSZ-IXPTR)
        LD (SECSZE),A           ;Set sector size if invert required
        XOR A                   ;No errors
        RET

;Can't run unassinged NULLF
QUITSU: POP HL
        POP HL                  ;Clear stack
        XOR A
        DEC A                   ;NZ 0FFH as error
        RET

;Convert logical track to physical
CONVT:  LD A,(LTRAK)            ;Logical track
        LD (PTRAK),A            ;Default is logical
        BIT 5,(IX+YNOPT-IXPTR)  ;Test for track cylindrical
        JR NZ,TCYL
        CP (IX+SIDTRK-IXPTR)    ;Test for side change
        RET C                   ;Side 0 or not TRKCRY or TRKREV
;Track carry
        SUB (IX+SIDTRK-IXPTR)   ;Take off side 0
        BIT 4,(IX+YNOPT-IXPTR)  ;Test for TRKREV
        JR Z,T2
;Track reversed
        CPL
        ADD A,(IX+SIDTRK-IXPTR)
T2:     CALL SET2
T1:     LD (PTRAK),A
        RET
;Track cylindrical
TCYL:   SRL A                   ;Logical to physical, side in C
        JR C,T2                 ;Side 1
        JR T1                   ;..else side 0

;Convert logical sector to physical
CONVS:  LD A,(LSECT)
        CP (IX+SIDSEC-IXPTR)    ;Test for change with sector
        JR C,CS1                ;Side 0 or not SECCYL
        SUB (IX+SIDSEC-IXPTR)   ;Take off side 0
        CALL SET2               ;And set up for side 1
CS1:    LD E,A
        LD HL,(SKTAB)
        LD A,H
        OR L                    ;Test for skew table
```

```
        LD A,E
        JR Z,CS2                ;No logical skew
;HL points at skew table

;       LD A,(IX+TKSKEW-IXPTR)
;       OR A                    ;Test for track skew
;       JR Z,CS0                ;No track skew
;Track skew, convert sector to track skewed, before applying sector skew
;       LD A,(PTRAK)
;       OR A
;       JR Z,CS0                ;Track 0
;       LD B,A                  ;Track count
;NXTT:  LD A,E
;       ADD A,(IX+TKSKEW-IXPTR)
;       LD E,A
;       SUB (IX+SIDSEC-IXPTR)   ;Test for overflow
;       JR C,NOVF               ;No overflow
;       LD E,A                  ;Update E
;NOVF:  DJNZ NXTT

CS0:    LD D,0
        ADD HL,DE               ;Add in to table
        LD A,(HL)               ;Skewed sector
CS2:    BIT 2,(IX+YNOPT-IXPTR)  ;Test for sector offset
        JR Z,CS3
        INC A
CS3:    LD (PSECT),A
        BIT 6,(IX+YNOPT-IXPTR)  ;Test for sector carry on side 1
        RET Z
;Physical sector numbers on side 1 carry on from side 0
        LD HL,DRVLOG
        BIT 7,(HL)              ;Test hard side flag
        RET Z                   ;Side 0 anyway
        ADD A,(IX+SIDSEC-IXPTR) ;Add in side 0
        LD (PSECT),A
        RET

;Set side 1 flags
SET2:   LD HL,DRVLOG
        SET 7,(HL)              ;Hard side select
        IF MPILNK               ;If MPI is linked for hard side select
        BIT 3,(IX+YNOPT-IXPTR)  ;..don't set soft side select
        RET NZ                  ;..if bad side 2 flag
        ENDIF
        LD HL,COMM
        SET 1,(HL)
        RET

;Floppy read/write
FRDWR:  LD HL,DRVLOG
        RES 7,(HL)              ;Reset hard side flag
        LD HL,COMM
        RES 1,(HL)              ;Reset soft side flag
;First convert Logical to Physical
        CALL CONVT              ;Convert logical track to physical
        CALL CONVS              ;Convert logical sector to physical
;Read/write physical with retries
        LD B,8                  ;Up to 8 retries
NEXTRY: LD A,(DRVLOG)           ;Physical drive code
        OUT (DRVPRT),A          ;Keep them motors running, select side
        PUSH BC                 ;Save retry counter
        LD A,(PTRAK)
        CALL SEEK               ;Get to required track
;Head is at the right place
        LD A,(PSECT)
        OUT (FDCSEC),A          ;Select sector
        CALL WAITON             ;Wait until drives ready
        JR C,DERR1              ;Drives have timed out
        LD C,STAPRT
        LD HL,(DMAADR)          ;Sector load address
        CALL RWFLOP             ;All ready so do read/write in common memory
        POP BC                  ;Retry counter
        IN A,(FDCSTA)
        AND 11011111B           ;Record type flag is not an error during read
        RET Z                   ;All OK
```

