HDBL
Hard Disk Boot Loader User's Manual

Martin Eberhard

13 August 2014

Revision History

Rev	Date	Author	Notes
1.00	15SEP2013	M. Eberhard	Created
1.01	25MAR2014	M. Eberhard	Fixed alternate platter load bug
1.02	26MAR2014	M. Eberhard	Platter number in signon, jump to XENTER when done, so message completes
1.03	05JUN2014	M. Eberhard	Use sense switch A11 to select platter. Don’t allow selecting unit.
2.00	13AUG2014	M. Eberhard	Turnkey Module compatibility: copy code to RAM before running.

INTRODUCTION

HDBL is a 256-byte PROM program that loads the boot file from either platter of an 88-HDSK hard disk, and executes the successfully loaded code. Progress and error messages are printed on a "standard" 6850-based Altair Terminal port, such as port A of an 88-2SIO, the serial port on a Turnkey Module, or the serial port of an 88-UIO.

CONTENTS

1. INSTALLATION

The HDBL PROM must be installed at address 176000 octal. It is inserted in slot E of an 88-PMC memory card (with its base address at 174000 octal), or in slot L1 of a Turnkey Module.

2. SYSTEM REQUIREMENTS

The standard build of HDBL requires at least 48K of RAM in the Altair, because its stack and relocated code is located in the 256-byte page that begins at address BF00h.

The 88-HDSK controller must be installed at the I/O standard addresses, 280 octal through 287 octal (A0h through A7h).

HDBL expects a Motorola 6850 ACIA-type device as its Terminal. This can be one of the following devices: an 88-2SIO (port A), an 8800b Turnkey Module, or an 88-UIO. This I/O device must be addressed at address 020 and 021 octal (10h and 11h).

3. SELECTING THE BOOT PLATTER

MITS’s software only boots from the removable cartridge of the disk drive. HDBL allows you to boot also from the fixed platter. You can select from which platter to boot using Sense Switch 3[1] (labeled A11 on an Altair front panel). If this switch is down (0), then HDBL will boot from the removable cartridge. If this switch is up (1), then HDBL will load from the fixed platter.

[1] This particular switch was chosen to minimize collisions with the normal use of sense switches <A11:A8>. 0000 through 0110 select the input device for MBL and for the various checksum loaders on tape (and values above 0110 will generate an error). So if you have the sense switches (which may be inaccessible on an 8800b Turnkey board) set for loading from e.g. cassette, then the machine will still boot from the removable cartridge when booting via HDBL.

4. <u>OPERATING PROCEDURES</u>

5.1 Start the Computer

1. Power on or Reset the Altair.
2. If you have TURMON or UBMON, then start this program, if the autostart did not do so already.
3. If you do not have TURMON or UBMON or equivalent:
   a. STOP and RESET the Altair.
   b. EXAMINE address 176000 octal

5.1 START THE HARD DISK

1. Power on the disk controller
2. Power on the disk drive
3. When the SAFE light comes on, press the RUN switch on the disk drive, and wait for the READY light to come on.

5.3 INITIATING THE LOAD

1. Select the boot platter by setting Sense Switch 3
   a. Set Switch 3 down to boot from the removable cartridge
   b. Set Switch 3 up to boot from the fixed platter
2. Start HDBL
   a. If you are using TURMON, type “J176000”.
   b. If you are using UBMON, type “L”
   c. If you are starting with a front panel, press the RUN switch.

5.4 COMPLETING THE LOAD

1. HDBL will immediately announce its revision number.

2. HDBL will announce “LOADING FROM {0 or 1}”, where 0 is the removable cartridge, and 1 is the fixed platter. This message is printed after HDBL has successfully loaded the Pack Descriptor Page from the hard disk. The Pack Descriptor page (track 0, side 0, sector 0) informs HDBL the starting load sector number, and the total number of sectors to load.

3. Loading will continue until the specified number of sectors has been loaded, or until an error occurs. HDBL does not retry on errors.

5. ERROR INDICATIONS

The Interrupt Enable light remains off if loading is proceeding properly. If an error occurs, the Interrupt Enable light comes on and an error message is printed on the Console. The 88-HDSK error code is stored in address 0000, and then the Altair will hang in an infinite loop. (You can reset the Altair, and then examine this location.)

