AA–P310A–TV

Rainbow™ 100

CP/M–86/80 BIOS Listings

digital equipment corporation

First Printing, March 1983

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

CP/M®, CP/M®–80 and CP/M®–86 are registered trademarks of Digital Research Inc.

Z80® is a registered trademark of Zilog, Inc.

8088® is a registered trademark of Intel Corporation.

The following are trademarks of Digital Equipment Corporation:

digital™

DEC	MASSBUS	UNIBUS
DECmate	PDP	VAX
DECsystem–10	P/OS	VMS
DECSYSTEM–20	Professional	VT
DECUS	Rainbow	Work Processor
DECwriter	RSTS	
DIBOL	RSX	

The postage-prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

Printed in U.S.A.

The following pages contain the CP/M–86/80 operating system BIOS listings. Note the following differences:

- The BIOS has a missing POP instruction in a section of code which is apparently never invoked. It should have been at location 40:2E3C.
- The PRMTVPVT code (actually the Z80code.ASM) has an incorrectly labeled argument in the multi-sector read routine. The BIOS never calls for multi-sectored reads, so it does not normally cause problems. (The operand at location 023A (hex) should be SECTN rather than SECNUM.)
- The diskette read/write code in Z80code.ASM has a timing "window" which can give false "disk not ready" error messages.
- The SUBMIT command does not work correctly from a non-boot drive.
- The transient program area for a 64K Rainbow™ 100 computer is 47–1/2K bytes.

```
CP/M MACRO ASSEM 2.0    #001    PC-100 COLD BOOT (TRACK 0, SECTOR 1)

                ; COLD BOOT PROGRAM FOR DEC PC-100
                ;
                ; LOADS CPL LOADER INTO MEMORY FROM SYSTEM TRACKS
                ; READ INTO SPECIFIED LOCATION BY ROM BOOT
                ;
                ;
                        MACLIB  Z80
                ;
                ; CONDITIONALIZE FOR TRACK 0 OR TRACK 1
                ;
 00FF =         BOOT100 EQU     0FFH    ;(FF=BOOT100, 0=BOOT101)
                IF BOOT100
                        TITLE   'PC-100 COLD BOOT (TRACK 0, SECTOR 1)'
                ENDIF
                IF NOT BOOT100
                        TITLE   'RX-50 STANDARD COLD BOOT (TRACK 1, SECTOR 1)'
                ENDIF
                ;
                ; ERROR AND COMPLETION CODES FOR THE 8088 ROM
                ;
 0008 =         QERCOD  EQU     08H     ;ERROR CODE (CHECK THIS)
 000A =         QSTCOD  EQU     0AH     ;COMPLETION CODE (CHECK THIS)
                ;
                ;
                ; SYSTEM ADDRESSES FOR LOAD AND START
                ;
 0000 =         QLLOAD  EQU     0       ;LOADER OFFSET
 0DA0 =         QLLDSEG EQU     0DA0H   ;LOADER SEGMENT
 DA00 =         QALLOAD EQU     QLLDSEG*16+QLLOAD       ;ABSOLUTE LOAD ADDRESS
 0000 =         QPRIMS  EQU     0       ;PRIMITVES
 0002 =         QLDSEC  EQU     2       ;LOADER SECTOR (MINUS 1)
 0FFF =         QSEMA4  EQU     0FFFH   ;SEMAPHORE ADDRESS
 0FFD =         QLDSSEG EQU     0FFDH   ;LOADER SEGMENT STORED
 0FFB =         QLDSOFF EQU     0FFBH   ;LOADER OFFSET STORED
 0FFA =         QDRNO   EQU     0FFAH   ;STORE DRIVE NUMBER
 0100 =         PRIMST  EQU     100H    ;PRIMITIVE START
                ;
                ; DISK CONSTANTS
                ;
 000A =         QKSECTRK EQU    10      ; SECTORS PER TRACK
 0200 =         QKSECSIZ EQU    512     ; BYTES PER SECTOR
                ;
                ; PC-100 PORT ADDRESSES
                ;
 0060 =         QPCOMD  EQU     60H     ; FDC COMMAND/STATUS REG
 0061 =         QPTRKRG EQU     61H     ; FDC TRACK REG
 0062 =         QPSECRG EQU     62H     ; FDC SECTOR REG
 0063 =         QPDATA  EQU     63H     ; FDC DATA REG
```

```
0040 =          QPSTAT  EQU     40H     ; GENERAL FD STATUS REG
                ;
                ; DISK CONTROLLER COMMANDS
                ;
0008 =          QCREST  EQU     08H     ; RESTORE (HEAD LOAD)
005C =          QCSTEPIN EQU    5CH     ; STEP IN
0080 =          QCREADS EQU     80H     ; READ SECTOR
00D0 =          QCTERM  EQU     0D0H    ; TERMINATE COMMAND (FORCE INT.)
CP/M MACRO ASSEM 2.0    #002    PC-100 COLD BOOT (TRACK 0, SECTOR 1)

                ;
                ; DISK CONTROLLER STATUS - TYPE 1 COMMANDS
                ;
0080 =          QMNRDY  EQU     80H     ; NOT READY
0040 =          QMWPROT EQU     40H     ; WRITE PROTECTED
0020 =          QMHLT   EQU     20H     ; HEAD LOADED
0010 =          QMSKERR EQU     10H     ; SEEK ERROR
0008 =          QMCRC   EQU     8       ; CRC ERROR
0004 =          QMTZERO EQU     4       ; TRACK ZERO
0002 =          QMINDEX EQU     2       ; INDEX
0001 =          QMBUSY  EQU     1       ; CONTROLLER BUSY
                ;
                ; DISK CONTROLLER STATUS - TYPE 2 AND 3 COMMANDS
                ;
0020 =          QMWRFLT EQU     20H     ; WRITE FAULT
0010 =          QMRNF   EQU     10H     ; RECORD NOT FOUND
0004 =          QMLDATA EQU     4       ; LOST DATA
0002 =          QMDRQ   EQU     2       ; DATA REQUEST
                ;
                ; ERROR MASKS FOR OPERATIONS
                ;
009C =          QMREAD  EQU     QMNRDY+QMRNF+QMCRC+QMLDATA ; READ SECTOR OPN.
0098 =          QMSEEK  EQU     QMNRDY+QMSKERR+QMCRC  ; POSITIONING OPN.
                ;
                ; MISCELLANEOUS EQUATES
                ;
1000 =          BOOTORG EQU     1000H           ; WHERE AM I?
1100 =          QLSTACK EQU     BOOTORG+100H    ; STACK LOCATION
                ;
                ;
                ;*************************************************
                ;*      ASSERT Z-FLIP, THEN GO TO THE LOADER     *
                ;*      WITH A PCHL COMMAND                      *
                ;*************************************************

8000 =          FLIP    EQU     8000H
00E9 =          GOWAN   EQU     PCHL
0021 =          ZFLIP   EQU     21H
1000            ORG BOOTORG
                IF NOT BOOT100
                ;*************************************************
                ;*      RX-50 STANDARD PREFIX BYTES              *
                ;*************************************************
                ;
                        DB      0                       ; ** TYPE 2 BOOT BLOCK **
```

```
                        DB      0
                        DB      3                       ; ** OFFSET TO IDENT BLOCK (WORDS)
                                                        ;       (6 BYTES, N=6) **
                        DB      1                       ; ** SYSTEM=1, DATA=0 **
                        DB      0
                DB      0
                        DB      40H                     ; ** 100 OCTAL = Z80 **
                        DB      18H                     ; ** 30 OCTAL = RAINBOW 100 **
                        DB      40H                     ; ** 100 OCTAL = CP/M **
                        DB      67H                     ; ** CHECKSUM FOR 96H **
                        DB      0
CP/M MACRO ASSEM 2.0    #003    PC-100 COLD BOOT (TRACK 0, SECTOR 1)


                ENDIF   ;START HERE ...
 1000 F3                DI                              ;NO INTERRUPTS, PLEASE
 1001 3EE9              MVI A,GOWAN                     ;LOAD PCHL INSTRUCTION
 1003 320D10            STA RSTSPOT                     ;MOVE IT OUT
 1006 210E10            LXI H,BOOT00                    ;GET BOOT START IN HL
 1009 3EF9              MVI A,0F9H                      ;TURN OF L.E.D.S
 100B D321              OUT ZFLIP                       ;FLIP
 100D E9        RSTSPOT:        PCHL                    ;GO TO BOOT

                ;=============================================================
                ;
                ;
                ;
                BOOT00:
 100E 310011            LXI     SP,QLSTACK      ; SET UP STACK POINTER
                BOOT10:
 1011 DB60              IN      QPCOMD          ; GET FDC STATUS
 1013 E601              ANI     QMBUSY          ; READY FOR COMMAND? (QMNRDY+)
                        JRNZ    BOOT10          ;   NOT YET - WAIT
 1015+20FA              DB      20H,BOOT10-$-1
                ;
                ; MAIN LOOP -- RESTORE DISK AND INITIALIZE FOR READING
                ;
 1017 DB40              IN      QPSTAT          ; GET DRIVE NUMBER
 1019 E603              ANI     3               ; MASK OUT OTHER BITS
 101B 32FA0F            STA     QDRNO           ; STORE IT FOR LATER
 101E 3E08              MVI     A,QCREST        ;
 1020 CD8C10            CALL    BOOTCMD         ; RESTORE DISK
 1023 3E02              MVI     A,QLDSEC        ;; FIRST SECTOR
 1025 32B410            STA     BOOTSEC         ;; STORE
 1028 0614              MVI     B,QKSECTRK*2    ;; SECTORS IN TWO TRACKS
 102A 0E63              MVI     C,QPDATA        ;; PORT NUMBER
 102C 21B510            LXI     H,ADTAB         ;; LOAD ADDRESS TABLE START
 102F 5E        BOOT11: MOV     E,M             ;; GET LOAD ADDRESS FORM TABLE
 1030 23                INX     H               ;;
 1031 56                MOV     D,M             ;; HIGH ORDER ADDRESS
 1032 23                INX     H               ;;
 1033 E5                PUSH    H               ;; SAVE ADDRESS FOR LATER
 1034 EB                XCHG                    ;; ADDRESS IN HL
 1035 C5                PUSH    B               ;; SAVE COUNT AND PORT NUMBER
 1036 7D                MOV     A,L             ;; LOOK AT LOW-ORDER
```

```
 1037 FEFF              CPI     0FFH            ;; LOW ORDER ILLEGAL (FLAGGED)?
                        JRZ     BOOT12          ;; SKIP IF SO
 1039+2806              DB      28H,BOOT12-$-1
 103B 3AB410            LDA     BOOTSEC         ;;
 103E CD6110            CALL    BOOT20          ;; LOAD A SECTOR
 1041 3AB410    BOOT12: LDA     BOOTSEC         ;; GET THE SECTOR NUMBER
 1044 3C                INR     A               ;; BUMP IT
 1045 32B410            STA     BOOTSEC         ;; STORE IT
 1048 C1                POP     B               ;; RETURN REGISTERS
 1049 E1                POP     H               ;;
 104A 05                DCR     B               ;; BUMP THE COUNT
                        JRNZ    BOOT11          ;; LOOP BACK UNTIL DONE
 104B+20E2              DB      20H,BOOT11-$-1
 104D 210000            LXI     H,QLLOAD        ; LOADER START
CP/M MACRO ASSEM 2.0    #004    PC-100 COLD BOOT (TRACK 0, SECTOR 1)

 1050 22FB0F            SHLD    QLDSOFF         ; STORED OFFSET
 1053 21A00D            LXI     H,QLLDSEG       ; LOADER SEGMENT
 1056 22FD0F            SHLD    QLDSSEG         ; TO INFO WORD
 1059 3E0A              MVI     A,QSTCOD        ; COMPLETION CODE
 105B 32FF0F            STA     QSEMA4          ; FLAG 8088 FOR COMPLETION
 105E C30001            JMP     PRIMST          ; GO START PRIMITIVES
                ;
                ; PRIMARY SUBROUTINE - READS AND LOADS SECTORS
                ;
 1061 FE0B      BOOT20: CPI     QKSECTRK+1      ;; HAVE WE GONE PAST LAST TRACK?
                        JRC     BOOT21          ;; JUMP AHEAD IF NOT
 1063+3813              DB      38H,BOOT21-$-1
 1065 D60A              SUI     QKSECTRK        ;; RESET TO LOW SECT FOR NEXT TRACK
 1067 32B410            STA     BOOTSEC         ;; RE-STORE IT
 106A 3E5C              MVI     A,QCSTEPIN      ;; STEP IN TO NEXT TRACK
 106C CD8C10            CALL    BOOTCMD         ;;
 106F DB60              IN      QPCOMD          ;; GET FDC STATUS
 1071 E698              ANI     QMSEEK          ;; ERROR?
                        JRNZ    BOOT90          ;; GO TAKE CARE OF IT IF SO
 1073+200D              DB      20H,BOOT90-$-1
 1075 3AB410            LDA     BOOTSEC         ;; GET SECTOR NUMBER

 1078 CD9710    BOOT21: CALL    BOOTREAD        ;; READ A SECTOR
 107B DB60              IN      QPCOMD          ;; GET FDC STATUS
 107D E69C              ANI     QMREAD          ;; READ ERROR?
                        JRNZ BOOT90             ;; TAKE CARE OF IT IF SO
 107F+2001              DB      20H,BOOT90-$-1
 1081 C9                RET                     ;; ELSE RETURN
                ;
                ; HANDLE ERROR CONDITIONS - RESTART BOOT
                ;
                BOOT90:
 1082 3ED0              MVI     A,QCTERM
 1084 D360              OUT     QPCOMD          ; TERMINATE COMMAND
 1086 3E08              MVI     A,QERCOD        ; ERROR CODE
 1088 32FF0F            STA     QSEMA4          ; IN SEMAPHORE LOCATION
 108B 76                HLT                     ; GIVE UP
                ;
                ; SUBROUTINE BOOTCMD
```

```
                ;    SEND COMMAND (NO DATA) TO FDC AND WAIT FOR COMPLETION
                ;    COMMAND IS IN REG. A
                ;
                BOOTCMD:
 108C D360              OUT     QPCOMD          ; SEND COMMAND TO FDC
 108E CDAD10            CALL    DLY28M          ; DELAY ABOUT 28 MICROSECONDS
                BOOTCMD1:
 1091 DB40              IN      QPSTAT          ; GET GENERAL STATUS
 1093 87                ADD     A               ; SHIFT INT. BIT TO SIGN
 1094 F8                RM                      ; RETURN ON INTERRUPT
                        JR      BOOTCMD1        ; WAIT FOR COMPLETION
 1095+18FA              DB      18H,BOOTCMD1-$-1
                ;
                ;
                ; SUBROUTINE BOOTREAD
                ;    READ A SECTOR INTO (HL)
                ;    SECTOR NUMBER IS IN REG. A
CP/M MACRO ASSEM 2.0    #005    PC-100 COLD BOOT (TRACK 0, SECTOR 1)

                ;
                BOOTREAD:
 1097 D362              OUT     QPSECRG         ; WRITE SECTOR NUMBER
 1099 CDB010            CALL    DLY14M          ; DELAY ABOUT 14 MICROSECONDS
 109C 3E80              MVI     A,QCREADS
 109E D360              OUT     QPCOMD          ; SEND READ SECTOR COMMAND
 10A0 CDAD10            CALL    DLY28M          ; DELAY ABOUT 28 MICROSECONDS
                BOOTRD10:
 10A3 DB40              IN      QPSTAT          ; GET GENERAL STATUS
 10A5 87                ADD     A               ; SHIFT INT. BIT TO SIGN
 10A6 F8                RM                      ; RETURN ON INTERRUPT
                        JRNC    BOOTRD10        ; LOOP IF NO DRQ
 10A7+30FA              DB      30H,BOOTRD10-$-1
                        INI                     ; LOAD A BYTE
 10A9+EDA2              DB      0EDH,0A2H
                        JR      BOOTRD10        ; GO WAIT FOR NEXT BYTE
 10AB+18F6              DB      18H,BOOTRD10-$-1
                ;
                ; RECURSIVE SUBROUTINES TO DELAY 7,14, OR 28 MICROSECONDS (APPROXIMATELY)
                ;       BASED ON THE APPROXIMATION THAT A CALL AND A RETURN TAKE ABOUT
                ;       7 MICROSECONDS.
 10AD CDB010    DLY28M: CALL    DLY14M          ; 28 MICROSECONDS
 10B0 CDB310    DLY14M: CALL    DLY7M           ; 14 MICROSECONDS
 10B3 C9        DLY7M:  RET                     ; 7 MICROSECONDS
                ;
                ;
                ; TEMPORARY STORAGE
                ;
 10B4           BOOTSEC: DS     1               ; CURRENT SECTOR NUMBER
                ;
 10B5 =         ADTAB   EQU     $               ;ADDRESS TABLE
 10B5 00DA              DW      QALLOAD                                 ; 1 LOADER START
 10B7 00DC              DW      QALLOAD+QKSECSIZ                        ; 2
 10B9 00DE              DW      QALLOAD+QKSECSIZ*2                      ; 3
 10BB 00E0              DW      QALLOAD+QKSECSIZ*3                      ; 4
 10BD 00E2              DW      QALLOAD+QKSECSIZ*4                      ; 5
```

```
 10BF 00E4              DW      QALLOAD+QKSECSIZ*5                      ; 6
 10C1 00E6              DW      QALLOAD+QKSECSIZ*6                      ; 7
 10C3 00E8              DW      QALLOAD+QKSECSIZ*7                      ; 8
 10C5 00EA              DW      QALLOAD+QKSECSIZ*8                      ; 9
 10C7 FFFF              DW      0FFFFH                                  ; SECOND BOOT
 10C9 00EC              DW      QALLOAD+QKSECSIZ*9                      ; 10
 10CB 00EE              DW      QALLOAD+QKSECSIZ*10                     ; 11
 10CD 00F0              DW      QALLOAD+QKSECSIZ*11                     ; 12
 10CF 00F2              DW      QALLOAD+QKSECSIZ*12                     ; 13
 10D1 00F4              DW      QALLOAD+QKSECSIZ*13                     ; 14
 10D3 00F6              DW      QALLOAD+QKSECSIZ*14                     ; 15 LOADER END
 10D5 0000              DW      QPRIMS                                  ; 16 PRIMITIVE START
 10D7 0002              DW      QPRIMS+QKSECSIZ                         ; 17
 10D9 0004              DW      QPRIMS+QKSECSIZ*2                       ; 18
 10DB 0006              DW      QPRIMS+QKSECSIZ*3                       ; 19 PRIMITIVE END
 10DD FFFF              DW      0FFFFH                                  ; NULL (JIC)
 10DF FFFF              DW      0FFFFH                                  ; NULL (JIC)
 10E1 FFFF              DW      0FFFFH                                  ; NULL (JIC)
 10E3 FFFF              DW      0FFFFH                                  ; NULL (JIC)
 10E5 FFFF              DW      0FFFFH                                  ; NULL (JIC)
CP/M MACRO ASSEM 2.0    #006    PC-100 COLD BOOT (TRACK 0, SECTOR 1)

 10E7 FFFF              DW      0FFFFH                                  ; NULL (JIC)

                ;
 10E9                   END     BOOT00
CP/M MACRO ASSEM 2.0    #007    PC-100 COLD BOOT (TRACK 0, SECTOR 1)

10B5 ADTAB      0000 BC         100E BOOT00     00FF BOOT100    1011 BOOT10
102F BOOT11     1041 BOOT12     1061 BOOT20     1078 BOOT21     1082 BOOT90
108C BOOTCMD    1091 BOOTCMD1   1000 BOOTORG    10A3 BOOTRD10   1097 BOOTREAD
10B4 BOOTSEC    0002 DE         10B0 DLY14M     10AD DLY28M     10B3 DLY7M
8000 FLIP       00E9 GOWAN      0004 HL         0004 IX         0004 IY
0100 PRIMST     DA00 QALLOAD    0080 QCREADS    0008 QCREST     005C QCSTEPIN
00D0 QCTERM     0FFA QDRNO      0008 QERCOD     0200 QKSECSIZ   000A QKSECTRK
0002 QLDSEC     0FFB QLDSOFF    0FFD QLDSSEG    0DA0 QLLDSEG    0000 QLLOAD
1100 QLSTACK    0001 QMBUSY     0008 QMCRC      0002 QMDRQ      0020 QMHLT
0002 QMINDEX    0004 QMLDATA    0080 QMNRDY     009C QMREAD     0010 QMRNF
0098 QMSEEK     0010 QMSKERR    0004 QMTZERO    0040 QMWPROT    0020 QMWRFLT
0060 QPCOMD     0063 QPDATA     0000 QPRIMS     0062 QPSECRG    0040 QPSTAT
0061 QPTRKRG    0FFF QSEMA4     000A QSTCOD     100D RSTSPOT    0021 ZFLIP
```

```
        title  'CP/M-86 Loader'

;       This module revised for DEC RAINBOW 100
;                       by CPL
;                       July 1982
;
;       The CPMLDR consists of this module along with the
;       LDRBDOS and LBIOS.

;       CPMLDR is loaded by a load routine which resides
;       on the first sector of the first track of a CP/M-86/80
;       system diskette.  CPMLDR itself resides on the
;       remainder of the first track and on the next track.

;       CPMLDR first opens the file 'CPM.SYS' using the
;       LDRBDOS and LBIOS and then reads it into memory.
;       Pointers and the memory region table
;       are then initialized.  Finally a jump to the BIOS
;       initialization entry point starts CP/M-86/80.

;       The first 128 byte record of the CPM.SYS file is a header
;       with the following format:
```

```
                ;       ty      rb      1               ;seg type (not used here)
                ;       len     rw      1               ;length (not used here)
                ;       abs     dw      ldrstart        ;absolute segment address for LOADER
                ;       min     rw      1               ;minimum mem (not used here)
                ;       max     rw      1               ;max mem needed (not used here)

                ;       (This header record is constructed automatically by the
                ;       GENCMD utility).

                ;       CPMLDR may be read into any segment that does not
                ;       overlap the desired system load segment as it makes
                ;       all memory references using copies of the CS: register
                ;       it is entered with.

0000            false           equ     0
FFFF            true            equ     not false

000D            cr              equ     0dh
000A            lf              equ     0ah


0040            CPM_SEG         EQU     0040H           ; STARTING SEGMENT OF
                                                        ;   CP/M-86/80 (CPM.SYS)
3600            CPM_LEN         EQU     3600H           ; MAX LENGTH (IN BYTES) OF
                                                        ;   CP/M-86/80
3A00            PTRSBUFS_ADR    EQU     (CPM_SEG*16)+CPM_LEN
                                                        ; STARTING ADDRESS OF
```

```
                                                        ;   POINTERS/BUFFERS DATA BLOCK
   0300                 PTRSBUFS_LEN    EQU     0300H       ; LENGTH (IN BYTES) OF
CP/M ASM86 1.1  SOURCE: CPLLDCPM.A86  CP/M-86 Loader


                                                        ;   POINTERS/BUFFERS DATA BLOCK
   03D0                 STARTTPA_SEG    EQU     (PTRSBUFS_ADR+PTRSBUFS_LEN)/16
                                                        ; STARTING SEGMENT OF TPA


   0FFA                 BOOTDRV_ADR     EQU     0FFAH       ; ADDRESS OF BYTE CONTAINING
                                                        ;   BOOT DRIVE# (FROM CPL BOOT)

CP/M ASM86 1.1  SOURCE: CPLLDCPM.A86  CP/M-86 Loader


                        EJECT

   0406                 LBDOSOFF        EQU     0406H       ; OFFSET OF LOADER BDOS
   1200                 LBIOS_OFFSET    EQU     1200H       ; OFFSET OF LOADER BIOS
   2500                 BIOSOFF         EQU     2500H       ; OFFSET OF BIOS --
                                                        ;   THIS IS THE ENTRY POINT INTO CPM


=                               INCLUDE DEFBUF.LIB
=                       ; ************************************************************************
=                       ; OFFSETS FROM START OF POINTERS/BUFFERS DATA BLOCK
```

```
=
=  FFA0                 XDPCX           EQU     -60H            ; DISK PARAMETER STORAGE (60H)
=  0000                 XDEFBUF         EQU     000H            ; MISC. BUFFER (LENGTH=80H)
=  0086                 XPACKET         EQU     086H            ; BIOS MESSAGE PACKET (LENGTH=0EH)
=  0086                 XSTPKT          EQU     086H            ; START PACKET BUFFER (LENGTH=0EH)
=  0094                 XADCPKT         EQU     094H            ; DATA PACKET (LENGTH=0EH)
=  0094                 XMVPKT          EQU     094H            ; MOVE PACKET BUFFER (LENGTH=0EH)
=  00A2                 XSHRBUF         EQU     0A2H            ; SEGMENT BUFFER (LENGTH=200H)
=  02F8                 XMEMSIZE        EQU     2F8H            ; MEMORY SIZE (WORD)
=  02FA                 XPCPMADR        EQU     2FAH            ; PSEUDO CP/M ADDRESS (WORD)
=  02FC                 XZ80PKT         EQU     2FCH            ; PACKET POINTER FROM Z80 (WORD)
=  02FE                 XI88PKT         EQU     2FEH            ; PACKET POINTER FROM 8088 (WORD)
=  02F0                 XTTRACK         EQU     2F0H            ;TRACK TABLE
=  02F4                 XTFORMAT        EQU     2F4H            ; FORMAT TABLE
=  02E7                 XCSFLAG         EQU     2E7H            ; CONSOLE STATUS FLAG
=
=                       ; OFFSETS FROM ZOT FOR CONVENIENCE
=  0000                 ZOTP            EQU     0               ; Z80 FLAG
=  FFFE                 Z80FLAGPT       EQU     -2              ; Z80-RUNNING FLAG
=  FFFB                 CICCK           EQU     -5              ; CONSOLE STATUS FLAG CHECK
=
=                       ; OTHER USEFUL EQUATES
=  0002                 BDCS            EQU     2               ; BDOS CHARACTER READY BIT
=  0001                 BIOCS           EQU     1               ; BIOS CONSOLE STATUS BIT
=  0017                 BIOS_JMPS       EQU     23              ; NUMBER OF FUNCTIONS IN JUMP TABLE
=
=                       ; ***************************************************************************
```

```
0045                SEGTBL_OFFSET   EQU     BIOS_JMPS*3     ; OFFSET (FROM START OF BIOS)
                                                            ;   OF WORD WITH OFFSET
                                                            ;   (FROM START OF CPM) TO
                                                            ;   MEMORY SEGMENT TABLE
0047                PBADR_OFFSET    EQU     SEGTBL_OFFSET+2 ; OFFSET (FROM START OF BIOS)
                                                            ;   OF WORD THAT WILL CONTAIN
                                                            ;   ADDRESS OF POINTERS/BUFFERS
                                                            ;   DATA BLOCK

02FA                PCPMADR_OFFSET  EQU     XPCPMADR        ; OFFSET (FROM START OF PTRSBUFS)
                                                            ;   OF WORD CONTAINING ADDRESS
                                                            ;   OF PSEUDO CP/M
```



```
                                                            ;   0=PSEUDO CP/M NOT LOADED
02F8                MEMSIZE_OFFSET  EQU     XMEMSIZE        ; OFFSET (FROM START OF PTRSBUFS)
                                                            ;   OF WORD INDICATING
                                                            :   #PARAGRAPHS ADDITIONAL MEMORY


                    ;
                    ; TO DETERMINE MEMORY SIZE
                    ;
1000                FSTMEM_SEG      EQU     1000H           ; START OF OPT ADDITIONAL MEMORY
1000                MEMINC          EQU     1000H           ; SIZE OF MEMORY INCREMENTS
```

```
                                                    ;   (IN PARAGRAPHS)
000E            MEMCNT          EQU     14          ; MAX# MEMORY INCREMENTS
00A5            PATTERN1        EQU     0A5H
0096            PATTERN2        EQU     096H


                ;
                ; INTERRUPTS
                ;
00E0            BDOS_INT        EQU     224         ; LBDOS INTERRUPT NUMBER


                ; bdos function numbers

0002            coutf           equ     2
0009            pstrf           equ     9
000E            seldsk          equ     14
000F            openf           equ     15
0014            readsf          equ     20
001A            dmaf            equ     26
0033            dmabf           equ     51


                ;*******************************
                ;*
                ;*      CPMLDR starts here
                ;*
                ;*******************************
```

```
                                cseg
                                org     0                       ; JMPF to here from boot ROM

0000 E9FD11             1200    jmp     LBIOS                   ; allow loader BIOS to
                                                                ;       initialize

                        start:                                  ; loader BIOS jumps here
0003 33C0                       XOR     AX,AX
0005 8EC0                       MOV     ES,AX
0007 BBFA0F                     MOV     BX,BOOTDRV_ADR          ; GET BOOT DRIVE# ...
000A 268A07                     MOV     AL,ES:[BX]
000D 2EA2B601                   MOV     BOOTDRV,AL              ; ... AND SAVE IT
CP/M ASM86 1.1  SOURCE: CPLLDCPM.A86  CP/M-86 Loader



0011 E8F100             0105    call initlbdos                  ;warm up lbdos and lbios


                        ; READ CPM.SYS
                        ;
0014 E8FA00             0111    call openfnc                    ;open CPM.SYS
0017 3CFF7509           0024    cmp al,255 ! jne perr           ; insure good file
001B BA3601E80501       0126    mov dx,offset nofile ! call msg ; no CPM.SYS file
0021 E90F01             0133    jmp stop                        ;  then halt the machine
                        perr:
```

```
0024 8CCAE8F500    011E         mov dx,cs ! call setdmab
0029 BAD801E8EA00  0119         mov dx,offset page1 ! call setdma       ;read first page of CPM.SYS
002F E8F900        012B         call read                               ; AND IGNORE IT
0032 BA4000E8E600  011E         mov dx,cpm_seg ! call setdmab           ; set DMA segment for disk IO

                                ;
0038 BA0000                     mov dx,0                                ;offset of CPM in segment
                                readit:
003B E8DB00        0119         call setdma                             ; set DMA offset for
003E 52E8E900      012B         push dx ! call read                     ; next sector read
0042 5A                         pop dx
0043 3C017407      004E         cmp al,01H ! je done                    ; check for EOF
0047 81C28000                   add dx,80h                              ; address for next record
004B E9EDFF        003B         jmp readit
                                done:
004E 2E8916B401                 MOV     LDLEN,DX                ; SAVE LENGTH

                                ;
                                ; DETERMINE MEMORY SIZE
                                ;
0053 BB0000                     MOV     BX,0
0056 B80010                     MOV     AX,FSTMEM_SEG
0059 8EC0                       MOV     ES,AX
005B B10E                       MOV     CL,MEMCNT
                                TESTMEM:
005D 26C607A5                   MOV     ES:BYTE PTR [BX],PATTERN1
0061 26803FA5                   CMP     ES:BYTE PTR [BX],PATTERN1
0065 7515          007C         JNE     DONEMEM
```

```
0067 26C60796                   MOV     ES:BYTE PTR [BX]',PATTERN2
006B 26803F96                   CMP     ES:BYTE PTR [BX]',PATTERN2
006F 750B       007C            JNE     DONEMEM
0071 8CC0                       MOV     AX,ES                   ; TRY NEXT MEMORY AREA
0073 050010                     ADD     AX,MEMINC
0076 8EC0                       MOV     ES,AX
0078 FEC9                       DEC     CL
007A 75E1       005D            JNZ     TESTMEM
                        DONEMEM:
007C 8CC0                       MOV     AX,ES
007E 2D0010                     SUB     AX,MEMINC
0081 50                         PUSH    AX                      ; SAVE #PARAGRAPHS ADDITIONAL MEMORY
```



```
                        ;
                        ; UPDATE POINTERS
                        ;
                                ; UPDATE BIOS' SEGMENT TABLE FIRST
0082 2E8E06B201                 MOV     ES,LDSEG
0087 BB4525                     MOV     BX,BIOSOFF+SEGTBL_OFFSET
008A 268B1F                     MOV     BX,ES:[BX]                      ; GET PTR TO SEGMENT TABLE
008D 26C60701                   MOV     ES:BYTE PTR [BX]',1             ; SET FOR 1 SEGMENT
0091 26C74701D003               MOV     ES:WORD PTR 1[BX]',STARTTPA_SEG ; SET START OF 1ST SEG
0097 05300C                     ADD     AX,FSTMEM_SEG-STARTTPA_SEG      ; COMPUTE LEN OF 1ST SEG ...
009A 26894703                   MOV     ES:3[BX]',AX                    ; ... AND SET IT
```

```
                                ; PLACE ADDRESS OF POINTERS/BUFFERS DATA BLOCK FOR BIOS' USE
009E BB4725                     MOV     BX,BIOSOFF+PBADR_OFFSET
00A1 26C707003A                 MOV     ES:WORD PTR [BX],PTRSBUFS_ADR

                                ; INITIALIZE PSEUDO CP/M ADDRESS
00A6 33C0                       XOR     AX,AX
00A8 8EC0                       MOV     ES,AX
00AA BBFA3C                     MOV     BX,PTRSBUFS_ADR+PCPMADR_OFFSET
00AD 26C7070000                 MOV     ES:WORD PTR [BX],0

                                ; SET SYSTEM SIZE
00B2 BBF83C                     MOV     BX,PTRSBUFS_ADR+MEMSIZE_OFFSET
00B5 58                         POP     AX
00B6 268907                     MOV     ES:[BX],AX

00B9 BA8301                     MOV     DX,OFFSET SEGMENT
00BC E86700       0126          CALL    MSG
00BF 2EA1B201                   MOV     AX,LDSEG
00C3 E81A00       00E0          CALL    PHEX                    ; PRINT BASE SYSTEM SEGMENT

00C6 BA9801E85A00 0126          mov dx,offset lenmsg ! call msg         ; print length message
00CC 2EA1B401                   mov     ax,ldlen
00D0 E80D00       00E0          call    phex                      ; print last address
00D3 E84D00       0123          call pcrlf                        ; and a crlf
00D6 2E8A0EB601                 mov     cl,bootdrv                ; pass boot drive#
00DB 2EFF2EB001                 jmpf dword ptr bios               ; leap to BIOS initialization
```

```
                        ;*****************************
                        ;*
                        ;*      subroutines
                        ;*
                        ;*****************************

                        ;******
                        phex:                           ;print 4 hex characters from ax
 00E0 B90404                    mov cx,0404h            ; 4 in both CH and CL
                        lhex:
 00E3 D3C0                      rol ax,cl               ; rotate left 4
 00E5 5150                      push cx ! push ax       ; save crucial registers
 00E7 E80700        00F1        call pnib               ; print hex nibble
CP/M ASM86 1.1  SOURCE: CPLLDCPM.A86  CP/M-86 Loader


 00EA 5859                      pop ax ! pop cx         ; restore registers
 00EC FECD75F3      00E3        dec ch ! jnz lhex       ; and loop four times
 00F0 C3                        ret
                        pnib:                           ;print low nibble in AL as hex char
 00F1 240F3C09                  and al,0fh ! cmp al,9
 00F5 7705          00FC        ja pl0                  ;above 9 ?
 00F7 0430                      add al,'0'              ;digit
 00F9 E90200        00FE        jmp prn
 00FC 0437              pl0:    add al,'A'-10           ;char a-e
 00FE 8AD0              prn:    mov dl,al
```

```
                        ;******
                        putchar:
0100 B102                       mov cl,coutf
0102 E92B00       0130          jmp sys_vec

                        ;******
                        initlbdos:
0105 2E8A16B601                 mov     dl,bootdrv                      ; select boot disk
010A 2AF6                       sub     dh,dh                           ; clear top half
010C B10E                       mov     cl,seldsk
010E E91F00       0130          jmp sys_vec

                        ;******
                        openfnc:
0111 B10F                       mov cl,openf
0113 BAB701                     mov dx,offset fcb       ; fcb already initialized
0116 E91700       0130          jmp sys_vec

                        ;********
                        ;
                        setdma:                         ;set new dma addr in dx
0119 B11A                       mov cl,dmaf
011B E91200       0130          jmp sys_vec

                        ;********
                        ;
                        setdmab:                        ; set new dma segment base from DX
011E B133                       mov cl,dmabf
```

```
0120 E90D00       0130          jmp sys_vec

                        ;******
                        ;
0123 BAAD01             pcrlf:  mov dx,offset crlf      ;print carriage return, line feed

                        ;******
                        ;
                        msg:                            ;print msg starting at dx until $
0126 B109                       mov cl,pstrf            ;print string function
0128 E90500       0130          jmp sys_vec

                        ;*****
CP/M ASM86 1.1  SOURCE: CPLLDCPM.A86  CP/M-86 Loader


                        ;
                        read:
012B BAB701B114                 mov dx,offset fcb ! mov cl,readsf
                        ;       jmp sys_vec

                        ;******
                        ;
                        sys_vec:
0130 CDE0                       int bdos_int
0132 C3                         ret
```

```
                        ;******
                        ;
                        stop:
0133 F4                         HLT
0134 EBFD          0133         JMPS    STOP

                        ;*********************************
                        ;*
                        ;*      DATA AREA
                        ;*
                        ;*********************************

0136 0D0A54686520       nofile          db      cr,lf,'The File CPM.SYS Not Found On This Disk'
     46696C652043
     504D2E535953
     204E6F742046
     6F756E64204F
     6E2054686973
     204469736B
015F 0D0A446F2061                       db      cr,lf,'Do a system reset with a new disk$'
     207379737465
     6D2072657365
     742077697468
     2061206E6577
     206469736B24
0183 0D0A5365676D       segment         db      cr,lf,'Segment Address = $'
     656E74204164
     647265737320
```

```
      3D2024
0198  0D0A20202020      lenmsg          db      cr,lf,'    Last Offset = $'
      4C617374204F
      666673657420
      3D2024
01AD  0D0A24            crlf            db      cr,lf,'$'


                                ; vector for jmpf indirect to start CP/M

01B0                    bios            rb      4
                                        org     offset bios     ;  overlay preceding with DW's
01B0  0025              biosstart       dw      biosoff         ; first word is BIOS offset
CP/M ASM86 1.1  SOURCE: CPLLDCPM.A86  CP/M-86 Loader


01B2  4000              ldseg           dw      cpm_seg         ; second is segment to put CPM.SYS

01B4                    ldlen           rw      1

01B6                    bootdrv         rb      1

01B7  0043504D2020      fcb     db      0,'CPM     ','SYS',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
      202020535953
      000000000000
      000000000000
      000000000000
```

```
      000000

01D8                  page1   rb      128

                                                ; dummy section for BIOS init label
                              org     lbios_offset
                      lbios:

                              end
```