```
;Head is at the right place
        LD A,(PSECT)
        OUT (FDCSEC),A          ;Select sector
        CALL WAITON             ;Wait until drives ready
        JR C,DERR1              ;Drives have timed out
        LD C,STAPRT
        LD HL,(DMAADR)          ;Sector load address
        CALL RWFLOP             ;All ready so do read/write in common memory
        POP BC                  ;Retry counter
        IN A,(FDCSTA)
        AND 11011111B           ;Record type flag is not an error during read
        RET Z                   ;All OK
        JP M,DERR               ;Motor off so don't retry
        PUSH AF                 ;Save error code
        LD A,B
        CP 4
        CALL Z,TRACK0           ;Restore head on 4th retry
        POP AF
        DJNZ NEXTRY
        JR DERR                 ;Tries exhausted

;Handle floppy error
DERR1:  POP BC                  ;Clear retry count
DERR:   CALL SETERR             ;Display error to user and re-set up
        RET NZ                  ;Return error to BDOS
        JR FRDWR                ;Full retry

;Display error and prompt for retry
SHWERR: LD HL,DNR               ;Find approp. message
        RLCA
        JR C,FFOUND
        LD HL,DWP
        RLCA
        JR C,FFOUND
        LD HL,WF
        RLCA
        JR C,FFOUND
        LD HL,RNF
        RLCA
        JR C,FFOUND
        LD HL,CRC
        RLCA
        JR C,FFOUND
        RLCA
        JR NC,NDEFF
        XOR A
        RET                     ;Auto retry lost data
NDEFF:  LD HL,NDWF              ;Shouldn't happen
FFOUND: CALL GOTMSG             ;Print error and track/sector message
        CALL CLEARF             ;Clear the 2797, drive and track logs
;Prompt for retry
        LD HL,RETRY             ;Prompt reply
        CALL PMSG
WRONG:  CALL ?CONIN             ;Get reply
        LD C,A
        AND 5FH                 ;Ensure u/c
        CP "Y"
        JR Z,RIGHT
        CP "N"
        JR NZ,WRONG
        DEC A                   ;Make NZ
RIGHT:  PUSH AF                 ;Save flag Z="Y" NZ="N"
        CALL ?CONOT
        LD HL,CRLF              ;CRLF
        CALL PMSG
        POP AF                  ;Z means retry
        LD A,1                  ;In case error to be returned to BDOS
        RET
```

```
;Print Error, Drive, Track and Sector
GOTMSG: PUSH HL                 ;Save error type
        LD HL,ERMSG
        CALL PMSG               ;Error header
        POP HL
        CALL PMSG               ;Print message
TSMSG:  LD HL,ONMSG
        CALL PMSG
        LD A,(LDRIV)            ;Get drive
        ADD A,"A"
        LD C,A
DRVINC: CALL ?CONOT             ;Drive code
        LD HL,TRKMSG
        CALL PMSG               ;Track header
        LD HL,(LTRAK)
        CALL PDEC               ;Track number
        LD HL,SECMSG
        CALL PMSG               ;Sector header
        LD HL,(LSECT)
        CALL PDEC               ;Sector number
        RET

;Return NC if ready
WAITON: LD DE,2500              ;Motor start up allowance (2.5secs)
WAIT1:  IN A,(FDCSTA)
        RLCA                    ;Look at ready bit
        RET NC                  ;Ready
        LD A,1                  ;1 ms delay
        CALL DELAY
        DEC DE                  ;Decrement time out
        LD A,D
        OR E
        JR NZ,WAIT1             ;Keep waiting
        LD A,80H                ;Not ready error
        SCF                     ;Signal error
        RET

