FOX


```
                    2           .ident  ZSBPRM
                    4   ;
0004                5   version ==       4       ; recombine proms
0017                6   revision==       23      ; last work 06/22/83 DSB
0044                7   patch     =       'D'
03D5                8   llb       =       llowbyt         ; proglammers ref
04CA                9   lb        =       lastbyt         ; prog ref too
                   10   ;
                   11   ; User enters response to select assembly options
                   12   ;
0000               13   HOST =\"
                   14   Enter  0  for DMS-3 or DMS-4
                   15          1  for DMS-3/F 'Fox'
                   16          2  for DMS-15
                   17   ;
                   18   Selection "
0000               19   DSC3    ==      HOST
FFFF               20   FOX     ==      HOST-1
FFFE               21   DMS15   ==      HOST-2
                   22   ;
0001               24   TAPEopt =\"
                   25   Enter  0  for tape boot
                   26          1  otherwise"
                   31           .pabs
                   32           .phex
```

```
;----------
; DSC-3, DSC-4 PROM Monitor
;
; Features of the DSC/3:
;
;    Z-80 at 4MHz
;    64K Dynamic RAM
;    2K PROM
;    4 Ports - 3 RS-232
;              1 RS-422  (optionally RS-232)
;    Real time clock
;    Optional floppy disk controller
;    2 8-bit parallel ports
;
; The PROM Monitor occupies the 2K PROM on all ZSBC-3
; CPU boards and can execute the following commands:
;
; Set <addr>
;   set memory locations, starting at <addr>
;
; Dump [ <addr1> [<addr2>] ]
;   dump memory from <addr1> to <addr2>
;
; Fill <addr1> <addr2> <byte>
;   fill memory with <byte> from <addr1> to <addr2>
;
; Go <addr>
;   Jump to <addr>
;
; In <port>
;   input from Z-80 <port>
;
; Out <port> <byte>
;   output <byte> to Z-80 <port>
;
; Test <addr1> <addr2>
;   test memory from <addr1> to <addr2>
;
; Boot <device> [<combuf>]
;   read and execute program from floppy, network,
;   xebec harddisk, test floppy, or tape
;
; Emulate <port> <baudrate> (Fox only)
;  Fox will emulate a CRT by sending keyboard
; input out the specified port, and passing all
; port input along to the fox CRT controller.
;
```

```
 84  ; Computing the Checksum for 2K PROMS
 85  ;
 86  ; DSC3, Fox and DMS15 PROMS do a checksum test before
 87  ; booting.  The checksum can be plugged in under ZDTI
 88  ; for testing, but once settled upon, it should be
 89  ; written into this source code so that PROM.HEX can
 90  ; be used in the usual way.
 91  ; If changes are made to this code the checksum will
 92  ; have to be recalculated.  This can be done using ZDTI
 93  ; to load the PROM image into the TPA, then writing a
 94  ; short program to add (16-bit quantity) all the bytes
 95  ; int the PROM.  The two FF's of the serial number are
 96  ; then subtracted out so the final version will be
 97  ; independent of the serial number.  The resulting 16-
 98  ; bit value, plus the checksum, should equal zero.  So,
 99  ; for example, if the sum of all bytes other than the
100  ; serial number is 88CA, the command "H 0 88CA" will
101  ; show the correct checksum to be plugged into 7FE and
102  ; 7FF.
103  ;*****************************************************************
104  ; BEFORE LOADING PROM.HEX IT IS NECESSARY TO INITIALIZE
105  ; THE BLOCK OF MEMORY IT WILL OCCUPY TO CONTAIN ALL FFs!

106  ; THIS IS BECAUSE LOADING THE .HEX FILE WILL LEAVE SOME
107  ; LOCATIONS IN THE RANDOM STATE THEY WERE ALREADY IN,
108  ; AND THESE RANDOM VALUES WILL BE BLOWN INTO THE PROM,
109  ; LOUSING UP THE CHECKSUM. ******************************
110  ;
```

```
113  ;----------
114  ; Update history:
115  ;
116  ; 1.07 09/01/80 Initial release
117  ; 1.08  10/01   Changed autoboot jumpers
118  ; 1.09  10/27   Changed autoboot jumpers (AGAIN!)
119  ;               Boot from any floppy drive
120  ;               Boot from hard disk
121  ; 1.09T 6/21/81 Boot from tape (option)
122  ; 4.09          version for fox
123  ; 4.15  27nov82 FOX boot from xebec hard disk
124  ; 4.17  06dec82 FOX xebec track/sector numbering
125  ;               changed so logical numbers are
126  ;               same as  regular hard disk
127  ; 4.18          Retries HD until ready
128  ;               Includes Dave Stein's emulate code
129  ; 4.18a         changed DVC to DRVCHR
130  ;               made a comment out of
131  ;               lowadr: .byte 128
132  ;               since it seemed unnecessary and
133  ;               caused errors in assembly.
134  ; 4.18b         Dave B. is back and fixed lowadr
135  ; 4.18c         No he didn't; but this time for sure
136  ; 4.19  04Jan83 Works: reads HD firmware from HD;
137  ;               tested on cmi and miniscribe
138  ; 4.20  18Jan83 Initializes HD (if present) on :BN
139  ; 4.21  23mar83 Fixed emulate (e) bugs.
140  ;               Installed assbly-time ovrrun prot.
141  ; 4.22  23may83 Improved Memory test per SCR #0004
142  ;               (Betsy Tanner's test for floats)
143  ;               Change parallel port status detect
144  ;               (SCR #0018) to avoid bus collisions
145  ;               Add self-test on reset (SCR #0022):
146  ;               PROM checksum, excluding serial no.;
147  ;               RAM check (4000-FFFF) for stuck bits
148  ; 4.23  27may83 Combined versions for various ZSBC3
149  ;               boards; i.e. choice of DSC-3, Fox or
150  ;               DMS15 is made at assby time.  All
151  ;               have the new mem test, and all do
152  ;               self-test since the new DSC-3 boards
153  ;               allow 2k proms.
```

```
156  ;  'A'  06/08/83 Clean up some comments, .page the
157  ;                 listing.  DRB
158  ;  'B'  06/16/83 Revised mem test (better test for un-
159  ;                 inserted address lines).  This test is
160  ;                 now also done if you boot from the
161  ;                 monitor by saying "B(something)".
162  ;  'C'  06/21/83 Yet another mem test (Joe DuVivier's
163  ;                 walking-bit version) since last one runs
164  ;                 like molasses.  Changed cpi's in emulate
165  ;                 command parsing since the value in A at
166  ;                 that point is (set this) the top two BCD
167  ;                 digits of a four-digit quantity, i.e.
168  ;                 06 for 600 baud, 01 for 110 baud, etc.
169  ;  'D'  06/22/83 Fixed bugs in memory test and 110 baud
170  ;                 emulate.