END OF ASSEMBLY. NUMBER OF ERRORS: 0. USE FACTOR: 8%

```
        title   'Customized Loader Basic I/O System'

;***********************************************
;*                                             *
;* This Customized BIOS adapts CP/M-86 to      *
;* the following hardware configuration        *
;*      Processor: PC-100 Rainbow              *
;*      Brand:     DEC                         *
;*      Controller:                            *
;*      System: CP/M 86/80                     *
;*                                             *
;*                                             *
;*      Programmer:rdk/CPL                     *
;*      Revisions :                            *
;*                                             *
;*                                             *
;*      Release 1.0                            *
;*                                             *
;*      8/25/82 BIOS modified to operate with  *
;*              PC-100 Rainbow hardware, and to*
;*              operate with Z80 second CPU.   *
;*              ROM date: 8/17/82              *
;*                                             *
;***********************************************
```

```
FFFF                    true            equ -1
0000                    false           equ not true
000D                    cr              equ 0dh ;carriage return
000A                    lf              equ 0ah ;line feed
0043                    l~ts            equ     43h
0041                    ldata           equ     41h
0042                    csts            equ     42h
0040                    cdata           equ     40h
0013                    ctrl_s          equ     13h
0011                    ctrl_q          equ     11h

                        ;*****************************************
                        ;*                                       *
                        ;* Loader_bios is true if assembling the *
                        ;* LOADER BIOS, otherwise BIOS is for the*
                        ;* CPM.SYS file.                         *
                        ;*                                       *
                        ;*****************************************

FFFF                    loader_bios     equ true
00E0                    bdos_int        equ 224 ;reserved BDOS interrupt
                        ;DEBLOCK                EQU TRUE        ;do deblocking

                                IF      not loader bios
                        ;---------------------------------------
                        ;|                                     |
```

```
                        bios_code       equ 2500h
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy



                        ccp_offset      equ 0000h
                        bdos_ofst       equ 0B06h ;BDOS entry point
                        ;|                                          |
                        ;-------------------------------------------
                                ENDIF   ;not loader_bios
   FD00                 pb2_adr         equ 0fd00h      ;

                                IF      loader_bios
                        ;-------------------------------------------
                        ;|                                          |
   1200                 bios_code       equ 1200h ;start of LDBIOS
   0003                 ccp_offset      equ 0003h ;base of CPMLOADER
   0406                 bdos_ofst       equ 0406h ;stripped BDOS entry
                        ;|                                          |
                        :-------------------------------------------
                                ENDIF   ;loader_bios
                        ;
=                               INCLUDE DEFBUF.LIB
=                       ; *****************************************************************************
=                       ; OFFSETS FROM START OF POINTERS/BUFFERS DATA BLOCK
=
=  FFA0                 XDPCX           EQU     -60H            ; DISK PARAMETER STORAGE (60H)
=  0000                 XDEFBUF         EQU     000H            ; MISC. BUFFER (LENGTH=80H)
=  0086                 XPACKET         EQU     086H            ; BIOS MESSAGE PACKET (LENGTH=0EH)
```

```
=  0086                 XSTPKT          EQU     086H            ; START PACKET BUFFER (LENGTH=0EH)
=  0094                 XADCPKT         EQU     094H            ; DATA PACKET (LENGTH=0EH)
=  0094                 XMVPKT          EQU     094H            ; MOVE PACKET BUFFER (LENGTH=0EH)
=  00A2                 XSHRBUF         EQU     0A2H            ; SEGMENT BUFFER (LENGTH=200H)
=  02F8                 XMEMSIZE        EQU     2F8H            ; MEMORY SIZE (WORD)
=  02FA                 XPCPMADR        EQU     2FAH            ; PSEUDO CP/M ADDRESS (WORD)
=  02FC                 XZ80PKT         EQU     2FCH            ; PACKET POINTER FROM Z80 (WORD)
=  02FE                 XI88PKT         EQU     2FEH            ; PACKET POINTER FROM 8088 (WORD)
=  02F0                 XTTRACK         EQU     2F0H             TRACK TABLE
=  02F4                 XTFORMAT        EQU     2F4H            ; FORMAT TABLE
=  02E7                 XCSFLAG         EQU     2E7H            ; CONSOLE STATUS FLAG
=
=                       ; OFFSETS FROM ZOT FOR CONVENIENCE
=  0000                 ZOTP            EQU     0               ; Z80 FLAG
=  FFFE                 Z80FLAGPT       EQU     -2              ; Z80-RUNNING FLAG
=  FFFB                 CICCK           EQU     -5              ; CONSOLE STATUS FLAG CHECK
=
=                       ; OTHER USEFUL EQUATES
=  0002                 BDCS            EQU     2               ; BDOS CHARACTER READY BIT
=  0001                 BIOCS           EQU     1               ; BIOS CONSOLE STATUS BIT
=  0017                 BIOS_JMPS       EQU     23              ; NUMBER OF FUNCTIONS IN JUMP TABLE
=
=                       ; ************************************************************************

=                               INCLUDE CPLBIOS1.A86
=                       ;
=                               cseg
```

```
=                               org     ccpoffset
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=                       cc :
=                               org     bios_code
=
=                       ;*****************************************
=                       ;*                                       *
                        ;* BIOS Jump Vector for Individual Routines  *
=                       ;*                                       *
=                       ;*****************************************
=
=1200 E90D01        1310  jmp INIT        ;Enter from BOOT ROM or LOADER
=1203 E97101        1377  jmp WBOOT       ;Arrive here from BDOS call 0
=1206 E9A304        16AC  jmp CONST       ;return console keyboard status
=1209 E9A504        16B1  jmp CONIN       ;return console keyboard char
=120C E9A704        16B6  jmp CONOUT      ;write char to console device
=120F E9AE04        16C0  jmp LISTOUT     ;write character to list device
=1212 E9B004        16C5  jmp PUNCH       ;write character to punch device
=1215 E9B204        16CA  jmp READER      ;return char from reader device
=1218 E93005        174B  jmp HOME        ;move to trk 00 on cur sel drive
=121B E9F604        1714  jmp SELDSK      ;select disk for next rd/write
=121E E93B05        175C  jmp SETTRK      ;set track for next rd/write
=1221 E93E05        1762  jmp SETSEC      ;set sector for next rd/write
=1224 E94105        1768  jmp SETDMA      ;set offset for user buff (DMA)
=1227 E95A05        1784  jmp READ        ;read a 128 byte sector
=122A E97205        179F  jmp WRITE       ;write a 128 byte sector
```

```
=122D E98B04        16BB  jmp LISTST      ;return list status
=1230 E94105        1774  jmp SECTRAN     ;xlate logical->physical sector
=1233 E93805        176E  jmp SETDMAB     ;set seg base for buff (DMA)
=1236 E9AF04        16E8  jmp GETSEGT     ;return offset of Mem Desc Table
=1239 E9A104        16DD  jmp GETIOBF     ;return I/O map byte (IOBYTE)
=123C E9A304        16E2  jmp SETIOBF     ;set I/O map byte (IOBYTE)
=123F E9D606        1918  jmp RWMOVE      ;move block of data (* added for 86/80 *)
=1242 E9A704        16EC  jmp VIDEO       ;direct video output (* added for 86/80 *)
=
=                               IF      not loader_bios
=                         ;
=                         ;
=                         ; Segment Table address is placed here immediately after
=                         ; the BIOS jumps to help the loader find the segment table
=                         ; and set it up.
=                         ;
=                               DW      OFFSET SEGTABLE
=                         ;
=                         ;
=                         DBPTR   DW      0               ; POINTER TO DATA BLOCK
=                         ;                               (FILLED BY LOADER OR MOVE ROUTINE)
=                                  IF
=                               IF      loader bios
=  3A00                   DBPTR   EQU     3A00H           ;( illed by loader)
=                               ENDIF
                          ;
=                         ;*****************************************************************************
```

```
=                       ;*
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=                       ;*              INTERPROCESSOR COMMUNICATION ROUTINES
=                       ;*
=                       ;*****************************************************************************
=                       ;
=                       ; EQUATES
=                       ;
=                       ;
=  0000                 INTZ80   EQU     0               ; PORT TO INTERRUPT Z80
=  0002                 GSCR     EQU     2               ; INTERRUPT STATUS PORT
=  0080                 BIT7     EQU     80H             ; Z80 INTERRUPT STATUS BIT (0 = PENDING)
=                       ;
=1245 4A14              SIOINIT          DW      TPRTISQ         ; LOCATION TO FURNISH INIT TABLE START
=1247 00                CONINCHECK       DB      0               ; CONSOLE STATUS CHECK
=1248 0000              Z80PKT           DW      0               ; TEMP STORAGE FOR PACKET ADDRESS
=                       ;
=
```

```
                                SERV:
=124E 50                        PUSH    AX              ; SAVE REGISTERS
=124F 53                        PUSH    BX
=1250 BB003A                    MOV     BX,DBPTR        ; GET POINTER TO DATA BLOCK
=1253 1E                        PUSH    DS
=1254 33C0                      XOR     AX,AX
=1256 8ED8                      MOV     DS,AX           ; SET ZERO DS
=1258 8B9FFC02                  MOV     BX,XZ80PKT[BX]  ; GET PACKET POINTER
=125C 85DB                      TEST    BX,BX
=125E 740C          126C        JZ      TYPE_39_EXIT    ; IGNORE ZERO FOR PACKET ADDRESS
=1260 2E891E4812                MOV     Z80PKT,BX       ; STORE PACKET ADDRESS
=1265 2EC7064C12FF              MOV     ZOT,TRUE        ; SET THE FLAG
      FF
=                       TYPE_39_EXIT:
=126C E400                      IN      AL,INTZ80       ; CLEAR THE INTERRUPT
=126E 1F                        POP     DS              ; RESTORE DS
=126F 5B                        POP     BX              ; RESTORE REGISTERS
=1270 58                        POP     AX
=1271 CF                         RET
=                       if not loader_bios
=                       ;---------------------------------------------------------------
=                       ;
=                       ;       interrupt handler for type 44 (line frequency clock)
=                       ;
=                       type_44_serv:
=                               mov     CONINCHECK,0ffh ;set flag
=                               int 100                 ;see below
```

```
=                               iret
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=                       ;
=                       ;
=                       ;       software interrupt for line frequency clock (type 100)
=                       ;
=                       tvpe_100_serv:
=                               iret                    ;user mav intercept for use
=                       ;-----------------------------------------------------------------
=                       endif   ;not loader_bios
=                       ;
=                       ;
=                       ; SENDPKT  -  SEMD A PACKET TO THE Z80
=                       ;
=                       ; ENTRY:  BX = POINTER TO PACKET (ABSOLUTE)
=                       ;
=                       ; EXIT:  N/A
=                       ;
=                       SENDPKT:
=1272 1E                        PUSH    DS
=1273 53                        PUSH    BX              ; SAVE POINTER
=1274 BB003A                    MOV     BX,DBPTR        ; POINT TO DATA BLOCK
=1277 33C0                      XOR     AX,AX
=1279 8ED8                      MOV     DS,AX           ; SET ZERO DS
=127B 8F87FE02                  POP     WORD PTR XI88PKT[BX]    ; STORE PACKET POINTER
=                       ; SIGNAL Z80 AND WAIT FOR ACKNOWLEDGEMENT
```

```
=127F E600                      OUT     INTZ80,AL       ; INTERRUPT THE Z80
=                       SENDPK10:
=1281 E402                      IN      AL,GSCR         ; GET Z80 STATUS
=1283 A880                      TEST    AL,BIT7         ; INTERRUPT STILL PENDING?
=1285 74FA          1281        JZ      SENDPK10        ;  YES - CHECK AGAIN
=1287 C787FE020000              MOV     WORD PTR XI88PKT[BX],0  ; ZERO THE PACKET POINTER
=128D 1F                        POP     DS              ; RESTORE DS
=128E C3                        RET
=                       ;
=                       ;
=                       ;***********************************************
=                       ;*                                             *
=                       ;*      Routine to wait for an interrupt       *
=                       ;*      (type 39) from Z80                     *
=                       ;*                                             *
=                       ;***********************************************
=
=                       WAITZ80:
=128F FA                        cli             ;no interrupts please
=1290 2E833E4C12FF              cmp ZOT,TRUE
=1296 7404          129C        je  WAITRET     ;
=1298 FB                        sti             ;must allow interrupts now
=1299 F4                        hlt             ;wait quietly until interrupt
=129A EBF3          128F        jmps WAITZ80    ;loop back until non-zero
=129C FB                WAITRET: sti            ;allow interrupts
=129D 2E8B1E4812                mov bx,z80pkt   ; get packet address
=12A2 C3                        ret             ;bye
```

```
=                       ;
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy



=                       ;
=                       PACKER:         ;take items off the stack and put them
=12A3 1E                        push ds         ;into the message packet
=12A4 2EA34C12                  mov %OT,ax      ;Z80 flag set
=12A8 2E891EA51C                mov BXHLD,bx    ;save bx
=12AD 33DB                      xor bx,bx       ;set zero data segment
=12AF 8EDB                      mov ds,bx       ;
=12B1 2E8F06201E                pop SEGHLD      :save data segment
=12B6 2E8F06221E                pop RTNHLD      ;return address
=12BB BB003A                    mov bx,DBPTR    ; POINT TO DATA BLOCK
=12BE 81C38600                  add bx,xpacket  ; point to 88 packet
=12C2 03D9                      add bx,cx       ;end of packet
=12C4 03D9                      add bx,cx       ;=bx+2*cx
=12C6 2E890E241E                mov COUNT,cx    ;save count for later
=                       pklp:                   ;loop to do packing
=12CB 4B                        dec bx          ;back up bx
=12CC 4B                        dec bx          ;twice
=12CD 58                        pop ax          ;get t.o.s.
=12CE 8907                      mov [bx],ax     ;pack it
=12D0 E2F9          12CB        loop pklp       ;loop until done
=12D2 2EC7064C1200              mov %OT,false   ; clear "done" flag
      00
=12D9 E896FF        1272        call sendpkt    ; send packet to z80
=12DC E8B0FF        128F        call waitz80    ;wait if z80 is working
```

```
=12DF 2E8B0E241E              mov cx,COUNT    ;get the count again
=                      repak:                 ;take stuff out of packet
=12E4 8B07                    mov ax.[bx]     ;and
=12E6 50                      push ax         ;push it on the stack
=12E7 43                      inc bx          ;bump the pointer
=12E8 43                      inc bx          ;twice
=12E9 E2F9         12E4       loop repak      ;loop until done
=
=12EB 2EFF36221E              push RTNHLD
=12F0 2E8E1E201E              mov ds,SEGHLD   ;restore return and segment
=12F5 2E8B1EA51C              mov bx,BXHLD    ;restore bx
=12FA C3                      ret
=
=
=                      pmsg:
=12FB 8A07                    mov al,[BX]     ;get next char from message
=12FD 84C0                    test al,al
=12FF 740A         130B       jz pmretn       ;if zero return
=1301 8AC8                    mov CL,AL
=1303 53                      push bx         ; preserve pointer
=1304 E8AF03       16B6       call CONOUT     ;print it
=1307 5B                      pop bx
=1308 43                      inc BX
=1309 EBF0         12FB       jmps pmsg       ;next character and loop
=130B B083             pmretn: mov al,83h      ;make sure that crt is initiallized
=130D E60A                    out 0ah,al
=130F C3                      ret
```

```
=
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=
=                       ;*****************************************
=                       ;*                                       *
=                       ;* INIT Entry Point, Differs for LDBIOS and  *
=                       ;* BIOS, according to "Loader_Bios" value    *
=                       ;*                                       *
=                       ;*****************************************
=
=                       INIT:   ;print signon message and initialize hardware
=                               if loader_bios                  ; set up a stack
=                       ;---------------------------------------;
=1310 FA                        cli                                 interrupts, please
=1311 8CC8                      mov ax cs                       ; use cs for stack too
=1313 8ED0                      mov ss,ax                       ;
=1315 BC5F1E                    mov sp,offset stkbase           ; set up a local stack
=1318 FB                        sti                             ; interrupts ok now
=1319 51                        push cx
=                       ;---------------------------------------;
=                               endif
=
=131A 8CC8                      mov ax,cs       ;we entered with a JMPF so use
=131C 8ED8                      mov ds,ax       ;CS: as the initial value for DS:,
=131E 8EC0                      mov es,ax       ;and ES:
=
```

```
=               IF      not loader_bios
=           ;-------------------------------------------
=           ;|                                         |
=               ; This is a BIOS for the CPM.SYS file.
=               ; Setup all interrupt vectors in low
=               ; memory to address trap
=               :use local stack during initialization
=               cli             ;no interrupts while doing the stack
=               mov ss,ax       ;CS: as the initial value of SS:,
=               mov sp,offset stkbase
=               sti             ;interrupts ok now
=               cld             ;set forward direction
=               push cx
=               mov cl,95H      ;set IOBYTE to lst=lpt,con=crt
=               call SETIOBF    ;
=               mov 780



           :-------------------------------------------
=               ENDIF   ;not loader_bios
=
=               IF      loader_bios
=           ;-------------------------------------------
=           ;|                                         |
=               ;This is a BIOS for the LOADER
```

```
=1320 FC                        cld             ;set forward direction
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=1321 E82300        1347        call REVECTOR   ;set up interrupts
=                       :|                                          |
=                       ;-------------------------------------------
=                               ENDIF   ;loader_bios
=                               if      not loader_bios
=                               STI
=                               endif
=1324 BBA81C                    mov bx,offset signon
=1327 E8D1FF        12FB        call pmsg       ;print signon message
=
=                       ;set up track and format tables
=132A BB003A                    mov bx,dbptr    ;get address of table area
=132D 1E                        push ds         ;
=132E 33C0                      xor ax,ax       ;zero for data seg
=1330 8ED8                      mov ds,ax       ;
=1332 B90400                    mov cx,nrdisks  ;how many disks?
=1335 F6D4                      not ah          ;need ff in ah
=1337 88A7F002          init10: mov xttrack[bx],ah      ;tracks to ff
=133B 8887F402                  mov xtformat[bx],al     ;formats to 00
=133F 43                        inc bx
=1340 E2F5          1337        loop init10     ;loop back until done
=1342 1F                        pop ds          ;restore data seg
=
=1343 59                        pop cx          ;restore drive etc
```

```
=                       if not loader_bios
=                       ;---------------------------------------------------------
=                               mov al,cl       ;let's take a look at that drive
=                               and al,0fh      ;make sure it's valid, then ...
=                               mov b te ptr .curdrvs,al        ;store drive for submit files
=                       ;---------------------------------------------------------
=                       endif   ;not loader_bios
=1344 E9BCEC       0003         jmp ccp         ;jump to cold start entry of CCP
=                       IF loader_bios
=                       ;-----------------------------------------
=                       REVECTOR:
=1347 1E                        push ds         ;save data segment
=1348 B80000                    mov ax,0
=134B 8ED8                      mov ds,ax       ;point to segment zero
=                               ;BDOS interrupt offset
=134D C70680030604              mov bdos_offset,bdos_ofst
=1353 8C0E8203                  mov bdos_segment,CS ;bdos interrupt segment
=                       ;       (additional LOADER initialization)
=1357 C7069C004E12              MOV     Z80_OFFSET,OFFSET TYPE_39_SERV
=135D 8C0E9E00                  MOV     Z80_SEG,cs      ;use current code segment (why not?
=1361 C70690004F15              mov sio_offset,offset I232RX
=1367 8C0E9200                  mov sio_seg,cs
=136B C70694006B15              mov sio2_offset,offset I232RX2
=1371 8C0E9600                  mov sio2_seg,cs
=1375 1F                        pop ds          ;restore data segment
=1376 C3                        ret
=                       ;-----------------------------------------
```

```
=                       ENDIF ;loader_bios
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=
=                       ; ------------------------------------------------
=                               F not loader_bios
=                       ; SET UP INTERRUPT VECTORS
=                       REVECTOR:
=                               push ds         ;save the DS register
=                               CLI             ; DISABLE WHILE CHANGING VECTORS
=                       ;
=                       ;*************************************************************
=                       ;**     Firmware initializing                                ***
=                               xor dl,dl                                            ;**
=                               mov di,16h                                           ;**
=                               int 40                                               ;**
=                               mov di,0ch                                           ;**
=                               int 40                                               ;**
=                               cli                                                  ;**
=                       ;*************************************************************
=                               mov ax,0
=                               mov ds,ax
=                               mov es,ax       ;set ES and DS to zero
=                               ;BDOS offset to proper interrupt
=                               mov bdos_offset,bdos_ofst
=                               MOV     BDOS_SEGMENT,CS
=                               MOV     Z80_OFFSET,OFFSET TYPE_39_SERV
```

```
=                               MOV     Z80_SEG,cs      ;use current code segment (why not?)
=                               mov tp_44_offset,offset type_44_serv
=                               mov tp_44_seg,cs
=                               mov tp_100_offset,offset type_100_serv
=                               mov tp_100_seg,cs
=                               mov sio_offset,offset I232RX
=                               mov sio_seg,cs
=                               mov sio2_offset,offset I232RX2
=                               mov sio2_seg,cs
=
=                               STI             ; RE-ENABLE INTERRUPTS
=                       ;
=                               pop ds          ;restore the DS register
=                               jmp P232INIT    ;initialize sio
=                       ENDIF   ;not loader_bios
=                       ;-----------------------------------------------
=
=1377 2EC7064A1200      WBOOT:  mov Z80FLAG,false       ;set z80 not running
      00
=137E E8C6FF       1347         call revector
=1381 E985EC       0009         jmp ccp+6       ;direct entry to CCP at command level
=
=
=                       ;*********************************************
=                       ;*                                           *
=                       ;*   CP/M Character I/O Interface Routines   *
=                       ;*                                           *
```

```
=                       ;*****************************************
