

```

;
;*****
;*
;*          UNIVERSAL BASIC I/O SYSTEM (BIOS)          *
;*          Vers. 1.6                                   *
;*
;*          A,B = 5 Inc. 32,10 sec/trk 256 byte/sec    *
;*          C,D = 5 Inc. 32,10 sec/trk 256 byte/sec    *
;*          E,F = 5 Inc. 17 sec/trk   128 byte/sec (old Lg) *
;*
;*          nota: aggiungere diagnostica errori in fdd i/o *
;*
;*****
;
;          title   Bios 1.6 for NE CP/M 2.2 with Hard-Disk Basf 6182
;          subttl  Copyright Studio Lg, Genova - Last rev 15/08/1984 09:06
;          Programmers: Martino Stefano & Gallerani Paolo
;
;          include SIZE.CPM ; Get cp/m size
003C      C      msize equ 60 ; CP/M memory size in kilobyte
;
;          vers equ 'HS' ; Single side version
4853      C      rev equ 16 ; CBIOS revision number
0010
;
;          Boolean scalar constants
;
0000      false equ 0
00FF      true  equ 0ffh
;
;          *** I/O Devices ***
0001      TTY equ 01b ; CON:
0000      RDR equ false ; Undefined
0000      PUN equ false ; Undefined
0002      LST equ 10b ; LST:
;
;          Default Value for I/O byte
00B1      DftI.O equ (LST shl 6) or (RDR shl 4) or (PUN shl 2) or (TTY)
;
;          page 62

```

```

;
C      Include ASCII.LIB      ; Ascii Table
C      ;
C      ;
C      ;*****
C      ;*
C      ;*      ASCII EQUIVALENTS
C      ;*
C      ;*****
;
0007   C      bell      equ      'G'->'@'      ; ring beeper
0008   C      backsp    equ      'H'->'@'      ; back space char.
0009   C      tab      equ      'I'->'@'      ; tabulation char.
000A   C      lf       equ      'J'->'@'      ; line-feed char.
000C   C      ffeed    equ      'L'->'@'      ; form feed char.
000D   C      cr       equ      'M'->'@'      ; carriage-return char.
0013   C      pfx      equ      'S'->'@'      ; attributes pfx
0048   C      rever    equ      'H'          ; Reverse On      (^SH)
0043   C      flash    equ      'C'          ; Flash On       (^SC)
0040   C      norm     equ      '@'          ; Normal         (^SE)
0020   C      space    equ      ' '          ; space char.
0024   C      endmsg   equ      '$'          ; end of print message
C      ;
C      Include ROMENTRY.LIB    ; load rom entry points
C      ;
C      ;
C      ;*****
C      ;*
C      ;*      Rom routines address
C      ;*
C      ;*****
;
F000   C      rom      equ      0F000h      ; (--- rom starting address
F003   C      cin      equ      rom+3      ; console input
F006   C      cout     equ      rom+6      ; console output
F009   C      csts     equ      rom+9      ; console status
F00C   C      lout     equ      rom+12     ; printer output
F00F   C      lsts     equ      rom+15     ; printer status
F012   C      fdios    equ      rom+18     ; fdd I/O 128 byte
F015   C      fdiod    equ      rom+21     ; fdd I/O 256 byte
F018   C      wdini    equ      rom+24     ; wdd initialization
F01B   C      wdio     equ      rom+27     ; wdd I/O 256 byte
F01E   C      strout   equ      rom+30     ; print string .DE until $
F01E   C      print    equ      strout     ; sinonime
F021   C      bootrom  equ      rom+33     ; load BIOS and go to wboote
F024   C      printat  equ      rom+36     ; print str. -> DE at -> HL cursor
F027   C      movcurs  equ      rom+39     ; move cursor at -> HL
F02A   C      vidinit  equ      rom+42     ; initialize video
F02D   C      CompFlg  equ      rom+45     ; Version Number
C      ;
;
page

```

```

;
;
;
;*****
;*          SYSTEM CONSTANTS          *
;*****
;
;
003C      csize equ      msize ; cp/m size in kbyte
;
;      bias is address offset from 3400h for memory system
;      than 16k (referred to as "b" throughout the next)
;
;
A000      bias equ      (csize-20)*1024;
D400      ccp equ      3400h+bias      ; base of ccp
DC06      bdos equ     ccp+806h        ; base of bdos
EA00      bios equ     ccp+1600h       ; base of bios
;
;
1600      cpml equ     bios-ccp        ; lenght (in bytes) of cp/m system (ccp + bdos)
0600      biosl equ    600h            ; lenght (in bytes) of standard bios
0016      cpmsiz equ   cpml/secsiz     ; lenght (sector numbers) of cp/m (ccp + bdos)
0006      biossiz equ  biosl/secsiz    ; lenght (sector numbers) of bios
0003      iobyte equ   0003h          ; intel I/O byte
0004      CurDsk equ   0004h          ; cp/m logical disk number
0080      stack equ    0080h          ; wboot stack pointer
0080      defdma equ    0080h          ; cp/m default dma adrs
1000      stack1 equ   1000h          ; ipl stack pointer
;
;
;      page
    
```

```

;
;*****
;*
;*          Disk constants
;*
;*
;*****
;
0002 fddsiz equ 2 ; fdd number on system (10 sec/trk -256 byte-)
0002 wddsiz equ 2 ; wdd number on system (32 sec/trk -256 byte-)
0002 extfds equ 2 ; fds number on system (17 sec/trk -128 byte-)
0006 maxdsk equ fddsiz+wddsiz+extfds ; max disk on system
;
000A fddsec equ 10 ; fdd sec/trk (10)
0020 wddsec equ 32 ; wdd sec/trk (32)
0100 secsiz equ 256 ; byte/sector (256)
;
0002 cpmbk equ secsiz/128 ; r/w buffer size
0001 secmsk equ cpmbk-1 ; sector mask
0014 fddspt equ fddsec*cpmbk ; cp/m fdd sec/trk (20)
0040 wddspt equ wddsec*cpmbk ; cp/m wdd sec/trk (64)
;
;*****
;*
;*          BDOS constants on entry to write
;*
;*
;*****
;
0000 wrall equ 0 ; write to allocated
0001 wrdir equ 1 ; write to directory
0002 wrual equ 2 ; write to unallocated
;
;
page
    
```

```

;
; INCLUDE IPL.DFT ; Default IPL
;
;*****
;*
;* IPL for NEW BIOS
;*
;*
;*****
;
; this program loaded in ram by rom boot, load the cp/m
; bios, set bios sysflag and go to wboote
;
; subttl IPL for NE BIOS 1.4 with Hard-Disk BASF 6182
;
0000' Aseg
; .phase 1000h ; origin of IPL
;
;
1000 wdbboot:
; ; entry point for bios boot from hard disk
1000 C3 1009 jp wdbbt1 ; jump to hard bios boot
;
1003 fdbboot:
; ; entry point for bios boot from floppy disk
1003 C3 103A jp fdbbt1 ; jump to floppy bios boot
1006 iplmsg:
; ; message for ipl checking
1006 49 50 4C defb 'IPL'
;
;
;
1009 wdbbt1:
; ; load bios from hard disk
1009 21 105F ld hl,bbtdsk ; H.L = bios boot r/w para pointer
100C CD F01B call wdio ; read bios
100F B7 or a ; read error ?
1010 20 1B jr nz,bbterr ; yes, then reinitialize system
; ; A = 0 because not error occurs
1012 bbtok:
; ; bios has been loaded
1012 32 EE4E ld (sysflag),a ; set bios system flag
1015 DD 21 EE4F ld ix,vidares ; init video table
1019 CD F02A call vidinit
101C 21 0003 ld hl,iobyte ; point to iobyte
101F 36 81 ld (hl),DftI.O ; value for i/o byte (1st:=1pt:)
1021 23 inc hl ; point to logdsk
1022 36 00 ld (hl),0 ; set cp/m logical disk = 0
1024 11 106D ld de,cpmmsg ; D.E = cp/m message
1027 CD F01E call strout ; print it
102A C3 EA03 jp wboote ; jump to bios wboote
;
; ; error in reading BIOS
;
102D bbterr:
102D 11 10A3 ld de,bbtermmsg ; D.E = bios boot error message
1030 CD F01E call strout ; print it
1033 wait1cr:

```



```

1033 CD F003 C call cin ; wait one char.
1036 FE 0D C cp cr ; cr ?
1038 20 F9 C jr nz,wait1cr ;
C ;
103A C fdbbt1:
C ; load bios from floppy disk
103A 21 1060 C ld hl,bbttrk ; H.L = track para pointer
103D 36 02 C ld (hl),2 ; floppy disk bios is in track 2
103F 06 06 C ld b,biossiz ; bios size in sector number
1041 11 1067 C ld de,bbtxlt ; D.E = sector translate table for bios boot
1044 C fdbbt2:
1044 C5 C push bc ; save sector count
1045 D5 C push de ; save xlt1 pointer
1046 1A C ld a,(de) ; load physical sector
1047 32 1062 C ld (bbtsec),a ; set physical sector number
104A 21 105F C ld hl,bbtdsk ; H.L = boot para adrs
104D CD F015 C call fdiod ; read 256 byte
1050 D1 C pop de ;
1051 C1 C pop bc ;
1052 B7 C or a ; read error ?
1053 20 D8 C jr nz,bbterr ; yes, then reinitialize system
1055 21 1064 C ld hl,bbtdma+1 ; HL = point to high byte current dma adrs
1058 34 C inc (hl) ; DMA add=DMA add + 256
1059 13 C inc de ; xlt1 pointer + 1
105A 10 E8 C djnz fdbbt2 ; loop until end
105C 3C C inc a ; A -> 1 for setting system flag
105D 18 B3 C jr bbtok ; boot done
C ;
C ;
C ; bios boot r/w para table (initially for wdd)
C ;
105F 00 C bbttdsk: defb 0 ; dsk-0 sid 0
1060 0000 C bbttrk: defw 0 ; cylinder number
1062 18 C bbtsec: defb 24 ; sector number (for wdd)
1063 EA00 C bbtdma: defw bios ; bios start address
1065 00 C btpw: defb 0 ; read operation
C ;
1066 06 C wdbloc: defb biossiz ; for wdd boot (6 sec. to load)
C ;
1067 C bbtxtlt:
C ; sector translate table for floppy disk (256 byte/sec)
C ; the first two sector are occupied by ccp + bdos
C ; than bbtxtlt starts at 4' sector
C ;
1067 09 05 02 08 C defb 9,5,2,8,4,10
106B 04 0A C ;
C ;
106D C cpmsg:
106D 0C 0D 0A 0A C defb ffeed,cr,lf,lf,pfx,'H',(cmsgsize+1)/10+'0',(cmsgsize+1) mod 10+'0','K N.E. New Disk 5
em - '
1071 13 48 36 31 C
1075 48 20 4E 2E C
1079 45 2E 20 4E C
107D 65 77 20 44 C
1081 69 73 6B 20 C
1085 53 79 73 74 C
1089 65 6D 20 2D C
108D 20 C

```

```

1092 20 48 53 C
1095 20 72 65 76 C defb ' rev ',rev/10+'0','.',rev mod 10+'0',pfx,'@'
1099 20 31 2E 36 C
109D 13 40 C
109F 0D 0A 07 24 C defb cr,lf,bell,endmsg
C
C ;
C ;
10A3 C bbttermg:
10A3 07 0D 0A 43 C defb bell,cr,lf,'Cannot load your BIOS.'
10A7 61 6E 6E 6F C
10AB 74 20 6C 6F C
10AF 61 64 20 79 C
10B3 6F 75 72 20 C
10B7 42 49 4F 53 C
10BB 2E C
10BC 0D 0A 53 65 C defb cr,lf,'Set new system diskette in disk A,'
10C0 74 20 6E 65 C
10C4 77 20 73 79 C
10C8 73 74 65 6D C
10CC 20 64 69 73 C
10D0 6B 65 74 74 C
10D4 65 20 69 6E C
10D8 20 64 69 73 C
10DC 6B 20 41 2C C
10E0 0D 0A 6F 6B C defb cr,lf,'ok push return. ',endmsg
10E4 20 70 75 73 C
10E8 6B 20 72 65 C
10EC 74 75 72 6E C
10F0 2E 20 24 C
C
C ;
10F3 C freipl equ $  $\rightarrow \phi = 10F3 \text{ Hex}$ 
C if $ lt wdbboot+256  $\rightarrow \text{wdbboot} + 256 = 11$ 
000D C frebIPL equ wdbboot+256-$ ; free space on IPL ram
10F3 C defs frebIPL
C else
C if2
C .printx *** WARNING: IPL overflow reserved space ***
C endif
C endif
C ;
C .dephase
C ;

```

For IPL only 10
bytes are reserved

*Nella versione 6PK il
seguente BIOS si carica
dalla locazione E600
in poi.*

```

E600 EA00 C3 EB17
      EA03
      EA06 C3 EB1D
      EA09 C3 EBD0
      EA0C C3 EBE8
      EA0F C3 EBF9
      EA12 C3 EC0E
      EA15 C3 EC2A
      EA18 C3 EC2B
      EA1B C3 EC5D
      EA1E C3 EC3C
      EA21 C3 EC60
      EA24 C3 EC71
      EA27 C3 EC76
      EA2A C3 EC7B
      EA2D C3 EC80
      EA30 C3 EC1C
      EA30 C3 EC65

;
;*****
;*                                     *
;*                               BIOS *
;*                                     *
;*****
;
;      jump vector for individual subroutines
;
;      .phase bios          ; origin of this program
;
;      jp      boot          ; cold start
wboot:
;      jp      wboot         ; warm start
;      jp      const        ; console status
;      jp      conin         ; console character in
;      jp      conout        ; console character out
;      jp      list          ; list character out
;      jp      punch         ; punch character out
;      jp      reader        ; reader character in
;      jp      home          ; move head to home position
;      jp      sel disk      ; select disk
;      jp      settrk        ; set track number
;      jp      setsec        ; set sector number
;      jp      setdma        ; set dma address
;      jp      read          ; read disk
;      jp      write         ; write disk
;      jp      listst        ; return list status
;      jp      sectran       ; sector translate
;
;      page