The 88-HDSK error code is an 8-bit value, with each bit indicating a different error when set. These bits are interpreted as follows:

Bit	Meaning
0	Drive not ready
1	Illegal sector requested
2	CRC error in sector data
3	CRC error in sector header
4	Wrong sector detected
5	Wrong cylinder detected
6	Wrong head detected
7	Write protect (not an error)

See the 88-HDSK documentation for further details.

6. 8800b TURNKEY MODULE COMPATIBILITY

The 8800b Turnkey Modules (except for Rev 0 Turnkey Modules that do not have the 88-SYS-CLG mods) disable PROM and enable RAM whenever software inputs from the Sense Switch port. (Some versions of this board will disable the PROMs when any IN instruction is executed!)

This means that any PROM that reads the Sense switches will not work with the Turnkey Module - and any PROM that ever uses the IN instruction will fail with some Turnkey Modules.

Starting with version 2.00, HDBL will work with all Turnkey Modules, because it relocates its own code to RAM before ever reading the Sense Switches. Code is relocated to the same page of memory that HDBL uses for its stack, starting at address BF00h in the standard build. (This requires at least 48K of RAM in the Altair.)

APPENDIX A - DISK STRUCTURE

There are 24 256-byte sectors per track, and these are numbered 0 through 23 on each track. Each platter has 2 sides, numbered 0 and 1. Data on each platter is organized as a sequence of Disk Pages, where each Page is one sector. Pages are numbered sequentially starting at 0 (on track 0, side 0), through the 24 sectors on track 0, side 0, and then on to track 0, side 1, where sector 0 is page 24. Page 47 is the first sector on track 1, side 0, and page numbering continues this way through all the tracks.

Page 0 (which is track 0, side 0, sector 0) is the Pack Descriptor Page, containing various information about the particular disk platter. Bytes 40-43 of this Page are the "Opsys Pointers." Bytes 40 & 41 are the page number of the starting boot page, Bytes 42 & 43 are the number of pages to load during boot. HDBL assumes that the boot file is to be loaded into memory starting at address 0000.