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=                       ;
=                       ; TERMINAL DEVICE DRIVERS
=                       ;
=                       CRTOUT:
=1384 8AC1                      MOV     AL,CL           ; Move character for output
=1386 BF0000                    MOV     DI,0            ; Function code
=1389 06                        PUSH    ES
=138A CD28                      INT     40
=138C 07                        POP     ES
=138D C3                        RET
=                       ;
=                       CRTIN:
=138E 06                        PUSH    ES
=138F BF0200            CRTIN1: MOV     DI,2            ; Function code
=1392 CD28                      INT     40
=1394 84C9                      TEST    CL,CL           ; Character available?
=1396 74F7          138F        JZ      CRTIN1          ;  No - retry
=1398 2EF7064A12FF              test z80flag,true       ; Is Z80 running?
      FF
=139F 741C          13BD        jz CRTIN2               ; go away if not
=13A1 50                        push ax
=13A2 BF0400                    mov di,4
=13A5 CD28                      int 40
=13A7 33DB                      xor bx,bx
```

```
=13A9 8EC3                      mov es,bx
=13AB BB003A                    mov bx,DBPTR
=13AE 80E101                    and cl,BIOCS
=13B1 2680A7E702FE              and es:byte ptr xcsflag[bx],not BIOCS   ;clear status flag
=13B7 26088FE702                or es:byte ptr xcsflag[bx],cl
=13BC 58                        pop ax
=13BD 07                CRTIN2: pop es
=13BE C3                        RET                     ; Return character in AL
=                       ;
=                       CRTSTI:
=13BF BF0400                    MOV     DI,4            ; Function code
=13C2 06                        PUSH    ES
=13C3 CD28                      INT     40
=13C5 07                        POP     ES
=13C6 8AC1                      MOV     AL,CL           ; Move status for return
=13C8 C3                        RET
=                       ;
=                       CRTSTO:
=13C9 B0FF                      MOV     AL,0FFH         ; Always ready
=13CB C3                        RET
=                       ;
=                       ;
=                       ; SERIAL I/O ROUTINES FOR DEC RAINBOW 100
=                       ;
=                       ;
=  0080                 HIPAR   EQU     80H             ; HIGH PARITY BIT
=                       ;
```

```
=                       ; CONTROL BLOCK OFFSETS
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=                       ;
=  0000                 QTPORT  EQU     BYTE PTR 0      ; CONTROL PORT ADDRESS
=  0001                 QTFLAGS EQU     BYTE PTR 1      ; DEVICE FLAGS (SEE MEANINGS BELOW)
=  0002                 QTNRCHR EQU     BYTE PTR 2      ; NR. OF CHARACTERS CURRENTLY IN BUFFER
=  0003                 QTCAP   EQU     BYTE PTR 3      ; BUFFER CAPACITY IN BYTES (CONSTANT)
=  0004                 QTINPTR EQU     BYTE PTR 4      ; BUFFER INPUT POINTER
=  0005                 QTOTPTR EQU     BYTE PTR 5      ; BUFFER OUTPUT POINTER
=  0006                 QTDEND  EQU     BYTE PTR 6      ; OFFSET OF LAST DATA BYTE (CONSTANT)
=  0007                 QTDEVID EQU     7               ; OFFSET OF PHYS DEVICE ID FOR ERROR MESSAGE
=  000A                 QTDATA  EQU     10              ; BUFFER DATA AREA
=                       ;
=                       ; BIT ASSIGNMENTS FOR "QTFLAGS"
=                       ;
=  0001                 QMSUSP  EQU     1       ; 1 = OUTPUT SUSPENDED
=  0002                 QMTYPE  EQU     2       ; 0   XON/XOFF, PARITY ERROR CHECKING
=  0004                 QMINIT  EQU     4       ; 1 = DEVICE REQUIRES INITIALIZATION
=  0008                 QMISUSP EQU     8       ; 1 = INPUT SUSPENDED
=  0010                 QMBREAK EQU     10H     ; 1 = BREAK DETECTED
=                       ;
=                       ; BUFFER LENGTHS IN CONTROL BLOCKS
=                       ;
=  0020                 QKPRTBL EQU     32      ; PRINTER CONTROL BLOCK
=  0020                 QKCOMBL EQU     32      ; COMM PORT CONTROL BLOCK
=  0020                 QKCOM2BL EQU    32      ; OPTIONAL COMM PORT CTL BLOCK
```

```
=                       ;
=                       ; CONTROL PORT ADDRESSES
=                       ;
=  0043                 QPPRT   EQU     43H     ; PRINTER PORT
=  0042                 QPCOM   EQU     42H     ; COMM PORT
=  0022                 QPCOM2  EQU     22H     ; OPTIONAL COMM PORT
=                       ;
=                       ; ASCII CONTROL CHARACTERS
=                       ;
=  0007                 QKBEL   EQU     7       ; BEL
=  0011                 QKXON   EQU     17      ; XON (CTL-Q)
=  0013                 QKXOFF  EQU     19      ; XOFF (CTL-S)
=  0091                 QKXONP  EQU     17+HIPAR        ; XON+ (CTL-Q)
=  0093                 QKXOFFP EQU     19+HIPAR        ; XOFF+ (CTL-S)
=  001A                 QKSUB   EQU     26      ;  'JB
=                       ;
=                       ; SIO STATUS BITS - RR0
=                       ;
=  0001                 QMRXR   EQU     1       ; RECEIVED CHAR. READY
=  0004                 QMTXR   EQU     4       ; TRANSMIT READY
=  0080                 QMBRK   EQU     80H     ; BREAK
=                       ;
=                       ; SIO STATUS BITS - RR1
=                       ,
=  0010                 QMPARE  EQU     10H     ; PARITY ERROR
=  0020                 QMOVRE  EQU     20H     ; OVERRUN ERROR
=                       ;
```

```
=                       ; SIO COMMANDS - WR0
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=                       ;
=  0038                 QKEOI EQU       38H     ; END OF INTERRUPT
=  0018                 QKCHRST EQU     18H     ; CHANNEL RESET
=  0030                 QKRESERR EQU    30H     ; RESET ERRORS
=  0010                 QKRESI  EQU     10H     ; RESET EXT/STATUS INT.
=                       ;
=                       ; SIO COMMANDS - WR1
=                       ;
=  0018                 QMMRXI  EQU     18H     ; INTERRUPT ON ALL RX CHAR.
=                       ;
=                       ; SIO COMMANDS - WR3
=                       ;
=  0040                 QMR7BIT EQU     40H     ; RX 7 BITS/CHAR
=  00C0                 QMR8BIT EQU     0C0H    ; RX 8 BITS/CHAR
=  0001                 QMMRXE  EQU     1       ; RX ENABLE
=                       ;
=                       ; SIO COMMANDS - WR4
=                       ;
=  0040                 QMMX16  EQU     40H     ; X16 CLOCK
=  000C                 QMMST2  EQU     0CH     ; 2 STOP BITS
=  0004                 QMMST1  EQU     4       ; 1 STOP BIT
=                       ;
=                       ; SIO COMMANDS - WR5
=                       ;
```

```
=  0080                 QMMDTR  EQU     80H     ; DTR ON
=  0002                 QMMRTS  EQU     2       ; RTS ON
=  0008                 QMMTXE  EQU     8       ; TX ENABLE
=  0020                 QMT7BIT EQU     20H     ; TX 7 BITS/CHAR
=  0060                 QMT8BIT EQU     60H     ; TX 8 BITS/CHAR
=                       ;
=                       ;
=                       ;
=                       ;
=                       ; CONTROL BLOCKS FOR EACH DEVICE
=                       ;
=                       ; PRINTER CONTROL BLOCK
=                       ;
=13CC                   TPRTCB  RS      0
=13CC 43                        DB      QPPRT   ; PORT ADDRESS
=13CD 04                        DB      QMINIT  ; USES XON/XOFF
=13CE 00                        DB      0       ; NUMBER OF CHARS.
=13CF 20                        DB      QKPRTBL ; CAPACITY
=13D0 0A                        DB      QTDATA  ; INPUT POINTER
=13D1 0A                        DB      QTDATA  ; OUTPUT POINTER
=13D2 29                        DB      QTDATA-1+QKPRTBL        ; OFFSET OF LAST DATA BYTE
=13D3 545459                    DB      'TTY'   ; DEVICE ID
=13D6                           RS      QKPRTBL ; DATA BUFFER
=                       ;
=                       ; COMM PORT CONTROL BLOCK
=                       ;
=13F6                   TCOMCB  RS      0
```

```
=13F6 42                        DB      QPCOM   ; PORT ADDRESS
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=13F7 06                        DB      QMTYPE+QMINIT   ; NO XON/XOFF
=13F8 00                        DB      0       ; NUMBER OF CHARACTERS
=13F9 20                        DB      QKCOMBL ; CAPACITY
=13FA 0A                        DB      QTDATA  ; INPUT POINTER
=13FB 0A                        DB      QTDATA  ; OUTPUT POINTER
=13FC 29                        DB      QTDATA-1+QKCOMBL        ; OFFSET OF LAST DATA BYTE
=13FD 505450                    DB      'PTP'   ; DEVICE ID
=1400                           RS      QKCOMBL ; DATA BUFFER
=                       ;
=                       ;OPTIONAL COMM PORT CONTROL BLOCK
=                       ;
=1420                   TCOM2CB RS      0
=1420 22                        DB      QPCOM2  ; PORT ADDRESS
=1421 02                        DB      QMTYPE  ; NO XON/XOFF
=1422 00                        DB      0       ; NUMBER OF CHARACTERS
=1423 20                        DB      QKCOM2BL        ; CAPACITY
=1424 0A                        DB      QTDATA  ; INPUT POINTER
=1425 0A                        DB      QTDATA  ; OUTPUT POINTER
=1426 29                        DB      QTDATA-1+QKCOM2BL       ; OFFSET OF LAST DATA BYTE
=1427 554331                    DB      'UC1'           ; DEVICE ID
=142A                           RS      QKCOM2BL        ; DATA BUFFER
=                       ;
=                       ;
=                       ; INITIALIZATION SEQUENCES FOR SIO
```

```
=                       ;
=                       ; PRINTER PORT
=                       ;
=144A                   TPRTISQ RS      0
=144A 18                        DB      QKCHRST                 ; CHANNEL RESET
=144B 14                        DB      4+QKRESI                ; WR4
=144C 4C                        DB      QMMX16+QMMST2           ; X16 CLOCK, 2 STOP BITS
=144D 13                        DB      3+QKRESI                ; WR3
=144E 41                        DB      QMR7BIT+QMMRXE          ; RX ENABLE, 7 BITS/CH
=144F 15                        DB      5+QKRESI                ; WR5
=1450 AA                        DB      QMMDTR+QMMRTS+QMT7BIT+QMMTXE  ; TX ENABLE, 7 BITS, RTS, DTR
=1451 11                TPRTISI DB      1+QKRESI                ; WR1
=1452 18                        DB      QMMRXI                  ; INTERRUPT ON ALL RX CHAR
=1453 00                        DB      0                       ; END OF SEQUENCE
=                       ;
=                       ; COMM PORT
=                       ;
=1454                   TCOMISQ RS      0
=1454 18                        DB      QKCHRST                 ; CHANNEL RESET
=1455 14                        DB      4+QKRESI                ; WR4
=1456 44                        DB      QMMX16+QMMST1           ; X16 CLOCK, 1 STOP BIT
=1457 13                        DB      3+QKRESI                ; WR3
=1458 C1                        DB      QMR8BIT+QMMRXE          ; RX ENABLE, 8 BITS/CH
=1459 15                        DB      5+QKRESI                ; WR5
=145A EA                        DB      QMMDTR+QMMRTS+QMT8BIT+QMMTXE  ; TX ENABLE, 8 BITS, RTS, DTR
=145B 11                TCOMISI DB      1+QKRESI                ; WR1
=145C 18                        DB      QMMRXI                  ; INTERRUPT ON ALL RX CHAR
```

```
=145D 00                        DB      0                       ; END OF SEQUENCE
```



```
=                       ;
=                       ; OPTIONAL COMM PORT
=                       ;
=145E                   TCOM2ISQ        RS      0
=145E 18                        DB      QKCHRST                 ; CHANNEL RESET
=145F 14                        DB      4+QKRESI                ; WR4
=1460 44                        DB      QMMX16+QMMST1           ; X16 CLOCK, 1 STOP BIT
=1461 13                        DB      3+QKRESI                ; WR3
=1462 C1                        DB      QMR8BIT+QMMRXE          ; RX ENABLE, 8 BITS/CH
=1463 15                        DB      5+QKRESI                ; WR5
=1464 EA                        DB      QMMDTR+QMMRTS+QMT8BIT+QMMTXE  ; TX ENABLE, 8 BITS, RTS, DTR
=1465 11                        DB      1+QKRESI                ; WR1
=1466 18                        DB      QMMRXI                  ; INTERRUPT ON ALL RX CHAR
=1467 00                        DB      0                       ; END OF SEQUENCE
=                       ;
=                       ;
=                       ;
=                       ; P232INIT - INITIALIZE RS232 (SIO) PORTS
=                       ;
=                       ; FUNCTION: INITIALIZES ALL RS232 PORTS FOR WHICH INITIALIZATION
=                       ;  IS REQUIRED.
=                       ;
=                       ; ENTRY AND EXIT PARAMETERS: N/A
=                       ;
```

```
=                       P232INIT:
=1468 BBCC13                    MOV     BX,OFFSET TPRTCB        ; PRINTER CONTROL BLOCK
=146B BE5114                    MOV     SI,OFFSET TPRTISI       ; PRINTER INIT SEQUENCE
=146E E81200        1483        CALL    P232IPR         ; INITIALIZE
=1471 BBF613                    MOV     BX,OFFSET TCOMCB        ; COMM CONTROL BLOCK
=1474 BE5B14                    MOV     SI,OFFSET TCOMISI       ; COMM INIT SEQUENCE
=1477 E80900        1483        CALL    P232IPR         ; INITIALIZE
=147A BB2014                    MOV     BX,OFFSET TCOM2CB       ; OPT COMM CONTROL BLOCK
=147D BE5E14                    MOV     SI,OFFSET TCOM2ISQ      ; OPT COMM INIT SEQUENCE
=1480 E90000        1483        JMP     P232IPR         ; INITIALIZE
=                       ;
=                       ;
=                       ; P232IPR - INITIALIZE AN SIO PORT
=                       ;
=                       ; FUNCTION: INITIALIZES AN SIO PORT ASSOCIATED WITH A CONTROL BLOCK.
=                       ;  CHECKS CONTROL BLOCK FIRST TO SEE IF INITIALIZATION IS REQUIRED.
=                       ;  (DOES NOT INITIALIZE BAUD RATES.)
=                       ;
=                       ; ENTRY:
=                       ;       BX = POINTER TO CONTROL BLOCK
=                       ;       SI = POINTER TO INITIALIZATION SEQUENCE
=                       ;            (SEQUENCE ENDS WITH A ZERO BYTE)
=                       ;
=                       ; EXIT : N/A
=                       ;
=                       P232IPR:
=1483 F6470104                  TEST    QTFLAGS[BX],QMINIT      ; INITIALIZATION REQUIRED?
```

```
=1487 7501        148A          JNZ     P232IP10                ;  YES
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=1489 C3                        RET                             ; ELSE RETURN
=                       P232IP10:
=148A 8A17                      MOV     DL,QTPORT[BX]           ; GET PORT ADDRESS
=148C 32F6                      XOR     DH,DH                   ; MAKE 16 BIT ADDR
=148E FC                        CLD                             ; SET FORWARD DIRECTION
=                       P232IP20:
=148F AC                        LODSB                           ; GET A BYTE
=1490 84C0                      TEST    AL,AL                   ; END OF SEQUENCE?
=1492 7501        1495          JNZ     P232IP30                ;  NO
=1494 C3                        RET                             ; EXIT IF END
=                       P232IP30:
=1495 EE                        OUT     DX,AL                   ; SEND IT TO PORT
=1496 EBF7        148F          JMPS    P232IP20                ; GET NEXT BYTE
=                       ;
=                       ;
=                       ;
=                       ; P232IN - READ A CHARACTER
=                       ;
=                       ; FUNCTION: RETURNS ONE BYTE FROM CIRCULAR BUFFER FOR A SPECIFIED
=                       ;  DEVICE.  WAITS IF THE BUFFER IS EMPTY.  IF XOFF HAS SUSPENDED INPUT,
=                       ;  AND THE BUFFER IS EMPTY, SEND AN XON.
=                       ;
=                       ; ENTRY:
=                       ;       BX = POINTER TO CONTROL BLOCK
```

```
=                       ;
=                       ; EXIT:
=                       ;       AL = CHARACTER
=                       ;
=                       P232IN:
=1498 8A4702                    MOV     AL,QTNRCHR[BX]          ; GET NUMBER OF CHARS. IN BUFFER
=149B 84C0                      TEST    AL,AL                   ; IS IT ZERO?
=149D 74F9          1498        JZ      P232IN                  ; LOOP BACK IF SO
=149F FA                        CLI                             ; DISABLE WHILE MANIPULATING BUFFER
=14A0 FE4F02                    DEC     QTNRCHR[BX]             ; DECREMENT CHARACTER COUNT
=14A3 7526          14CB        JNZ     P232IN1                 ; JUMP AHEAD IF NOT ZERO        **
=14A5 FB                        STI                             ; TURN ON INTERRUPTS            **
=14A6 2EF687010002              TEST    CS:QTFLAGS[BX],QMTYPE   ; DO WE SUPPORT XON?            **
=14AC 751D          14CB        JNZ     P232IN1                 ; NO. SKIP AHEAD                **
=14AE 2EF687010008              TEST    CS:QTFLAGS[BX],QMISUSP  ; YES. IS INPUT SUSPENDED?      **
=14B4 7415          14CB        JZ      P232IN1                 ; NO, BUFFER IS JUST EMPTY      **
=                   P232IN2:                                    ;                               **
=14B6 2EF687010001              TEST    CS:QTFLAGS[BX],QMSUSP   ; IS OUTPUT SUSPENDED?          **
=14BC 75F8          14B6        JNZ     P232IN2                 ; YES, WAIT FOR IT              **
=14BE 2E80A70100F7              AND     CS:QTFLAGS[BX],NOT QMISUSP      ;INPUT NOT SUSPENDED    **
=14C4 51                        PUSH    CX                      ; (JIC)                         **
=14C5 B111                      MOV     CL,QKXON                ; OUTPUT AN ...                 **
=14C7 E81900        14E3        CALL    P232OUT                 ; XON.                          **
=14CA 59                        POP     CX                      ;                               **
=                   P232IN1:                                    ;                               **
=14CB FA                        CLI                             ; NO MORE INTERRUPTS            **
=14CC 8A4705                    MOV     AL,QTOTPTR[BX]          ; GET OUTPUT POINTER
```

```
=14CF 32E4                      XOR     AH,AH                   ; MAKE 16 BIT OFFSET
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=14D1 8BF0                      MOV     SI,AX                   ; MOVE TO INDEX REG.
=14D3 FEC0                      INC     AL                      ; INCREMENT OUTPUT POINTER
=14D5 3A4706                    CMP     AL,QTDEND[BX]           ; PAST END OF BUFFER?
=14D8 7602          14DC        JBE     P232IN4                 ;  NO
=14DA B00A                      MOV     AL,QTDATA               ; SET TO START OF BUFFER
=                       P232IN4:
=14DC 884705                    MOV     QTOTPTR[BX],AL          ; STORE UPDATED POINTER
=14DF 8A00                      MOV     AL,[BX+SI]              ; GET CHARACTER FROM BUFFER
=14E1 FB                        STI                             ; RE-ENABLE
=14E2 C3                        RET                             ; EXIT
=                       ;
=                       ;
=                       ; P232OUT - WRITE A CHARACTER
=                       ;
=                       ; FUNCTION: WRITES A CHARACTER TO A SPECIFIED DEVICE.
=                       ;
=                       ; ENTRY:
=                       ;       BX = POINTER TO CONTROL BLOCK
=                       ;       CL = CHARACTER TO BE WRITTEN
=                       ;
=                       ; EXIT: N/A
=                       ;
=                       P232OUT:
=14E3 8AE1                      MOV     AH,CL                   ; SAVE THE CHARACTER
```

```
=                       P232OUT0:
=14E5 8A17                      MOV     DL,QTPORT[BX]           ; GET PORT ADDRESS
=14E7 32F6                      XOR     DH,DH                   ; MAKE 16 BIT ADDRESS
=14E9 F6470102                  TEST    QTFLAGS[BX],QMTYPE      ; OK TO SEND XOFF?
=14ED 7509          14F8        JNZ     P232OUT1                ;  YES
=14EF 8AC4                      MOV     AL,AH                   ; GET CHARACTER FOR TESTS
=14F1 247F                      AND     AL,07FH                 ; STRIP PARITY
=14F3 3C13                      CMP     AL,QKXOFF               ; IS IT XOFF?
=14F5 7501          14F8        JNE     P232OUT1                ;  NO - SEND IT
=14F7 C3                        RET                             ; EXIT WITHOUT SENDING
=                       P232OUT1:
=14F8 33C9                      XOR     CX,CX                   ; INIT TIMEOUT LOOP COUNTER
=                       P232OUT2:
=14FA EC                        IN      AL,DX                   ; GET PORT STATUS
=14FB A804                      TEST    AL,QMTXR                ; TX READY?
=14FD 7522          1521        JNZ     P232OUT3                ;  YES - DO IT
=14FF 51                        PUSH    CX                      ; THESE 2 INSTRUCTIONS ARE FILLERS
=1500 59                        POP     CX                      ;  TO MAKE A VALID TIMEOUT
=1501 E2F7          14FA        LOOP    P232OUT2                ; TRY IT AGAIN IF NO TIMEOUT YET
=1503 50                        PUSH    AX                      ; TIMEOUT - SOMETHING'S WRONG WITH
=1504 53                        PUSH    BX                      ;  UART
=1505 8B4707                    MOV     AX,QTDEVID[BX]          ; PUT DEVICE ID INTO MESSAGE
=1508 2EA31B1E                  MOV     P232TO1,AX
=150C 8A4709                    MOV     AL,QTDEVID+2[BX]
=150F 2EA21D1E                  MOV     P232TO2,AL
=1513 BB0E1E                    MOV     BX,OFFSET P232TO
=1516 E8E2FD        12FB        CALL    PMSG                    ; WRITE TIMEOUT MESSAGE
```

```
=1519 E8A104       19BD         CALL    KQERY                    ; GET USER OPTION
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=151C 5B                        POP     BX                       ;  SUBROUTINE WILL NOT RETURN IF CTL-C
=151D 58                        POP     AX
=151E 74C5         14E5         JZ      P232OUT0                 ; TRY AGAIN IF SPACE BAR
=1520 C3                        RET                              ; PRETEND IT'S OK IF ANY OTHER KEY
=                       P232OUT3:
=1521 F6470101                  TEST    QTFLAGS[BX],QMSUSP       ; IS OUTPUT SUSPENDED?
=1525 75FA         1521         JNZ     P232OUT3                 ;  YES - WAIT UNTIL ENABLED
=1527 80EA02                    SUB     DL,2                     ; POINT TO DATA PORT
=152A 8AC4                      MOV     AL,AH                    ; GET CHARACTER
=152C EE                        OUT     DX,AL                    ; SEND IT
=152D C3                        RET                              ; EXIT
=                       ;
=                       ;
=                       ;
=                       ; P232STI  -  RETURN INPUT STATUS
=                       ;
=                       ; FUNCTION:  RETURNS A VALUE INDICATING WHETHER AN INPUT CHARACTER
=                       ;  IS AVAILABLE FROM A SPECIFIED DEVICE.
=                       ;
=                       ; ENTRY:
=                       ;       BX = POINTER TO CONTROL BLOCK
=                       ;
=                       ; EXIT:
=                       ;       AL = 0     IF NO CHARACTER IS READY (BUFFER EMPTY)
```

```
=                       ;       AL = X'FF' IF ONE OR MORE CHARACTERS ARE READY
=                       ;
=                       P232STI:
=152E 8A4702                    MOV     AL,QTNRCHR[BX]          ; GET CURRENT NR OF CHARS.
=1531 84C0                      TEST    AL,AL                   ; IS IT ZERO?
=1533 7501          1536        JNZ     P232STI1                ;  NO
=1535 C3                        RET                             ; RETURN "NOT READY"
=                       P232STI1:
=1536 B0FF                      MOV     AL,0FFH                 ; SET "READY"
=1538 C3                        RET
=                       ;
=                       ;
=                       ;
=                       ; P232STO  -  RETURN OUTPUT STATUS
=                       ;
=                       ; FUNCTION: RETURNS A VALUE INDICATING WHETHER THE SPECIFIED DEVICE IS
=                       ;  READY TO ACCEPT AN OUTPUT CHARACTER.
=                       ;
=                       ; ENTRY:
=                       ;       BX = POINTER TO CONTROL BLOCK
=                       ;
=                       ; EXIT:
=                       ;       AL = 0      IF DEVICE IS NOT READY TO ACCEPT A CHARACTER
=                       ;       AL = X'FF'  IF DEVICE IS READY TO ACCEPT A CHARACTER
=                       ;
=                       P232STO:
=1539 8A17                      MOV     DL,QTPORT[BX]           ; GET PORT ADDRESS
```

```
=153B 32F6                      XOR     DH,DH                   ; MAKE 16 BIT ADDRESS
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=153D EC                        IN      AL,DX                   ; GET PORT STATUS
=153E 2404                      AND     AL,QMTXR                ; IS TX READY?
=1540 7501          1543        JNZ     P232STO1                ;  YES
=1542 C3                        RET                             ; RETURN "NOT READY"
=                       P232STO1:
=1543 32C0                      XOR     AL,AL                   ; SET "NOT READY" VALUE
=1545 F6470101                  TEST    QTFLAGS[BX],QMSUSP      ; IS OUTPUT SUSPENDED?
=1549 7401          154C        JZ      P232STO3                ;  NO
=154B C3                        RET                             ; RETURN "NOT READY"
=                       P232STO3:
=154C F6D0                      NOT     AL                      ; SET "READY"
=154E C3                        RET
=                       ;
=                       ;
=                       ;
=                       ; I232RX  -  HANDLE SIO RECEIVE INTERRUPTS
=                       ;
=                       ; FUNCTION:  PROCESS RECEIVE INTERRUPTS FROM ONE OR MORE SIO PORTS.
=                       ;  SET CONTROL BLOCK POINTERS AND CALL PORT INTERRUPT PROCESSOR.
=                       ;
=                       ; ENTRY AND EXIT PARAMETERS: N/A
=                       ;
=                       I232RX:
=154F 50                        PUSH    AX                      ; SAVE REGISTERS
```

```
=1550 53                        PUSH    BX
=1551 52                        PUSH    DX