```



```

;
; INCLUDE DPB/DPH.DFT
;
;
;*****
; * D P B T A B L E *
; *
; * W/F Size B/S S/T Trk Hds R/T Capacity *
; * - - - - - - - - - - - - - - - - *
; * A: = wdd 5" 256 32 153 2 2 2432 Kbyte *
; * B: = wdd 5" 256 32 153 2 0 2448 Kbyte *
; * C: = fdd 5" 256 10 40 1 3 92 Kbyte *
; * D: = fdd 5" 256 10 40 1 3 92 Kbyte *
; * E: = fdd 5" 128 17 40 1 3 78 Kbyte *
; * F: = fdd 5" 128 17 40 1 3 78 Kbyte *
; *
;*****
;
;
EA33 C dpbase equ $ ; base of disk parameter header
;
; ; dpe0,dpe1 = disk parameter header for hard disk
EA33 C dpe0:
EA33 0000 0000 C defw xlt0,0000h ; no translate table
EA37 0000 0000 C defw 0000h,0000h ; scratch area
EA3B F100 EACC C defw dirbuf,dpb0 ; dir buff,param block
EA3F F218 F180 C defw csv0,alv0 ; check,alloc vector
EA43 C dpe1:
EA43 0000 0000 C defw xlt0,0000h ; no translate table
EA47 0000 0000 C defw 0000h,0000h ; scratch area
EA4B F100 EADB C defw dirbuf,dpb01 ; dir buff,param block
EA4F F2B1 F218 C defw csv1,alv1 ; check,alloc vector
;
; ; dpe2,dpe3 = disk parameter header for floppy disk (256 byte/sec)
EA53 C dpe2: ; 256 byte/sec - Single Side
EA53 EA93 0000 C defw xlt1,0000h ; translate table
EA57 0000 0000 C defw 0000h,0000h ; scratch area
EA5B F100 EAEA C defw dirbuf,dpb1 ; dir buff,param block
EA5F F2BD F2B1 C defw csv2,alv2 ; check,alloc vector
;
;
EA63 C dpe3: ; 256 byte/sec - Single Side
EA63 EA93 0000 C defw xlt1,0000h ; translate table
EA67 0000 0000 C defw 0000h,0000h ; scratch area
EA6B F100 EAF9 C defw dirbuf,dpb12 ; dir buff,param block
EA6F F2D9 F2CD C defw csv3,alv3 ; check,alloc vector
;
;
; ; dpe4,dpe5 = disk parameter header for floppy disk (128 byte/sec)
EA73 C dpe4: ; 128 byte/sec - Single Side
EA73 EABB 0000 C defw xlt2,0000h ; translate table
EA77 0000 0000 C defw 0000h,0000h ; scratch area
EA7B F100 EB08 C defw dirbuf,dpb2 ; dir buff,param block
EA7F F2F3 F2E9 C defw csv4,alv4 ; check,alloc vector
;
;
EA83 C dpe5: ; 128 byte/sec - Single Side
EA83 EABB 0000 C defw xlt2,0000h ; translate table

```

EA87	0000 0000	C	defw	0000h,0000h	; scratch area
EA8B	F100 EB08	C	defw	dirbuf,dpb2	; dir buff,param block
EA8F	F305 F2FB	C	defw	csv5,alv5	; check,alloc vector
		C			
		C			
		C	page	← <i>logliava</i>	

```

C
C
0000 C xlt0 equ 0 ; no sector translate for hard disk
C
EA93 C xlt1:
C ; sector translate table for floppy disk (256 byte/sec)
EA93 01 02 0D 0E C defb 1,2,13,14,5,6,17,18,9,10,3,4,15,16,7,8,19,20,11,12
EA97 05 06 11 12 C 0D 0E 11 12 0A 0F 10 13 14 B C
EA9B 09 0A 03 04 C
EA9F 0F 10 07 08 C
EAA3 13 14 0B 0C C 25 10 17 23 18 27 1F
EAA7 15 16 21 22 C defb 21,22,33,34,25,26,37,38,29,30,23,24,35,36,27,28,39,40,31,32
EAB3 19 1A 25 26 C 15 16 22 2A 26 1E 18 0 24 16 28 20
EAB7 1D 1E 17 18 C 41 19
EAB3 23 24 1B 1C C
EAB7 27 28 1F 20 C
C
C
EABB C xlt2:
C ; sector translate table for floppy disk (128 byte/sec)
EABB 01 07 0D 02 C defb 1,7,13,2,8,14,3,9,15
EABF 08 0E 03 09 C
EAC3 0F C
EAC4 04 0A 10 05 C defb 4,10,16,5,11,17,6,12
EAC8 0B 11 06 0C C
C
C
EACC C dpb0:
C ; disk parameter block for hard disk 0 (256 byte/sector 1 res. trk)
EACC 0080 C defw 128 ; SPT (sec/trk) (32 sect * (256/128) * 2 side)
EACE 05 C defb 5 ; BSH
EACF 1F C defb 31 ; BLM
EAD0 01 C defb 1 ; EXM (extent mask)
EAD1 025F C defw 607 ; DSM (disk size in BLS units - 1) (2432 kbyte)
EAD3 03FF C defw 1023 ; DRM (directory elements - 1)
EAD5 FF C defb 11111111b ; AL0
EAD6 00 C defb 00000000b ; AL1
EAD7 0000 C defw 0 ; CKS disk fixed, no dir. check vector
EAD9 0001 C defw 1 ; OFF (track offset)
C
C
EADB C dpb01:
C ; disk parameter block for hard disk 1 (256 byte/sector no res. trk)
EADB 0080 C defw 128 ; SPT (sec/trk) (32 sect * (256/128) * 2 side)
EADD 05 C defb 5 ; BSH
EADE 1F C defb 31 ; BLM
EADF 01 C defb 1 ; EXM (extent mask)
EAE0 0263 C defw 611 ; DSM (disk size in BLS units - 1) (2448 kbyte)
EAE2 03FF C defw 1023 ; DRM (directory elements - 1)
EAE4 FF C defb 11111111b ; AL0
EAE5 00 C defb 00000000b ; AL1
EAE6 0000 C defw 0 ; CKS disk fixed, no dir. check vector
EAE8 0000 C defw 0 ; OFF (no track offset)
C
C
EAEA C dpb1:
C ; disk parameter block for floppy disk
C ; 256 byte/sector - Single Side

```

```

EAEA 0014      C      defw 20      ; SPT (sec/trk) (10 sect * (256/128) * 1 side)
EAEC 04        C      defb 4        ; BSH
EAED 0F        C      defb 15       ; BLM
EAEF 01        C      defb 1        ; EXM (extent mask)
EAF1 002D      C      defw 45       ; DSM (disk size in BLS unit) (90 kbyte)
EAF3 003F      C      defw 63       ; DRM (directory elements - 1)
EAF5 80        C      defb 10000000b ; ALO
EAF7 00        C      defb 00000000b ; AL1
EAF9 0010      C      defw 16       ; CKS = (DRM + 1)/4 (size dir. check vect.)
EAF7 0003      C      defw 3        ; OFF (track offset)
;
;
EAF9          C      dpb12:
; disk parameter block for floppy disk
; 256 byte/sector - Single Side
EAF9 0014      C      defw 20      ; SPT (sec/trk) (10 sect * (256/128))
EAFB 04        C      defb 4        ; BSH
EAFD 0F        C      defb 15       ; BLM
EAF1 01        C      defb 1        ; EXM (extent mask)
EAF3 002D      C      defw 45       ; DSM (disk size in BLS unit) (90 kbyte)
EAF5 003F      C      defw 63       ; DRM (directory elements - 1)
EAF7 80        C      defb 10000000b ; ALO
EAF9 00        C      defb 00000000b ; AL1
EAFB 0010      C      defw 16       ; CKS = (DRM + 1)/4 (size dir. check vect.)
EAFD 0003      C      defw 3        ; OFF (track offset)
;
;
EB08          C      dpb2:
; disk parameter block for floppy disk (128 byte/sector)
EB08 0011      C      defw 17      ; SPT (sec/trk)
EB0A 03        C      defb 3        ; BSH
EB0B 07        C      defb 7        ; BLM
EB0C 00        C      defb 0        ; EXM (extent mask)
EB0D 004D      C      defw 77       ; DSM (disk size in BLS unit) (77 kbyte)
EB0F 001F      C      defw 31       ; DRM (directory elements - 1)
EB11 80        C      defb 10000000b ; ALO
EB12 00        C      defb 00000000b ; AL1
EB13 0008      C      defw 8        ; CKS = (DRM + 1)/4 (size dir. check vect.)
EB15 0003      C      defw 3        ; OFF (track offset)
;
;
page