```

```
                    173  ;----------
                    174  ; Memory equates
                    175  ;
9000                176  RAM      ==      09000h  ; beginning of RAM
9000                177  BOOTloc  ==      RAM     ; beginning of bootstrap
9400                178  stack    ==      RAM+400h; top of RAM
93E0                179  Daddr    ==      RAM+3E0h; Dump address
9380                180  resbuf   ==      RAM+380h; Floppy result buffer
9390                181  homeF    ==      RAM+390h; home command
9393                182  readF    ==      RAM+393h; read command
D000                183  tapboot  ==      0D000h  ; write PROM here for tapeboot
D800                184  tapstk   ==      tapboot+800h; so stack not overwritten
                    185  ;
                    186  ; ASCII equates
                    187  ;
000D                188  cr       ==      0Dh
000A                189  lf       ==      0Ah
003F                190  qmark    ==      3Fh
0007                191  bell     ==      07h
                    192  ;
                    193  ; other equates
                    194  ;
0000                195  RxRDY    ==      0       ; receiver ready status bit
0002                196  TxRDY    ==      2       ; transmitter ready status bit
                    197  ;
                    198  ;     Port definitions:
                    199  ;
0038                200  DMA      ==      38h     ; DMA chip
0030                201  CTC0     ==      30h     ; CTC channel 0 (port 0)
0032                202  CTC2     ==      32h     ; CTC channel 2 (ports 2,3)
002A                203  SIO1AC   ==      2Ah     ; SIO-1 channel A, control
0028                204  SIO1AD   ==      28h     ; SIO-1 channel A, data -port 0
002B                205  SIO1BC   ==      2Bh     ; SIO-1 channel B, control
0029                206  SIO1BD   ==      29h     ; SIO-1 channel B, data -port 1
0022                207  SIO2AC   ==      22h     ; SIO-2 channel A, control
0020                208  SIO2AD   ==      20h     ; SIO-2 channel A, data -port 2
0023                209  SIO2BC   ==      23h     ; SIO-2 channel B, control
0021                210  SIO2BD   ==      21h     ; SIO-2 channel B, data -port 3
000A                211  PIOAC    ==      0Ah     ; PIO channel A, control
0008                212  PIOAD    ==      08h     ; PIO channel A, data
000B                213  PIOBC    ==      0Bh     ; PIO channel B, control
0009                214  PIOBD    ==      09h     ; PIO channel B, data
0010                215  FLOPSR   ==      10h     ; Floppy status register
0011                216  FLOPDR   ==      11h     ; Floppy data register
0001                217  HARDP    ==      01h     ; Hard disk parallel port
0003                218  SETMAP   ==      03h     ; Set memory map register
0003                219  STOPFLOP==       03h     ; Stop floppy controller
0002                220  PARADATA==       02H     ; J4 PARALLEL PORT
0008                221  PARASTAT==       08H     ; PIO CHAN A DATA
```

```
                        224  ;       -- XEBEC s1410 controller equates --
                        225  ;
0003                    226  CMDmode          ==      00011b  ; command mode
0001                    227  wrDATAmode       ==      00001b  ; write mode
0009                    228  rdDATAmode       ==      01001b  ; read data mode
000B                    229  rdERRmode        ==      01011b  ; read error mode
000F                    230  LASTmode         ==      01111b  ; ret zero byte
0000                    231  xBUSY            ==      0       ; busy true high
0004                    232  xREQxfer         ==      4       ; data xfer request bit
0001                    233  DAT$CMDport      ==      1       ; mode is set by toggle
0000                    234  toggle           ==      0
                        235  ;
                        236  ; output to port0 toggles between cmd and data modes
                        237  ;
0000                    238  xDRVready        ==      0       ; test drive ready
0008                    239  xREAD            ==      08      ; read command
0003                    240  xSENSE           ==      03      ; status sense
000C                    241  xINITdrv         ==      0Ch     ; initialize drive
0001                    242  xRECAL           ==      1       ; recalibrate head
0002                    243  xERROR           ==      02      ; illegal command
0040                    244  xIDLE            ==      40h     ; idle command
0000                    245  xINITcont        ==      0       ; reset controller
00C1                    246  xSELcont         ==      0C1h    ; select controller
0041                    247  xDESELcont       ==      041h    ; deselect controller
0100                    248  secSIZE          ==      256
0006                    249  lenXEBcmd        ==      6
9400                    250  xOPcode          ==      9400h   ; cmd table moved to RAM
9401                    251  HIGHadr          ==      xOPcode+1
9402                    252  MIDadr           ==      HIGHadr+1
9403                    253  LOWadr           ==      MIDadr+1
9404                    254  skew$blk         ==      LOWadr+1
9405                    255  ctl$fld          ==      skew$blk+1
9406                    256  xERRbuf          ==      ctl$fld+1
                        257
```

```
                    260  ;---------
                    261  ; Come here on master reset
  0000              262          .loc    00h     ; RST 0
  0000  31 9400     263          lxi     SP,stack; you always need a stack
  0003  C3 00DB     264          jmp     initial ; intialize the PIO and SIO1
                    265  ;
                    266  ; Use these two bytes to store the DMA vector
  0006  03A8        267  DMAvect:.word   DMARdone; interrupt on DMA done
```

```
                        270  ;----------
                        271  ; Utility routines
                        272  ;
                        273  ; Callable by an RST instruction:
                        274  ;
                        275  ; RST 1 - in0:      Input a character from port 0
                        276  ; RST 2 - out0:     Output a character to port 0
                        277  ; RST 3 - outst0:   Output a string to port 0
                        278  ; RST 4 - crlf:     Output <return>, <linefeed> to port0
                        279  ; RST 5 - space:    Output <space> to port 0
                        280  ; RST 6 - getbyt:   Input a byte from port 0
                        281  ; RST 7 - restart:  Restart the PROM monitor
                        282  ;
                        283  ; Callable through jump vector at top of ROM
                        284  ; (these first 4 are for DMS-15's only)
                        285  ; cmdFOXhard:    Send a command string to hard disk
                        286  ; xSTATrcv:      Get a result string from hard disk
                        287  ; xERRrcv:       Get error bytes from the hard disk
                        288  ; JustREAD:      Get bytes from hard disk
                        289  ;
                        290  ; command: Output a command string to the floppy contlr
                        291  ; result:  Input a result string from the floppy contlr
                        292  ; ioerr:   Output an I/O error message on port 0
                        293  ;
                        294  ; Jump vector table:
0007                    296  .ife    DMS15&DSC3, [numvec = 7]
03E7                    297          .loc    3FCh-(3*numvec)
03E7                    298  vectab:
03E7  C3 047A           306          jmp     cmdHARD
03EA  C3 0496           307          jmp     resHARD
03ED  C3 04A6           308          jmp     sendHARD
03F0  C3 04B3           309          jmp     recHARD
03F3  C3 00A1           312          jmp     command         ; hard disk write func
03F6  C3 00B1           313          jmp     result
03F9  C3 00D2           314          jmp     ioerr
03FC  04                323          .byte   version
03FD  17                324          .byte   revision
03FE  FFFF              325  serial: .word   0FFFFh  ; serial number
                        326
07FE                    327          .loc    7FEh                    B543
07FE  B1C5              330          .ife DSC3 ! (1-TAPEopt),[.word 0B1C5h]
```

```
                        334  ;----------
                        335  ; Input a character from console port
                        336  ;  Regs in:   none
                        337  ;  Regs out:  A = input character
0008                    338          .loc    08h     ; RST 1
0008                    339  in0:
0008                    340  waiti:
0008  DB2A              342          in      SIO1AC  ; read status
000A  CB47              343          bit     RxRDY,A
000C  28FA              344          jrz     waiti
000E  DB28              345          in      SIO1AD  ; get char.
0010                    361          .loc    10h     ; RST 2
0010                    362  out0:
0010  E67F              363          ani     7Fh     ; strip parity bit
0012  F5                364          push    PSW     ; save output char
0013  182D              365          jmpr    outc    ; continue elsewhere
                        366  ;----------
                        367  ; Flush an input character from port 0
                        368  ;  (jumped to from 'out')
0015                    369  flush:
0015  DB28              370          .ife    DSC3,       [in SIO1AD]
0017  FF                372          rst     restart/8 ; restart the monitor
                        373  ;----------
                        374  ; Output a string to port 0
                        375  ;  Regs in:   HL = address of string (ended by null)
                        376  ;  Regs out:  none
                        377  ;  Destroyed: A, HL
0018                    378          .loc    18h     ; RST 3
0018                    379  outst0:
0018  7E                380          mov     A,M     ; get next output char
0019  B7                381          ora     A       ; string terminated by null
001A  C8                382          rz
001B  D7                383          rst     out0/8  ; output this char
001C  23                384          inx     H       ; point to next char
001D  18F9              385          jmpr    outst0  ; repeat until null found
```

```
                        388  ;----------
                        389  ; Output <return>, <linefeed> to port 0
                        390  ;  Regs in:   none
                        391  ;  Regs out:  none
                        392  ;  Destroyed: A
  0020                  393          .loc    20h     ; RST 4
  0020                  394  crlf:
  0020  3E0D            395          mvi     A,cr
  0022  D7              396          rst     out0/8
  0023  3E0A            397          mvi     A,lf
  0025  18E9            398          jmpr    out0    ; return from out0
                        399  ;----------
                        400  ; Output <space> to port 0
                        401  ;  Regs in:   none
                        402  ;  Regs out:  none
                        403  ;  Destroyed: A
  0028                  404          .loc    28h     ; RST 5
  0028                  405  space:
  0028  3E20            406          mvi     A,' '
  002A  18E4            407          jmpr    out0    ; return from out0
                        408  ;----------
                        409  ; Handle an input error
                        410  ;  Regs in:   none
                        411  ;  Regs out:  none
  002C  3E3F            412  error:  mvi     A,qmark
  002E  D7              413          rst     out0/8  ; type qmark on input error
  002F  FF              414          rst     restart/8 ; restart the monitor
                        415  ;--------
                        416  ; Input a byte from port 0
                        417  ;  Regs in:   none
                        418  ;  Regs out:  A = byte read
                        419  ;             B = 1 if number was typed, 0 otherwise
                        420  ;  Destroyed: DE
  0030                  421          .loc    30h     ; RST 6
  0030                  422  getbyt:
  0030  E5              423          push    H       ; save HL (courtesy to caller)
  0031  CD 0058         424          call    getadr  ; let 'getadr' do the work
  0034  7D              425          mov     A,L     ; result in low byte of HL
  0035  E1              426          pop     H       ; restore HL
  0036  C9              427          ret
                        428  ;----------
                        429  ; Restart the main monitor loop
  0038                  430          .loc    38h     ; RST 7
  0038                  431  restart:
  0038  C3 01AE         432          jmp     prompt  ; jump here to restart monitor
```

```
                        435  ;----------
                        436  ; Input a byte from parallel port
                        437  ;  (Jumped to from 'in0')
                        438
003B  CB77              439  inc:    bit     6,A     ; check for receiver ready
003D  28C9              440          jrz     waiti   ; wait until ready
003F  DB02              441          in      PARADATA; read the char
0041  F5                442          push    PSW     ; save it
                        443                          ; echo by falling into outc
                        444  ;----------
                        445  ; Output a byte to parallel port
                        446  ;  (Jumped to from 'out0', falle into from 'inc')
                        447  ; Return with A= the byte we put out, xlated into
                        448  ; upper case (in case it is an input echo.)