=                       ; POINT TO EACH CONTROL BLOCK AND CHECK RECEIVER
=1552 BBCC13                    MOV     BX,OFFSET TPRTCB        ; PRINTER CONTROL BLOCK
=1555 E81B00    1573            CALL    I232RPT
=1558 BBF613                    MOV     BX,OFFSET TCOMCB        ; COMM PORT CONTROL BLOCK
=                       I232RXX:
=155B E81500    1573            CALL    I232RPT
=                       ; SEND END-OF-INTERRUPT
=155E 80CA02                    OR      DL,2
=1561 80E2FE                    AND     DL,0FEH
=1564 B038                      MOV     AL,QKEOI
=1566 EE                        OUT     DX,AL
=                       ; RESTORE REGISTERS AND EXIT
=1567 5A                        POP     DX
=1568 5B                        POP     BX
=1569 58                        POP     AX
=156A CF                        IRET
=                       ;
=                       ;
=                       ;
=                       ; I232RX2  -  HANDLE OPTIONAL SIO RECEIVE INTERRUPTS
=                       ;
=                       ; FUNCTION:  PROCESS RECEIVE INTERRUPTS FROM ONE OR MORE SIO PORTS.
=                       ;  SET CONTROL BLOCK POINTERS AND CALL PORT INTERRUPT PROCESSOR.
=                       ;
=                       ; ENTRY AND EXIT PARAMETERS: N/A
```

```
=                       ;
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=                       I232RX2:
=156B 50                        PUSH    AX                      ; SAVE REGISTERS
=156C 53                        PUSH    BX
=156D 52                        PUSH    DX
=                       ; POINT TO EACH CONTROL BLOCK AND CHECK RECEIVER
=156E BB2014                    MOV     BX,OFFSET TCOM2CB       ; COMM PORT CONTROL BLOCK
=1571 EBE8          155B        JMPS    I232RXX                 ;SAVE REGISTERS AND EXIT
=                       ;
=                       ;
=                       ;
=                       ; I232RPT  -  PROCESS RECEIVED CHARACTER
=                       ;
=                       ; FUNCTION:  IF A CHARACTER HAS BEEN RECEIVED AT A PORT, READS IT
=                       ;  AND PLACES IT IN A CIRCULAR BUFFER.  IF REQUIRED, PROCESSES
=                       ;  XON/XOFF PROTOCOL AND SUBSTITUTES A 'SUB' CHARACTER WHEN A
=                       ;  PARITY ERROR OCCURS.
=                       ;
=                       ; ENTRY:
=                       ;       BX = POINTER TO CONTROL BLOCK
=                       ;
=                       ; EXIT:  N/A
=                       ;
=                       ; REGISTER USE:
=                       ;       AL      PORT INPUT/OUTPUT
```

```
=                       ;       AH      ERROR STATUS STORAGE
=                       ;       BX      CONTROL BLOCK POINTER
=                       ;       DX      PORT ADDRESS AND INPUT POINTER WORKING REG.
=                       ;
=                       I232RPT:
=1573 2E8A970000                MOV     DL,CS:QTPORT[BX]        ; GET PORT ADDRESS
=1578 32F6                      XOR     DH,DH                   ; MAKE 16 BIT ADDRESS
=157A EC                        IN      AL,DX                   ; GET PORT STATUS
=157B A801                      TEST    AL,QMRXR                ; RECEIVE READY?
=157D 7501          1580        JNZ     I232RP05                ;  YES
=                       I232RPTX:
=157F C3                        RET                             ; EXIT - NO PROCESSING NECESSARY
=                       I232RP05:
=1580 A880                      TEST    AL,QMBRK                ; BREAK?
=1582 7409          158D        JZ      I232RP10                ;
=1584 2E808F010010              OR      CS: QTFLAGS[BX],QMBREAK ; SIGNAL IT
=158A E9D100        165E        JMP     I232RP40                ;
=                       ; PROCESS A RECEIVED CHARACTER
=                       I232RP10:
=158D 2EF687010010              TEST    CS:QTFLAGS[BX],QMBREAK  ; DO WE HAVE A BREAK?
=1593 7409          159E        JZ      I232RP11                ; SKIP IF NOT
=1595 2E80A70100EF              AND     CS:QTFLAGS[BX],NOT QMBREAK      ;ELSE CLEAR FLAG
=159B E9C000        165E        JMP     I232RP40                ; AND GO AWAY
=                       I232RP11:
=159E B001                      MOV     AL,1                    ; POINT TO RR1
=15A0 EE                        OUT     DX,AL                   ; WRITE TO SIO
=15A1 EC                        IN      AL,DX                   ; READ RR1
```

```
=15A2 8AE0                      MOV     AH,AL                   ; SAVE IT
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=15A4 B030                      MOV     AL,QKRESERR             ; RESET SIO ERRORS (IF ANY)
=15A6 EE                        OUT     DX,AL
=15A7 80EA02                    SUB     DL,2                    ; POINT TO DATA PORT
=15AA EC                        IN      AL,DX                   ; READ RECEIVED CHARACTER
=                       ; CHECK FOR RECEIVE ERRORS
=15AB F6C420                    TEST    AH,QMOVRE               ; OVERRUN ERROR?
=15AE 7406          15B6        JZ      I232RP12                ;  NO
=15B0 2E80A70100FE              AND     CS:QTFLAGS[BX],NOT QMSUSP ; ENABLE OUTPUT (ASSUME XON LOST)
=                       I232RP12:
=15B6 F6C410                    TEST    AH,QMPARE               ; PARITY ERROR?
=15B9 7402          15BD        JZ      I232RP14                ;  NO
=15BB B01A                      MOV     AL,QKSUB                ; CHANGE RX CHAR. TO 'SUB'
=                       I232RP14:
=15BD 2EF687010002              TEST    CS:QTFLAGS[BX],QMTYPE   ; XON/XOFF APPLICABLE?
=15C3 7530          15F5        JNZ     I232RP20                ;  NO - STORE CHARACTER
=                       ; PROCESS XON/XOFF PROTOCOL
=15C5 84C0                      TEST    AL,AL                   ; CHECK FOR NULL
=15C7 7501          15CA        JNZ     I232RP15                ; NOT A NULL
=15C9 C3                        RET                             ; IGNORE NULL CHARACTERS
=                       I232RP15:
=15CA 3C93                      CMP     AL,QKXOFFP              ; IS IT XOFF+?
=15CC 7404          15D2        JZ      I232RP15A               ;  YES
=15CE 3C13                      CMP     AL,QKXOFF               ; IS IT XOFF?
=15D0 7507          15D9        JNE     I232RP16                ;  NO
```

```
=                       I232RP15A:
=15D2 2E808F010001               OR      CS:QTFLAGS[BX],QMSUSP   ; SUSPEND OUTPUT
=15D8 C3                         RET                             ;  AND EXIT
=                       I232RP16:
=15D9 3C91                       CMP     AL,QKXONP               ; IS IT XON?
=15DB 7404          15E1         JZ      I232RP16A               ;  NO - STORE IT
=15DD 3C11                       CMP     AL,QKXON                ; IS IT XON?
=15DF 7514          15F5         JNE     I232RP20                ;  NO - STORE IT
=                       I232RP16A:
=15E1 F6C420                     TEST    AH,QMOVRE               ; WAS THERE AN OVERRUN ERROR?
=15E4 7508          15EE         JNZ     I232RP18                ;  YES - ASSUME AN XOFF WAS LOST
=15E6 2EF687010001               TEST    CS:QTFLAGS[BX],QMSUSP   ; IS OUTPUT SUSPENDED?
=15EC 7407          15F5         JZ      I232RP20                ;  NO - STORE XON IN BUFFER
=                       I232RP18:
=15EE 2E80A70100FE               AND     CS:QTFLAGS[BX],NOT QMSUSP       ; ENABLE OUTPUT
=15F4 C3                         RET                             ;  AND EXIT
=                       ; STORE CHARACTER IN BUFFER
=                       I232RP20:
=15F5 2E8A970200                 MOV     DL,CS:QTNRCHR[BX]       ; GET NR. OF CHARS. IN BUFFER
=15FA 2E3A970300                 CMP     DL,CS:QTCAP[BX]         ; IS BUFFER FULL?
=15FF 744C          164D         JE      I232RP30                ;  YES
=1601 2EFE870200                 INC     CS:QTNRCHR[BX]          ; INCREMENT CHARACTER COUNT
=1606 2E8A970400                 MOV     DL,CS:QTINPTR[BX]       ; GET INPUT POINTER
=160B 2EFE870400                 INC     CS:QTINPTR[BX]          ; INCREMENT POINTER
=1610 2E3A970600                 CMP     DL,CS:QTDEND[BX]        ; PAST END OF BUFFER?
=1615 7206          161D         JB      I232RP26                ;  NO
=1617 2EC68704000A               MOV     CS:QTINPTR[BX],QTDATA   ; SET TO START OF BUFFER
```

```
=                       I232RP26:
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=161D 53                        PUSH    BX                      ; NEED TO SAVE POINTER          **
=161E 03DA                      ADD     BX,DX                   ; STORE UPDATED POINTER
=1620 2E88870000                MOV     CS:[BX],AL              ; STORE CHARACTER
=                       ; NOW SEE IF BUFFER IS MORE THAN HALF FULL AND IF XOFF/XON              **
=                       ; IS SUPPORTED.  IF BOTH, SEND AN XOFF.                                 **
=1625 5B                        POP     BX                      ;                               **
=1626 2EF687010002              TEST    CS:QTFLAGS[BX],QMTYPE   ; XOFF SUPPORTED?               **
=162C 751E          164C        JNZ     I232RP28                ; GO AWAY IF NOT                **
=162E 2E8A970200                MOV     DL,CS:QTNRCHR[BX]       ; GET NO. OF CHARACTERS         **
=1633 02D2                      ADD     DL,DL                   ; DOUBLE IT                     **
=1635 2E3A970300                CMP     DL,CS:QTCAP[BX]         ; MORE THAN HALF-FULL?          **
=163A 7610          164C        JBE     I232RP28                ; GO AWAY IF NOT                **
=                       I232RP27:                               ; OUTPUT A XOFF                 **
=163C 2EF687010008              TEST    CS:QTFLAGS[BX],QMISUSP  ; UNLESS INPUT IS ALREADY SUSP- **
=1642 7508          164C        JNZ     I232RP28                ; ENDED.                        **
=1644 2E808F010008              OR      CS:QTFLAGS[BX],QMISUSP  ; SUSPEND INPUT                 **
=164A EB01          164D        JMPS    I232RP32                ; OUTPUT IT                     **
=                       I232RP28:                               ;                               **
=164C C3                        RET                             ;  AND EXIT
=                       ;
=                       ; ERROR CONDITION ROUTINES
=                       ;
=                       ; BUFFER IS FULL, OR HALF-FULL, OR SOMETHING
                        ;
```

```
=                       I232RP30:
=                       I232RP32:                               ; ENTER HERE FOR XOFF          **
=164D 2E8A970000                MOV     DL,CS:QTPORT[BX]
=                       I232RP33:
=1652 EC                        IN      AL,DX                   ;
=1653 A804                      TEST    AL,QMTXR                ;
=1655 74FB          1652        JZ      I232RP33                ;
=1657 B013                      MOV     AL,QKXOFF               ;
=1659 80EA02                    SUB     DL,2
=165C EE                        OUT     DX,AL
=165D C3                        RET                             ;  AND EXIT
=                       ;
=                       ; BREAK CONDITION
=                       ;
=                       I232RP40:
=165E B010                      MOV     AL,QKRESI               ; RESET BREAK CONDITION
=1660 EE                        OUT     DX,AL
=1661 B030                      MOV     AL,QKRESERR             ; RESET ANY OTHER ERRORS
=1663 EE                        OUT     DX,AL
=1664 80EA02                    SUB     DL,2                    ; POINT TO DATA PORT
=1667 EC                        IN      AL,DX                   ; CLEAR INPUT
=1668 C3                        RET
=                       ;
=                       ; ************************************************************************
=                       ;
=                       ;               PHYSICAL DEVICE DRIVERS
=                       ;
```

```
=                       ; ************************************************************************.
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=                       ;
=                       ;
=                       ; PTR: AND PTP: DRIVERS
=                       ;
=                       PTRIN:
=1669 BBF613                    MOV     BX,OFFSET TCOMCB        ; POINT TO CONTROL BLOCK
=166C E929FE       1498         JMP     P232IN
=                       ;
=                       PTPOUT:
=166F BBF613                    MOV     BX,OFFSET TCOMCB        ; POINT TO CONTROL BLOCK
=1672 E96EFE       14E3         JMP     P232OUT
=                       ;
=                       PTRST:
=1675 BBF613                    MOV     BX,OFFSET TCOMCB        ; POINT TO CONTROL BLOCK
=1678 E9B3FE       152E         JMP     P232STI
=                       ;
=                       ;
=                       ; TTY: DRIVERS
                        ;
=                       TTYIN:
=167B BBCC13                    MOV     BX,OFFSET TPRTCB        ; POINT TO CONTROL BLOCK
=167E E917FE       1498         JMP     P232IN
=                       ;
=                       TTYOUT:
```

```
=1681 BBCC13                    MOV     BX,OFFSET TPRTCB        ; POINT TO CONTROL BLOCK
=1684 E95CFE        14E3        JMP     P232OUT
=                       ;
=                       TTYSTI:
=1687 BBCC13                    MOV     BX,OFFSET TPRTCB        ; POINT TO CONTROL BLOCK
=168A E9A1FE        152E        JMP     P232STI
=                       ;
=                       TTYSTO:
=168D BBCC13                    MOV     BX,OFFSET TPRTCB        ; POINT TO CONTROL BLOCK
=1690 E9A6FE        1539        JMP     P232STO
=                       ;
=                       ;
=                       ;
=                       ; NULL DEVICE DRIVERS
=                       ;
=  001A                 QKCTLZ  EQU     26              ; END OF FILE (CTL-Z)
=                       ;
=                       ;
=                       PNULIN:
=1693 B01A                      MOV     AL,QKCTLZ       ; INDICATE END OF FILE
=1695 C3                        RET
=                       ;
=                       PNULOUT:
=1696 C3                        RET                     ; ACCEPT ALL CHARACTERS
=                       ;
=                       PNULSTI:
=1697 B0FF                      MOV     AL,0FFH         ; INDICATE DEVICE READY
```

```
=1699 C3                        RET
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=                       ;
=  1697                 PNULSTO EQU     PNULSTI
=                       ;
=                       ;
=                       ; BATCH drivers
=                       ;
=  1675                 BATST   EQU     PTRST
=  1669                 BATIN   EQU     PTRIN
=  1384                 BATOUT  EQU     CRTOUT
=                       ;
=                       ;
=                       ; LPT: DRIVERS
=                       ;
=  1681                 LPTOUT  EQU     TTYOUT
=  168D                 LPTSTO  EQU     TTYSTO
=                       ;
=                       ;
=                       ; UC1: DRIVERS
=                       ;
=                       UC1IN:
=169A BB2014                    MOV     BX,OFFSET TCOM2CB       ; POINT TO CONTROL BLOCK
=169D E9F8FD       1498         JMP     P232IN
=                       ;
=                       UC1OUT:
```

```
=16A0 BB2014                    MOV     BX,OFFSET TCOM2CB       ; POINT TO CONTROL BLOCK
=16A3 E93DFE        14E3        JMP     P232OUT
=                       ;
=                       UC1ST:
=16A6 BB2014                    MOV     BX,OFFSET TCOM2CB       ; POINT TO CONTROL BLOCK
=16A9 E982FE        152E        JMP     P232STI
=                       ;
=                       ;
=                       ;
=                       ; UR1: DRIVERS
=                       ;
=  169A                 UR1IN   EQU     UC1IN
=  16A6                 UR1ST   EQU     UC1ST
=                       ;
=                       ; UR2: DRIVERS
=                       ;
=  1693                 UR2IN   EQU     PNULIN
=  1697                 UR2ST   EQU     PNULSTI
=                       ;
=                       ; UP1: AND UP2: DRIVERS
=                       ;
=  16A0                 UP1OUT  EQU     UC1OUT
=  1696                 UP2OUT  EQU     PNULOUT
=                       ;
=                       ; UL1: DRIVERS
=                       ;
=  1696                 UL1OUT  EQU     PNULOUT
```

```
= 1697                  ULlST   EQU     PNULSTO
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=                       ;
=                       ;
=
=16AC 2EFF26CF16        CONST:  jmp word ptr constjmp           ;console status
=
=                       CONIN:                  ;console input
=16B1 2EFF26D116                jmp word ptr coninjmp
=
=16B6 2EFF26D316        CONOUT: jmp word ptr conoutjmp          ;console output
=
=16BB 2EFF26D916        LISTST: jmp word ptr liststjmp          ;list device status
=
=16C0 2EFF26DB16        LISTOUT:jmp word ptr listoutjmp         ;output to list device
=
=16C5 2EFF26D716        PUNCH:  jmp word ptr punchjmp           ;output to punch
=
=16CA 2EFF26D516        READER: jmp word ptr readerjmp          ;input from reader
=
=                       ;Indirect jump table for I/O
=16CF BF13              constjmp        dw      CRTSTI
=16D1 8E13              coninjmp        dw      CRTIN
=16D3 8413              conoutjmp       dw      CRTOUT
=16D5 6916              readerjmp       dw      PTRIN
=16D7 6F16              punchjmp        dw      PTPOUT
```

```
=16D9 8D16              liststjmp       dw      LPTSTO
=16DB 8116              listoutjmp      dw      LPTOUT
=
=
=
=
=
=                       GETIOBF:
=16DD 2EA0A41C                  mov al,IOBYTE
=16E1 C3                        ret
=
=                       SETIOBF:
=16E2 2E880EA41C                mov IOBYTE,cl   ;set iobyte
=                       IF      NOT LOADER_BIOS
=                       ;-----------------------------------------------;
=                       ;|                                             |
=                               mov di,offset constjmp
=                               mov si,offset iojtbl
=                               call cioset                     ;low two bits
=                               call iojset                     :const
=                               call iojset                     ;conin
=                               call iojset                     ;conout
=                               call cioset                     ;second two bits
=                               call iojset                     ;reader
=                               call cioset                     ;third two bits
=                               call iojset                     ;punch
=                               call cioset                     ;high two bits
```

```
=                               call iojset                     ;liststst
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=                               call iojset                     ;listout
=                               ret
=
=                       iojtbl  dw      TTYSTI  ;CONST
=                               dw      CRTSTI
=                               dw      BATST
=                               dw      UC1ST
=
=                               dw      TTYIN   ;CONIN
=                               dw      CRTIN
=                               dw      BATIN
=                               dw      UC1IN
=
=                               dw      TTYOUT  ;CONOUT
=                               dw      CRTOUT
=                               dw      BATOUT
=                               dw      UC1OUT
=
=                               dw      TTYIN   ;READER
=                               dw      PTRIN
=                               dw      UR1IN
=                               dw      UR2IN
=
=                               dw      TTYOUT  ;PUNCH
```

```
=                       dw      PTPOUT
=                       dw      UP1OUT
=                       dw      UP2OUT
=
=                       dw      TTYSTO  ;LISTST
=                       dw      CRTSTO
=                       dw      LPTSTO
=                       dw      ULlST
=
=                       dw      TTYOUT  ;LPTOUT
=                       dw      CRTOUT
=                       dw      LPTOUT
=                       dw       L1OUT
=
=
=
=               cioset: ;Set the offsets in bx according to the low two bits in cl
=                       mov bx,3        ;and then rotate cl twice
=                       and bl,cl       ;
=                       add bx,bx       ;twice the number for offset
=                       shr cl,1        ;shift cl right ...
=                       shr cl,1        ;twice
=                       ret
=
=               iojset: ;Move the appropriate entry from the jump list to the indirect
=                       mov cs:ax,[bx+si]       ;jump table and then increment the pointers
=                       mov cs:[di],ax          ;for the next call
```

```
=                               add si,8                ;
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=                               add di,2                ;
=                       ;|                                                  |
=                       ;|--------------------------------------------------|
=                       ENDIF   ;not loader_bios
=16E7 C3                        ret
=
=                       GETSEGT:  ;return address of physical memory table
=16E8 BB261E                    mov bx,offset seg_table
=16EB C3                        ret
=
=                       ;
=                       VIDEO:  ;Output video directly to PC-100 via int 40
=16EC 1E                        push ds
=16ED 06                        push es
=16EE 51                        push cx
=16EF 52                        push dx
=16F0 BF0800                    mov di,8
=16F3 CD28                      int 40          ; turn cursor off
=16F5 5D                        pop bp
=16F6 8EDD                      mov ds,bp
=16F8 5E                        pop si
=16F9 8B04                      mov ax,[si]
=16FB 8B5C02                    mov bx,2[si]
=16FE 8B4C04                    mov cx,4[si]
```

```
=1701 8B5406                    mov dx,6[si]
=1704 8B7408                    mov si,8[si]
=1707 BF1400                    mov di,14h
=170A CD28                      int 40          ; move in the video
=170C BF0A00                    mov di,0ah
=170F CD28                      int 40          ; turn cursor on
=1711 07                        pop es
=1712 1F                        pop ds
=1713 C3                        ret
=                       ;

=                               INCLUDE CPLBLOK.LIB
=                       ;       SECTOR BLOCKING/DEBLOCKING
=
=                       ;       Modified for CP/M 86/80
=                       ;       May 1982  by CPL (RK)
=
=                       ;*****************************************************
=                       ;*                                                   *
=                       ;*         CP/M to host disk constants               *
=                       ;*                                                   *
=                       ;*****************************************************
=  0000                 una     equ     byte ptr [BX]   ;name for byte at BX
=                       ;
=  0004                 nrdisks equ     4               ;four disks allowed
=  0800                 blksiz  equ     2048            ;CP/M allocation size
=  0200                 hstsiz  equ     512             ;host disk sector size
```

```
=  000A                 hstspt  equ     10              ;host disk sectors/trk
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=  0004                 hstblk  equ     hstsiz/128      ;CP/M sects/host buff
=  FD00                 xfrbuf  equ     pb2_adr+xdefbuf ;buffer in shared memory
=                       ;
=                       ;*******************************************************
=                       ;*                                                     *
=                       ;* secshf is log2(hstblk), and is listed below for     *
=                       ;* values of hstsiz up to 2048.                        *
=                       ;*                                                     *
=                       ;*           hstsiz    hstblk    secshf                *
=                       ;*              256         2         1                *
=                       ;*              512         4         2                *
=                       ;*             1024         8         3                *
=                       ;*             2048        16         4                *
=                       ;*                                                     *
=                       ;*******************************************************
=  0002                 secshf  equ     2               ;log2(hstblk)
=  0028                 cpmspt  equ     hstblk * hstspt ;CP/M sectors/track
=  0003                 secmsk  equ     hstblk-1        ;sector mask
=                       ;
=                       ;*******************************************************
=                       ;*                                                     *
=                       ;*        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
=                       ;
=                       ;***************************************************
=                       ;*                                                 *
=                       ;*      BDOS function table beginning              *
=                       ;*                                                 *
=                       ;***************************************************
=                               if not loader_bios
=                       ;_____________________________________________________;
=                       bdos_ftbl       equ     0a80h
=                       bdos_dlog       equ     22eah
=                       ;_____________________________________________________;
=                               endif   ;not loader_bios
=                       ;
=                       ;***************************************************
=                       ;*                                                 *
=                       ;*      The BIOS entry points given below show the *
=                       ;*      code which is relevant to deblocking only. *
=                       ;*                                                 *