```

EB17	3A EE4E
EB1A	C3 F021

EB1D	
EB1D	31 0080
EB20	CD ED3F
EB23	21 0004
EB26	7E
EB27	E6 0F
EB29	FE 06
EB2B	38 02
EB2D	
EB2D	36 00

EB2F

EB2F	21	0000
EB32	22	EE69
EB35	26	02
EB37	22	EE6B
EB3A	26	16
EB3C	22	EE6F
EB3F	21	D400
EB42	22	EE6D

EB45	3A EE4E
EB48	B7
EB49	20 0B
EB4B	

EB4B	21	EE69
EB4E	CD	F01B
EB51	B7	
EB52	20	7F
EB54	18	2C

EB56

```

;*****
;* B O O T                                     *
;*                               Exec a Cold Boot                               *
;******
;
boot:
; set A = sysflag and go to bootrom
ld    a,(sysflag)    ; if A = 0 then load IPL from WDD
jp    bootrom        ; else from FDD
;
INCLUDE WBOOT.DFT    ; Default Wboot
;
;*****
;* W B O O T                                     *
;*                               Load bdos + ccp                               *
;*                               From wdd or Single/Side Fdd                     *
;******
;
wboot:
ld    sp,stack        ; set stack pointer
call  WrtPng          ; Write any pending sector
ld    hl,CurDsk       ; point to cp/m log disk
ld    a,(hl)          ; load cp/m logical disk
and   00001111b       ; mask out User
cp    maxdsk          ; disk overflow ?
jr    c,wb_1          ; no, then go to wboot1

wb_0:
ld    (hl),0          ; else clear cp/m log disk
; H=0

wb_1:
;
; Set parameter,
; then load from Hard or Floppy Disk
ld    hl,0            ; Hl=0
ld    (PrePhy),hl     ; Dsk 0 - side 0 & low Track=0
ld    h,2             ; Sector #2
ld    (PreTrk+1),hl   ; Set High Trk=0 & Sector #2
ld    h,cpmsiz        ; ccp + bdos size in sectors number
ld    (PreR.W),hl     ; set Read op. and # of sec (for wdd)
ld    hl,ccp          ; Cp/m starting add
ld    (PreDma),hl     ; set it
;
; Hard or Floppy ?
;
ld    a,(sysflag)     ; load system flag
or    a               ; sysflag = 0 ?
jr    nz,fd_wb        ; no, load from floppy

wd_wb:
; load from hard disk
ld    hl,PrePhy       ; H.L = wdd boot para adrs
call  wdio            ; call wdd read
or    a               ; wdd i/o error ?
jr    nz,exboot       ; yes, then retry
jr    syschk          ; no, then go to system check
;
fd_wb:

```



```

; load cp/m from floppy disk
EB56 06 16 C ld b,cpmisz ; ccp + bdos size in sector number
EB58 11 EBCA C ld de,wbxlt+1 ; D.E = sector translate table
EB5B C fd_wb.3:
EB5B C5 C push bc ; save sector count
EB5C D5 C push de ; save xlt1 pointer
EB5D 1A C ld a,(de) ; load physical sector
EB5E 32 EE6C C ld (PreSec),a ; set physical sector number
EB61 21 EE69 C ld hl,PrePhy ; H.L = boot para adrs
EB64 CD F015 C call fdiod ; read 256 byte
EB67 D1 C pop de ;
EB68 C1 C pop bc ;
EB69 B7 C or a ; read error ?
EB6A 20 67 C jr nz,exboot ; yes, then retry
EB6C 21 EE6E C ld hl,PreDma+1 ; HL = high current dma adrs
EB6F 34 C inc (hl) ; DMA=DMA+256
EB70 05 C dec b ; warm boot end ?
EB71 28 0F C jr z,syschk ; yes, then go to system check
EB73 13 C inc de ; xlt1 pointer + 1
EB74 7B C ld a,e ;
EB75 FE D3 C cp low (wbxlt+fddsec); end of track ?
EB77 20 E2 C jr nz,fd_wb.3 ; no, load next sector
EB79 21 EE6A C ld hl,PreTrk ; H.L = track para adrs
EB7C 34 C inc (hl) ; track = track + 1
EB7D 11 EBC9 C ld de,wbxlt ; point to start xlt table
EB80 18 D9 C jr fd_wb.3 ; load first sector to next track
C ;
C ; CP/M has been loaded
EB82 C syschk:
C ; cp/m system check
EB82 3A D402 C ld a,(ccp+2) ; load third data of cp/m
EB85 FE D7 C cp high (ccp+35Ch) ; check for correct jp address
EB87 20 4A C jr nz,exboot ; no, error
EB89 3E C3 C ld a,0c3h ; jump command
EB8B 32 0000 C ld (0000),a ; location 0000h
EB8E 21 EA03 C ld hl,wboot ; wboot address
EB91 22 0001 C ld (0001),hl ;
EB94 32 0005 C ld (0005),a ; location 0005h
EB97 21 DC06 C ld hl,bdos ; bdos address
EB9A 22 0006 C ld (0006),hl ;
EB9D 3E FF C ld a,0ffh ; A = 0ffh
EB9F 32 EE68 C ld (PreDsk),a ; Physic disk para -> 'ff'
EBA2 21 F000 C ld hl,defbuf ; Default Buffer
EBA5 22 EE6D C ld (PreDma),hl ; set it
EBA8 01 0080 C ld bc,defdma ; BC = default dma adrs
EBAB CD EC76 C call setdma ; cp/m dma = 0080h
EBAE 3A 0004 C ld a,(CurDsk) ; load cp/m Default disk
EBB1 4F C ld c,a ;
EBB2 C3 D400 C jp ccp ; and jump to ccp
C ;
EBB5 C defs 20 ; fre space
C ;
EBC9 C ;
EBC9 01 07 03 09 C defb 1,7,3,9,5,2,8,4,10,6 ; Skew factor table for fdd wboot
EBCD 05 02 08 04 C
EBD1 0A 06 C
C ;
C ;

```



```
EBD3      C      exboot:
EBD3      C      exbot1:
EBD3      11 EDB8  C      ld      de,nosysmsg    ; D.E = no system message
EBD6      CD EDAD  C      call    msgcr        ; print it and wait cr
EBD9      C3 EB2F  C      jp      wb_1         ; retry wboot
           C      ;
           C      ;
           C      page
```

EBDC
 EBDC 3A 0003
 EBD F E6 03
 EBE1 FE 02
 EBE3 DA F009
 EBE6 18 46

EBE8
 EBE8 3A 0003
 EBE B E6 03
 EBE D FE 02
 EBE F 30 3D
 EBF1 DD E5
 EBF3 CD F003
 EBF6 DD E1
 EBF8 C9

EBF9
 EBF9 3A 0003
 EBF C E6 03
 EBF E FE 02
 EC00 30 2C
 EC02 DD E5
 EC04 FD E5
 EC06 CD F006
 EC09 FD E1
 EC0B DD E1
 EC0D C9

```

;
;*****
;*                                     *
;*      *** Logical Peripheral Device Sub ***      *
;*                                     *
;*****
;
;
;
;*****
;* C o n S t                                     *
;*      Return console status (A=-1 if char ready) *
;*****
;
const:
        ld      a,(iobyte)      ; load intel i/o byte
        and     00000011b      ; mask bit 0,1
        cp      2               ;
        jp      c,csts         ; jump rom console status
        jr      notdev         ; jump no device
;
;
;*****
;* C o n I n                                     *
;*      Read char from console *
;*****
;
conin:
        ld      a,(iobyte)      ; load intel i/o byte
        and     00000011b      ; mask bit 0,1
        cp      2               ;
        jr      nc,notdev       ; >1, no device
;
;
        push    IX              ; preserve IX for Z80's prgm
        call    cin             ; rom console input
        pop     IX              ; restore
        ret                     ; done
;
;
;*****
;* C o n O u t                                     *
;*      Write C caracter on console *
;*****
;
conout:
        ld      a,(iobyte)      ; load intel i/o byte
        and     00000011b      ; mask bit 0,1
        cp      2               ; >1, then
        jr      nc,notdev       ; jump no device
;
;
        push    IX              ; Save
        push    IV              ;      Register
        call    cout            ; call console output
        pop     IV              ; Restore
        pop     IX              ;      Registers
        ret                     ; done