                        449  ;
0042                    450  outc:
0042                    451  wait0:
0042  DB2A              453          in      SIO1AC  ; read status
0044  CB57              454          bit     TxRDY,A ; check for xmtr ready
0046  28FA              455          jrz     wait0
0048  CB47              456          bit     RxRDY,A ; check for rcvr ready
004A  20C9              457          jrnz    flush   ; allow interrupt by user
004C  F1                458          pop     PSW     ; get output char
004D  D328              459          out     SIO1AD  ; put it out
004F  FE61              476          cpi     'a'     ; return if less than 'a'
0051  F8                477          rm
0052  FE7B              478          cpi     'z'+1   ; return if greater than 'z'
0054  F0                479          rp
0055  D620              480          sui     'a'-'A' ; convert lower to upper case
0057  C9                481          ret
```

```
                        484  ;----------
                        485  ; Input an address from port 0
                        486  ;  Regs in:   none
                        487  ;  Regs out:  HL = address read
                        488  ;              B  = 1 if number was typed, 0 otherwise
                        489  ;  Destroyed: A, DE
                        490  ;
                        491  ; Note:  An address contains zero or more spaces,
                        492  ;        followed by one or more HEX digits, and
                        493  ;        terminated by a <space> or <return>
0058                    494  getadr:
0058  21 0000           495          lxi     H,0       ; HL = 0
005B  44                496          mov     B,H       ; B = 0
005C  54                497          mov     D,H       ; D = 0
005D  CF                498  get0:   rst     in0/8     ; skip leading spaces
005E  FE20              499          cpi     ' '
0060  28FB              500          jrz     get0
0062  FE0D              501  get1:   cpi     cr        ; terminate if <return>
0064  C8                502          rz
0065  FE47              503          cpi     'G'
0067  30C3              504          jrnc    error     ; jump if digit > F
0069  FE41              505          cpi     'A'
006B  3806              506          jrc     num       ; jump if digit < A
006D  D607              507          sui     'A'-('9'+1) ; partially convert A..F
006F  FE3A              508          cpi     '9'+1
0071  38B9              509          jrc     error     ; jump if 9 < digit < A
0073  D630              510  num:    sui     '0'       ; convert ASCII to HEX
0075  38B5              511          jrc     error     ; jump if digit < 0
0077  29                512          dad     H         ; shift HL left 4 bits
0078  29                513          dad     H
0079  29                514          dad     H
007A  29                515          dad     H
007B  5F                516          mov     E,A       ; DE = current HEX digit
007C  19                517          dad     D         ; add current digit to addr
007D  0601              518          mvi     B,1       ; set 'number typed' flag
007F  CF                519          rst     in0/8
0080  FE20              520          cpi     ' '       ; terminate if <space>
0082  C8                521          rz
0083  18DD              522          jmpr    get1      ; continue collecting chars
                        523  ;----------
                        524  ; Output an address to port 0
                        525  ;  Regs in:   HL = address to be printed
                        526  ;  Regs out:  none
                        527  ;  Destroyed: A
0085                    528  prtadr:
0085  7C                529          mov     A,H
0086  CD 008A           530          call    prtbyt
0089  7D                531          mov     A,L
                        532  ;----------
                        533  ; Output a byte to port 0
                        534  ;  Regs in:   A = byte to be printed
                        535  ;  Regs out:  none
                        536  ;  Destroyed: A
008A                    537  prtbyt:
```

```
008A  F5          538           push    PSW
008B  0F          539           rrc             ; print high nibble of A
008C  0F          540           rrc
008D  0F          541           rrc
008E  0F          542           rrc
008F  CD 0093     543           call    prtnbl  ; print low nibble of A
0092  F1          544           pop     PSW
0093  E60F        545  prtnbl:  ani     0Fh     ; mask lower nibble
0095  C630        546           adi     '0'     ; convert to ASCII
0097  FE3A        547           cpi     '9'+1
0099  DA 0010     548           jc      out0    ; jump if 0..9
009C  C607        549           adi     'A'-('9'+1) ; convert A..F to ASCII
009E  C3 0010     550           jmp     out0    ; use return from out0
```

```
00A1                559  command:
00A1  7E            560          mov     A,M     ; get a command byte
00A2  FEFF          561          cpi     endcom  ; check if end-of-string
00A4  C8            562          rz              ; return if all bytes sent
00A5  CD 00BE       563          call    waitdr  ; wait for data register
00A8  DC 00D2       564          cc      ioerr   ; I/O error if wrong direction
00AB  7E            565          mov     A,M     ; get the command byte
00AC  D311          566          out     FLOPDR  ; write to data register
00AE  23            567          inx     H       ; point to next command byte
00AF  18F0          568          jmpr    command
                    569  ;----------
                    570  ; Collect result string from the floppy controller
                    571  ;  Regs in:   none
                    572  ;  Regs out:  none
                    573  ;  Destroyed: A, HL
00B1                574  result:
00B1  21 9380       575          lxi     H,resbuf; point to result buffer
00B4  CD 00BE       576  readDR: call    waitdr  ; wait from data register
00B7  D0            577          rnc             ; return if all bytes received
00B8  DB11          578          in      FLOPDR  ; get the result byte
00BA  77            579          mov     M,A     ; store it in 'flopbuf'
00BB  23            580          inx     H       ; point to next result byte
00BC  18F6          581          jmpr    readDR
                    582  ;----------
                    583  ; Wait until floppy data register is ready
                    584  ;  Regs in:   none
                    585  ;  Regs out:  A = status register, rotated left by 1
00BE                586  waitdr:
00BE  DB10          587          in      FLOPSR  ; read floppy status register
00C0  07            588          rlc
00C1  30FB          589          jrnc    waitdr  ; bit 7 = 1 if ready
00C3  07            590          rlc             ; bit 6 = 1 if CPU receiving
00C4  C9            591          ret             ;         0 if CPU sending
00C5  0D0A492F4F    597  errmsg: .asciz  [cr][lf]'I/O error '
00D2  21 00C5       598  ioerr:  lxi     H,errmsg
00D5  DF            599          rst     outst0/8 ; output a message
00D6  E1            600          pop     H       ; get the caller's address
00D7  CD 0085       601          call    prtadr  ; print the caller's address
00DA  FF            602          rst     restart/8 ; restart the monitor
```

```
                        605  ;----------
                        606  ; Initialization code - executed on master reset
                        607  ; Initialize the PIO chip
  00DB                  608  initial:
  00DB  3ECF            609          mvi     A,0CFh
  00DD  D30A            610          out     PIOAC   ; reset PIO channel A
  00DF  3EF8            611          mvi     A,0F8h  ;  top    5 lines = input
  00E1  D30A            612          out     PIOAC   ;  bottom 3 lines = output
  00E3  3E03            613          mvi     A,3
  00E5  D30A            614          out     PIOAC   ; disable interrupts from PIO
                        615                          ; (We have no reason to want to
                        616                          ; be interrupted by the PIO.
                        617                          ; Remember we are in the default
                        618                          ; int mode so any ints make us
                        619                          ; restart the monitor.  We don't
                        620                          ; need to do this on transitions
                        621                          ; of the PIO status lines.)