APPENDIX B - SOURCE CODE LISTING

The following pages list the source code for HDBL. This code was assembled using Digital Research’s ASM assembler. As such, all values are in hexadecimal, rather than in octal as is normal for MITS software.

```
;=================================================================
; Hard Disk Boot Loader (HDBL)
; By Martin Eberhard
;
; HDBL is a 256-byte PROM program that loads the boot file from
; an 88-HDSK hard disk, and executes the successfully loaded
; code. Progress and error messages are printed on a "standard"
; 6850-based Altair Terminal port at address 10H and 11h, such
; as port A of an 88-2SIO, the serial port on a Turnkey Module,
; or the serial port of an 88-UIO.
;
; The standard 88-HDSK system uses a Pertec D3422 disk drive,
; which contains 2 platters - one is in a removable cartridge,
; the other is a fixed platter. However, The 88-HDSK controller
; can actually support up to 4 platters, supporting the Pertec
; D3462 disk drive, which has one removable platter, and 3
; fixed platters.
;
; Altair software normally boots only from the removable
; cartridge of a D3422 disk drive. HDBL allows you to boot also
; from the fixed platter. You can select from which platter to
; boot using Sense Switch 3 (A11 on the front panel). 0 (down)
; selects the removable cartridge, 1 (up) selects the fixed
; platter. This switch was chosen to minimize collisions with
; the normal use of sense switches <A11:A8>. 0000 through 0110
; select the input device for MBL and for the various checksum
; loaders on tape (and values above 0110 will generate an
; error). So if you have the sense switches (which may be
; inaccessible on an 8800b Turnkey board) set forloading from
; e.g. cassette, then the machine will still boot from the
; removable cartridge when booting via HDBL.
;
; There are 24 256-byte sectors per track, and these are
; numbered 0 through 23 on each track. Each platter has 2
; sides, numbered 0 and 1. Data on each platter is organized as
; a sequence of Disk Pages, where each Page is one sector.
; Pages are numbered sequentially starting at 0 (on track 0,
; side 0), through the 24 sectors on track 0, side 0, and then
; on to track 0, side 1, where sector 0 is page 24.  Page 47 is
; the first sector on track 1, side 0, and page numbering
; continues this way through all the tracks.
;
; Page 0 (which is track 0, side 0, sector 0) is the Pack
; Descriptor Page, containing various information about the
; particular disk platter. Bytes 40-43 of this Page are the
; "Opsys Pointers." Bytes 40 & 41 are the Page number of the
; starting boot Page, Bytes 42 & 43 are the number of Pages to
; load during boot. HDBL assumes that the boot file is to be
; loaded into memory starting at address 0000, and executed
; there.
;
; During loading, the INTE (Interrupt Enabled) LED on the front
; panel will be off. Any error during loading will cause the
; INTE LED to light and a "LOAD ERR" message to be printed
; on the Terminal. The error code is stored in memory at
; address 0. HDBL will then hwng in a loop until Reset.
;
; Because HDBL may be running stand-alone, the Terminal port
; gets initialized during HDBL initialization. But if control
; came from UBMON (with an "L" command) or from TURMON (with a
; "J 176000" command) then the Terminal port's ACIA will still
```

```
                ; be transmitting the last two character-echos of the command
                ; when HDBL begins. HDBL will stall 6.5 uS (one character time
                ; above 1800 baud) before initializing the Terminal port, so
                ; that the ACIA will have time to finish transmitting this last
                ; characters before it gets reset.
                ;
                ; Newer Turnkey Modules (and older ones with the 88-SYS-CLG
                ; modification) will disable the PROMs when an IN instruction
                ; reads port FFh (the Sense Switches). Some of these boards
                ; will disable the PROM when *any* IN instruction is
                ; executed. For this reason, HDBL copies itself to RAM, and
                ; runs from there, before executing any IN instructions.
                ;
                ; This code is written to assemble with ASM by Digital Research
                ;
                ;===============================================================
                ; Revision History
                ;   1.00  15SEP2013  M.Eberhard
                ;     Created
                ;   1.01  25MAR2014  M.Eberhard
                ;     Fixed bug with loading from alternate platters, code
                ;     squeeze, improve comments
                ;   1.02  26MAR2014  M.Eberhard
                ;     Further compression. Print platter number in signon msg.
                ;     Jump to XENTER instead of XMON on error, so ACIA reset
                ;     doesn't hose the last chr of error code.
                ;   1.03  05JUN2013  M.Eberhard
                ;     Use sense switch A11 to select boot platter. Eliminate