```

Bios 1.6 for NE CP/M 2.2 with Hard-Disk Basf 6182
IPL for NE BIOS 1.4 with Hard-Disk BASF 6182

MACRO-80 3.36 17-Mar-80

PAGE 2-9

;

page ← *soybiere*

EC0E

EC0E 3A 0003
 EC11 E6 C0
 EC13 FE 80
 EC15 DA F006
 EC18 CA F00C
 EC1B C9

```

;
;
;
;*****
;*   P R I N T E R   S u b r o u t i n e   *
;*****
;
;
;
;*****
;* L i s t                               *
;*   Write C caracter on printer         *
;*****
;
;
list:
        ld      a,(iobyte)      ; load intel i/o byte
        and     11000000b       ; mask bit 6,7
        cp      080h           ;
        jp      c,cout          ; jump rom console output
        jp      z,lout          ; jump printer output
        ret                      ; no device, data lost
        jr      notdev          ; jump no device
;
;
;
;*****
;* L i s t S t                           *
;*   Return printer status               *
;*****
;
;
listst:
        ld      a,(iobyte)      ; load intel i/o byte
        and     11000000b       ; mask bit 6,7
        cp      080h           ;
        jp      c,csts          ; jump rom console status
        jp      z,lsts          ; jump printer status
        ret                      ; no device, now ret 11000000b
        jr      notdev          ; jump no device
;

```

EC1C

EC1C 3A 0003
 EC1F E6 C0
 EC21 FE 80
 EC23 DA F009
 EC26 CA F00F
 EC29 C9

```

;
;
;
;*****
;* L i s t S t                           *
;*   Return printer status               *
;*****
;
;
listst:
        ld      a,(iobyte)      ; load intel i/o byte
        and     11000000b       ; mask bit 6,7
        cp      080h           ;
        jp      c,csts          ; jump rom console status
        jp      z,lsts          ; jump printer status
        ret                      ; no device, now ret 11000000b
        jr      notdev          ; jump no device
;
;
page

```

EC2A

EC2A C9

EC2B

EC2B 3E 1A

EC2D C9

EC2E

EC2E 3E 81

EC30 32 0003

EC33 11 EE3D

EC36 CD F01E

EC39 C3 EB1D

```

;
;*****
;* SERIAL DEVICES Subroutine *
;*****
;
;
;
;*****
;* P u n c h *
;*   Puncher output *
;*****
;
;
punch:
    if     PLIN          ; if PUNcher exists
    ld     a,(iobyte)    ; load intel i/o byte
    and    00110000b    ; mask bit 4,5
    cp     00010000b    ;
    jp     c,cout        ; = TTY: jump rom console output
                        ; = PTR:
    jp     nz,notdev     ; else no device exist
    jp     0000          ; spare jump
    ; start of PTR: dev subroutine
    ret
    else          ; no puncher devices
    ret          ; data lost
    endif
;
;
;*****
;* R e a d e r *
;*   Reader input *
;*****
;
;
reader:
    if     RDR          ; if ReaDeR exists
    ld     a,(iobyte)    ; load intel i/o byte
    and    00001100b    ; mask bit 2,3
    cp     00000100b    ;
    jp     c,cin        ; = TTY: jump rom console input
                        ; = PTR:
    jr     nz,notdev     ; else no device exists
    ; start of PTR: dev subroutine
    jp     0000          ; spare jump
    ELSE          ; if no device
    ld     a,'Z'-'@'     ; set ^z = EOF
    ret          ; end
    endif
;
;
notdev:
    ; print not device message and go to cpm
    ld     a,DftI.O      ; set default i/o byte
    ld     (iobyte),a    ;
    ld     de,ndevmsg     ; D.E = no device msg
    call   strout         ; print it
    jp     wboot          ; return to cp/m
;

```


page ← *copy here*

```

;
;*****
;*          Disk I/O Subroutine          *
;*****
;
;
;*****
;* S e l D s k                          *
;*          Select logical disk from reg. C      *
;*          Ret HL=DPB or 0 if error            *
;*****
;
SelDsk:
        ld    hl,0          ; return 0000h if error
        ld    a,c           ;
        cp    maxdsk        ; too large ?
        ret    nc           ; leave HL = 0000
;
        ld    a,(sysflag)   ; load system flag
        or    a             ; if system flag = 0 then disk
        ld    a,c           ; restore disk # on a
        jr    z,SDsk.1      ; A,B = hard disk; C,D = floppy disk
        cp    wddsiz+fddsiz ; Disk # > D:
        jr    nc,SDsk.1     ; yes, no exchange
        xor    00000010b    ; A,B -> C,D and vice-versa
                          ; A,B = floppy disk; C,D = hard disk

SDsk.1:
        ld    (LogDsk),a    ; set logical disk number
        ld    l,a           ; L = disk number
        rept 4
        add    hl,hl        ; HL = disk number * 16
        endm
        add    hl,hl        ; HL = disk number * 16
        add    hl,hl        ; HL = disk number * 16
        add    hl,hl        ; HL = disk number * 16
        ld    de,dpbase    ;
        add    hl,de        ; H.L disk table adrs
        ret
;
page

```

EC3C
 EC3C 21 0000
 EC3F 79
 EC40 FE 06
 EC42 D0

EC43 3A EE4E
 EC46 B7
 EC47 79
 EC48 28 06
 EC4A FE 04
 EC4C 30 02
 EC4E EE 02

EC50
 EC50 32 EE60
 EC53 6F

EC54 29 +
 EC55 29 +
 EC56 29 +
 EC57 29 +
 EC58 11 EA33
 EC5B 19
 EC5C C9

```

;
;
;*****
;* H O M E *
;*          Select logical track 0 *
;*****
;
;
EC5D      Home:
EC5D      01 0000          ld      bc,0          ; Track #0000
;
;
;*****
;* S e t T r k *
;*          Select logical track from reg.s BC *
;*****
;
EC60      SetTrk:
;
EC60      ED 43 EE62      ld      (LogTrk),bc    ; Save low and high byte
EC64      C9              ret                    ;
;
;
;*****
;* S e c T r a n *
;*          Translate the BC sector using trans *
;*          table pointed by DE *
;*****
;
EC65      SecTran:
EC65      EB              ex      de,hl          ; H.L = sectran table adrs
EC66      7D              ld      a,l          ; check for -> 0000
EC67      B4              or      h            ; this means no sec tran
EC68      09              add     hl,bc         ; compute sector (BC = sec num)
EC69      28 04           jr      z,Strn_5      ; no sec tran
EC6B      6E              ld      l,(hl)        ; get trans sector
EC6C      26 00           ld      h,0          ; high = 0
EC6E      C9              ret                    ; done
EC6F      2C              Strn_5: inc     l      ; convert to base 1
EC70      C9              ret
;
;
;*****
;* S e t S e c *
;*          Set sector from registers BC *
;*****
;
EC71      SetSec:
EC71      79              ld      a,c          ; Only low byte
EC72      32 EE5F         ld      (LogSec),a    ; because sector < 256
EC75      C9              ret
;
;
;*****
;* S e t D M A *
;*          Set DMA address from registers BC *
;*****

```

```
EC76      ;  
EC76      SetDma:  
EC76      ED 43 EE65      ld      (LogDMA),bc      ; set logical DMA  
EC7A      C9              ret  
          ;  
          page
```