  00E7  3ECF            622          mvi     A,0CFh
  00E9  D30B            623          out     PIOBC   ; reset PIO channel B
  00EB  3EC0            624          mvi     A,0C0h  ;  top    2 lines = input
  00ED  D30B            625          out     PIOBC   ;  bottom 6 lines = output
  00EF  3E03            626          mvi     A,3
  00F1  D30B            627          out     PIOBC   ; disable interrupts
  00F3  3E00            628          mvi     A,0
  00F5  D309            629          out     PIOBD   ; unload the floppy head
  00F7  3EC3            630          mvi     A,0C3h
  00F9  D338            631          out     DMA     ; reset the DMA chip
                        632  ;
                        633  ; Initialize port 0 to be a standard CRT at 9600 baud
  00FB  3E45            634          mvi     A,45h   ; set counter mode
  00FD  D330            635          out     CTC0    ; next byte is time constant
  00FF  3E0D            636          mvi     A,13    ; 13 = 2Mhz/(16*baudrate)
  0101  D330            637          out     CTC0    ; port 0 is 9600 baud
  0103  21 0183         638          lxi     H,rs232 ; port 0 is RS-232
  0106  01 092A         639          lxi     B,rs232$<8+SIO1AC
  0109  EDB3            640          outir
                        641  ;
                        642  ; Prepare to enter monitor if front-panel switch is on
  010B  DB08            643          in      PIOAD
  010D  CB7F            644          bit     7,A     ; check front-panel switch
  010F  C2 014F         645          jnz     ramok   ; don't do self-test if set.
```

```
                        648  ; Self-Test.  Since we aren't using the Monitor it's ok
                        649  ; to trash memory inthe course of testing it.  First we
                        650  ; do a ROM checksum calculation, then test the RAM for
                        651  ; stuck bits and for address line 15 being shorted to
                        652  ; one of the others.
                        653  ;
0112  21 0000           654          lxi     H,0       ; start from the bottom
0115  01 07FE           655          lxi     B,cksum ; don't add this in too
0118  AF                656          xra     A
0119  57                657          mov     D,A       ; zero partial result ress
011A  5F                658          mov     E,A
011B  7B                659  ..add:  mov     A,E       ;add to previous partial result
011C  86                660          add     M         ;the byte pointed to by HL.
011D  5F                661          mov     E,A       ;do 16-bit add this way so we
011E  3001              662          jrnc    ..nc      ;don't need a stack
0120  14                663          inr     D
0121  23                664  ..nc:   inx     H         ; point to next byte in PROM
0122  0B                665          dcx     B
0123  78                666          mov     A,B
0124  B1                667          ora     C
0125  20F4              668          jrnz    ..add     ; add more to partial sum
                        669  ;
0127  EB                670          xchg              ;partial sum in HL
0128  AF                671          xra     A         ;clear carry for sure
0129  3A 03FE           672          lda     serial    ;remove serial from calculation
012C  4F                673          mov     C,A       ;gotta subtract out serial#
012D  0600              674          mvi     B,0
012F  ED42              675          dsbc    B         ; subtract one byte of serial
0131  AF                676          xra     A
0132  3A 03FF           677          lda     serial+1
0135  4F                678          mov     C,A
0136  0600              679          mvi     B,0
0138  ED42              680          dsbc    B         ; subtract 2nd byte of serial#
013A  ED4B 07FE         681          lbcd    cksum     ; now HL plus checksum
013E  09                682          dad     B         ; should add up to 0!
013F  7D                683          mov     A,L
0140  B4                684          ora     H
0141  2806              685          jrz     ckram
                        686  ;
0143  3E07              687          mvi     A,bell    ; if bad, camp on conout port
0145                    688  ..bell:
0145  D328              689  .ife DSC3, [out SIO1AD]
0147  18FC              691          jmpr    ..bell    ; possible, no status check.
```

```
                        694  ; Now check out the RAM (up where the code has room)
                        695  ;
0149                    696  ckram:
0149  21 014F           697          lxi     H,ramok ; return addr
014C  C3 0400           698          jmp     ckram2
                        699  ;
                        700  ;-------------------------
                        701  ; Since the RAM (above 4000h) has passed the test we
                        702  ; assume,naively, that it's ok to map out the PROM.
                        703  ;
014F                    704  ramok:
014F  21 0000           705          lxi     H,0     ; move to ram, rel rom
0152  11 8000           706          lxi     D,8000h
0155  01 0800           707          lxi     B,800h
0158  EDB0              708          ldir
015A  C3 815D           709          jmp     movbak+8000h
015D  DB02              710  movbak: in      2       ; release rom
015F  21 8000           711          lxi     H,8000h
0162  11 0000           712          lxi     D,0
0165  01 0800           713          lxi     B,800h
0168  EDB0              714          ldir
016A  C3 016D           715          jmp     contin
                        716  ;
016D                    717  contin:                 ; place to continue
016D  DB08              718          in      PIOAD   ; if front panel int switch is
016F  CB7F              719          bit     7,A     ; on, so to the Moniter
0171  C2 01AA           720          jnz     Monitor ; ELSE, check for auto-boot
                        721  ;
0174  DB2A              722          in      SIO1AC  ;  DCD  STS
0176  CB5F              723          bit     3,A     ;  ---  ---
0178  C2 037E           724          jnz     Network ;   1    -   Network boot
017B  CB6F              725          bit     5,A     ;   0    0   Floppy boot
017D  C2 0460           726          jnz     Harddisk;   0    1   Harddisk boot
                        727  ;
0180  C3 02EC           728  .ifn TAPEopt,[jmp Floppy]
0183  18               733  rs232:  .byte   18h             ; channel reset
0184  144C             734          .byte   14h,01001100b ; 2 stops bits, no parity
0186  03C1             735          .byte    3 ,11000001b ; receiver enable
0188  05EA             736          .byte    5 ,11101010b ; transmitter enable
018A  1100             737          .byte   11h,00000000b ; no interrupts
0009                    738  rs232$  ==      .-rs232   ; length of SIO command strins
```

```
                        741  ;----------
                        742  ; Main monitor loop
                        743  ;
                        744  ; Wait for a user request, then jump to the
                        745  ; appropriate routine to process it
                        746  ;
 018C  0D0A             747  PROMmsg:.ascii  [cr][lf]
 018E  444D532033       748          .ife    DSC3, [.ascii "DMS 3/4 "]
 0196  2050524F4D       751          .ascii  " PROM Monitor "
 01A4  342E             752          .byte   version+'0','.'