                ;     selection of boot drive.
                ;   2.00  15AUG2014 M.Eberhard
                ;     Copy code to RAM before execution, for Turnkey Module
                ;     compatibility. Remove RAMCOD option. Eliminate return
                ;     to UBMON/TURMON on load error (Must reset on error.)
                ;
                ; (Remember to update the Version String below)
                ;===============================================================
0000 =          FALSE   equ     0
FFFF =          TRUE    equ     not FALSE

                ; PROM address and Entry point for HDBL. UBMON assumes FC00h.

FC00 =          HDBL    equ     0FC00h  ;Beginning of HDBL PROM

                ; RAM address for moved code. Exactly one of these
                ; should be used.

BF00 =          RAMPAG  equ     0BF00h  ;beginning of RAM page (48K system)
                ;RAMPAG equ     0F700h  ;beginning of RAM page (62K system)
                ;RAMPAG equ     0FB00h  ;beginning of RAM page (63K system)
                                        ;(e.g. Turnkey Module's RAM)

3D00 =          ROF     equ     HDBL-RAMPAG     ;RAM relocation offset

                ; Sense Switch assignment for selecting the boot platter & unit

00FF =          SSWTCH  equ     0FFh            ;Sense Switch address
0008 =          PSWTCH  equ     008h            ;mask for platter Switches
                                                ;Code assumes 008h (bit 3)

                ; Terminal port equates - same for 88-2SIO port 0, Turnkey
                ; Module, and 88-UIO (all based on the Motorola 6850 ACIA)
                ; Note: transmitting with 2 stop bits is also compatible with a
                ; receiver that is programmed for 1 stop bit.
```

```
0010 =          ACCTRL  EQU     10h             ;ACIA Control output port
0010 =          ACSTAT  EQU     10h             ;ACIA Status input port
0011 =          ACTXD   EQU     11h             ;ACIA TX Data register
0011 =          ACRXD   EQU     11h             ;ACIA RX Data register

0003 =          ACRSET  EQU     00000011b       ;Master reset
0001 =          ACRDF   EQU     00000001b       ;RX Data register full
0002 =          ACTDE   EQU     00000010b       ;TX Data register empty
0011 =          ACINIT  EQU     00010001b       ;/16, 8-bit, No Parity, 2Stops

                ; 88-HDSK ports (The interface board is actually an 88-4PIO.)

00A0 =          CREADY  equ     0A0h    ;IN: Ctlr ready for command (bit7)
00A1 =          CSTAT   equ     0A1h    ;IN: error flags, reset CREADY
00A2 =          ACSTA   equ     0A2h    ;IN: Command Ack (bit 7)
00A3 =          ACMD    equ     0A3h    ;IN: reset Command Ack
                                        ;OUT: Command high byte/initiate
00A4 =          CDSTA   equ     0A4h    ;IN: data/stat availablr at CDATA
00A5 =          CDATA   equ     0A5h    ;IN: Disk data or status from Ctlr
00A6 =          ADSTA   equ     0A6h    ;IN: ADATA Port Available (bit 7)
00A7 =          ADATA   equ     0A7h    ;OUT: Command low byte

                ; 88-HDSK ACMD:ADATA Commands

0000 =          CSEEK   equ     00h     ;Bits 15:12 = 0000b
                                        ;Bits 11:10 = Unit #
                                        ;Bits  9:0  = Cylinder #

0030 =          CRDSEC  equ     30h     ;Bits 15:12 = 0011b
                                        ;Bits 11:10 = Unit #
                                        ;Bits  9:8  = Buffer #
                                        ;Bit   7:6  = Platter #
                                        ;Bits    5  = Side #
                                        ;Bits  4:0  = Sector #

0020 =          CSIDE    equ      020h    ;Side select for CRDSEC
00C0 =          CFPLTR   equ      0C0h    ;platter mask for CRDSEC
000C =          CUNIT    equ      00Ch    ;Unit mask for CSEEK & CRDSEC

0050 =          CRDBUF  equ     50h     ;Bits 15:12 = 0101b
                                        ;Bits 11:10 = not used
                                        ;Bits  9:8  = buffer #
                                        ;Bits  7:0  = # bytes to transfer
                                        ;(00 means 256)

                ; 88-HDSK CSTAT error bits

0001 =          ERDNR   equ     01h     ;drive not ready
0002 =          ERBADS  equ     02h     ;illegal sector
0004 =          ERSCRC  equ     04h     ;CRC error during sector read
0008 =          ERHCRC  equ     08h     ;CRC error during header read
0010 =          ERSWRG  equ     10h     ;header has wrong sector
0020 =          ERCWRG  equ     20h     ;header has wrong cylinder
0040 =          ERHWRG  equ     40h     ;header has wrong head
0080 =          WPROT   equ     80h     ;Write Protect
007F =          ERMASK  equ     7Fh     ;all the actual error bits

                ; 88-HDSK Constants

0028 =          OSOFF   equ     40      ;Page 0 offset to opsys pointers
0018 =          HDSPT   equ     24      ;Sectors per track
0000 =          DBUFR   equ     0       ;Default controller buffer: 0-3
```

```