EC7B

EC7B AF

EC7C 0E 02

EC7E 18 02

```

;
;*****
;* R e a d                                     *
;*          Read sector specified by prev param *
;*          @ spec DMA (ret A=-1 if error)      *
;*****
;
read:
        xor     a                ; set disk read operation
        ld      c,wrua1         ; write type (to unallocated)
        jr      rw00            ;
;
;
;*****
;* W r i t e                                     *
;*          Write sector specified by prev param *
;*          from spec DMA (ret A=-1 if error)    *
;*****
;
write:
        ld      a,1             ; set write operation
rw00:   ld      (LogR.W),a       ; set read or write operation
        ld      de,LogDsk       ; DE. LogDsk
        ld      a,(de)          ; A = Logical Disk number
        cp      wddsiz+fdssiz   ; check for 256 byte/sec dsk
        jr      c,RW256         ; yes, jump to it
;
;
;*****
;* R W 1 2 8 - Read o Write 128 byte/sec dsk *
;*          Write pending sectors              *
;*          just read or write sector          *
;*          set no sector buffered              *
;*****
;
RW128:
        and     1               ; mask bit 1 for unit select
        ld      (PhyDsk),a      ; set Disk Unit & Side 0
        call    WrtPng          ; Write Pending Sectors
        ld      a,(LogSec)      ;
        ld      (PhySec),a      ; Physical sector = Logical
;
RW128.1:
        ld      hl,PhyDsk       ; Point to Operation Table
        call    fdios           ; read o write 128 byte
        ld      hl,PreDsk       ; Point to Sector Buffered tbl
        ld      (hl),Offh       ; set no sector buffered
        or      a               ; fdd i/o error ?
        ret     z               ; no, then normal return
        ld      de,ioerrmsg     ; D.E = Disk err message
        call    strout          ; print it
        call    cin             ; wait one char.
        cp      cr              ; is return ?
        jr      z,RW128.1       ; yes, then retry
        cp      'C'- '@'        ; in cntrl C ?
        jp      z,wboot         ; yes, wboot
        ld      a,1             ; Set Error

```

EC80

EC80 3E 01

EC82 32 EE67

EC85 11 EE60

EC88 1A

EC89 FE 04

EC8B 38 31

```

;
;
;*****
;* R W 1 2 8 - Read o Write 128 byte/sec dsk *
;*          Write pending sectors              *
;*          just read or write sector          *
;*          set no sector buffered              *
;*****
;
RW128:
        and     1               ; mask bit 1 for unit select
        ld      (PhyDsk),a      ; set Disk Unit & Side 0
        call    WrtPng          ; Write Pending Sectors
        ld      a,(LogSec)      ;
        ld      (PhySec),a      ; Physical sector = Logical
;
RW128.1:
        ld      hl,PhyDsk       ; Point to Operation Table
        call    fdios           ; read o write 128 byte
        ld      hl,PreDsk       ; Point to Sector Buffered tbl
        ld      (hl),Offh       ; set no sector buffered
        or      a               ; fdd i/o error ?
        ret     z               ; no, then normal return
        ld      de,ioerrmsg     ; D.E = Disk err message
        call    strout          ; print it
        call    cin             ; wait one char.
        cp      cr              ; is return ?
        jr      z,RW128.1       ; yes, then retry
        cp      'C'- '@'        ; in cntrl C ?
        jp      z,wboot         ; yes, wboot
        ld      a,1             ; Set Error

```

EC8D

EC8D E6 01

EC8F 32 EE61

EC92 CD ED3F

EC95 3A EEF5

EC98 32 EE64

EC9B

EC9B 21 EE61

EC9E CD F012

ECA1 21 EE68

ECA4 36 FF

ECA6 B7

ECA7 C8

ECA8 11 EDFC

ECA8 CD F01E

ECAE CD F003

ECB1 FE 0D

ECB3 28 E6

ECB5 FE 03

ECB7 CA EB1D

ECBA 3E 01

```

;
;*****
;* R W 1 2 8 - Read o Write 128 byte/sec dsk *
;*          Write pending sectors              *
;*          just read or write sector          *
;*          set no sector buffered              *
;*****
;
RW128:
        and     1               ; mask bit 1 for unit select
        ld      (PhyDsk),a      ; set Disk Unit & Side 0
        call    WrtPng          ; Write Pending Sectors
        ld      a,(LogSec)      ;
        ld      (PhySec),a      ; Physical sector = Logical
;
RW128.1:
        ld      hl,PhyDsk       ; Point to Operation Table
        call    fdios           ; read o write 128 byte
        ld      hl,PreDsk       ; Point to Sector Buffered tbl
        ld      (hl),Offh       ; set no sector buffered
        or      a               ; fdd i/o error ?
        ret     z               ; no, then normal return
        ld      de,ioerrmsg     ; D.E = Disk err message
        call    strout          ; print it
        call    cin             ; wait one char.
        cp      cr              ; is return ?
        jr      z,RW128.1       ; yes, then retry
        cp      'C'- '@'        ; in cntrl C ?
        jp      z,wboot         ; yes, wboot
        ld      a,1             ; Set Error

```


ECBC B7
ECBD C9

or a
ret

; set flag
; ret with operation status on A

;

page

```

;
;*****
;* R W 2 5 6 - Read o Write 256 byte/sec dsk *
;*****
;
;
ECBE RW256:
ECBE 26 40 ld h,wddspt ; if disk number is 0 or 1
ECC0 FE 02 cp wddsiz ; then H = wdd sector/track
ECC2 38 02 jr c,R256.1 ;
ECC4 26 14 ld h,fddspt ; else H = fdd sector/track
ECC6 R256.1:
ECC6 79 ld a,c ; get &
ECC7 32 EE72 ld (WrType),a ; set CP/M write type
ECCA 1B dec de ; DE. LogSec
ECCB 1A ld a,(de) ; Get Logical Sector
ECCC 3D dec a ; to base 0
ECCD 2E 00 ld l,0 ; initial side = 0
ECCF R256.2:
ECCF BC cp h ; repeat until
ECD0 38 04 jr c,R256.3 ; log sec ( sec/trk
ECD2 2C inc l ; side up
ECD3 94 sub h ; log sec = log sec - sec/trk
ECD4 18 F9 jr R256.2 ; retry
ECD6 R256.3:
ECD6 B7 or a ; carry = 0
ECD7 1F rra ; A = A/2
ECD8 3C inc a ; to base 1
ECD9 32 EE64 ld (PhySec),a ; Set physical sector
rept 4 ; move side number
sla l ; to bit 4
endm ; L = side number
EDEC CB 25 + sla l ; to bit 4
EDEE CB 25 + sla l ; to bit 4
EDE0 CB 25 + sla l ; to bit 4
ECE2 CB 25 + sla l ; to bit 4
ECE4 13 inc de ; DE. LogDsk
ECE5 1A ld a,(de) ; get LogDsk
ECE6 E6 01 and 1 ; only unit number
ECE8 B5 or 1 ; merge side
ECE9 32 EE61 ld (PhyDsk),a ; set unit and side
;
EDEC 06 05 ld b,5 ; byte count for old-new para compare
; D.E => CP/M Disk para (new)
; H.L => Disk para (old)
EDEE 21 EE68 ld hl,PreDsk
ECF1 rw01:
; compare old para with new para (dsk,sid,trk,sec)
ECF1 1A ld a,(de) ; A = new para
ECF2 BE cp (hl) ; (hl) = old para
ECF3 20 06 jr nz,wtchk ; new () old
ECF5 23 inc hl ; next para adrs
ECF6 13 inc de ;
ECF7 10 F8 djnz rw01 ; repeat until end para
ECF9 18 14 jr match ; dsk,sid,trk,sec equ
ECFB wtchk:
ECFB CD ED3F call WrtPng ; Write Pending Sectors
ECFE C0 ret nz ; return if error