=                       ;***************************************************
=                       seldsk:
=1714 80F904                    cmp cl,nrdisks          ;valid disk number?
=1717 7204          171D        jb seldskl              ;go ahead if ok
=1719 BB0000                    mov bx,0                ;else zero bx
=171C C3                        ret                     ;and let bdos take care of it
```

```
=                               ;select disk
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=                               ;is this the first activation of the drive?
=                       seldsk1:
=171D 2EA0A01A                  mov al,hstdsk           ;which disk?
=1721 3CFF                      cmp al,0ffh             ;absolutely first activation?
=1723 7409          172E        je seldsk2              ;go clear host buffer
=1725 F6C201                    test DL,1               ;lsb = 0?
=1728 7510          173A        jnz selset
=                               ;if this is the first activation, clear host buff
=172A 3AC1                      cmp al,cl               ;but is this the same disk?
=172C 750C          173A        jne selset              ;if not, don't
=                       seldsk2:
=172E 2EC606A51A00              mov hstact,0
=1734 2EC606A71A00              mov unacnt,0
=                       selset:
=173A 8AC198                    mov al,cl ! cbw         ;put in AX
=173D 2EA29C1A                  mov sekdsk,al           ;seek disk number
=1741 B104D2E0                  mov cl,4 ! shl al,cl    ;times 16
=1745 05B51A                    add ax,offset dpbase
=1748 8BD8                      mov bx,ax
=                       if not loader_bios
=                       ;_____________________________________________________;
=                       ;check for Robin media on any unallocated disk called
=                               push dx                 ;save dx and bx
                                push bx
```

```
=                mov bx,word ptr .bdos_dlog      ;get allocation vector
=                or bx,bx                ;any allocated?
=                jz seldl                ;skip ahead if none
=                pop bx                  ;else find out ...
=                push bx                 ;
=                add bx,0ah              ;if this disk has been checked
=                mov bx,[bx]             ;get dpb offset
=                or bx,bx                ;zero? not checked yet
=                jz seld2                ;go check it if so
=                mov mediatype,0         ;assume Rainbow
=                cmp bx,dpb0             ;is it really?
=                jz seld0                ;move on if so
=                mov mediatype,2         ;else mark as Robin
=       seld0:   pop bx                  ;then ...
=                pop dx                  ;
=                ret                     ;return
=
=       seldl:   xor ax,ax               ;zero all dpb's
=                mov bx,offset dpbase+0ah        ;dpb offset from base
=                mov cx,nrdisks          ;loop for all disks
=
=       seldll:  mov [bx],ax             ;zero a dpb
=                add bx,10h              ;get the next one
=                loop seldll             ;loop till done
=
=       seld2:   mov ax,15h              ;get media type
=                push ax                 ;via packer
```

```
=                               mov al,sekdsk           ;get disk number back again
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=                               mov cl,5                ; and put in proper
=                               rol ax,cl               ;  position for packet
=                               push ax                 ;
=                               mov cx,2                ;move 2 words
=                               call packer
=                               pop ax                  ;discard drive number
=                               pop ax                  ;ah has status/media type
=                               xor bx,bx               ;clean out a register
=                               mov mediatype,ah        ;mark the media type
=                               mov bl,ah               ;get the type (Rainbow = 0, robin = 2
=                               add bx,mediatbl         ;add the table address
=                               mov dx,[bx]             ;get contents of table
=                               pop bx                  ;
=                               push bx                 ;get dpb pointer
=                               add bx,0ah              ;
=                               mov [bx],dx             ;put address ilnto pointer
=                               pop bx                  ;get return data
=                               pop dx                  ;
=                       ; ______________________________________________________;
=                               endif   ;not loader_bios
=174A C3                        ret
=                       ;
=                       home:
=                               ;home the selected disk
```

```
=174B 2EA0A61A                  mov al,hstwrt           ;check for pending write
=174F 84C0                      test al,al
=1751 7506          1759        jnz homed
=1753 2EC606A51A00              mov hstact,0            ;clear host active flag
=                       homed:
=1759 B90000                    mov cx,0                ;now, set track zero
=                       ;       (continue HOME routine)
=                       ;
=                       settrk:
=                               ;set track given by registers CX
=175C 2E890E9D1A                mov sektrk,CX           ;track to seek
=1761 C3                        ret
=                       ;
=                       setsec:
=                               ;set sector given by register cl
=1762 2E880E9F1A                mov seksec,cl           ;sector to seek
=1767 C3                        ret
=                       ;
=                       setdma:
=                               :set dma address given by CX
=1768 2E890EB21A                mov dma_off,CX
=176D C3                        ret
=                       ;
=                       setdmab:
=                               ;set segment address given by CX
=176E 2E890EB01A                mov dma_seg,CX
=1773 C3                        ret
```

```
=                       ;
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=                       sectran:
=                               ;translate sector number CX with table at [DX]
=1774 85D2                      test DX,DX      ;test for hard skewed
=1776 7409          1781        jz notran       ;(blocked must be hard skewed)
=1778 8BD9                      mov BX,CX
=177A 03DA                      add BX,DX
=177C 8A1F                      mov BL,[BX]
=177E B700                      mov BH,0        ;*** be compatible with 86/80  (fix 5/21/82) ***
=1780 C3                        ret
=                       no_tran:
=                               ;hard skewed disk, physical = logical sector
=1781 8BD9                      mov BX,CX
=1783 C3                        ret
=                       ;
=                       read:
=                               ;read the selected CP/M sector
=1784 2EC606A71A00              mov unacnt,0            ;clear unallocated counter
=178A 2EC606AE1A01              mov readop,1            ;read operation
=1790 2EC606AD1A01              mov rsflag,1            ;must read data
=1796 2EC606AF1A02              mov wrtype,wrual        ;treat as unalloc
=179C E98D00        182C        jmp rwoper              ;to perform the read
=                       ;
=                       write:
=                               :write the selected CP/M sector
```

```
=                               if not loader_bios
=                       ;-----------------------------------------------;
=                               test mediatype,0ffh     ;is this Rainbow media?
=                               jz okwrite              ;go write if so
=                               mov bx,offset unabl     ;'unable to write...'
=                               call pmsg               ;print it
=                               call prthst             ;print drive no.
=                               mov bx,offset endlin    ;
=                               call pmsg               ;print cr,lf.
=                               mov ax,1                ;mark for error
=                               ret                     ;return
=                       okwrite:
=                       ;_______________________________________________;
=                               endif   ;not loader_bios
=179F 2EC606AE1A00              mov readop,0            ;write operation
=17A5 2E880EAF1A                mov wrtype,cl
=17AA 80F902                    cmp cl,wrual            ;write unallocated?
=17AD 751E          17CD        jnz chkuna              ;check for unalloc
=                       ;
=                       ;       write to unallocated, set parameters
=                       ;
=17AF 2EC606A71A10              mov unacnt,(blksiz/128) ;next unalloc recs
=17B5 2EA09C1A                  mov al,sekdsk           ;disk to seek
=17B9 2EA2A81A                  mov unadsk,al           ;unadsk = sekdsk
=17BD 2EA19D1A                  mov ax,sektrk
=17C1 2EA3A91A                  mov unatrk,ax           ;unatrk = sektrk
=17C5 2EA09F1A                  mov al,seksec
```

```
=17C9 2EA2AB1A                  mov unasec,al           ;unasec = seksec
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=                       ;
=                       chkuna:
=                               ;check for write to unallocated sector
=                       ;
=17CD BBA71A                    mov bx,offset unacnt    ;point "UNA" at UNACNT
=17D0 8A0784C0                  mov al,una ! test al,al ;any unalloc remain?
=17D4 744A          1820        jz alloc                ;skip if not
=                       ;
=                       ;       more unallocated records remain
=17D6 FEC8                      dec al                  ;unacnt = unacnt-1
=17D8 8807                      mov una,al
=17DA 2EA09C1A                  mov al,sekdsk           ;same disk?
=17DE BBA81A                    mov BX,offset unadsk
=17E1 3A07                      cmp al,una              ;sekdsk = unadsk?
=17E3 753B          1820        jnz alloc               ;skip if not
=                       ;
=                       ;       disks are the same
=17E5 2EA1A91A                  mov AX, unatrk
=17E9 2E3B069D1A                cmp AX, sektrk
=17EE 7530          1820        jnz alloc               ;skip if not
=                       ;
=                       ;       tracks are the same
=17F0 2EA09F1A                  mov al,seksec           ;same sector?
=                       ;
```

```
=17F4 BBAB1A                    mov BX,offset unasec    ;point una at unasec
=                       ;
=17F7 3A07                      cmp al,una              ;seksec = unasec?
=17F9 7525          1820        jnz alloc               ;skip if not
=                       ;
=                       ;       match, move to next sector for future ref
=                       ;        (Code modified for skewed sectors)
=                       ;
=17FB 8CD9                      mov cx,ds               ; set up ES
=17FD 8EC1                      mov es,cx
=17FF FC                        cld                     ; scan forward
=1800 B92700                    mov cx,cpmspt-1         ; set count for scan
=1803 BF041B                    mov di,xlt0             ; point to translate table
=1806 F2AE                      repne scasb             ; scan for sector number
=1808 E306          1810        jcxz ovf                ;  didn't find it
=180A 8A05                      mov al,[di]             ; get nr of next sector
=180C 8807                      mov una,al
=180E EB08          1818        jmps noovf
=                       ;
=                       ;       overflow to next track
=                       ovf:
=1810 C60700                    mov una,0               ;unasec = 0
=1813 2EFF06A91A                inc unatrk              ;unatrk=unatrk+1
=                       ;
=                       noovf:
=                               ;match found, mark as unnecessary read
=1818 2EC606AD1A00              mov rsflag,0            ;rsflag = 0
```

```
=181E EB0C         182C         jmps rwoper             ;to perform the write
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=                       ;
=                       alloc:
=                               ;not an unallocated record, requires pre-read
=1820 2EC606A71A00              mov unacnt,0            ;unacnt = 0
=1826 2EC606AD1A01              mov rsflag,1            ;rsflag = 1
=                                                       ;drop through to rwoper
=                       ;
=                       ;*******************************************************
=                       ;*                                                     *
=                       ;*      Common code for READ and WRITE follows         *
=                       ;*                                                     *
=                       ;*******************************************************
=                       rwoper:
=                               ;enter here to perform the read/write
=182C 2EC606AC1A00              mov erflag,0            ;no errors (yet)
=1832 2EA09F1A                  mov al, seksec          ;compute host sector
=1836 B102                      mov cl, secshf
=1838 D2E8                      shr al,cl
=183A FEC0                      inc al                  ;*** added for 1-based sectors ***
=183C 2EA2A41A                  mov sekhst,al           ;host sector to seek
=                       ;
=                       ;       active host sector?
=1840 B001                      mov al,1
=1842 2E8606A51A                xchg al,hstact          ;always becomes 1
```

```
=1847 84C0                      test al,al              ;was it already?
=1849 7437          1882        jz filhst               ;fill host if not
=                       ;
=                       ;       host buffer active, same as seek buffer?
=184B 2EA09C1A                  mov al,sekdsk
=184F 2E3A06A01A                cmp al,hstdsk           ;sekdsk = hstdsk?
=1854 7516          186C        jnz nomatch
=                       ;
=                       ;       same disk, same track?
=1856 2EA1A11A                  mov ax,hsttrk
=185A 2E3B069D1A                cmp ax,sektrk           ;host track same as seek track
=185F 750B          186C        jnz nomatch
=                       ;
=                       ;       same disk, same track, same buffer?
=1861 2EA0A41A                  mov al,sekhst
=1865 2E3A06A31A                cmp al,hstsec           ;sekhst = hstsec?
=186A 744A          18B6        jz match                ;skip if match
=                   nomatch:
=                               ;proper disk, but not correct sector
=186C 2EA0A61A                  mov al, hstwrt
=1870 84C0                      test al,al              ;"dirty" buffer ?
=1872 740E          1882        jz filhst               ;no, don't need to write
=1874 E8A500        191C        call writehst           ;yes, clear host buff
=                       ;       (check errors here)
=1877 2EA0AC1A                  mov al,erflag
=187B 0AC0                      or al,al                ;any?
=187D 7403          1882        jz filhst               ;skip if none
```

```
=187F E99500       1917         jmp return_rw           ;exit if so
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=                       ;
=                       filhst:
=                               ;may have to fill the host buffer
=1882 2EA09C1A2EA2              mov al,sekdsk ! mov hstdsk,al
      A01A
=188A 2EA19D1A2EA3              mov ax,sektrk ! mov hsttrk,ax
      A11A
=1892 2EA0A41A2EA2              mov al,sekhst ! mov hstsec,al
      A31A
=189A 2EA0AD1A                  mov al,rsflag
=189E 84C0                      test al,al              ;need to read?
=18A0 740E         18B0         jz filhstl
=                       ;
=18A2 E8CD00       1972         call readhst            ;yes, if 1
=                       ;       (check errors here)
=18A5 2EA0AC1A                  mov al,erflag
=18A9 0AC0                      or al,al                ;any?
=18AB 7403         18B0         jz filhstl              ;skip if none
=18AD E96700       1917         jmp return_rw           ;exit if so
=                       ;
=                       filhstl:
=18B0 2EC606A61A00              mov hstwrt,0            ;no pending write
=                       ;
=                       match:
```

```
=                               ;copy data to or from buffer depending on "readop"
=18B6 2EA09F1A                  mov al,seksec           ;mask buffer number
=18BA 250300                    and ax,secmsk           ;least signif bits are masked
=18BD B107D3E0                  mov cl, 7 ! shl ax,cl   ;shift left 7 (* 128 = 2**7)
=
=                       ;       ax has relative host buffer offset
=                       ;
=18C1 05003A                    add ax,dbptr            ; address of data block
=18C4 05A200                    add ax, xshrbuf         ;*** new buffer for CP/M 86/80 ***
=18C7 8BF0                      mov si,ax               ;put in source index register
=18C9 2E8B3EB21A                mov di,dma_off          ;user buffer is dest if readop
=                       ;
=18CE 1E06                      push DS ! push ES       ;save segment registers
=                       ;
=18D0 2E8E06B01A                mov ES,dma_seg          ;set destseg to the users seg
=                                                       ;SI/DI and DS/ES is swapped
=                                                       ;if write op
=18D5 2BC0                      sub ax,ax               ;*** added for CP/M 86/80 ***
=18D7 8ED8                      mov ds,ax               ;which needs ds=0 (*** end add ***)
=18D9 B98000                    mov cx,128              ;length of move in bytes
=18DC 2EA0AE1A                  mov al,readop
=18E0 84C0                      test al,al              ;which way?
=18E2 7511          18F5        jnz     rwmovx          ;skip if read
=                       ;
=                       ;       write operation, mark and switch direction
=18E4 2EC606A61A01              mov hstwrt,1            ;hstwrt = 1 (dirty buffer now)
=18EA 87F7                      xchg si,di              ;source/dest index swap
```

```
=18EC 8CD8                      mov ax,DS
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=18EE 8EC0                      mov ES,ax
=18F0 2E8E1EB01A                mov DS,dma_seg          ;setup DS,ES for write
=18F5 E82000       1918 rwmovx: call rwmove
=                       ;
=
=18F8 071F                      pop ES ! pop DS         ;restore segment registers
=                       ;
=                       movdone:
=                       ;       data has been moved to/from host buffer
=18FA 2E803EAF1A01              cmp wrtype,wrdir        ;write type to directory?
=1900 2EA0AC1A                  mov al,erflag           ;in case of errors
=1904 7511         1917         jnz return_rw           ;no further processing
=                       ;
=                       ;       clear host buffer for directory write
=1906 84C0                      test al,al              ;errors?
=1908 750D         1917         jnz return_rw           ;skip if so
=190A 2EC606A61A00              mov hstwrt,0            ;buffer written
=1910 E80900       191C         call writehst
=1913 2EA0AC1A                  mov al,erflag
=                       return_rw:
=1917 C3                        ret
=                       ;
=                       ;***********************************************
=                       ;*      this subroutine is made available      *
```

```
=                       ;*      for other parts of CP/M.                *
=                       ;************************************************
=
=                       rwmove:
=                               if not loader_bios
=                       ;-----------------------------------------
=                       ;       Added for PC100 -- RK/CPL -- 4/13/82
=                               test Z80FLAG,true       ; is z80 running?
=                               jnz rwml                ;
=                               jmp move88              ;normal move if z80 not running
=
=                       rwml:   push cx                 ;preserve the count
=                               mov ax,cx               ;get the count into accum
=                               mov bx,ds               ;check the source segment
=                               mov dx,es               ;and destination segment
=                               test bx,0ff80h          ;out of Z80 private?
=                               jnz nsz80p              ;jump ahead if it is
=                               mov cl,4                ;set for shl
=                               shl bx,cl               ;mult by 16
=                               add bx,si               ;get absolute address for Z80
=                               cmp bx,800h             ;in z80 pvt?
=                               jb sz80p                ;jump ahead so
=
=                       nsz80p: test dx,0ff80h          ;source is not z80 private.
=                               jz nszl                 ;
=                               jmp notz80              ;jump ahead if destination isn't, either
=                       nszl:   mov cl,4                ;set for shl
```

```
=                               shl dx,cl               ;mul dest seg by 16
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy



=                               add dx,di               ;create absolute address
=                               cmp dx,800h             ;dest in z80 pvt?
=                               jb dz80p                ;
=                               jmp notz80              ;jump ahead if not
=
=                       dz80p:  mov bx,ds               ;destination is in z80 pvt, source isn't
=                               test bx,0f000h          ;check for source in 8088 pvt
=                               jnz dzs88p              ;dest z80, source 8088.
=                               mov cl,4                ;
=                               shl bx,cl               ;
=                               add bx,si               ;absolute source
=                               jc dzs88p               ;dest z80, source 8088.
=                               add bx,ax               ;how about the end of the source?
=                               jnc not88               ;z80 move if low enough
=
=                       dzs88p: pop cx                  ;dest in z80 pvt, source in 8088 pvt
=                               cmp cx,128              ;is count too big?
=                               jbe dzs1                ;
=                               mov cx,128              ;make it maximum if so
=                       dzs1:   push ds                 ;save ds for later
=                               push cx                 ;save count for later
=                               push es                 ;
=                               push di                 ;
=                               xor ax,ax               ;zero ...
```

```
=                       mov es,ax               ;the destination segment
=                       mov di,xfrbuf           ;destination is buffer
=                       call move88             ;move source to buffer
=                       pop di                  ;
=                       pop es                  ;
=                       mov ds,ax               ;zero the source segment
=                       mov si,xfrbuf           ;source is buffer
=                       pop cx                  ;
=                       call movz80             ;move buffer to destination after restoring cx
=                       pop ds                  ;restore ds
=                       ret                     ;
=
=               sz80p:  mov dx,es               ;source is in z80 pvt, destination isn't
=                       test dx,0f000h          ;check for destination in 8088 pvt
=                       jnz d88szp              ;dest 8088, source z80.
=                       mov cl,4                ;
=                       shl dx,cl               ;
=                       add dx,di               ;absolute destination
=                       jc d88szp               ;source z80, destination 8088.
=                       add dx,ax               ;how about the end of the destination?
=                       jnc not88               ;z80 move if low enough
=
=               d88szp: pop cx                  ;dest in z80 pvt, source in 8088 pvt
=                       cmp cx,128              ;is count too big?
=                       jbe d88sl               ;
=                       mov cx,128              ;make it maximum if so
=               d88sl:  push ds                 ;save ds for later
```

```
=                               push cx                 ;save count for later
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=                               push es                 ;
=                               push di                 ;
=                               xor ax,ax               ;zero ...
=                               mov es,ax               ;the destination segment
=                               mov di,xfrbuf           ;destination is buffer
=                               call movz80             ;move source to buffer
=                               pop di                  ;
=                               pop es                  ;
=                               mov ds,ax               ;zero the source segment
=                               mov si,xfrbuf           ;source is buffer
=                               call move88             ;move buffer to destination after restoring cx
=                               pop ds                  ;restore ds
=                               ret                     ;
=
=                       not88:  pop cx                  ;do a z80 move and exit
=
=                       movz80: mov bx,ds               ;Z80 move to emulate 8088 move
=                               mov dx,es               ;