                                              ;Code gets longer if <>0

               ; ASCII characters

000D =         CR      equ     0Dh
000A =         LF      equ     0Ah

               ;==============================================================
               ; Start of HDBL PROM
FC00                   org     HDBL
               ;==============================================================
FC00 F3                di                      ;front panel INTE light off

               ;---------------------------------------------------------
               ; Copy code to RAM. This will provide 6.5 mS of delay.
               ; This will allow a UART that is running at 1700 baud or
               ; faster to complete transmission. (This also leaves 18
               ; bytes in RAM for the stack.)
               ;---------------------------------------------------------
FC01 2100C0            lxi     h,RAMPAG+100h           ;last RAM address+1
FC04 16FC              mvi     d,(HDBL/256)            ;PROM code page

FC06 2B        COPLUP: dcx     h               ;(5+1)
FC07 5D                mov     e,l             ;(4+1)
FC08 1A                ldax    d               ;(7+1)
FC09 77                mov     m,a             ;(7+1)
FC0A 7D                mov     a,l             ;(4+1)
FC0B D612              sui     RAMCOD and 0FFh ;(7+2)ends with a=0
FC0D C206FC            jnz     COPLUP          ;(10+3)

               ;54 cycles per pass X (256-18) /2 = 6.426 mS

               ;------------------------------------------------------
               ; Set up system stack immediately below RAM code image
               ;------------------------------------------------------
FC10 F9                sphl

               ;----------------------------
               ;go to loaded code (with a=0)
               ;----------------------------
FC11 E9                pchl

               ;=============================================================
               ; All of the following code gets copied to RAM and run there.
               ; On Entry:
               ;   a = 0
               ;=============================================================
FC12 67        RAMCOD: mov     h,a             ;set load initial page
FC13 6F                mov     l,a             ;hl=0

               ;----------------------------------------
               ; Initialize 88-HDSK interface board
               ; (Actually ports 0 and 1 of an 88-4PIO)
               ; On Entry:
               ;   a = 0
               ;  hl = 0
               ; On Exit:
               ;  hl = 0
               ;----------------------------------------
FC14 D3A0              out     0A0h            ;Select port 0Ah DDR
FC16 D3A2              out     0A2h            ;Select port 0Bh DDR
FC18 D3A4              out     0A4h            ;Select port 1Ah DDR
FC1A D3A6              out     0A6h            ;Select port 1Bh DDR
```

```
FC1C D3A1               out     0A1h            ;Port 0Ah is an input port
FC1E D3A5               out     0A5h            ;Port 1Ah is an input port

FC20 2F                 cma
FC21 D3A3               out     0A3h            ;Port 0Bh is an output port
FC23 D3A7               out     0A7h            ;Port 1Bh is an output port

FC25 3E2C               mvi     a,2Ch           ;set up input port handshakes
FC27 D3A0               out     0A0h
FC29 D3A4               out     0A4h
FC2B D3A6               out     0a6h            ;output port 1Bh handshakes

FC2D 3E24               mvi     a,24h           ;set up port 0Bh handshakes
FC2F D3A2               out     0A2h

FC31 DBA1               in      CSTAT           ;clear Controller Ready bit

               ;---------------------------------------------
               ; Reset and initialize the Terminal port ACIA
               ; On Entry & Exit:
               ;  hl = 0
               ;---------------------------------------------
FC33 3E03               mvi     A,ACRSET
FC35 D310               out     ACCTRL
FC37 3E11               mvi     A,ACINIT
FC39 D310               out     ACCTRL

               ;----------------------------
               ; Print HDBL version message
               ; On Entry & Exit:
               ;  hl = 0
               ;----------------------------
FC3B CDE5BF             call    PRINTF-ROF      ;print the following string
FC3E 0D0A484442         db      CR,LF,'HDBL 2.0','0'+80h

               ;-------------------------------------------------
               ; Read the Pack Descriptor Page (Disk Page 0)
               ; to get the Opsys Pointers:
               ;   Bytes 41:40 = Initial Disk Page number
               ;   Bytes 43:42 = Disk Page count (Byte 43=MSB=0)
               ; On Entry:
               ;  hl = 0
               ;-------------------------------------------------
FC49 062B               mvi     b,OSOFF+3       ;byte count to end of pointers
FC4B CD82BF             call    GETPAG-ROF      ;Seek, read page hl into buffer
                                                ;set up to read b buffer bytes