```

```

;
ECFF 01 0005      ld    bc,5          ; 5 parameters
ED02 21 EE60      ld    hl,LogDsk     ; H.L = new para adrs
ED05 11 EE68      ld    de,PreDsk     ; D.E = old para adrs
ED08 ED B0        ldir              ; new para -> old para
ED0A CD ED4B      call   diskrd       ; disk read
ED0D B7           or     a            ; read error ?
ED0E C0           ret    nz          ; error return
ED0F
match:
ED0F 3A EE5F      ld    a,(LogSec)    ; load logical sector
ED12 3D           dec    a            ; convert to base 0
ED13 E6 01        and    secmsk       ; sector mask
ED15 67           ld    h,a
ED16 2E 00        ld    l,0          ; get high or low buff adrs
ED18 CB 3C        srl    h            ; HL=HL*128=(*256)/2
ED1A CB 1D        rr     l
ED1C 11 F000      ld    de,defbuf     ; D.E = phys sector buff start adrs
ED1F 19           add    hl,de        ; H.L = log sector buff start adrs
ED20 ED 5B EE65   ld    de,(LogDma)   ; D.E = user dma adrs
ED24 01 0080      ld    bc,128       ; BC = moving count
ED27 3A EE67      ld    a,(LogR.W)    ; load r/w flag
ED2A B7           or     a            ; read ?
ED2B 28 04        jr     z,rwbuf      ;
ED2D 32 EE71      ld    (WrtFlg),a    ; write flag on (A=1)
ED30 EB           ex     de,hl        ; H.L = user dma adrs
ED31
rwbuf:
ED31 ED B0        ldir              ; move (hl) to (de)
ED33 3A EE72      ld    a,(WrType)    ; load write type
ED36 FE 01        cp     wrdir        ; directory write ?
ED38 3E 00        ld    a,0          ; prepare no errors
ED3A CC ED3F      call   z,WrtPng     ; yes, write Phys sector
ED3D B7           or     a            ; set flags
ED3E C9           ret                ; return status (A)
;
page

```

ED3F
ED3F 21 EE71
ED42 7E
ED43 36 00
ED45 B7
ED46 C8
ED47 CD ED4E
ED4A C9

```

;
;
;*****
;* W r t P n g *
;*          Check for pending Sectors *
;*          Write if active *
;*****
;
WrtPng:
    ld    hl,WrtFlg
    ld    a,(hl)        ; get flag
    ld    (hl),0        ; & clear
    or    a              ; was active ?
    ret    z             ; no, return
    call  diskwt         ; yes, write flush data
    ret                    ; return status & flag
;
;
;*****
;* D i s k R d *
;*          Read Physical Sector *
;*****
;
diskrd:
    ; disk read
    xor    a              ; 0 = read
    jr     rdwt
;
;
;*****
;* D i s k W t *
;*          Write Physical Sector *
;*****
;
diskwt:
    ; disk write
    ld    a,1            ; 1 = write
    rdwt:
    ld    (PreR.W),a      ; set r/w para
    rdwt0:
    ld    hl,PrePhy       ; H.L = i/o para adrs
    ld    a,(PreDsk)      ; load i/o unit number
    cp    wddsiz          ; wdd i/o ?
    jr    nc,fdrdwt       ; no, then fdd i/o
    wdrdwt:
    ld    a,1            ; one sector to wdd i/o
    ld    (PreBlk),a      ; set wdd sector block
    call  wdio            ; exec. wdd i/o
    or    a              ; i/o error ?
    ret    z             ; no, then normal return
    call  SendErr         ; Print Error Code
;
;
NoBuff:
    rdwterr:
    ld    a,Offh         ; set no sector buffered
    ld    (PreDsk),a
;

```

ED4B

ED4B AF
ED4C 18 02

ED4E

ED4E 3E 01
ED50
ED50 32 EE6F
ED53
ED53 21 EE69
ED56 3A EE68
ED59 FE 02
ED5B 30 15
ED5D
ED5D 3E 01
ED5F 32 EE70
ED62 CD F01B
ED65 B7
ED66 C8
ED67 CD ED8B

ED6A
ED6A
ED6A 3E FF
ED6C 32 EE68

```

ED6F E6 01          and 1          ; A=1
ED71 C9            ret
ED72              fdrdw1:
ED72              fdrw1:
ED72 CD F015        call fdiod      ; r/w 256 byte
ED75              fdrw2:
ED75 B7            or a            ; fdd i/o error ?
ED76 C8            ret z          ; no, then normal return
ED77 11 ED7C        ld de,ioerrmsg ; D.E = Disk err message
ED7A CD F01E        call strout     ; print it
ED7D CD F003        call cin       ; wait one char.
ED80 FE 0D          cp cr         ; is return ?
ED82 28 CF          jr z,rdwt0     ; yes, then retry
ED84 FE 03          cp 'C'-'@'     ; in ctrl C ?
ED86 20 E2          jr nz,NoBuff   ; Set Error and no sector buff
ED88 C3 EA03        jp wboot       ; else go to wboot

```

;

page


```

;
;
;*****
;*          Send Error Message on console          *
;*****
;
SendErr:
ED8B      F5          push    af          ; save character
ED8B      F5          rept    4
ED8C      0F          rrca          ; get 4 msb's
ED8D      0F          rrca          ; get 4 msb's
ED8E      0F          rrca          ; get 4 msb's
ED8F      0F          rrca          ; get 4 msb's
ED90      CD EDA4      call    HxChar      ; print 4 msb's
ED93      32 EDF6      ld      (ErrHig),a  ; store high Error code
ED96      F1          pop     af          ; get 4 lsb's
ED97      CD EDA4      call    HxChar
ED9A      32 EDF7      ld      (ErrLow),a  ; store low Error code
ED9D      11 EDEC      ld      de,ErrMsg
EDA0      CD F01E      call    strout      ; print it
EDA3      C9          ret              ; done
;
; Convert A low nibble in Hex ASCII Char
;
EDA4      E6 0F      HxChar: and     0Fh      ; keep 4 lsb's
EDA6      C6 90      add     a,90h      ; develop a supplement of 6
EDAB      27          daa              ; and carry
EDA9      CE 40      adc     a,'@'      ; sum ASCII offset
EDAB      27          daa
EDAC      C9          ret
;
;
;
EDAD      msgcr:
EDAD      CD F01E      ; print string pointed by DE and wait cr
EDB0      call    strout      ; print it
EDB0      waitcr:
EDB0      CD F003      call    cin          ; wait one char.
EDB3      FE 0D      cp      cr          ; cr ?
EDB5      20 F9      jr      nz,waitcr
EDB7      C9          ret
;
; page

```

```
;
;*****
;*                                         *
;*           Initialized RAM data areas          *
;*                                         *
;*-----*
;
nosysmsg:
    defb     cr,lf,bell,'Set system diskette in disk A,',cr,lf

;
;
defb         'ok push return. ',endmsg

;
ErrMsg:
    defb     cr,lf,bell,'Error #'

ErrHig: defb      '0'
ErrLow: defb      '0'
        defb     ' - ',endmsg

;
;
ioerrmsg:
    defb     cr,lf,bell,'DISK I/O ERROR',cr,lf

;
;
defb         '(RETURN) retry, ^C abort, any key to continue'

;
defb         endmsg

;
ndevmsg:
    defb     cr,lf,bell.'.NO Device.',cr,lf,endmsg
```

Bios 1.6 for NE CP/M 2.2 with Hard-Disk Basf 6182
IPL for NE BIOS 1.4 with Hard-Disk BASF 6182