 01A6  32334400         753          .byte   revision/10+'0',revision@10+'0',patch,0
 01AA                   754  Monitor:
 01AA  21 018C          755          lxi     H,PROMmsg
 01AD  DF               756          rst     outst0/8 ; print a boot-up message
 01AE                   757  prompt:
 01AE  31 9400          758          lxi     SP,stack ; start with a fresh stack
 01B1  E7               759          rst     crlf/8  ; on a fresh line
 01B2  3E3A             760          mvi     A,':'   ; output the command prompt
 01B4  D7               761          rst     out0/8
 01B5  CF               762          rst     in0/8   ; input a command
 01B6  FE53             763          cpi     'S'
 01B8  282A             764          jrz     Set     ; Set memory
 01BA  FE44             765          cpi     'D'
 01BC  CA 0202          766          jz      Dump    ; Dump memory
 01BF  FE47             767          cpi     'G'
 01C1  CA 024E          768          jz      Go      ; Go to a program
 01C4  FE46             773          cpi     'F'
 01C6  CA 0252          774          jz      Fill    ; Fill memory
 01C9  FE49             775          cpi     'I'
 01CB  CA 0267          776          jz      In      ; Input from a port
 01CE  FE4F             777          cpi     'O'
 01D0  CA 0270          778          jz      Out     ; Output to a port
                        779  ;.ife (DSC3 ! (1-TAPEopt)),[
                        780  ;       cpi     'M'
                        781  ;       jz      Map     ; not enough room for
                        782  ;       ]               ; DSC-4 map register set
 01D3  FE54             783          cpi     'T'
 01D5  CA 0278          784          jz      Test    ; Test memory
 01D8  FE42             785          cpi     'B'
 01DA  CA 02C4          786          jz      Boot    ; Boot
 01DD  FE0D             787          cpi     cr
 01DF  2819             788          jrz     Dumpcr  ; <cr> - dump memory
 01E1  C3 002C          789          jmp     error
```

```
                    792  ;----------
                    793  ; Set memory
                    794  ;
                    795  ; S <addr>
  01E4              796  Set:
  01E4  CD 0058     797          call    getadr  ; address is in HL
  01E7  E7          798  Set1:   rst     crlf/8
  01E8  CD 0085     799          call    prtadr  ; print HL
  01EB  EF          800          rst     space/8
  01EC  7E          801          mov     A,M     ; get old byte from memory
  01ED  CD 008A     802          call    prtbyt  ; print it
  01F0  EF          803          rst     space/8
  01F1  F7          804          rst     getbyt/8 ; get new byte from CRT
  01F2  CB40        805          bit     0,B     ; test if number entered
  01F4  2801        806          jrz     Set2    ; jump if no number entered
  01F6  77          807          mov     M,A     ; replace byte in memory
  01F7  23          808  Set2:   inx     H       ; point to next byte
  01F8  18ED        809          jmpr    Set1
                    810  ;----------
                    811  ; Dump memory
                    812  ;
                    813  ; D  [ <addr1> [<addr2] ]
  01FA              814  Dumpcr:
  01FA  21 9000     815          lxi     H,RAM   ; default starting address
  01FD  22 93E0     816          shld    Daddr
  0200  1818        817          jmpr    default
  0202              818  Dump:
  0202  CD 0058     819          call    getadr  ; get starting address
  0205  CB40        820          bit     0,B
  0207  2005        821          jrnz    Daddr1  ; jump if number entered
  0209  2A 93E0     822          lhld    Daddr   ; resume from previous dump
  020C  180C        823          jmpr    default
  020E  22 93E0     824  Daddr1: shld    Daddr   ; store starting address
  0211  FE0D        825          cpi     cr      ; see if <return> was typed
  0213  2805        826          jrz     default ; jump if no end addr entered
  0215  CD 0058     827          call    getadr  ; get ending address
  0218  1804        828          jmpr    Daddr2
  021A  11 0080     829  default:lxi     D,128   ; default dump range
  021D  19          830          dad     D
  021E  EB          831  Daddr2: xchg            ; DE = ending address
```

```
                    834  ;
 021F  2A 93E0      835  Dump0:  lhld    Daddr   ; Top of dump loop -
 0222  E7           836          rst     crlf/8  ; print dump address
 0223  CD 0085      837          call    prtadr
                    838  ;
 0226  EF           839  Dump1:  rst     space/8 ; Dump 16 bytes in hex format
 0227  7E           840          mov     A,M     ; get a byte
 0228  CD 008A      841          call    prtbyt  ; print it in hexadecimal
 022B  23           842          inx     H       ; point to next byte
 022C  7D           843          mov     A,L     ; check if multiple of 16
 022D  E60F         844          ani     0Fh
 022F  20F5         845          jrnz    Dump1   ; Jump if not
                    846  ;
 0231  EF           847          rst     space/8 ; Dump 16 bytes in ASCII format
 0232  2A 93E0      848          lhld    Daddr
 0235  7E           849  Dump2:  mov     A,M     ; get a byte
 0236  E67F         850          ani     7Fh     ; mask out parity bit
 0238  FE20         851          cpi     20h     ; printable?
 023A  3002         852          jrnc    .+4     ; skip next instr if printable
 023C  3E2E         853          mvi     A,'.'   ; print <dot> if unprintable
 023E  D7           854          rst     out0/8  ; print it in ASCII
 023F  23           855          inx     H       ; point to next byte
 0240  7D           856          mov     A,L     ; check if multiple of 16
 0241  E60F         857          ani     0Fh
 0243  20F0         858          jrnz    Dump2   ; Jump if not
                    859  ;
 0245  22 93E0      860          shld    Daddr   ; Check if dump is finished
 0248  B7           861          ora     A       ; clear carry flag
 0249  ED52        862          dsbc    D       ; clear carry flag if done
 024B  38D2         863          jrc     Dump0   ; Jump if not done
 024D  FF           864          rst     restart/8
                    865  ;----------
                    866  ; Go to a memory location
                    867  ;
                    868  ; G <addr>
 024E               869  Go:
 024E  CD 0058      870          call    getadr  ; address is in HL
 0251  E9           871          pchl            ; Jump to address in HL
```

```
                        874  ;----------
                        875  ; Fill memory
                        876  ;
                        877  ; F <addr1> <addr2> <byte>
0252                    878  Fill:
0252  CD 0058           879          call    getadr  ; starting address
0255  E5               880          push    H
0256  CD 0058           881          call    getadr  ; ending address
0259  F7               882          rst     getbyt/8 ; fill char
025A  D1               883          pop     D       ; begin addr in DE
025B  B7               884          ora     A       ; clear carry
025C  ED52             885          dsbc    D       ; HL = HL - DE
025E  4D               886          mov     C,L     ; BC = byte count
025F  44               887          mov     B,H
0260  6B               888          mov     L,E     ; HL = source address
0261  62               889          mov     H,D
0262  77               890          mov     M,A     ; put fill char in memory
0263  13               891          inx     D       ; DE = destination address
0264  EDB0             892          ldir
0266  FF               893          rst     restart/8
                        894  ;----------
                        895  ; Input from a port
                        896  ;
                        897  ; I <port>
0267                    898  In:
0267  F7               899          rst     getbyt/8
0268  4F               900          mov     C,A     ; C = port number
0269  E7               901          rst     crlf/8
026A  ED78             902          inp     A       ; input byte to A
026C  CD 008A          903          call    prtbyt  ; print the byte
026F  FF               904          rst     restart/8
                        905  ;----------
                        906  ; Output to a port
                        907  ;
                        908  ; O <port> <byte>
0270                    909  Out:
0270  F7               910          rst     getbyt/8
0271  F5               911          push    PSW     ; save port number
0272  F7               912          rst     getbyt/8 ; get output byte
0273  C1               913          pop     B       ; restore port number
0274  48               914          mov     C,B     ; C = port number
0275  ED79             915          outp    A       ; output byte from A
0277  FF               916          rst     restart/8
```

```
919  ;----------
920  ; set memory map register
921  ;
922  ; M <reg> <byte>
923  ;
924  ; Note: <reg> = 0,1,...,F
925  ;       <byte>: bit 7 = 0 local mem
926  ;               bit 7 = 1 external mem
927  ;               bits 6 to 0 = page number
928  ;                (bits 6 to 4 are low-true)
929  ;
930  ; For exemple, the command "M 2 F3" will map logical
931  ; addresses 2000-2FFF to physical addresses 3000-3FFF
932  ;
933  ; To enable the map, do command "0 0 8".
934  ;.ife (DSC3 ! (1-TAPEopt)),[
935  ;Map:
936  ;       rst     getbyt/8
937  ;       push    PSW
938  ;       rst     getbyt/8
939  ;       mov     H,A
940  ;       pop     PSW
941  ;       rlc
942  ;       rlc
943  ;       rlc
944  ;       rlc
945  ;       mov     B,A
946  ;       mvi     C,SETMAP
947  ;       outp    H
948  ;       rst     restart/8
949  ;       ]       ; end "DSC-3 without Tape"
```

```
                        952  ;----------
                        953  ; Test memory
                        954  ;
                        955  ; T <addr1> <addr2>
                        956  ;
                        957  ; Repetitively write an increasing data pattern,
                        958  ; and read it back for verification.
                        959  ; For each error detected , the address, byte written,
                        960  ; and byte read are displayed.
  00FF                  961  pattern =       0FFh    ; initial pattern offset
  0278                  962  Test:
  0278  CD 0058         963          call    getadr  ; read starting address
  027B  E5              964          push    H
  027C  CD 0058         965          call    getadr  ; read ending address
  027F  D1              966          pop     D       ; DE = start, HL = end
  0280  06FF            967          mvi     B,pattern
  0282  E7              968          rst     crlf/8
  0283  E5              969  retry:  push    H       ; save ending address
  0284  E5              970          push    H
  0285  0E7B            971          mvi     C,7Bh   ; seed
  0287  CD 02BB         972  fillmem:call    tstpatn
  028A  77              973          mov     M,A     ; fill memory with random
  028B  B7              974          ora     A       ; clear carry bit
  028C  ED52            975          dsbc    D       ; set Z flag if done
  028E  19              976          dad     D       ; restore HL
  028F  2B              977          dcx     H       ; go to next location
  0290  20F5            978          jrnz    fillmem
  0292  E1              979          pop     H       ; restore ending address
  0293  0E7B            980          mvi     C,7Bh   ; seed
  0295  CD 02BB         981  testmem:call    tstpatn
  0298  7E              982          mov     A,M
  0299  B9              983          cmp     C
  029A  C4 02AC         984          cnz     memerr  ; call if memory error
  029D  B7              985          ora     A
  029E  ED52            986          dsbc    D
  02A0  19              987          dad     D
  02A1  2B              988          dcx     H
  02A2  20F1            989          jrnz    testmem
  02A4  05              990          dcr     B       ; use another pattern
  02A5  3E2E            991          mvi     A,'.'   ; tell user we're still alive
  02A7  D7              992          rst     out0/8
  02A8  E1              993          pop     H
  02A9  C3 0283         994          jmp     retry
  02AC  F5              995  memerr: push    PSW     ; save what was written
  02AD  E7              996          rst     crlf/8
  02AE  CD 0085         997          call    prtadr  ; print address of error
  02B1  EF              998          rst     space/8
  02B2  F1              999          pop     PSW
  02B3  CD 008A        1000          call    prtbyt  ; byte written
  02B6  EF             1001          rst     space/8
  02B7  7E             1002          mov     A,M
  02B8  C3 008A        1003          jmp     prtbyt  ; byte read
  02BB                 1004  tstpatn:              ; set pseudorandom # in C
  02BB  79             1005          mov     A,C
```

```
02BC  C631      1006          adi     31h
02BE  EED1      1007          xri     11010001b
02C0  0F        1008          rrc
02C1  80        1009          add     B
02C2  4F        1010          mov     C,A
02C3  C9        1011          ret
```

```
                    1014  ;----------
                    1015  ; Boot from floppy, network, harddisk, or tape.