FC4E E5                 push    h               ;execution address on stack
FC4F EB                 xchg                    ;load address into de

               ; Read from the controller buffer and discard everything until
               ; we get to the opsys pointers. Load the opsys pointers into
               ; C & HL. Note: no testing any handshake here - just assume
               ; the controller can keep up. (The controller can send a data
               ; byte every 2.5 uS.) This only reads the low byte of the
               ; page count, since the high byte must be 0 anyway.

FC50 DBA5      PTRLUP:  in      CDATA           ;read byte from controller

FC52 6C                 mov     l,h             ;shift everybody over...
FC53 61                 mov     h,c
FC54 4F                 mov     c,a             ;...and put it away
```

```
FC55 05                 dcr     b
FC56 C250BF             jnz     PTRLUP-ROF

               ; Announce 'LOADING FROM <platter>' on the Terminal

FC59 CDDCBF             call    LOADPF-ROF       ;CR,LF,'LOAD', then string
FC5C 494E472046         db      'ING FROM',' '+80h

FC65 DBFF               in      SSWTCH           ;read platter switch
                                                 ;(disables PROMS in Turnkey bd)
FC67 E608               ani     PSWTCH           ;mask off all others
FC69 0F                 rrc
FC6A 0F                 rrc
FC6B 0F                 rrc
FC6C C630               adi     '0'              ;make it ASCII
FC6E CDF3BF             call    PRINTA-ROF       ;and print it

               ;-----------------------------------------------
               ; Read c Pages from disk, starting at Page hl, into
               ; memory starting at the address on the stack
               ; On Entry:
               ;   b = 0
               ;   c = page count
               ;  de = LDADDR
               ;  hl = initial Disk page number
               ;-----------------------------------------------
FC71 CD82BF    PAGELP: call     GETPAG-ROF       ;Seek, read page hl into buffer
                                                 ;set up to read b buffer bytes
                                                 ;b=0 here always.

               ; Load 256 bytes of buffer data into memory at de (b=0 here)
               ; Note: no testing any handshake here - just assume the
               ; controller can keep up. (The controller can send a data byte
               ; every 2.5 uS.)

FC74 DBA5      BYTELP: in       CDATA            ;get a data byte
FC76 12                stax     d                ;write it to RAM
FC77 13                inx      d                ;next address
FC78 05                dcr      b                ;bump byte counter
FC79 C274BF            jnz      BYTELP-ROF       ;until done (b=0)

               ; Next Disk Page

FC7C 23                inx      h                ;Next Disk Page
FC7D 0D                dcr      c                ;bump Disk Page count
FC7E C271BF            jnz      PAGELP-ROF

               ;--------------------------------------------------
               ; Go execute loaded code, at the address on the stack
               ;--------------------------------------------------
FC81 C9                ret

               ;===Subroutine============================================
               ; Seek and read disk Page hl into 88-HDSK buffer 0
               ; On Entry:
               ;   b=number of bytes to transfer (0 means 256)
               ; On Exit:
               ;   a,flags trashed, all others preserved
               ;   Controller has specified sector data in its buffer
               ;=========================================================
FC82 E5        GETPAG: push     h                ;Save requested Page
FC83 D5                push     d                ;Save regs
FC84 C5                push     b                ;save byte count
```

```
               ;------------------------------------------------------------
               ; Compute cylinder and sectorX2  from Disk Page number in hl
               ;  hl := hl / (2*HDSPT) (Quotient=cylinder)
               ;   h := hl MOD (2*HDSPT) (Remainder=sectorx2)
               ; This is fast only if the cylinder number is low. MITS
               ; usually put the boot image starting at cylinder 0, side 1,
               ; so this will be faster and shorter than the 'fast'
               ; division of previous HDBL rev. This will become slower
               ; if the boot image is above cylinder 20 or so. But we will
               ; always miss the next sector anyway, so each sector will
               ; require a full disk rev (25 mS), plenty of time
               ;------------------------------------------------------------
FC85 01D0FF            lxi     b,-2*HDSPT
FC88 50                mov     d,b             ;de=FFFF=-1
FC89 58                mov     e,b             ;since loop goes 1 extra

FC8A 13        DIV1:   inx     d               ;compute quotient=cylinder
FC8B 09                dad     b               ;hl gets remainder
FC8C DA8ABF            jc      DIV1-ROF