=                               push cx                 ;
=                               mov cl,4                ;
=                               shl bx,cl               ;
=                               shl dx,cl               ;
=                               add bx,si               ;bx has source absolute
=                               add dx,di               ;dx has destination absolute
```

```
=                       mov ax,22h              ;function: transfer ...
=                       pop cx                  ;to Z80 move routine
=                       push ax                 ;via packer
=                       push bx                 ;
=                       push dx                 ;
=                       push cx                 ;
=                       mov cx,0004             ;4 words stacked
=                       xor al,al               ;zero for Z80 wait
=                       call packer             ;call interface layer
=                       pop ax                  ;after return,
=                       pop ax                  ;level the stack
=                       pop ax                  ;
=                       pop ax                  ;
=                       ret                     ;go past 8088 move
=
=                   notz80: pop cx              ;don't use z80 move
=
=                   move88:                     ;continue to normal -- end of additiion
=                   ;-------------------------------------------------------
=                       endif                   ;not loader_bios
=
=1918 FCF3A4                cld ! rep movs AL,AL    ;move as bytes
=191B C3                    ret                     ;end of move subroutine
=
=                   ;*******************************************************
=                   ;*                                                     *
=                   ;* WRITEHST performs the physical write to the host    *
```

```
=                       ;* disk, while READHST reads the physical disk.      *
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=                       ;*                                                   *
=                       ;*****************************************************
=
=                       ;|-----------------------------------------------------|
=                       ;| Code added for pcl00, CP/M 86/80 4/13/82            |
=
=                       writehst:
=191C B81400            rewhst: mov ax,0014h            ;try once (or retry)
=191F E83B01       1A5D         call rwhst              ;
=1922 F6C4FC                    test ah,0fch            ;check for error
=1925 7501         1928         jnz wragain             ;try if any error
=1927 C3                        ret                     ;else it's done
=
=                       wragain:                        ;If at first...
=1928 F6C480                    test ah,80h             ;is the drive ready?
=192B 7514         1941         jnz wnrdy               ;error if not
=192D F6C440                    test ah,40h             ;is drive write-protected?
=1930 7531         1963         jnz writeprot           ;error if so
=1932 F6C401                    test ah,1               ;seek error
=1935 7510         1947         jnz wrsker              ;
=1937 BB8F1D                    mov bx,offset biwmsg    ;then give up.
=193A E8BEF9       12FB         call pmsg               ;no sense making a fool of yourself
=193D E8BF00       19FF         call rwerr              ;print messages
=1940 C3                        ret
```

```
=
=1941 E87000      19B4 wnrdy:  call notready          ;disk not ready, print message
=1944 74D6        191C         jz rewhst              ;try again if so directed
=1946 C3                       ret                    ;else return with error
=
=1947 BBA71D           wrsker: mov bx,offset skmsg
=194A E8AEF9      12FB         call pmsg
=194D E8FC00      1A4C         call prthst            ;print drive number
=1950 BBFB1D                   mov bx,offset trkmsg   ;print 'track'
=1953 E8A5F9      12FB         call pmsg
=1956 2EA1A11A                 mov ax,hsttrk          ;get track number
=195A E8D200      1A2F         call decprt            ;print decimal
=195D E85D00      19BD         call kqery
=1960 74BA        191C         jz rewhst              ;try again?
=1962 C3                       ret                    ;or return with error
=
=                      writeprot:                     ;disk is write-protected
=1963 BB5C1D                   mov bx,offset wpmsg
=1966 E892F9      12FB         call pmsg              ;print error message
=1969 E8E000      1A4C         call prthst            ;print which one
=196C E84E00      19BD         call kqery             ;check keys
=196F 74AB        191C         jz rewhst              ;try again if directed
=1971 C3                       ret                    ;else return with bios error
=
=                      ;
=                      readhst:
=1972 B81300           rerhst: mov ax,0013h          ;ask interface layer
```

```
=1975 E8E500       1A5D         call rwhst              ;
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=1978 F6C4FC                    test ah,0fch            ;check for error
=197B 7501         197E         jnz rdagain             ;jump ahead if error
=197D C3                        ret                     ;else you're done
=
=                         rdagain:
=197E F6C480                    test ah,80h             ;drive ready?
=1981 750F         1992         jnz rdnrdy              ;go away if not
=1983 F6C401                    test ah,1               ;seek error?
=1986 7510         1998         jnz rdsker              ;take care of it
=1988 BB781D                    mov bx,offset birmsg    ;then give up.
=198B E86DF9       12FB         call pmsg               ;print an error message
=198E E86E00       19FF         call rwerr              ;print messages
=1991 C3                        ret
=
=1992 E81F00       19B4 rdnrdy: call notready           ;not ready -- print message, etc.
=1995 74DB         1972         jz readhst              ;try again if directed
=1997 C3                        ret                     ;else return with error
=
=1998 BBA71D            rdsker: mov bx,offset skmsg
=199B E85DF9       12FB         call pmsg
=199E E8AB00       1A4C         call prthst             ;print drive number
=19A1 BBFB1D                    mov bx,offset trkmsg    ;print 'track'
=19A4 E854F9       12FB         call pmsg
=19A7 2EA1A11A                  mov ax,hsttrk           ;get track number
```

```
=19AB E88100       1A2F         call decprt             ;print decimal
=19AE E80C00       19BD         call kqery
=19B1 74BF         1972         jz rerhst               ;try again?
=19B3 C3                        ret                     ;or return with error
=
=                         notready:                     ;routine to print "not ready"
=19B4 BBBE1D                    mov bx,offset nrmsg     ;get message address
=19B7 E841F9       12FB         call pmsg               ;print it then fall through ...
=19BA E88F00       1A4C         call prthst             ;print which one
=
=                         kqery:                        ;routine to check keyboard for ^C
=19BD BBCF1C                    mov bx,offset kqmsg     ;standard keyboard message
=19C0 E838F9       12FB         call pmsg               ;print it
=19C3 E8EBFC       16B1         call CONIN              ;get character
=19C6 3C03                      cmp al,03               ;ctrl c?
=19C8 7515         19DF         jnz kqnex               ;skip ahead if not
=19CA 2EC606A71A00              mov unacnt,0            ;clear write flags
=19D0 2EC606A51A00              mov hstact,0            ;
=19D6 2EC606AF1A02              mov wrtype,wrual        ;
=                         if not loader_bios
=                         ;-------------------------------------------------------
=                               mov Z80FLAG,false       ;clear z80 flag
=                               call revector           ;start over if so
=                               mov cl,byte ptr .curdrvs
=                               jmp ccp+3               ;go back to ccp
=                         ;-------------------------------------------------------
=                         endif ;not loader_bios
```

```
=                       if loader_bios
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=                       ;-------------------------------------------------------
=19DC E998F9        1377        jmp WBOOT
=                       ;-------------------------------------------------------
=                       endif ;loader_bios
=19DF 3C20              kqnex:  cmp al,' '              ;space bar for retry
=19E1 7503          19E6        jnz kqret               ;other key for error return
=19E3 32C0                      xor al,al               ;zero it
=19E5 C3                        ret                     ;
=19E6 2EC606AC1A01      kqret:  mov erflag,1            ;error code
=19EC 2EC606A71A00              mov unacnt,0            ;clear write flags
=19F2 2EC606A51A00              mov hstact,0            ;
=19F8 2EC606AF1A02              mov wrtype,wrual        ;
=19FE C3                        ret                     ;else return
=
=                       ;****************************************
=                       ;*
=                       ;*      print disk, track, sector
=                       ;*
=                       ;****************************************
=
=19FF E84A00        1A4C rwerr: call prthst             ;print disk":"
=1A02 BBFB1D                    mov bx,offset trkmsg    ;print" TRACK "
=1A05 E8F3F8        12FB        call pmsg
=1A08 2EA1A11A                  mov ax,hsttrk           ;print track no.
```

```
=1A0C E82000       1A2F          call decprt
=1A0F BB041E                     mov bx,offset secmsg    ;print " SECTOR "
=1A12 E8E6F8       12FB          call pmsg
=1A15 2EA0A31A                   mov al,hstsec           ;print sector no.
=1A19 32E4                       xor ah,ah
=1A1B E81100       1A2F          call decprt
=1A1E BB201D                     mov bx,offset kqxmsg    ;ask for options
=1A21 E8D7F8       12FB          call pmsg
=1A24 E88AFC       16B1          call CONIN              ;
=1A27 3C0D                       cmp al,cr               ;carriage return?
=1A29 7501         1A2C          jne rwmker              ;mark error if not
=1A2B C3                         ret                     ;ignore if return
=1A2C E9B7FF       19E6 rwmker:  jmp kqret               ;return with marked error
=
=                          decprt: ;enter with number to be printed in ax
=1A2F 33C9                       xor cx,cx               ;count digits in cx
=1A31 33D2                 rediv:  xor dx,dx             ;clear the remainder
=1A33 BB0A00                     mov bx,10
=1A36 F7F3                       div bx                  ;divide by 10
=1A38 41                         inc cx                  ;bump the count
=1A39 52                         push dx                 ;store digit
=1A3A 0BC0                       or ax,ax                ;any more?
=1A3C 75F3         1A31          jnz rediv               ;loop if so
=1A3E 5A                   redout: pop dx                ;the last shall be first
=1A3F 51                         push cx                 ;store the count
=1A40 8ACA                       mov cl,dl               ;remainder in dl to cl
=1A42 80C130                     add cl,'0'              ;ascii number
```

```
=1A45 E86EFC       16B6         call CONOUT             ;print digit
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=1A48 59                        pop cx                  ;get the count
=1A49 E2F3         1A3E         loop redout             ;loop back if more
=1A4B C3                        ret                     ;else return
=
=                       prthst: ;print disk, A:, B:, etc.
=1A4C 2E8A0EA01A                mov cl,hstdsk           ;get the number
=1A51 80C141                    add cl,'A'
=1A54 E85FFC       16B6         call CONOUT             ;Print it
=1A57 B13A                      mov cl,':'
=1A59 E85AFC       16B6         call CONOUT             ;print a colon
=1A5C C3                        ret
=
=                       ;****************************************
=                       ;*
=                       ;*      read/write from host
=                       ;*
=                       ;****************************************
=
=1A5D 50                rwhst:  push ax                 ;function set for packer
=1A5E B400                      mov ah,0                ;clear high
=1A60 B104                      mov cl,4                ;set for rol
=1A62 2EA0A01A                  mov al,hstdsk           ;disk number
=1A66 D3C0                      rol ax,cl               ;times 32
=1A68 BBB51A                    mov bx,offset dpbase    ;get disk data address
```

```
=1A6B 03D8                      add bx,ax               ;for this disk
=1A6D 8B17                      mov dx,[bx]             ;get pointer in dx
=1A6F 2E8A1EA31A                mov bl,hstsec           ;host sector
=1A74 D1C0                      rol ax,1                ;rotate drive once more
=1A76 80E31F                    and bl,1fh              ;mask sector number (jic)
=1A79 0AC3                      or al,bl                ;combine sector
=1A7B 2E8B1EA11A                mov bx,hsttrk           ;and track
=1A80 8AE3                      mov ah,bl               ;in one byte
=1A82 50                        push ax                 ;stack for packer (2)
=1A83 B8003A                    mov ax,dbptr            ; point to data block
=1A86 05A200                    add ax,xshrbuf          ;point to buffer
=1A89 50                        push ax                 ;(3)
=1A8A B80100                    mov ax,0001             ;
=1A8D 50                        push ax                 ;(4)
=1A8E B80000                    mov ax,0                ;wait for z80
=1A91 B90400                    mov cx,4                ;4 words for packer
=1A94 E80CF8       12A3         call packer             ;
=1A97 58                        pop ax                  ;level the stack
=1A98 58                        pop ax                  ;
=1A99 58                        pop ax                  ;
=1A9A 58                        pop ax                  ;
=1A9B C3                        ret                     ;go away when done
=                       ;|                                                      |
=                       ;| End of added code                                    |
=                       ;|-------------------------------------------------------|
=
=                       ;
```

```
=                       ;******************************************************
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=                       ;*                                                    *
=                       ;* Use the GENDEF utility to create disk def tables  *
=                       ;*                                                    *
=                       ;******************************************************
=                       ;dpbase equ     offset $
=                       ;       disk parameter tables go here
=                       ;
=
=                       ;******************************************************
=                       ;*                                                    *
=                       ;* Uninitialized RAM areas follow, including the     *
=                       ;* areas created by the GENDEF utility listed above. *
=                       ;*                                                    *
=                       ;******************************************************
=1A9C                   sek_dsk rb      1               ;seek disk number
=1A9D                   sek_trk rw      1               ;seek track number
=1A9F                   sek_sec rb      1               ;seek sector number
=                       ;
=1AA0 FF                hst_dsk db      0ffh            ;host disk number
=1AA1                   hst_trk rw      1               ;host track number
=1AA3                   hst_sec rb      1               ;host sector number
=                       ;
=1AA4                   sek_hst rb      1               ;seek shr secshf
=1AA5                   hst_act rb      1               ;host active flag
```

```
=1AA6                   hst_wrt rb      1               ;host written flag
=                       ;
=1AA7                   una_cnt rb      1               ;unalloc rec cnt
=1AA8                   una_dsk rb      1               ;last unalloc disk
=1AA9                   una_trk rw      1               ;last unalloc track
=1AAB                   una_sec rb      1               ;last unalloc sector
=                       ;
=1AAC                   erflag  rb      1               ;error reporting
=1AAD                   rsflag  rb      1               ;read sector flag
=1AAE                   readop  rb      1               ;1 if read operation
=1AAF                   wrtype  rb      1               ;write operation type
=1AB0                   dma_seg rw      1               ;last dma segment
=1AB2                   dma_off rw      1               ;last dma offset
=1AB4                   mediatype rb    1               ;Rainbow or Robin media
=                       ;hstbuf rb      hstsiz          ;host buffer (not in CP/M 86/80)
                                end
=                               INCLUDE CAT.LIB
=                       ;       DISKS 5
=  1AB5                 dpbase  equ     $               ;Base of Disk Parameter Blocks
=1AB5 041B0000          dpe0    dw      xlt0,0000h      ;Translate Table
=1AB9 00000000                  dw      0000h,0000h     ;Scratch Area
=1ABD 3F1BF51A                  dw      dirbuf,dpb0     ;Dir Buff, Parm Block
=1AC1 D81BBF1B                  dw      csv0,alv0       ;Check, Alloc Vectors
=1AC5 041B0000          dpe1    dw      xlt1,0000h      ;Translate Table
=1AC9 00000000                  dw      0000h,0000h     ;Scratch Area
=1ACD 3F1BF51A                  dw      dirbuf,dpb1     ;Dir Buff, Parm Block
=1AD1 111CF81B                  dw      csv1,alv1       ;Check, Alloc Vectors
```

```
=1AD5 041B0000          dpe2    dw      xlt2,0000h      ;Translate Table
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=1AD9 00000000                  dw      0000h,0000h     ;Scratch Area
=1ADD 3F1BF51A                  dw      dirbuf,dpb2     ;Dir Buff, Parm Block
=1AE1 4A1C311C                  dw      csv2,alv2       ;Check, Alloc Vectors
=1AE5 041B0000          dpe3    dw      xlt3,0000h      ;Translate Table
=1AE9 00000000                  dw      0000h,0000h     ;Scratch Area
=1AED 3F1BF51A                  dw      dirbuf,dpb3     ;Dir Buff, Parm Block
=1AF1 831C6A1C                  dw      csv3,alv3       ;Check, Alloc Vectors
=                       ;-- DUMMY DISK FOR ROBIN MEDIA REFERENCE
=                       ;       DISKDEF 0,0,39,1,2048,195,128,128,2
=  1AF5                 dpb0    equ     offset $        ;Disk Parameter Block
=1AF5 2800                      dw      40              ;Sectors Per Track
=1AF7 04                        db      4               ;Block Shift
=1AF8 0F                        db      15              ;Block Mask
=1AF9 01                        db      1               ;Extnt Mask
=1AFA C200                      dw      194             ;Disk Size - 1
=1AFC 7F00                      dw      127             ;Directory Max
=1AFE C0                        db      192             ;Alloc0
=1AFF 00                        db      0               ;Alloc1
=1B00 2000                      dw      32              ;Check Size
=1B02 0200                      dw      2               ;Offset
=  1B04                 xlt0    equ     offset $        ;Translate Table
=                       ;** Modified for Rainbow media ...
=1B04 00010203                  db      0,1,2,3
=1B08 08090A0B                  db      8,9,10,11
```

```
=1B0C 10111213              db      16,17,18,19
=1B10 18191A1B              db      24,25,26,27
=1B14 20212223              db      32,33,34,35
=1B18 04050607              db      4,5,6,7
=1B1C 0C0D0E0F              db      12,13,14,15
=1B20 14151617              db      20,21,22,23
=1B24 1C1D1E1F              db      28,29,30,31
=1B28 24252627              db      36,37,38,39
=  0019             als0    equ     25              ;Allocation Vector Size
=  0020             css0    equ     32              ;Check Vector Size
=                   ;       DISKDEF 1,0
=  1AF5             dpb1    equ     dpb0            ;Equivalent Parameters
=  0019             als1    equ     als0            ;Same Allocation Vector Size
=  0020             css1    equ     css0            ;Same Checksum Vector Size
=  1B04             xlt1    equ     xlt0            ;Same Translate Table
=                   ;       DISKDEF 2,0
=  1AF5             dpb2    equ     dpb0            ;Equivalent Parameters
=  0019             als2    equ     als0            ;Same Allocation Vector Size
=  0020             css2    equ     css0            ;Same Checksum Vector Size
=  1B04             xlt2    equ     xlt0            ;Same Translate Table
=                   ;       DISKDEF 3,0
=  1AF5             dpb3    equ     dpb0            ;Equivalent Parameters
=  0019             als3    equ     als0            ;Same Allocation Vector Size
=  0020             css3    equ     css0            ;Same Checksum Vector Size
=  1B04             xlt3    equ     xlt0            ;Same Translate Table
=                   ;       DISKDEF 4,0,35,1,1024,171,64,64,2
=  1B2C             dpb4    equ     offset $        ;Disk Parameter Block
```

```
=1B2C 2400                      dw      36              ;Sectors Per Track
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=1B2E 03                        db      3               ;Block Shift
=1B2F 07                        db      7               ;Block Mask
=1B30 00                        db      0               ;Extnt Mask
=1B31 AA00                      dw      170             ;Disk Size - 1
=1B33 3F00                      dw      63              ;Directory Max
=1B35 C0                        db      192             ;Alloc0
=1B36 00                        db      0               ;Alloc1
=1B37 1000                      dw      16              ;Check Size
=1B39 0200                      dw      2               ;Offset
=  1B04                 xlt4    equ     xlt0            ;Translate Table
=  000C                 als4    equ     12              ;Allocation Vector Size
=  0010                 css4    equ     16              ;Check Vector Size
=                       ;       ENDEF
=                       ;
=                       ;       Table for transfering dpb's between rainbow and robin media:
=                       ;
=  1B3B                 mediatbl        equ offset $
=1B3B F51A                              dw      dpb0    ;Rainbow PC-100
=1B3D 2C1B                              dw      dpb4    ;Robin
=                       ;
=                       ;       Uninitialized Scratch Memory Follows:
=                       ;
=  1B3F                 begdat  equ     offset $        ;Start of Scratch Area
=1B3F                   dirbuf  rs      128             ;Directory Buffer
```

```
=1BBF                   alv0    rs      als0            ;Alloc Vector
=1BD8                   csv0    rs      css0            ;Check Vector
=1BF8                   alv1    rs      als1            ;Alloc Vector
=1C11                   csv1    rs      css1            ;Check Vector
=1C31                   alv2    rs      als2            ;Alloc Vector
=1C4A                   csv2    rs      css2            ;Check Vector
=1C6A                   alv3    rs      als3            ;Alloc Vector
=1C83                   csv3    rs      css3            ;Check Vector
=  1CA3                 enddat  equ     offset $        ;End of Scratch Area
=  0164                 datsiz  equ     offset $-begdat ;Size of Scratch Area
=1CA3 00                        db      0               ;Marks End of Module

=                               INCLUDE CPLBIOS2.A86
=
=
=
=                       ;*********************************************
=                       ;*                                           *
=                       ;*               Data Areas                  *
=                       ;*                                           *
=                       ;*********************************************
=
=  1AB2                 dma_adr equ dma_off
=
=1CA4 00                IOBYTE  db      0
=1CA5 0000              BXHLD   dw      0               ;store bx here
=1CA7 00                xoff_flg db     0
```

```
=
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy



=                               IF      loader_bios
=                       ;---------------------------------------
=                       ;|                                     |
=1CA8 1B5B324A1B5B      signon  db      27,'[2J',27,'[3;1H'
      333B3148
=1CB2 0D0A0D0A                  db      cr,lf,cr,lf
=1CB6 43502F4D2D38              DB      'CP/M-86/80 Loading ...',CR,LF,0
      362F3830204C
      6F6164696E67
      202E2E2E0D0A
      00
=                       ;|                                     |
=                       ;---------------------------------------
=                               ENDIF   ;loader_bios
=
=                       ;error messages:
=
=1CCF 0D0A50726573      kqmsg   db      cr,lf,'Press CTRL-C to restart, ',cr,lf
      73204354524C
      2D4320746F20
      726573746172
      742C200D0A
=1CEC 737061636520              db      'space bar to retry, or any other key to continue.'
      62617220746F
```

```
       207265747279
       2C206F722061
       6E79206F7468
       6572206B6579
       20746F20636F
       6E74696E7565
       2E
=1D1D 0D0A00            endlin  db      cr,lf,0
=
=1D20 0D0A              kqxmsg  db      cr,lf
=1D22 507265737320              db      'Press Return to ignore error, any other key to continue'
       52657475726E
       20746F206967
       6E6F72652065
       72726F722C20
       616E79206F74
       686572206B65
       7920746F2063
       6F6E74696E75
       65
=1D59 0D0A00                    db      cr,lf,0
=
=1D5C 0D0A44726976      wpmsg   db      cr,lf,'Drive write-protected -- ',0
       652077726974
       652D70726F74
       656374656420
       2D2D2000
```