                    1016  ;
                    1017  ; B <device> [<combuf>]
                    1018  ;
                    1019  ; Read a bootstrap program from a device into RAM.
                    1020  ; If successful, jump to RAM.  If the device is 'T',
                    1021  ; then test the floppy disk controller.
  02C4              1022  Boot:
  02C4  CF          1023          rst     in0/8   ; set device name
  02C5  08          1024          exaf            ; save it
  02C6  21 02CC     1025          lxi     H,..ok  ; come back to here
  02C9  C3 0400     1026          jmp     ckram2  ; test memory!
  02CC  08          1027  ..ok:   exaf            ; set device back
  02CD  FE4E        1028          cpi     'N'
  02CF  CA 037E     1029          jz      Network ; Boot from network
  02D2  FE48        1030          cpi     'H'     ; boot from harddisk
  02D4  CA 0460     1031          jz      Harddisk
  02D7  FE46        1038          cpi     'F'
  02D9  2811        1039          jrz     Floppy  ; Boot from floppy unit 0
  02DB  FE54        1040          cpi     'T'
  02DD  CA 0361     1041          jz      Testflop; Test floppy
                    1042  ;
                    1043  ; Check for boot from specified floppy unit
  02E0  D630        1044          sui     '0'     ; unit must be >= 0
  02E2  FA 002C     1045          jm      error
  02E5  FE08        1046          cpi     7+1     ; unit must be <= 7
  02E7  F2 002C     1047          jp      error
  02EA  1801        1048          jmpr    goflop
```

```
                    1051  ;----------
                    1052  ; Floppy boot
                    1053  ;
                    1054  ; Read single density, track 0, sectors 1-2
                    1055  ; from drive 0, and jump to RAM
02EC                1056  Floppy:
02EC  97            1057          sub     A
02ED                1058  goflop:
02ED  21 0371       1059          lxi     H,homeFLOP; setup home, read commands
02F0  11 9390       1060          lxi     D,homeF ; must move to RAM so that
02F3  01 000D       1061          lxi     B,13    ; unitno can be selected
02F6  EDB0          1062          ldir
02F8  32 9391       1063          sta     homeF+1 ; put unitno in home, read
02FB  32 9394       1064          sta     readF+1
02FE  CBD7          1065          set     2,A     ; turn on drive select bit
0300  D309          1066          out     PIOBD   ; select drive
0302  97            1067          sub     A
0303  D308          1068          out     PIOAD   ; multiplex Floppy to DMA
0305  CD 00B1       1069          call    result  ; flush the floppy controller
0308  21 036B       1070          lxi     H,specFLOP
030B  CD 00A1       1071          call    command ; specify timing info
030E  21 9390       1072  reseek: lxi     H,homeF
0311  CD 00A1       1073          call    command ; recalibrate drive 0
0314  21 036F       1074  sense:  lxi     H,IsenseFLOP
0317  CD 00A1       1075          call    command ; sense interrupt status
031A  CD 00B1       1076          call    result  ; 2 result bytes returned
031D  3A 9380       1077          lda     resbuf  ; get first status byte
0320  CB5F          1078          bit     3,A
0322  20EA          1079          jrnz    reseek  ; retry if not ready
0324  CB6F          1080          bit     5,A
0326  28EC          1081          jrz     sense   ; retry until seek done
0328  E6D0          1082          ani     0D0h    ; check if seek successful
032A  C4 00D2       1083          cnz     ioerr
                    1084  ;
                    1085  ; This code is very timing sensitive.
                    1086  ; For 5 inch floppy, each data byte must be read from
                    1087  ; the floppy controller within 27 microsecs, or it is
                    1088  ; lost, and read operation will terminate prematurely.
                    1089  ;
032D  21 9393       1090          lxi     H,readF
0330  CD 00A1       1091          call    command ; read data
0333  21 9000       1092          lxi     H,BOOTloc ; boot address
0336  01 0011       1093          lxi     B,256<8+FLOPDR
0339  CD 00BE       1094  ..read: call    waitdr  ; wait for next byte available
033C  EDA2          1095          ini             ; get and store a byte
033E  20F9          1096          jrnz    ..read  ; continue until 1 sector read
0340  C3 02ED       1097          jmp     goflop  ; loop bootstrap loader
0343  DB03          1098          in      STOPFLOP; stop the floppy controller
                    1099                          ; and check result status
0345  CD 00B1       1100          call    result  ; 7 result bytes returned
0348  21 9380       1101          lxi     h,resbuf; get first result byte
034B  7E            1102          mov     a,m
034C  E6C0          1103          ani     0C0h    ; top 2 bits should be zero
034E  CA 9000       1104          jz      BOOTloc ; jump to boot code
```

```
0351  11 9070   1105          lxi     d,9070h ; mov result buf
0354  01 0010   1106          lxi     b,10h   ;  to to 9070 for display
0357  EDB0      1107          ldir
0359  21 9000   1108          lxi     h,9000h ; display result if err
035C  3E0D      1109          mvi     a,cr
035E  C3 020E   1110          jmp     daddr1  ; into display routine
```

```
                        1113  ;----------
                        1114  ; Test floppy
                        1115  ;
                        1116  ; Get command buffer address, and then
                        1117  ; perform a floppy command and result phase
  0361                  1118  Testflop:
  0361  CD 0058         1119          call    setadr  ; set command buffer address
  0364  CD 00A1         1120          call    command ; do a command phase
  0367  CD 00B1         1121          call    result  ; do a result phase
  036A  FF              1122          rst     restart/8
                        1123  ;----------
                        1124  ; Floppy command strings - see NEC765 chip description
                        1125  ;                           for explanation
  0000                  1126  drive   =       0       ; boot drive number
  0000                  1127  track   =       0       ; boot track number
  0001                  1128  sector  =       1       ; first boot sector
  00FF                  1129  endcom  =       0FFh    ; end-of-command character
  036B                  1130  specFLOP:
  036B  03              1131          .byte   3       ; 'specify' command
  036C  88              1139          .byte   88h     ; 8ms seek, 128 ms unload
  036D  81              1140          .byte   81h     ; load 128 ms, non-DMA
  036E  FF              1142          .byte   endcom
  036F                  1143  IsenseFLOP:
  036F  08              1144          .byte   8       ; 'sense interrupt status'
  0370  FF              1145          .byte   endcom
  0371                  1146  homeFLOP:
  0371  07              1147          .byte   7       ; 'recalibrate' command
  0372  00              1148          .byte   drive   ; drive select
  0373  FF              1149          .byte   endcom
  0374                  1150  readFLOP:
  0374  06              1164          .byte   6       ; single density
  0375  00              1165          .byte   drive   ; head, drive select
  0376  00              1166          .byte   track   ; track number
  0377  00              1167          .byte   00h     ; head address
  0378  01              1168          .byte   sector  ; sector number
  0379  00              1169          .byte   00h     ; 128 byte sector
  037A  1A              1170          .byte   1Ah     ; last sector number
  037B  07              1171          .byte   07h     ; gap length (in bytes)
  037C  80              1172          .byte   128     ; sector size
  037D  FF              1173          .byte   endcom
```

```
                1178  ;-----------------------------------------------
                1179  ; Network boot
  037E          1180  Network:
  037E  97      1192          sub     A       ; DMA vector stored in low core
  037F  ED47    1193          stai
  0381  ED5E    1194          im2             ; setup interrupt mode
  0383  3E01    1195          mvi     A,1
  0385  D308    1196          out     PIOAD   ; multiplex SIO1B to DMA
  0387  21 03C2 1197          lxi     H,DMANprog
  038A  01 1338 1198          lxi     B,DMAN$<8+DMA
  038D  EDB3    1199          outir           ; program the DMA chip
  038F  21 03B3 1200          lxi     H,RECprog
  0392  01 0F2B 1201          lxi     B,REC$<8+SIO1BC
  0395  EDB3    1202          outir           ; program the SIO1B chip
  0397          1203  ..wait:
  0397  DB2B    1204          in      SIO1BC  ; wait for data addressed to us
  0399  E601    1205          ani     1<RxRDY
  039B  28FA    1206          jrz     ..wait
  039D  DB29    1207          in      SIO1BD  ; throw away the user number
  039F  3E87    1208          mvi     A,87h   ; enable DMA
  03A1  D338    1209          out     DMA
  03A3  FB      1210          ei              ; remember ints are disabled
                1211                          ; by a hardware reset.