FC8F 7D                mov     a,l             ;fix remainder, since
FC90 91                sub     c               ;..loop went 1 extra

FC91 EB                xchg                    ;cylinder number to hl

               ;----------------------------------------------------
               ; Compute Sector & Side
               ; If sectorX2 > sectors/track then set CSIDE
               ; bit, and reduce sector number by sectors/track
               ;   hl= Quotient (cylinder)
               ;   a = Remainder (sectorX2, either for head 0 or 1)
               ;----------------------------------------------------
FC92 FE18              cpi     HDSPT           ;past end of side 0?
FC94 DA99BF            jc      SIDEOK-ROF      ;N: sector number is good

FC97 C608              adi     CSIDE-HDSPT     ;Compute sector mod HDSPT,
                                               ;..and set side 1 bit

FC99 47        SIDEOK: mov     b,a             ;save sector # with side

               ;---------------------------------------------------
               ; Seek Cylinder
               ;    b = sector number, with side bit set correctly
               ;    hl = cylinder number<9:0>
               ;---------------------------------------------------
                if CSEEK+DBUFR                 ;these are actually 00
                       mov     a,h             ;h<1:0>=cylinder<9:8>
                       ori     CSEEK+DBUFR     ;combine with SEEK cmd
                       mov     h,a
                endif

                                               ;hl=SEEK command with cyl #
FC9A CDB9BF            call    HDCMD-ROF       ;HCMD gets unit # from switches

               ;---------------------------------------------------------
               ; Get platter from sense switch, and combine with
               ; side and sector already in b
               ;   b<7:6> = 0
               ;     b<5> = side
               ;   b<4:0> = sector number
               ;   Sense Switch <A11> = platter number
               ;---------------------------------------------------------
```

```
FC9D 2630               mvi     h,CRDSEC+DBUFR  ;read command, high byte

FC9F DBFF               in      SSWTCH          ;read platter switches
FCA1 E608               ani     PSWTCH          ;mask off all others

FCA3 07                 rlc                     ;Shift to CFPLTR position
FCA4 07                 rlc                     ;..which are bits 7:6
FCA5 07                 rlc

FCA6 B0                 ora     b               ;combine w/ sect & side

                ;-----------------------------------------------------------
                ; Read Sector from current track into controller's buffer 0
                ;   a<7:6> = platter
                ;     a<5> = side
                ;   a<4:0> = sector number
                ;-----------------------------------------------------------
FCA7 CDBABF             call    HDCMDA-ROF      ;low command byte is in a

                ;----------------------------------------------
                ; Issue CRDBUF command to kick off read of 256
                ; bytes from the controller's buffer
                ; Note: this assumes the controller is ready.
                ; (and it is, because HDCMD left it that way.)
                ;----------------------------------------------
FCAA DBA5               in      CDATA           ;reset CDA in CDSTA
FCAC DBA3               in      ACMD            ;clear CMDACK in ACSTA

FCAE C1                 pop     b               ;b=requested byte count
FCAF 78                 mov     a,b
FCB0 D3A7               out     ADATA           ;..to controller

FCB2 3E50               mvi     a,CRDBUF+DBUFR  ;issue Read Buffer command
FCB4 D3A3               out     ACMD            ;..to controller

FCB6 D1                 pop     d               ;(10)
FCB7 E1                 pop     h               ;(10) 10 uS total from 'out'

                ; The 8x300 is ready to transmit data in 8 uS. This code takes
                ; 30 cycles (including the 'ret'), or 15 uS min to get around
                ; to reading the data - so there is no need to wait on CDSTA

                 if FALSE
                DATAWT: in      CDSTA           ;Wait for data port to be ready
                        rlc                     ;msb=CDA
                        jnc     DATAWT-ROF
                 endif

                ;----------------------------------
                ; Controller is ready to transfer
                ; 256 bytes of data from its buffer
                ;----------------------------------
FCB8 C9                 ret                     ;(10)done with GETPAG

                ;===Subroutine===============================================
                ; Issue a disk command, and then wait for the controller
                ; to complete it
                ;
                ; Note: this just assumes the controller is ready, which is OK
                ; since the last command was either a seek (where HDCMD waited