=

CP/M ASM86 1.1 SOURCE: CPLDBIOS.A86 Customized Loader Basic I/O Sy

```
=1D78 0D0A52656164      er1     db      cr,lf,'Read error on drive ',0
      206572726F72
      206F6E206472
      6976652000
=1D8F 0D0A57726974      er2     db      cr,lf,'Write error on drive ',0
      65206572726F
      72206F6E2064
      726976652000
=1DA7 0D0A5365656B      skmsg   db      cr,lf,'Seek error on drive ',0
      206572726F72
      206F6E206472
      6976652000
=1DBE 0D0A44726976      er5     db      cr,lf,'Drive not ready -- ',0
      65206E6F7420
      726561647920
      2D2D2000
=1DD4 0D0A43616E6E      unabl   db      cr,lf,'Cannot write on VT180 disk on drive ',0
      6F7420777269
      7465206F6E20
      565431383020
      6469736B206F
      6E2064726976
      652000
=1DFB 2C2074726163      trkmsg  db      ', track ',0
```

```
      6B2000
=1E04 2C2073656374      secmsg  db      ", sector ",0
      6F722000
=
=  1D78                birmsg  equ er1
=  1D8F                biwmsg  equ er2
=  1DBE                nrmsg   equ er5
=
=1E0E 0D0A54696D65      p232to  db      cr,lf,'Timeout on '
      6F7574206F6E
      20
=1E1B 4544              p232to1 dw      'DE'
=1E1D 563A00            p232to2 db      'V:',0
=
=
=                               if      not loader_bios
=                       dispst  db      0
=                               endif
=                       ;
=1E20                   SEGHLD  rw 1            ;save segment
=1E22                   RTNHLD  rw 1            ;save return address
=1E24                   COUNT   rw 1            ;
=                       ;
=                       ;       System Memory Segment Table
=
=1E26                   segtable rb 1   ;2 segments
=1E27                           rw 4    ;room for two segs
```

```
=
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=                               IF      not loader_bios
=                       ;---------------------------------------------
=                       ;|                                           |
=                       signon  db      27,'[2J',27,'[3;1H'
=                               DB      '     CP/M-86/80 Version '
=                               db      version_number+'0'
=                               db      '.'
=                               db      rev_number+'0'
=                       ;       db      '.'
=                       ;       db      mod_number+'0'
=                               db      ' (1.1)',CR,LF
=                               DB      '(c) Copyright 1981 Digital Research Inc.',CR,LF
=                               DB      '(c) Copyright 1982 Digital Equipment Corporation',cr,lf
=                               db      cr,lf,0
=
=                       ;|                                           |
=                       ;---------------------------------------------
=                               ENDIF   ;not loader_bios
=
=1E2F                   loc_stk rw  24  ;local stack for initialization
=  1E5F                 stkbase equ offset $
=
=  1E5F                 lastoff equ offset $
=  0226                 tpa_seg equ (lastoff+0400h+15) / 16
```

```
=  0DD1                 tpa_len equ 0FF7h - tpa_seg
=1E5F 00                        db 0    ;fill last address for GENCMD
=
=                       ;*****************************************
=                       ;*                                       *
=                       ;*          Dummy Data Section           *
=                       ;*                                       *
=                       ;*****************************************
=  0000                         dseg    0       ;absolute low memory
=                               org     0       ;(interrupt vectors)
=0000                   int0_offset     rw      1
=0002                   int0_segment    rw      1
=                       ;       pad to system call vector
=0004                           rw      2*(bdos_int-1)
=
=0380                   bdos_offset     rw      1
=0382                   bdos_segment    rw      1
=                               org 36*4                        ;type 36 service
=0090                   sio_offset      rw      1
=0092                   sio_seg         rw      1
=0094                   sio2_offset     rw      1
=0096                   sio2_seg        rw      1
=                               org 39*4                        ;type 39 service
=009C                   Z80_OFFSET      RW      1
=009E                   Z80_SEG         RW      1
=                               org 44*4
=00B0                   tp44_offset     rw      1
```

```
=00B2                   tp44_seg        rw      1
CP/M ASM86 1.1  SOURCE: CPLDBIOS.A86  Customized Loader Basic I/O Sy


=                               org 100*4
=0190                   tp100_offset    rw      1
=0192                   tp100_seg       rw      1
=
=                       ;*****************************************************************************
=                       ;
=                       ;       DUMMY CODE SECTION (FOR FAR CALL TO Z80CCP SERVICES)
=                       ;
=                       ;*****************************************************************************       END
=                       ;
=  0040                         CSEG    40h     ; ABSOLUTE LOCATION
=                               ORG     6h
=  0006                 CCPSERV EQU     $
=                       ;



END OF ASSEMBLY.  NUMBER OF ERRORS:   0.  USE FACTOR: 39%
```

```
CP/M MACRO ASSEM 2.0     #001     CODE FOR Z80 BASE PAGE

                         TITLE    'CODE FOR Z80 BASE PAGE'

                ; WRITTEN FOR DEC RAINBOW 100
                ;          BY CPL
                ;          JULY 1982
                ;       10/15/82 - ADDED UDELAY SUBROUTINE

                         MACLIB   Z80                    ; Z80 MNEMONICS

 0100                    ORG      100H

 0100 0000000000         DW       0,0,0,0,0,0,0,0        ; BYTES 00 THRU 0F

 0110 0000000000         DW       0,0,0,0                ; BYTES 10 THRU 1F
 0118 C30000             JMP      0                      ;   BYTES 18 THRU 1A = RST3
 011B 0000000000         DB       0,0,0,0,0

                                                         ; BYTES 20 THRU 2F
 0120 76                 HLT                             ;   BYTES 20 THRU 22 = RST4
                         JR       $-2
 0121+18FD               DB       18H,$-2-$-1
                                                         ;BYTES 23 THROUGH 25 CONTAIN
 0123 FB                 EI                                 SUBROUTINE FOR WAITING
 0124 76                 HLT
```

```
0125 C9                 RET

0126 0000000000         DB      0,0,0,0,0,0,0,0,0,0

                                                ; BYTES 30 THRU 3F
0130 C30000             JMP     0               ;   BYTES 30 THRU 32 = RST6
0133 0000000000         DB      0,0,0,0,0,0,0,0,0,0,0,0,0

                ; ENTRY CONDITIONS FOR DISK I/O LOOP
                ;       PORT NUMBER IS IN C
                ;       BUFFER ADDRESS IS IN HL

                LOOP:                           ; DISK I/O LOOP IS ORG'D AT 40H
0140 DB40               IN      40H             ; GET GENERAL FLOPPY STATUS
0142 87                 ADD     A               ; SHIFT INTERRUPT BIT TO SIGN
0143 F8                 RM                      ; EXIT IF DONE
                        JRNC    LOOP            ; LOOP IF NO DRQ
0144+30FA               DB      30H,LOOP-$-1
                        INI                     ; READ A BYTE
0146+EDA2               DB      0EDH,0A2H
                        JR      LOOP            ; TEST AGAIN
0148+18F6               DB      18H,LOOP-$-1
                ;
                ; UDELAY  -  SOFTWARE DELAY SUBROUTINE
                ;
                ; RETURNS AFTER NUMBER OF 0.500 MILLISECONDS SPECIFIED IN REG. C
                ;
```

```
                ; ENTRY: C = NUMBER OF 0.500 MILLISECONDS TO DELAY
                ;
                ; EXIT: REG C = 0;  OTHERS ARE PRESERVED
                ;
                UDELAY:
CP/M MACRO ASSEM 2.0    #002    CODE FOR Z80 BASE PAGE

 014A C5                PUSH    B       ; SAVE BC
 014B 0697              MVI     B,151   ; SET FOR 151 COUNTS
                UDEL1:
                        DJNZ    UDEL1   ; DELAY
 014D+10FE              DB      10H,UDEL1-$-1
 014F C1                POP     B       ; RESTORE BC
 0150 0D                DCR     C       ; DECREMENT 0.500 MS. COUNT
 0151 C8                RZ              ; RETURN IF DONE
                        JR      UDELAY  ; DELAY AGAIN
 0152+18F6              DB      18H,UDELAY-$-1
                ;

 0154 0000000000        DW      0,0,0,0,0,0             ; BYTES 54 THRU 5F
 0160 0000000000        DW      0,0,0,0,0,0,0,0         ; BYTES 60 THRU 6F
 0170 0000000000        DW      0,0,0,0,0,0,0,0         ; BYTES 70 THRU 7F
 0180 0000000000        DW      0,0,0,0,0,0,0,0         ; BYTES 80 THRU 8F
 0190 0000000000        DW      0,0,0,0,0,0,0,0         ; BYTES 90 THRU 9F
 01A0 0000000000        DW      0,0,0,0,0,0,0,0         ; BYTES A0 THRU AF
 01B0 0000000000        DW      0,0,0,0,0,0,0,0         ; BYTES B0 THRU BF
 01C0 0000000000        DW      0,0,0,0,0,0,0,0         ; BYTES C0 THRU CF
 01D0 0000000000        DW      0,0,0,0,0,0,0,0         ; BYTES D0 THRU DF
```

```
 01E0 0000000000          DW      0,0,0,0,0,0,0,0          ; BYTES E0 THRU EF
 01F0 0000000000          DW      0,0,0,0,0,0,0,0          ; BYTES F0 THRU FF


 0200                     END
CP/M MACRO ASSEM 2.0      #003    CODE FOR Z80 BASE PAGE

0000 BC          0002 DE          0004 HL          0004 IX          0004 IY
0140 LOOP        014D UDELl       014A UDELAY
```

```
CP/M MACRO ASSEM 2.0    #001    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION

                TITLE 'Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION'

 0000 =         FALSE   EQU     0
 FFFF =         TRUE    EQU     NOT FALSE

 FFFF =         PRIVATE EQU     TRUE                    ; ASSEMBLE PRIVATE VERSION



 0000 =         SHARE   EQU     NOT PRIVATE

                MACLIB  Z80
                PAGE
CP/M MACRO ASSEM 2.0    #002    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION


                IF      SHARE
               ;****************************************
               ;*                                      *
               ;*      PSEUDO-BIOS FOR Z80             *
               ;*      CP/M 86/80                      *
               ;*                                      *
               ;*      04/19/82        RDK             *
               ;*      10/19/82 1100   LZ              *
               ;*      11/22/82 1200   LZ              *
               ;*                                      *
               ;****************************************


               ORG     0F600H                  ;CHANGE THIS IF NECESSARY TO RE-ORG

               IOBYTE  EQU     3
               FDOSM   EQU     90H
               CSFLAG  EQU     0FFE7H
               BIOCS   EQU     1
               BDCS    EQU     2

               ; ** JUMP TABLE FOR STARTING:
               PSBDOS:
                       JMP PSBDOS1
                       JMP START               ;INITIALIZE RST 3 AND 6
               ; ** PSEUDO BDOS:

               PSBDOS1:
                       MOV A,C                 ;** VER 1.0.3
                       CPI 41                  ;** VER 1.0.4
                       RNC                     ;** VER 1.0.4 BAD BDOS CALL
```

```
                        CPI 11                  ;** VER 1.0.3
                        JRZ PCONS               ;** VER 1.0.3 CONSOLE STATUS
                        JRNC PSBD1              ;** VER 1.0.4
                        CPI 6                   ;** VER 1.0.3
                        JRZ PDRC                ;** VER 1.0.3 DIRECT I/O
                        CPI 7                   ;** VER 1.0.3
                        JZ PGIOB                ;** VER 1.0.3 GET IOBYTE
                        CPI 8                   ;** VER 1.0.3
                        JZ PSIOB                ;** VER 1.0.3 STORE IOBYTE
                PSBD1:
                        CALL PACKIT
                        LXI H,GTABLE            ;** VER 1.0.4 START CHECK TO SEE
                        MVI B,0                 ;** VER 1.0.4 IF WE CAN RETURN BEFORE
                        DAD B                   ;** VER 1.0.4 THE 8088 IS FINISHED
                        MOV A,M                 ;** VER 1.0.4 GET CODE
                        STA GONOW               ;** VER 1.0.4 STORE IT
                        MVI A,FDOSM             ;BDOS FUNCTION CODE
                PSEUX:  CALL PSEUD
                        CALL UNPSTAK            ;
                        RET

                PCONS:  LDA CSFLAG              ;** VER 1.0.3
                        ANI BDCS                ;** VER 1.0.3
                        JRNZ PCONS1             ;** VER 1.0.4
CP/M MACRO ASSEM 2.0    #003    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION

                        CALL IOBCHK             ;** VER 1.0.4
                        JRNZ PSBD1              ;** VER 1.0.4
                        LDA CSFLAG              ;** VER 1.0.4
                        ANI BIOCS               ;** VER 1.0.4
                PCONS1: RZ                      ;** VER 1.0.3
                        MVI A,0FFH              ;** VER 1.0.3
                        RET                     ;** VER 1.0.3

                PDRC:   MOV A,E                 ;** VER 1.0.3
                        CPI 0FFH                ;** VER 1.0.3
                        JRZ PDRC1               ;** VER 1.0.4
                        MOV C,E                 ;** VER 1.0.4
                        JMP CONOUT              ;** VER 1.0.4
                PDRC1:  CALL CONST              ;** VER 1.0.3
                        JRNZ PSBD1              ;** VER 1.0.3
                        RET                     ;** VER 1.0.3

                PSEUD:  STA PACKET
                        LDA GONOW               ;** VER 1.0.4
                        STA PACKET+1            ;** VER 1.0.4
                        LXI H,PACKET
                        CALL    I88SVC          ; REQUEST 8080 PROCESSING ( 1.0.3)
                        LDA GONOW               ;** VER 1.0.4
                        ORA A                   ;** VER 1.0.4
                        RZ                      ;** VER 1.0.4
                        CALL WAIT88             ;** VER 1.0.4
                PSEUD1: CALL UNPACKIT
                        RET
```

```
;** DATA AREA
PACKET  DB 0,0,0,0,0,0,0,0,0,0

;*
;*      THE MESSAGE PACKET WILL LOOK LIKE THIS:
;*      BYTE    USED            Z80     8088    USED
;*              FOR             REG     REG     AT
;*
;*      ---------------------------------------------------
;*      0       FUNCTION        --      AH      ENTRY
;*      1       RET VAL         A       AL      RETURN (GONOW ON ENTRY)
;*      2       FUNCTION NO.    C       CL      ENTRY
;*      3                       B       CH      ENTRY
;*      4                       E       DL      ENTRY
;*      5                       D       DH      ENTRY
;*      6                       L       BL      RETURN
;*      7                       H       BH      RETURN
;*      8       IOBYTE          --      --      BOTH
;*


; ** SUBROUTINE TO PUT REGISTERS IN PACKET FOR BIOS OR BDOS CALLS

PACKIT:
REPSTAK:                ;ROUTINE TO SHIFT TO NEW STACK
CP/M MACRO ASSEM 2.0    #004    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION

        SHLD HSAVE      ;HANG ON TO HL
        DI              ;NO INTERRUPTS, PLEASE
        POP H           ;GET RETURN
        SSPD PSTAKSAV   ;STASH THE STACK POINTER
        LXI SP,PSTACK   ;SET STACK TO NEW AREA
        PUSH H          ;SET UP RETURN
        EI              ;INTERUPTS OK
        LHLD HSAVE      ;RESTORE HL
        CALL WAIT88             ;** VER 1.0.4
        SBCD PACKET+2           :STORE REGISTERS
        SDED PACKET+4
        LDA IOBYTE
        STA PACKET+8
        RET

; ** SUBROUTINE TO PUT MESSAGE-PACKET DATA INTO REGISTERS

UNPACKIT:               ;
        PUSHIX          ;STORE INDEX REG
        LIXD IPKT       ;** VER 1.0.4
        LDX A,8         ;IOBYTE
        STA IOBYTE      ;(LOC 3)
        LDX A,1
        LDX L,6
        LDX H,7
        POPIX
```

```
                RET

GTABLE: ;TABLE OF CRITERIA FOR WAITING AFTER A BDOS CALL. ZERO = DON'T WAIT
                DB      1               ;0
                DB      1               ;1
                DB      0               ;2
                DB      1               ;3
                DB      0               ;4
                DB      0               ;5
                DB      1               ;6
                DB      1               ;7
                DB      1               ;8
                DB      1               ;9
                DB      1               ;10
                DB      1               ;11
                DB      1               ;12
                DB      1               ;13
                DB      1               ;14
                DB      1               ;15
                DB      1               ;16
                DB      1               ;17
                DB      1               ;18
                DB      1               ;19
                DB      1               ;20
                DB      1               ;21
                DB      1               ;22
                DB      1               ;23
                DB      1               ;24
                DB      1               ;25
                DB      0               ;26
CP/M MACRO ASSEM 2.0    #005    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION

                DB      1               ;27
                DB      1               ;28
                DB      1               ;29
                DB      1               ;30
                DB      1               ;1
                DB      0               ;32
                DB      1               ;33
                DB      1               ;34
                DB      1               ;35
                DB      1               ;36
                DB      1               ;37
                DB      1               ;38
                DB      1               ;39
                DB      1               ;40

                PAGE
                ORG     PSBDOS+100H     ; PAGE BOUNDARY FOR BIOS JUMPS
;
; ** BIOS JUMP TABLE                    ;FUNCT NO.

                JMP BOOT                ;0
                JMP WBOOT               ;1
                JMP CONST               ;2
```

```
                JMP CONIN               ;3
                JMP CONOUT              ;4
                JMP LIST                ;5
                JMP PUNCH               ;6
                JMP READER              ;7
                JMP HOME                ;8
                JMP SELDSK              ;9
                JMP SETTRK              ;A
                JMP SETSEC              ;B
                JMP SETDMA              ;C
                JMP READ                ;D
                JMP WRITE               ;E
                JMP LISTST              ;F
                JMP SECTRAN             ;10
                JMP VIDEO               ;11


        BOOT:   CALL PACKIT
                MVI A,40H
        XBOOT:  STA PACKET              ;0
                LXI H,PACKET            ;STORE CODE IN PACKET
        ;       CALL WAIT88             ;** VER 1.0.4
                CALL    I88SVC          ; REQUEST 8088 PROCESSING (1.0.3)
                RST 4                   ;BYE.

        WBOOT:  CALL PACKIT             ;1
                MVI A,41H
                JMP XBOOT               ;DO IT TO IT

        CONST:                          ;2
                CALL IOBCHK             ;** VER 1.0.3
                JRNZ CONST1             ;** VER 1.0.3
                LDA CSFLAG              ;** VER 1.0.3
CP/M MACRO ASSEM 2.0    #006    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION

                ANI BIOCS               ;** VER 1.0.3
                JMP PCONS1              ;** VER 1.0.3
        CONST1:
                CALL PACKIT
                CALL PSEUN              ;** VER 1.0.4
                DB      1,42H

        CONIN:  CALL PACKIT             ;3
                CALL PSEUN              ;** VER 1.0.4
                DB      1,43H

        CONOUT: CALL PACKIT             ;4
                CALL PSEUN              ;** VER 1.0.4
                DB      0,44H

        LIST:   CALL PACKIT             ;5
                CALL PSEUN              ;** VER 1.0.4
                DB      0,45H

        PUNCH:  CALL PACKIT             ;6
```

```
                CALL PSEUN              ;** VER 1.0.4
                DB      0,46H

READER:         CALL PACKIT             ;7
                CALL PSEUN              ;** VER 1.0.4
                DB      1,47H

HOME:           CALL PACKIT             ;8
                CALL PSEUN              ;** VER 1.0.4
                DB      0,48H

SELDSK:         CALL PACKIT             ;9
                CALL PSEUN              ;** VER 1.0.4
                DB      1,49H

SETTRK:         MVI B,0                 ;10
                CALL PACKIT
                CALL PSEUN              ;** VER 1.0.4
                DB      0,4AH

SETSEC:         CALL PACKIT             ;11
                CALL PSEUN              ;** VER 1.0.4
                DB      0,4BH

SETDMA:         CALL PACKIT             ;12
                CALL PSEUN              ;** VER 1.0.4
                DB      0,4CH

READ:           CALL PACKIT             ;13
                CALL PSEUN              ;** VER 1.0.4
                DB      1,4DH

WRITE:          CALL PACKIT             ;14
                CALL PSEUN              ;** VER 1.0.4
                DB      1,4EH
```



```
LISTST:         CALL PACKIT             ;15
                CALL PSEUN              ;** VER 1.0.4
                DB      1,4FH

SECTRAN:                                ;16
                MVI B,0                 ;DOUBLE PRECISION IN BC
                XCHG                    ;TRANSLATE TABLE ADDRESS TO HL
                DAD B                   ;TRANSLATE SECTOR ADDRESS
                MOV L,M                 ;RETURN SECTOR IN L
                MVI H,0                 ;
                RET                     ;DON'T BOTHER CP/M-86

VIDEO:                                  ;17
                MVI D,0                 ;FAST VIDEO. NEED 0 SEG
                MVI E,0                 ;** VER 1.0.4
                CALL PACKIT
                CALL PSEUN
```

```
                DB      1,56H