MACRO-80 3.36 17-Mar-80

PAGE 2-24

EE49 65 2E 0D 0A
EE4D 24

;

page

```

;
;
;*****
;*      B i o s  INPUT/OUTPUT  Tables      *
;*****
;
;
EE4E      sysflag:
EE4E      00          defb    0          ; system flag for disk assignment
;
EE4F      vidareas:
          ; video routine data areas
EE4F      defb    16
;
; Logical Parameter Table
;
EE5F      01      LogSec: defb    1          ; CP/M logical Sector number
EE60      00      LogDsk: defb    0          ; CP/M logical Disk number
EE61      00      PhyDsk: defb    0          ; Physical Disk Number
EE62      0000    LogTrk: defw    0000       ; Physical Track Number
EE64      01      PhySec: defb    1          ; Physical Sector Number
EE65      0080    LogDma: defw    0080h      ; CP/M logical Dma address
EE67      00      LogR.W: defb    0          ; CP/M logical R/W Flag
;
; Previous Parameter Table
;
EE68      FF      PreDsk: defb    0ffh       ; Previous CP/M Disk
EE69      00      PrePhy: defb    0          ; Previous Phys Disk
EE6A      0000    PreTrk: defw    0000       ; Previous Phys=logical Track
EE6C      01      PreSec: defb    1          ; Previous Phys Sector
EE6D      F000    PreDma: defw    defbuf     ; Physical DMA add
EE6F      00      PreR.W: defb    0          ; Phys R/W operation
EE70      01      PreBlk: defb    1          ; Phys # of Sectors (for wdd)
EE71      00      WrtFlg: defb    0          ; Write Pending Flag
EE72      01      WrType: defb    1          ; BDos Write Type
;
;
EE73      freeram equ    $
          if    $ lt bios+600h
018D      freebyt equ    bios+600h-$      ; free space on bios ram
EE73      defb    freebyt
          else
          if2
          .printx *** WARNING: BIOS overflow reserved space ***
          endif
          endif
          page

```

```

;
;
;*****
;*
;*          Disk data areas
;*
;*
;*****
;
F000      defbuf: defs      secsiz      ; default i/o dma address
F100      dirbuf: defs      128          ; directory buffer
;
;
;*****
;*          Allocation and check vectors
;*
;*****
;
;          ; wdd alloc and check vector
;
F180      alv0:  defs      152          ; alloc vector 0 (1215K/8)+1
F218      csv0:  defs      0            ; no check vector 0
;
F218      alv1:  defs      153          ; alloc vector 1 (1223K/8)+1
F2B1      csv1:  defs      0            ; no check vector 1
;
;
;          ; fdd alloc and check vector
;
F2B1      alv2:  defs      12            ; alloc vector 2
F2BD      csv2:  defs      16            ; check vector 2
;
F2CD      alv3:  defs      12            ; alloc vector 3
F2D9      csv3:  defs      16            ; check vector 3
;
;
;          ; extfdd alloc and check vector
;
F2E9      alv4:  defs      10            ; alloc vector 4
F2F3      csv4:  defs      8             ; check vector 4
;
F2FB      alv5:  defs      10            ; alloc vector 5
F305      csv5:  defs      8             ; check vector 5
;
; (** 769 bytes **)
O0F3      freelbyt equ      bios+600h+400h-$      ; free areas on data ram
;
F30D      defs      freelbyt      ; free space
;
;          .dephase
;          end      100h      ; end of bios + data areas

```

Macros:

Symbols:

ALV0	F180	ALV1	F218	ALV2	F2B1	ALV3	F2CD
ALV4	F2E9	ALV5	F2FB	BACKSP	0008	BBTDMA	1063
BBTDSK	105F	BBTERM	10A3	BBTERR	102D	BBTOK	1012
BBTSEC	1062	BBTRK	1060	BBTXLT	1067	BDOS	DC06
BELL	0007	BIAS	A000	BIOS	EA00	BIOSL	0600
BIOSI	0006	BOOT	EB17	BOOTRO	F021	BTPRW	1065
CCP	D400	CIN	F003	CMSIZE	003C	COMPFL	F02D
CONIN	EBE8	CONOUT	EBF9	CONST	EBDC	COUT	F006
CPMBLK	0002	CPML	1600	CPMSG	106D	CPMSIZ	0016
CR	000D	CSTS	F009	CSV0	F218	CSV1	F2B1
CSV2	F2BD	CSV3	F2D9	CSV4	F2F3	CSV5	F305
CURDSK	0004	DEFBUF	F000	DEFDMA	0080	DFTI.0	0081
DIRBUF	F100	DISKRD	ED48	DISKWT	ED4E	DPB0	EACC
DPB01	EADB	DPB1	EAEA	DPB12	EAF9	DPB2	EB08
DPBASE	EA33	DPE0	EA33	DPE1	EA43	DPE2	EA53
DPE3	EA63	DPE4	EA73	DPE5	EA83	ENDMSG	0024
ERRHIG	EDF6	ERRLOW	EDF7	ERRMSG	EDEC	EXBOOT	EBD3
EXBOT1	EBD3	EXTFDS	0002	FALSE	0000	FDBB00	1003
FDBBT1	103A	FDBBT2	1044	FDDSEC	000A	FDDSI2	0002
FDDSP1	0014	FDIOD	F015	FDIOS	F012	FDRDWT	ED72
FDRW1	ED72	FDRW2	ED75	FD WB	EB56	FD WB.	EB5B
FFEEED	000C	FLASH	0043	FREBIP	000D	FREE1B	00F3
FREEBY	018D	FREERA	EE73	FREIPL	10F3	H0ME	EC5D
HXCHAR	EDA4	IDBYTE	0003	IDERRM	EDFC	IPLMSG	1006
LF	000A	LIST	EC0E	LISTST	EC1C	LOGDMA	EE65
LOGDSK	EE60	LOGR.W	EE67	LOGSEC	EE5F	LOGTRK	EE62
LOUT	F00C	LST	0002	LSTS	F00F	MATCH	ED0F
MAXDSK	0006	MOVCUR	F027	MSBCR	EDAD	MSIZE	003C
NDEVMS	EE3D	NOBUFF	ED6A	NORM	0040	NOSYSM	EDB8
NOTDEV	EC2E	PFX	0013	PHYDSK	EE61	PHYSEC	EE64
PREBLK	EE70	PREDMA	EE6D	PREDSK	EE68	PREPHY	EE69
PRER.W	EE6F	PRESEC	EE6C	PRETRK	EE6A	PRINT	F01E
PRINTA	F024	PUN	0000	PUNCH	EC2A	R256.1	ECC6
R256.2	ECCF	R256.3	EC06	RDR	0000	RDWT	ED50
RDWTO	ED53	RDWTER	ED6A	READ	EC7B	READER	EC2B
REV	0010	REVER	0048	ROM	F000	RW00	EC82
RW01	ECF1	RW128	EC8D	RW128.	EC9B	RW256	ECBE
RWBUF	ED31	SDSK.1	EC50	SECMSK	0001	SECSIZ	0100
SECTRA	EC65	SELDISK	EC3C	SENDER	ED8B	SETDMA	EC76
SETSEC	EC71	SETTRK	EC60	SPACE	0020	STACK	0080
STACK1	1000	STRN_5	EC6F	STROUT	F01E	SYSCHK	EBB2
SYSFLA	EE4E	TAB	0009	TRUE	00FF	TTY	0001
VER8	4853	VIDARE	EE4F	VIDINI	F02A	WAIT1C	1033
WAITCR	ED80	WBOOT	EB1D	WBOOTE	EA03	WBXLT	EBC9
WB_0	EB2D	WB_1	EB2F	WDBB00	1000	WDBBT1	1009
WDBLOC	1066	WDDSEC	0020	WDDSI2	0002	WDDSP1	0040
WDINI	F018	WDIO	F01B	WORDWT	ED5D	WD WB	EB4B
WRALL	0000	WRDIR	0001	WRITE	EC80	WRTFLG	EE71
WRTPNG	ED3F	WRTPYE	EE72	WRUAL	0002	WCHK	ECFB
XLT0	0000	XLT1	EA93	XLT2	EABB		