  03A4  76      1212          hlt             ; wait for DMA done interrupt
  03A5  C3 9000 1213          jmp     BOOTloc ; execute the bootstrap
                1214  ;
                1215  ; Process interrupt on DMA done
  03A8          1216  DMARdone:
  03A8  F5      1217          push    PSW
  03A9  3EC3    1218          mvi     A,0C3h  ; reset DMA chip
  03AB  D338    1219          out     DMA
  03AD  3E18    1220          mvi     A,18h   ; reset the SIO1B chip
  03AF  D32B    1221          out     SIO1BC
  03B1  F1      1222          pop     PSW
  03B2  C9      1223          ret
```

```
                        1226  ;----------
                        1227  ; SDLC command strings
  0380                  1228  lenBOOT ==      380h    ; length of network boot code
  00FE                  1229  BOOTusr ==      254     ; boot pseudo user number
  03B3                  1230  RECprog:
  03B3  18              1231          .byte       00011000b ; channel reset
  03B4  0420            1232          .byte     4,00100000b ; SDLC mode
  03B6  1580            1233          .byte   15h,10000000b ; transmit disable
  03B8  03FC            1234          .byte     3,11111100b ; receiver info
  03BA  06FE            1235          .byte     6,BOOTusr   ; user number
  03BC  077E            1236          .byte     7,01111110b ; flag byte
  03BE  11E4            1237          .byte   11h,11100100b ; interrupt control
  03C0  23FD            1238          .byte   23h,11111101b ; enable receiver
  000F                  1239  REC$     ==      .-RECprog
                        1240  ;
  03C2                  1241  DMANprog:
  03C2  C3              1242          .byte   0C3h    ; master reset              2D
  03C3  C7              1243          .byte   0C7h    ; reset port A              2D
  03C4  CB              1244          .byte   0CBh    ; reset port B              2D
  03C5  7D              1245          .byte   7Dh     ; read from port B
  03C6  9000            1246          .word   BOOTloc ; DMA address
  03C8  037F            1247          .word   lenBOOT-1; DMA length - 1
  03CA  14              1248          .byte   14h     ; port A inc, memory        1B
  03CB  28              1249          .byte   28h     ; port B fixed, I/O         1B
  03CC  95              1250          .byte   95h     ; byte mode                 2B
  03CD  29              1251          .byte   SIO1BD  ; port B
  03CE  12              1252          .byte   12h     ; interrupt at end of block
  03CF  06              1253          .byte   DMAvect&0FFh ; interrupt vector
  03D0  92              1254          .byte   92h     ; stop at end of block
  03D1  CF              1255          .byte   0CFh    ; load starting address     2C
  03D2  01              1256          .byte   1       ; read
  03D3  CF              1257          .byte   0CFh    ; load starting address     1A
  03D4  AB              1258          .byte   0ABh    ; enable interrupts
  0013                  1259  DMAN$    ==      .-DMANprog
  03D5                  1260   llowbyt ==      .
```

```
                    1266  ;*******************************************************
                    1267
0400                1268          .LOC    400H    ;SECOND HALF OF ROM
                    1269
                    1270  ; RAM diagnostics.  First, check for data bits stuck
                    1271  ; high or low, by filling memory with 0's and FF's and
                    1272  ; reading them back.
                    1273
0400                1274  ckram2:
0400  D9            1275          exx             ; save HL (ret addr)
0401  3EFF          1276          mvi     A,0FFh  ; test with FF's first
0403  21 4000       1277  ..chk1: lxi     H,4000h ; fill memory, 4000 to top
0406  11 4001       1278          lxi     D,4001h
0409  01 BFFF       1279          lxi     B,(0-4001h)
040C  77            1280          mov     M,A     ; char to fill
040D  EDB0          1281          ldir            ; fill it on up
                    1282
040F  21 4000       1283          lxi     H,4000h
0412  11 0000       1284          lxi     D,0
0415  BE            1285  ..chk2: cmp     M       ; see if still same as A
0416  C2 0459       1286          jnz     stuck   ; if not, it's stuck.
0419  B7            1287          ora     A       ; clear carry
041A  23            1288          inx     H
041B  ED5A          1289          dadc    D       ; check for HL=0 this way
041D  20F6          1290          jrnz    ..chk2  ; (doesn't trash A)
                    1291
041F  B7            1292          ora     A       ; after doing FF, do 0
0420  2803          1293          jrz     across  ; after doing 0, go on
0422  AF            1294          xra     A
0423  18DE          1295          jmpr    ..chk1
                    1296
```

```
                    1299  ; Test for shorted address lines: Memory is now full of
                    1300  ; zeros from 4000h to FFFFh.  Test all address inputs to
                    1301  ; those three banks.  With one addr line high, write an
                    1302  ; FF, then check the addresses with any other line high
                    1303  ; (and the one with all low) to see if any bits are non-
                    1304  ; zero (i.e. an addr line to chip is shorted or open).
                    1305  ; Next rotate the high bit right in the write address
                    1306  ; and repeat, thus testing every line to the chip.  Do
                    1307  ; this entire operation for eah of the three high banks.
                    1308
                    1309
0425                1310  across:
0425  01 4000      1311          lxi     B,4000h ; bank offset saved in BC
0428  21 2000      1312  bnkloop:lxi     H,2000h ; highest addr pin to RAM is high

042B  09            1313  outloop:dad     B       ; HL now = write address
042C  7E            1314          mov     A,M     ; read byte to be written
042D  B7            1315          ora     A       ; should be 0 to start! if not,
042E  2029          1316          jrnz    cross   ; bank-select logic has problems
0430  2F            1317          cma             ; make FF
0431  77            1318          mov     M,A     ; stick it into test address
0432  54            1319          mov     D,H
0433  5D            1320          mov     E,L     ; set up write addr in DE too
                    1321                          ; then go & rotate bits once
                    1322
0434  7A            1323  inloop: mov     A,D     ; DE has actual test addr
0435  90            1324          sub     B       ; so subtract bank offset
0436  CB3F          1325          srlr    A       ; rotate it right (fill w/0)
0438  CB1B          1326          rarr    E       ; if 1 has got thru, then
043A  3808          1327          jrc     ..go    ; break out of inner loop
043C  80            1328          add     B       ; remember DE has just 1 bit high

043D  57            1329          mov     D,A     ; so adding bank offset gives
043E  1A            1330          ldax    D       ; test address.  Read it.
043F  B7            1331          ora     A       ; check for 1's
0440  2017          1332          jrnz    cross   ; bomb if any found
0442  18F0          1333          jmpr    inloop
                    1334
0444  0A            1335  ..go:   ldax    B       ; now check addr w/ all 0's
0445  B7            1336          ora     A
0446  2011          1337          jrnz    cross
0448  7C            1338          mov     A,H     ; HL has write addr, so subtract
0449  90            1339          sub     B       ; bank offset, and rotate
044A  CB3F          1340          srlr    A       ; rotate right, fill w/0
044C  67            1341          mov     H,A     ;  low bit -> CY
044D  CB1D          1342          rarr    L       ; rotate right, CY -> high bit
044F  30DA          1343          jrnc    outloop ; if 1 isn't thru go back again
0451  3E40          1344          mvi     A,40h   ; now move bank offset up to
0453  80            1345          add     B       ; next bank
0454  47            1346          mov     B,A
0455  20D1          1347          jrnz    bnkloop ; if zero, we're done
                    1348
                    1349
0457  D9            1350          exx             ; get ret addr back
```

```
0458  E9        1351          pchl            ; exit by jumping to where we
                1352                          ; left off.