;Restore the head and initialise track map
TRACK0: XOR A
;Seek track in A
SEEK:   PUSH BC
        LD (IX+TRKA-IXPTR),A    ;Update track map
        OR A                    ;Test for track 0
        LD A,(DRVLOG)
        PUSH AF                 ;Save drive for exit
        SET 5,A                 ;2 Mhz for step
        OUT (DRVPRT),A
;Ensure we get to track 0 from possible unknown or error
        LD BC,STPOUT            ;B = 256, C = step out command
        JR Z,STEP0              ;Track 0
;Get to required track using STEP IN or STEP OUT commands from known start
        IN A,(FDCTRK)           ;Current track
        SUB (IX+TRKA-IXPTR)     ;Subtract required
        JR Z,THERE
        LD C,STPOUT
        JP P,GOSTEP             ;Step out if required track is less
        LD C,STEPIN             ;Else step in
        NEG                     ;Twos complement
GOSTEP: LD B,A                  ;Number of steps
        BIT 0,(IX+YNOPT-IXPTR)
        JR Z,DOSTEP
        SLA B                   ;Double steps
        JR DOSTEP

;Get a type I status
STEP0:  CALL LOADH
STEP1:  IN A,(FDCSTA)
        AND 00000100B           ;Test for track 0
        JR NZ,STPDON
DOSTEP: LD A,C                  ;Get command
        OUT (FDCCOM),A          ;Issue it
```

```
        SLA B                   ;Double steps
        JR DOSTEP

;Get a type I status
STEP0:  CALL LOADH
STEP1:  IN A,(FDCSTA)
        AND 00000100B           ;Test for track 0
        JR NZ,STPDON
DOSTEP: LD A,C                  ;Get command
        OUT (FDCCOM),A          ;Issue it
        LD A,(IY+STPSPD-IYPTR)  ;Step delay
        CALL DELAY
        CALL WAITNB             ;Make sure 2797 has finished
        DJNZ STEP1

STPDON: LD A,(IX+TRKA-IXPTR)
        OUT (FDCTRK),A          ;Update track register
        LD A,(IY+SETTL-IYPTR)
        CALL DELAY              ;Settling time
THERE:  POP AF
        OUT (DRVPRT),A          ;Back to original clock
        POP BC
        RET

;Load heads
LOADH:  OUT (FDCTRK),A          ;Track and
        OUT (FDCDAT),A          ;...Data regs to any value
        LD A,SEKTRK             ;Seek to current track
        JR SENCOM               ;Do command

;Clear the track map and 2797
CLEARF: LD HL,FMTLOG
        LD (HL),0FFH
        INC HL
        INC HL                  ;On to DRVLOG
        LD B,NDRVF+1
INITRK: LD (HL),0FFH            ;Initialise track map and drive log
        INC HL
        DJNZ INITRK
;Clear the 2797
CLEAR:  LD A,CLRFDC             ;Load the command

;Send command to 2797 and wait til it's done
SENCOM: OUT (FDCCOM),A          ;Send command
        LD A,10
WAIT:   DEC A
        JR NZ,WAIT
WAITNB: IN A,(FDCSTA)
        RRCA                    ;Wait til finished
        JR C,WAITNB
        RET

;Print AF
B2HEX:  PUSH AF
        RRCA                    ;Find error type
        RRCA
        RRCA
        RRCA
        CALL B1HEX
        POP AF
B1HEX:  AND 0FH
        ADD A,90H
        DAA
        ADC A,40H
        DAA
        LD C,A
        JP ?CONOT

;Print binary number 0-65535 from <HL>
PDEC:   XOR A
        LD (LDG0),A
```

```
            LD BC,TBL10
            LD DE,-10000
NEXT:       LD A,"0"-1
PDECL:      PUSH HL
            INC A
            ADD HL,DE
            JR NC,STOPL
            INC SP
            INC SP
            JR PDECL
STOPL:      PUSH DE
            PUSH BC
            LD C,A
            CP "0"
            JR NZ,NOLDG
            LD A,(LDG0)
            OR A
            JR Z,ISLDG
NOLDG:      LD (LDG0),A
            CALL ?CONOT
ISLDG:      POP BC
            POP DE
NEXDIG:     POP HL
            LD A,(BC)
            LD E,A
            INC BC
            LD A,(BC)
            LD D,A
            INC BC
            LD A,E
            OR D
            JR NZ,NEXT
            LD A,(LDG0)
            OR A
            RET NZ
            LD C,"0"
            JP ?CONOT

TBL10:      DEFW -1000,-100,-10,-1,0
LDG0:       DEFB 0

            SUBTTL Common Floppy read/write
            PAGE
            CSEG                    ;Actual read/write must be common