                ; for the controller to become ready) or it was a CRDBUF, which
                ; ended with all bytes transferred - and the controller becomes
                ; ready very soon (1.5 uS) after the last byte is transferred.
                ; On Entry at HDCMD:
                ;   hl = complete command
                ; On Entry at HDCMDA:
                ;   a=low byte of command
                ;   h=high byte of command
                ; On Exit:
                ;   a,flags trashed, all others preserved.
                ;   The command is completed and the controller is ready.
                ;   Any errors will terminate the load, and print an error
                ;   message on the Terminal
                ;==============================================================
FCB9 7D         HDCMD:  mov     a,l             ;low byte of command

FCBA D3A7       HDCMDA: out     ADATA           ;..to data port

FCBC DBA1               in      CSTAT           ;reset CRDY flag just in case
FCBE DBA3               in      ACMD            ;clear CMDACK in ACSTA

FCC0 7C                 mov     a,h             ;command high byte
FCC1 D3A3               out     ACMD            ;issue command

FCC3 DBA0       HDWAIT: in      CREADY          ;Is the controller done?
FCC5 07                 rlc                     ;look at msb=CRDY
FCC6 D2C3BF             jnc     HDWAIT-ROF      ;N: keep waiting

FCC9 DBA1               in      CSTAT           ;reset CRDY flag
FCCB E67F               ani     ERMASK          ;and get A=error code
FCCD C8                 rz                      ;No errors: happy return

                ;       Fall into error exit

                ;===Error Exit=================================================
                ; Report a load error and store error code in RAM at 0.
                ; Hang here forever, with the INTE light lit.
                ; On Entry:
                ;   a=error flag bits
                ;==============================================================
FCCE 320000             sta     0               ;save a=error flags
FCD1 CDDCBF             call    LOADPF-ROF      ;CR,LF,'LOAD', then string
FCD4 204552D2           db      ' ER','R'+80h

FCD8 FB                 ei                      ;INTE is error indicator light
FCD9 C3D9BF     FOREVR: jmp     FOREVR-ROF      ;N: die here, INTE light lit

                ;===Subroutine=================================================
                ; Print inline string on the Terminal,
                ; preceeded by CR,LF,'LOAD'
                ; On Entry:
                ;    The string address is the "return address" on the stack.
                ;    The string is terminated by bit 7 set in its last chr.
                ;    The actual return address is the next address after the
                ;    last string character.
                ; On Exit:
                ;    Trashes a and flags, all other registers preserved.
                ;=============================================================
FCDC CDE5BF     LOADPF: call    PRINTF-ROF
FCDF 0D0A4C4F41         db      CR,LF,'LOA','D'+80h

                ;fall into PRINTF

                ;===Subroutine=================================================
                ; Print inline string on the Terminal
                ; On Entry:
```

```
                ;    The string address is the "return address" on the stack.
                ;    The string is terminated by bit 7 set in its last chr.
                ;    The actual return address is the next address after the
                ;    last string character.
                ; On Exit:
                ;    Trashes a and flags, all other registers preserved.
                ;==============================================================
FCE5 E3         PRINTF: xthl                    ;get string address, save hl

FCE6 7E         PRNTLP: mov     a,m             ;get string character
FCE7 E67F               ani     7Fh             ;strip end-of-string mark
FCE9 CDF3BF             call    PRINTA-ROF      ;and print it

FCEC BE                 cmp     m               ;end of string?
FCED 23                 inx     h               ;point to next chr
FCEE CAE6BF             jz      PRNTLP-ROF      ;No difference: keep going

FCF1 E3                 xthl                    ;restore hl, put return address
FCF2 C9                 ret                     ;..onto stack, and go there

                ;===Subroutine===================
                ; Print a the Terminal
                ; On Entry:
                ;   a=chr to print
                ; On Exit:
                ;    all registers preserved.
                ;================================
FCF3 F5         PRINTA: push    psw             ;save chr to print

FCF4 DB10       PALOOP: in      ACSTAT          ;Wait for TX to be ready
FCF6 E602               ani     ACTDE
FCF8 CAF4BF             jz      PALOOP-ROF

FCFB F1                 pop     psw
FCFC D311               out     ACTXD           ;and send chr
FCFE C9                 ret

FCFF                    end
```