0459            1353  stuck:
0459  7E        1354  cross:  mov     A,M     ; get culprit onto address lines
045A  3E07      1355          mvi     A,bell  ; if bad, camp on conout port
045C            1356  ..bell:
045C  D328      1357  .ife DSC3,[out SIO1AD]
045E  18F9      1359          jmpr    cross   ; possible, no status check.
                1360
```

```
0460                    1368  Harddisk:
0460  21 04C0           1369          lxi     H,selHARD
0463  CD 047A           1370          call    cmdHARD ; select unit 0
0466  21 04C2           1371          lxi     H,readHARD
0469  CD 047A           1372          call    cmdHARD ; read 1 sector
046C  CD 0496           1373          call    resHARD ; get result status
046F  21 9000           1374          lxi     H,BOOTloc
0472  0680              1375          mvi     B,128
0474  CD 04B3           1376          call    recHARD ; get data bytes
0477  C3 9000           1377          jmp     BOOTloc
                        1378  ;----------
                        1379  ; Send command to the hard disk
                        1380  ;  Regs in:   HL = address of command string
                        1381  ;  Regs out:  none
                        1382  ;  Destroyed: A, BC, HL
047A                    1383  cmdHARD:
047A  DB01              1384  ..1:    in      HARDP   ; clear status
047C  DB08              1385          in      PIOAD   ; wait for HDC receive
047E  CB5F              1386          bit     3,A
0480  20F8              1387          jrnz    ..1
0482  3E51              1388          mvi     A,51h   ; "request to send"
0484  D301              1389          out     HARDP
0486  DB08              1390  ..2:    in      PIOAD   ; wait for HDC send
0488  CB67              1391          bit     4,A
048A  28FA              1392          jrz     ..2
048C  DB01              1393          in      HARDP   ; check if "clear to send"
048E  FE52              1394          cpi     52h
0490  20E8              1395          jrnz    ..1     ; if not, retry
0492  0608              1396          mvi     B,8
0494  1810              1397          jmpr    sendHARD
                        1398  ;----------
                        1399  ; Receive status from hard disk controller
                        1400  ;  Regs in:   none
                        1401  ;  Regs out:  none
                        1402  ;  Destroyed: A, BC, HL
0496                    1403  resHARD:
0496  21 9380           1404          lxi     H,resbuf; put result bytes here
0499  0608              1405          mvi     B,8     ; always get 8 result bytes
049B  CD 04B3           1406          call    recHARD
049E  3A 9387           1407          lda     resbuf+7; check error status
04A1  B7                1408          ora     A
04A2  C4 00D2           1409          cnz     ioerr   ; abort on any error
04A5  C9                1410          ret
```

```
                    1413  ;----------
                    1414  ; Send data bytes to the hard disk
                    1415  ;  Regs in:   HL = block address
                    1416  ;              B  = byte count
                    1417  ;  Regs out:  none
                    1418  ;  Destroyed: A, BC, HL
04A6                1419  sendHARD:
04A6  0E01          1420          mvi     C,HARDP
04A8  DB08          1421  ..1:    in      PIOAD
04AA  CB5F          1422          bit     3,A
04AC  20FA          1423          jrnz    ..1
04AE  EDA3          1424          outi
04B0  20F6          1425          jrnz    ..1
04B2  C9            1426          ret
                    1427  ;----------
                    1428  ; Receive data bytes from the hard disk
                    1429  ;  Regs in:   HL = block address
                    1430  ;              B  = byte count
                    1431  ;  Regs out:  none
                    1432  ;  Destroyed: A, BC, HL
04B3                1433  recHARD:
04B3  0E01          1434          mvi     C,HARDP
04B5  DB08          1435  ..1:    in      PIOAD
04B7  CB67          1436          bit     4,A
04B9  28FA          1437          jrz     ..1
04BB  EDA2          1438          ini
04BD  20F6          1439          jrnz    ..1
04BF  C9            1440          ret
                    1441  ;----------
                    1442  ; Hard disk commands
04C0                1443  selHARD:
04C0  13            1444          .byte   13h     ; select
04C1  00            1445          .byte   0       ; unit
04C2                1446  readHARD:
04C2  11            1447          .byte   11h     ; read
04C3  01            1448          .byte   1       ; secotr
04C4  02            1449          .byte   2       ; track
04C5  00            1450          .byte   0
04C6  00            1451          .byte   0
04C7  00            1452          .byte   0
04C8  0A            1453          .byte   10      ; retries
04C9  00            1454          .byte   0
```

1635

```
04CA                2202  lastbyt ==      .
                    2203  .if2,[
                    2206        [.prntx 'Code is shorter than 2k (& will fit in th
                          e prom).']
                    2208         .end
```

BELL	0007	BNKLOO	0428	BOOT	02C4	BOOTLO	9000
BOOTUS	00FE	CKRAM	0149	CKRAM2	0400	CKSUM	07FE
CMDHAR	047A	CMDMOD	0003	COMMAN	00A1	CONTIN	016D
CR	000D	CRLF	0020	CROSS	0459	CTCO	0030
CTC2	0032	CTL$FL	9405	DADDR	93E0	DADDR1	020E
DADDR2	021E	DAT$CM	0001	DEFAUL	021A	DMA	0038
DMANPR	03C2	DMAN$	0013	DMARDO	03A8	DMAVEC	0006
DMS15	FFFE	DRIVE	0000	DSC3	0000	DUMP	0202
DUMP0	021F	DUMP1	0226	DUMP2	0235	DUMPCR	01FA
ENDCOM	00FF	ERRMSG	00C5	ERROR	002C	FILL	0252
FILLME	0287	FLOPDR	0011	FLOPPY	02EC	FLOPSR	0010
FLUSH	0015	FOX	FFFF	GET0	005D	GET1	0062
GETADR	0058	GETBYT	0030	GO	024E	GOFLOP	02ED
HARDDI	0460	HARDP	0001	HIGHAD	9401	HOMEF	9390
HOMEFL	0371	HOST	0000	IN	0267	INO	0008
INC	003B	INITIA	00DB	INLOOP	0434	IOERR	00D2
ISENSE	036F	LASTBY	04CA	LASTMO	000F	LB	04CA
LENBOO	0380	LENXEB	0006	LF	000A	LLB	03D5
LLOWBY	03D5	LOWADR	9403	MEMERR	02AC	MIDADR	9402
MONITO	01AA	MOVBAK	015D	NETWOR	037E	NUM	0073
NUMVEC	0007	OUT	0270	OUTO	0010	OUTC	0042
OUTLOO	042B	OUTSTO	0018	PARADA	0002	PARAST	0008
PATCH	0044	PATTER	00FF	PIOAC	000A	PIOAD	0008
PIOBC	000B	PIOBD	0009	PROMMS	018C	PROMPT	01AE
PRTADR	0085	PRTBYT	008A	PRTNBL	0093	QCROSS	0425
QMARK	003F	RAM	9000	RAMOK	014F	RDDATA	0009
RDERRM	000B	READDR	00B4	READF	9393	READFL	0374
READHA	04C2	RECHAR	04B3	RECPRO	03B3	REC$	000F
RESBUF	9380	RESEEK	030E	RESHAR	0496	RESTAR	0038
RESULT	00B1	RETRY	0283	REVISI	0017	RS232	0183
RS232$	0009	RXRDY	0000	SECSIZ	0100	SECTOR	0001
SELHAR	04C0	SENDHA	04A6	SENSE	0314	SERIAL	03FE
SET	01E4	SET1	01E7	SET2	01F7	SETMAP	0003
SIO1AC	002A	SIO1AD	0028	SIO1BC	002B	SIO1BD	0029
SIO2AC	0022	SIO2AD	0020	SIO2BC	0023	SIO2BD	0021
SKEW$B	9404	SPACE	0028	SPECFL	036B	STACK	9400
STOPFL	0003	STUCK	0459	TAPBOO	D000	TAPEOP	0001
TAPSTK	D800	TEST	0278	TESTFL	0361	TESTME	0295
TOGGLE	0000	TRACK	0000	TSTPAT	02BB	TXRDY	0002
VECTAB	03E7	VERSIO	0004	WAITO	0042	WAITDR	00BE
WAITI	0008	WRDATA	0001	XBUSY	0000	XDESEL	0041
XDRVRE	0000	XERRBU	9406	XERROR	0002	XIDLE	0040
XINITC	0000	XINITD	000C	XOPCOD	9400	XREAD	0008
XRECAL	0001	XREQXF	0004	XSELCO	00C1	XSENSE	0003
.BLNK.	0000:03 X	.DATA.	0000" X	.PROG.	0000' X		