COMM:       DEFB 0                  ;FDC command for read/write with flags

;Read/write floppy
RWFLOP:     LD A,(DMABNK)           ;Select DMA bank
            CALL BANK
            LD A,(RWFLAG)
            OR A                    ;Read or write ?
            JR Z,WRTOP
            LD A,(COMM)             ;Get command
            OUT (FDCCOM),A          ;Send command
            JR RDTEST

DATIN:      IN A,(FDCDAT)           ;Read byte
            LD (HL),A               ;Store buffer byte
            INC HL
RDTEST:     IN B,(C)                ;Get flags
            JR Z,RDTEST             ;Nothings happening
            JP M,DATIN              ;DRQ
            JR RWDONE               ;Finished, either IRQ or motor off

WRTOP:      CALL INVERT             ;Invert the data if necessary
            LD A,(COMM)             ;Get command
            OUT (FDCCOM),A          ;Send command
DATOUT:     LD A,(HL)               ;Get buffer byte
            INC HL
WRTEST:     IN B,(C)                ;Get flags
            JR Z,WRTEST             ;Nothings happening
            OUT (FDCDAT),A          ;Send byte
            JP M,DATOUT             ;DRQ
            DEC HL                  ;Back to end of buffer
            LD A,2
            CALL DELAY              ;Delay 2 ms after write
;Reselect page 0
```

```
        JP M,DATIN              ;DRQ
        JR RWDONE               ;Finished, either IRQ or motor off

WRTOP:  CALL INVERT             ;Invert the data if necessary
        LD A,(COMM)             ;Get command
        OUT (FDCCOM),A          ;Send command
DATOUT: LD A,(HL)               ;Get buffer byte
        INC HL
WRTEST: IN B,(C)                ;Get flags
        JR Z,WRTEST             ;Nothings happening
        OUT (FDCDAT),A          ;Send byte
        JP M,DATOUT             ;DRQ
        DEC HL                  ;Back to end of buffer
        LD A,2
        CALL DELAY              ;Delay 2 ms after write
;Reselect page 0
RWDONE: CALL INVERT             ;Invert the data if necessary
        XOR A
        CALL BANK
        RET

SECSZE: DEFB 4

;Invert data if reqd
INVERT: LD A,(SECSZE)
        OR A
        RET P                   ;Not required
        PUSH HL
        PUSH BC
        AND 7FH                 ;Strip inverse bit
        LD B,A                  ;Get physical sector size
        LD C,0
        SRL B
        RR C
        LD HL,(DMAADR)          ;DMA address
CPLOOP: LD A,(HL)
        CPL
        LD (HL),A
        CPI                     ;INC HL, DEC BC
        JP PE,CPLOOP
        POP BC
        POP HL
        RET

;Delay for (A) ms
DELAY:  OR A                    ;Return on 0 delay
        RET Z
        PUSH BC
        LD B,A                  ;Ms count to B
DLY1:   LD A,MSCNT
WTLP:   DEC A                   ;Main count
        JR NZ,WTLP
        DJNZ DLY1               ;Ms count
        POP BC
        RET

;Disk communication data items
LTRAK:  DEFS 2                  ;~ Logical track number
LSECT:  DEFS 2                  ;~ Logical sector number
PTRAK:  DEFS 1                  ;Physical track
PSECT:  DEFS 1                  ;Physical sector

DMAADR: DEFS 2                  ;Current DMA address
MULCNT: DEFW 0                  ;Count for multisector transfer and countdown
DMABNK: DEFB 0                  ;Bank for DMA operations
RWFLAG: DEFB 0                  ;0 = Write, 1 =Read
SKTAB:  DEFW 0                  ;Skew table address

        SUBTTL Disk messages
        PAGE
```

```
        DSEG

;Error message components
ERMSG:  DEFB 7
CRLF:   DEFB CR,LF,Ø
ONMSG:  DEFB " on ",Ø
TRKMSG: DEFB ": Track- ",Ø
SECMSG: DEFB " Sector- ",Ø

DNR:    DEFB "Drive Not Ready",Ø
DWP:    DEFB "Write Protect",Ø
WF:     DEFB "Write Fault",Ø
RNF:    DEFB "Record Not Found",Ø
CRC:    DEFB "CRC Fault",Ø
NDWF:   DEFB "????",Ø
RETRY:  DEFB " - Retry (Y/N)? ",Ø

CLEAN:  CALL CLEARF             ;Clean up 2797 and track map

        IF DRVW
        CALL CLEANW             ;Set up SHD controller
        ENDIF

        RET
```