IOBCHK:         LDA IOBYTE              ;** VER 1.0.3
                ANI 3                   ;** VER 1.0.3 CRT?
                CPI 1                   ;** VER 1.0.3 REALLY?
                RET                     ;** VER 1.0.3

PGIOB:          LDA IOBYTE              ;** VER 1.0.3 GET IOBYTE
                RET                     ;** VER 1.0.3

PSIOB:          MOV A,E                 ;** VER 1.0.3 STORE IOBYTE
                STA IOBYTE              ;** VER 1.0.3
                RET                     ;** VER 1.0.3

PSEUN:          POP H                   ;ROUTINE TO CARRY PARAMETERS FORWARD INTO
                MOV A,M                 ;THE PACKET FOR BIOS CALLS
                STA GONOW               ;STORE WAIT CRITERION
                INX H                   ;
                MOV A,M                 ;GET FUNCTION NUMBER
                JMP PSEUX               ;GO TO PSEUDO-BIOS EXIT

UNPSTAK:                        ;ROUTINE TO RESET TO OLD STACK
                SHLD HSAVE      ;HANG ON TO HL
                DI              ;NO INTERRUPTS, PLEASE
                POP H           ;GET RETURN ADDRESS
                LSPD PSTAKSAV   ;RESTORE USER STACK
                PUSH H          ;RESTORE RETURN
                EI              ;INTERRUPTS OK
                LHLD HSAVE      ;RESTORE HL
                RET             ;

HSAVE:          DW      0
PSTAKSAV:       DW      0
GONOW:          DB      0


                ;SPACE FOR A STACK
                DS      50
PSTACK:         DW      0
CP/M MACRO ASSEM 2.0    #008    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION


                ENDIF
                PAGE
CP/M MACRO ASSEM 2.0    #009    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION


                ; Z80 INTERFACE LAYER AND PRIMITIVE ROUTINES

                ; REWRITTEN FOR DEC RAINBOW 100
                ;           BY CPL
                ;           JULY 1982
```

```
                        IF      PRIVATE
0100 =          START   EQU     00100H                  ; STARTING ADDRESS OF CODE
3CFC =          INTFPTR EQU     03CFCH                  ; LOCATION OF PACKET POINTERS
                        ELSE
                START   EQU     $                       ; STARTING ADDRESS OF CODE
                INTFPTR EQU     0FFFCH                  ; LOCATION OF PACKET POINTERS
                        ENDIF

0018 =          RST3    EQU     18H                     ; ADDRESS OF RST 18H (RST 3) VECTOR
0020 =          RST4    EQU     20H                     ; ADDRESS OF RST 20H (RST 4) VECTOR
0030 =          RST6    EQU     30H                     ; ADDRESS OF RST 30H (RST 6) VECTOR

0000 =          I88INT  EQU     0                       ; 8088 INTERRUPT PORT
0020 =          INTSTA  EQU     20H                     ; CROSS-CPU INTERRUPT STATUS PORT

00C3 =          JUMP    EQU     0C3H                    ; OPCODE FOR JUMP INSTRUCTION

0004 =          INTBIT  EQU     4H                      ; INTERRUPT BIT
                                                        ;   WHEN LOW, SHOWS 8088 INTERRUPT
                                                        ;   FLOP IS INTERRUPTING THE 8088;
                                                        ;   ORIGINALLY SET BY Z80

00F0 =          FRANGE  EQU     0F0H                    ; FUNCTION RANGE MASK
                                                        ;   FOR HIGH NIBBLE OF FUNCTION CODE

0010 =          DSKFNC  EQU     10H                     ; DISK FUNCTION (HIGH NIBBLE VALUE)
0020 =          OTHFNC  EQU     20H                     ; OTHER Z80 FUNCTIONS (HIGH NIBBLE VALUE)
0040 =          USRFNC  EQU     40H                     ; USER-DEFINED FUNCTIONS

0021 =          Z80BGN  EQU     21H                     ; ALLOW Z80 TPA EXECUTION
0022 =          Z80MVE  EQU     22H                     ; MOVE Z80 MEMORY CONTENTS
                                                        ;   LOW VALUE LIMIT - 1

0007 =          LEGFUN  EQU     07H                     ; FUNCTION CODE MASK
                                                        ;   USE LOW 3 BITS

00FF =          FNCNG   EQU     0FFH                    ; FUNCTION CODE NO GOOD

0014 =          DKWRIT  EQU     14H                     ; PACKET DISK WRITE FUNCTION CODE

000A =          NUMSEC  EQU     10                      ; NUMBER OF SECTORS PER TRACK
0200 =          SECSIZ  EQU     512                     ; NUMBER OF BYTES PER DISK SECTOR

0002 =          STADRL  EQU     2                       ; START ADDRESS LSB PACKET OFFSET
0003 =          STADRH  EQU     3                       ; START ADDRESS MSB PACKET OFFSET

0002 =          SCADRL  EQU     2                       ; SOURCE ADDRESS LSB PACKET OFFSET
CP/M MACRO ASSEM 2.0    #010    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION

0003 =          SCADRH  EQU     3                       ; SOURCE ADDRESS MSB PACKET OFFSET
0004 =          DSADRL  EQU     4                       ; DESTINATION ADDRESS LSB PACKET OFFSET
0005 =          DSADRH  EQU     5                       ; DESTINATION ADDRESS MSB PACKET OFFSET
0006 =          BYCNTL  EQU     6                       ; BYTE COUNT ADDRESS LSB PACKET OFFSET
0007 =          BYCNTH  EQU     7                       ; BYTE COUNT ADDRESS MSB PACKET OFFSET
```

```
                        PAGE
CP/M MACRO ASSEM 2.0    #011    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION


 0100                   ORG     START

                ; ************************************************************
                ;
                ; NAME: INTSTR
                ;
                ; FUNCTION: THIS ROUTINE WILL RE-INITIALIZE THE RST 6
                ;           VECTORS, SET UP THE Z80 SERVICE STACK POINTER,
                ;           AND DO A RST 4.
                ;
                ; ENTRY: NONE
                ;
                ; EXIT: NONE
                ;
                INTSTR:

                ; SET UP RST VECTORS

 0100 211201            LXI     H,PKTPRO                ; SET UP RST 6 VECTOR
 0103 223100            SHLD    RST6+1

                ; SET UP Z80 SERVICE STACK POINTER

 0106 313005            LXI     SP,STACK

                ; INITIALIZE FLAGS

 0109 3E00              MVI     A,FALSE
 010B 323105            STA     DONEFL

                ; WAIT UNTIL NEEDED

 010E FB                EI
 010F C32000            JMP     RST4                    ; GO TO HALT AND WAIT FOR INTERRUPT
                        PAGE
CP/M MACRO ASSEM 2.0    #012    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION


                ; ************************************************************
                ;
                ; NAME: PKTPRO
                ;
                ; FUNCTION: THIS ROUTINE IS PASSED A PACKET ADDRESS IN I88PKT.  IT USES
                ;           THAT TO GET THE FUNCTION NUMBER WHICH IT USES TO JUMP
                ;           TO THE SPECIFIC FUNCTION HANDLER ROUTINE.
                ;
                ;           THIS ROUTINE IS ENTERED VIA AN INTERRUPT FROM THE 8088 AT RST 6 -
                ;           THEREFORE, INTERRUPTS ARE OFF.
                ;
                ;           FOR ALL FUNCTIONS EXCEPT Z80START AND SERIAL SUPPORT FROM 8088,
                ;           WE WILL INTERRUPT THE 8088 TO INDICATE WE HAVE FINISHED
```

```
                ;           THE FUNCTION
                ;
                ; ENTRY: I88PKT = PACKET ADDRESS
                ;
                ; EXIT: IX = PACKET ADDRESS
                ;
                PKTPRO:
                        PUSHIX                          ; SAVE
 0112+DDE5              DB      0DDH,0E5H
 0114 F5                PUSH    PSW

                        LIXD    I88PKT                  ; GET ADDRESS OF PACKET
 0115+DD2A              DB      0DDH,2AH
 0117+FE3C              DW      I88PKT

 0119 DB00              IN      I88INT                  ; CLEAR 8088 INTERRUPT FLAG

                        PUSHIY                          ; SAVE REMAINING REGISTERS
 011B+FDE5              DB      0FDH,0E5H
 011D E5                PUSH    H
                        PUSHIX                          ; CHECK FOR ZERO ADDRESS
 011E+DDE5              DB      0DDH,0E5H
 0120 E1                POP     H                       ; MOVE IX TO H
 0121 7C                MOV     A,H                     ; PUT IN  ACCUM
 0122 B5                ORA     L                       ; GET LOW-ORDER, TOO.
 0123 CA7901            JZ      HIFXIT                  ; GO AWAY IF ZERO
 0126 D5                PUSH    D
 0127 C5                PUSH    B

 0128 015201            LXI     B,PKTRET                ; PUSH RETURN ADDRESS ONTO STACK
 012B C5                PUSH    B

 012C FB                EI                              ; RE-ENABLE INTERRUPTS

                        LDX     A,FNCCOD        ; GET FUNCTION CODE
 012D+DD7E00            DB      0DDH,A*8+46H,FNCCOD
                ;
 0130 FE13              CPI     QKRDCOM
 0132 CACF01            JZ      DKREAD          ; READ FUNCTION
                ;
 0135 FE14              CPI     QKWTCOM
 0137 CACF01            JZ      DKWRITE         ; WRITE FUNCTION
CP/M MACRO ASSEM 2.0    #013    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION

                ;
 013A FE42              CPI     USRFNC+2        ; BIOS 42 - 4F, BDOS 90
                        JRNC    HIFNC           ; YES
 013C+3033              DB      30H,HIFNC-$-1
                ;
 013E FE22              CPI     Z80MVE          ; MOVE Z80 MEMORY
 0140 CADB04            JZ      ZMOVE
                ;
 0143 FE15              CPI     QKCMCOM
 0145 CA8601            JZ      DKCHECK         ; CHECK MEDIA FUNCTION
                ;
```

```
0148 FE21               CPI     Z80BGN          ; ALLOW Z80 TPA EXECUTION
014A CAC604             JZ      ZSTART
                ;
                ERROR:
014D C1                 POP     B               ; EMPTY STACK OF RETURN ADR
                        MVIX    FNCNG,STATUS    ; SET ERROR STATUS
014E+DD3601FF           DB      0DDH,36H,STATUS,FNCNG

                PKTRET:
0152 FB                 EI                              ; RE-ENABLE INTERRUPTS IN CASE
                                                        ;   THEY WERE OFF (DISK I/O)
0153 C1                 POP     B                       ; RESTORE REGISTERS
0154 D1                 POP     D
0155 E1                 POP     H
                        POPIY
0156+FDE1               DB      0FDH,0E1H

                        SIXD    Z80PKT                  ; SET UP RETURN PACKET POINTER
0158+DD22               DB      0DDH,22H
015A+FC3C               DW      Z80PKT
015C D300               OUT     I88INT                  ;   AND PASS IT

                PKTR1:
015E DB20               IN      INTSTA                  ; LOOP UNTIL 8088 CLEARS THE
0160 E604               ANI     INTBIT                  ;   INTERRUPT
0162 CA5E01             JZ      PKTR1

                        LXIX    0
0165+DD21               DB      0DDH,21H
0167+0000               DW      0
                        SIXD    Z80PKT                  ;ZERO ADDRESS (8088 HAS IT NOW)
0169+DD22               DB      0DDH,22H
016B+FC3C               DW      Z80PKT

016D F1                 POP     PSW
                        POPIX
016E+DDE1               DB      0DDH,0E1H

0170 C9                 RET                             ; AND RETURN

                ; PROCESS FUNCTION CODES  42H TO 4FH, OR 90H

                HIFNC:
0171 C1                 POP     B                       ; FIX STACK
0172 3EFF               MVI     A,TRUE                  ; SET DONE FLAG
CP/M MACRO ASSEM 2.0    #014    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION

0174 323105             STA     DONEFL

0177 C1                 POP     B                       ; RESTORE REGISTERS
0178 D1                 POP     D                       ;   EXCEPT HL,AF, AND IX
0179 E1         HIFXIT: POP     H
                        POPIY
017A+FDE1               DB      0FDH,0E1H
```

```
 017C F1                POP     PSW                     ; RESTORE
                        SIXD    IPKT                    ; STASH PACKET ADDRESS
 017D+DD22              DB      0DDH,22H
 017F+8401              DW      IPKT
                        POPIX
 0181+DDE1              DB      0DDH,0E1H
 0183 C9                RET
 0184 0000      IPKT    DW      0

                        PAGE
CP/M MACRO ASSEM 2.0    #015    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION


                        IF      SHARE
                ; *********************************************************
                ;
                ; NAME: I88SVC
                ;
                ; FUNCTION: THIS ROUTINE REQUESTS SERVICE FROM THE 8088 FOR A
                ;           Z80-RESIDENT ROUTINE.
                ;
                ; ENTRY: HL = PACKET ADDRESS
                ;
                ; EXIT: HL = PACKET ADDRESS
                ;
                I88SVC:

                        DI
                        SHLD    Z80PKT                  ; WRITE ADDRESS OF PACKET

                        MVI     A,FALSE                 ; CLEAR DONE FLAG
                        STA     DONEFL

                        OUT     I88INT                  ; SET 8088 INTERRUPT FLAG

                INTCLR:
                        IN      INTSTA                  ; LOOP UNTIL 8088 TURNS OFF
                        ANI     INTBIT                  ;   INTERRUPT FLAG
                        JZ      INTCLR
                        PUSH    H
                        LXI     H,0
                        SHLD    Z80PKT                  ; ZERO THE PACKET ADDRESS
                        POP     H
                ;
                        EI                              ; RE-ENABLE INTERRUPTS
                        RET

                ;
                ; ROUTINE TO WAIT FOR THE DONE FLAG
                ;
                WAIT88:
                        DI
                        LDA     DONEFL                  ; JUMP IF DONE
                        CPI     TRUE
                        JRZ     WAITDN
```

```
                        CALL    23H                     ; WAIT UNTIL INTERRUPTED

                        JR      WAIT88                  ; JUMP TO MAKE SURE DONE FLAG IS TRUE

                WAITDN: EI
                        RET

                        PAGE

                        ENDIF
                ; ***************************************************************
                ;
                ; NAME: DSKPRO
CP/M MACRO ASSEM 2.0    #016    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION

                ;
                ; FUNCTION: THIS ROUTINE IS THE ENTRY POINT FOR ALL OF THE
                ;           DISK I/O FUNCTIONS.  IT WILL DETERMINE WHICH
                ;           PRIMITIVE TO CALL, AND PASS CONTROL TO THAT PRIMITIVE.
                ;
                ; ENTRY: IX = PACKET ADDRESS
                ;
                ; EXIT: IX = PACKET ADDRESS
                ;
                ; ***************************************************************
                ;
                ; PACKET OFFSETS
                ;
 0000 =         FNCCOD  EQU     0       ; FUNCTION CODE
 0001 =         STATUS  EQU     1       ; RETURNED STATUS
 0002 =         DRIVEN  EQU     2       ; DRIVE NUMBER
 0002 =         SECTN   EQU     2       ; SECTOR NUMBER
 0003 =         TRACKN  EQU     3       ; TRACK NUMBER
 0004 =         DMALOW  EQU     4       ; DMA ADDRESS (LSB)
 0005 =         DMAHI   EQU     5       ; DMA ADDRESS (MSB)
 0006 =         NSECT   EQU     6       ; NUMBER OF SECTORS
                ;
 0060 =         DRVNUM  EQU     60H     ; MASK FOR DRIVE NUMBER
 001F =         SECNUM  EQU     1FH     ; MASK FOR SECTOR NUMBER
                ;
                ;  FUNCTION CODES
 0013 =         QKRDCOM EQU     13H     ; READ FUNCTION CODE
 0014 =         QKWTCOM EQU     14H     ; WRITE FUNCTION CODE
 0015 =         QKCMCOM EQU     15H     ; CHECK MEDIA FUNCTION CODE
                ;
                ; FDC COMMAND FLAGS
                ;
 0008 =         QMHLD   EQU     8       ; HEAD LOAD FLAG
 0004 =         QMVERF  EQU     4       ; VERIFY FLAG
 0004 =         QMEFLG  EQU     4       ; HEAD LOAD FLAG FOR READ/WRITE
 0008 =         QMSSEL  EQU     8       ; SIDE SELECT FLAG
 0002 =         QMSCOM  EQU     2       ; SIDE COMPARE FLAG
 0010 =         QMUPDT  EQU     10H     ; UPDATE TRACK REG FLAG
                ;
                ; FDC STEP RATE
```

```
                        ;
0000 =                  QKRATE  EQU     0       ; 6 MS RATE
                        ;
                        ; FDC COMMANDS
                        ;
0008 =                  QCREST  EQU     0+QMHLD+QKRATE          ; RESTORE (RECAL)
001C =                  QCSEEK  EQU     10H+QMHLD+QKRATE+QMVERF ; SEEK (WITH VERIFY)
0018 =                  QCSEEKN EQU     10H+QMHLD+QKRATE        ; SEEK (NO VERIFY)
0010 =                  QCSEKH0 EQU     10H+QKRATE              ; SEEK - NO HEAD LOAD
0048 =                  QCSTEPIN EQU    40H+QMHLD+QKRATE        ; STEP IN
0068 =                  QCSTEPOT EQU    60H+QMHLD+QKRATE        ; STEP OUT
0080 =                  QCREADS EQU     80H                     ; READ SECTOR
00A0 =                  QCWRTS  EQU     0A0H                    ; WRITE SECTOR
00F0 =                  QCWRTRK EQU     0F0H                    ; WRITE TRACK
00C0 =                  QCRDADR EQU     0C0H                    ; READ ADDRESS
00D0 =                  QCTERM  EQU     0D0H                    ; TERMINATE COMMAND
CP/M MACRO ASSEM 2.0    #017    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION

                        ;
                        ; DISK CONTROLLER STATUS - TYPE 1 COMMANDS
                        ;
0080 =                  QMNRDY  EQU     80H     ; NOT READY
0040 =                  QMWPROT EQU     40H     ; WRITE PROTECTED
0020 =                  QMHLT   EQU     20H     ; HEAD LOADED
0010 =                  QMSKERR EQU     10H     ; SEEK ERROR
0008 =                  QMCRC   EQU     8       ; CRC ERROR
0004 =                  QMTZERO EQU     4       ; TRACK ZERO
0002 =                  QMINDEX EQU     2       ; INDEX
0001 =                  QMBUSY  EQU     1       ; BUSY
                        ;
                        ; DISK CONTROLLER STATUS - TYPE 2 AND 3 COMMANDS
                        ;
0020 =                  QMWRFLT EQU     20H     ; WRITE FAULT
0010 =                  QMRNF   EQU     10H     ; RECORD NOT FOUND
0004 =                  QMLDATA EQU     4       ; LOST DATA
0002 =                  QMDRQ   EQU     2       ; DATA REQUEST
0020 =                  QMDELDM EQU     20H     ; DELETED DATA MARK
                        ;
                        ; ERROR MASKS FOR OPERATIONS
                        ;
0091 =                  QMREST  EQU     QMNRDY+QMSKERR+QMBUSY           ; RESTORE
0099 =                  QMSEEK  EQU     QMNRDY+QMSKERR+QMCRC+QMBUSY     ; SEEK
00BD =                  QMREAD  EQU     QMNRDY+QMRNF+QMCRC+QMLDATA+QMDELDM+QMBUSY       ; READ SECTOR
00FD =                  QMWRITE EQU     QMNRDY+QMRNF+QMCRC+QMLDATA+QMWRFLT+QMWPROT+QMBUSY  ; WRITE SECTOR
009D =                  QMRDADR EQU     QMNRDY+QMRNF+QMCRC+QMLDATA+QMBUSY       ; READ ADDRESS
                        ;
                        ; GENERAL STATUS PORT (QPSTAT) EQUATES
                        ;
00C0 =                  QMPRECMP EQU    0C0H    ; PRECOMPENSATION BITS
0020 =                  QMPSIDE EQU     20H     ; SIDE SIGNAL
0010 =                  QMON1   EQU     10H     ; MOTOR 1 ON
0008 =                  QMON0   EQU     8       ; MOTOR 0 ON
0018 =                  QMON    EQU     QMON0+QMON1
0004 =                  QMTG42  EQU     4       ; TG 42 SIGNAL
0003 =                  QMDRNR  EQU     3       ; DRIVE NUMBER
```

```
                        ;
                        ; MISCELLANEOUS EQUATES
                        ;
0004 =                  QKDRETRY EQU    4       ; DISK OPERATION RETRIES
9C40 =                  QKDRCNT EQU     40000   ; COUNT FOR DISK READY TIMING (.5 SEC)
003D =                  QKPCTRK EQU     61      ; PRECOMP REQUIRED STARTING AT THIS TRACK
0040 =                  QKPCBIT EQU     40H     ; PRECOMP BIT TO USE AFTER TRACK 60
004F =                  QKMXTRK EQU     79      ; MAX TRACK NUMBER ON RX50
000A =                  QKMXSECT EQU    10      ; MAX PHYSICAL SECTOR NO. ON RAINBOW DISKS
0002 =                  QKROBIN EQU     2       ; STATUS RETURNED FOR ROBIN MEDIA ON CHECK FUNCTION
                        ;
                        ;
                        ;
                        ;
                        ;
                        ; EQUATES FOR PHYSICAL DISK CONTROL
                        ;
0040 =                  QPSTAT  EQU     40H     ; GENERAL DISK CONTROL/STATUS REG.
0060 =                  QPCOMD  EQU     60H     ; FDC COMMAND/STATUS REG.
CP/M MACRO ASSEM 2.0    #018    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION

0061 =                  QPTRKRG EQU     61H     ; FDC TRACK REG.
0062 =                  QPSECRG EQU     62H     ; FDC SECTOR REG.
0063 =                  QPDATA  EQU     63H     ; FDC DATA REG.
                        ;
                        ;
                        ; EQUATES FOR PRIVATE RAM I/O ROUTINE
                        ;
0040 =                  RDORWR  EQU     0040H   ; ADDRESS OF ROUTINE
0046 =                  INOROUT EQU     0046H   ; ADDRESS OF INI OR OUTI INSTR.
004A =                  UDELAY  EQU     004AH   ; DELAY IN 0.500 MSEC INCREMENTS, C = COUNT
                        ;
A2ED =                  INII    EQU     0A2EDH  ; INI INSTRUCTION
A3ED =                  OUTII   EQU     0A3EDH  ; OUTI INSTRUCTION
63DB =                  ININ    EQU     QPDATA SHL 8 + IN ; IN QPDATA INSTR.
                        ;
                                PAGE
CP/M MACRO ASSEM 2.0    #019    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION


                        ; DKCHECK  -  CHECK MEDIA TYPE
                        ;
                        ; FUNCTION:     ATTEMPTS TO DETERMINE WHETHER A RAINBOW OR ROBIN DISKETTE
                        ;               IS MOUNTED IN THE SPECIFIED DRIVE.
                        ;               INVOKED BY THE BIOS IN THE 8088 WHENEVER DRIVE IS
                        ;               SELECTED FOR THE FIRST TIME AFTER A DISK SYSTEM RESET -
                        ;               MOST COMMONLY AFTER A ^C AT THE CCP LEVEL.
                        ; RULES:        NO ERRORS ARE RETURNED. A RESTORE OPERTION IS DONE,
                        ;               FOLLOWED BY A READ OF SECTOR 10. IF A RECORD NOT FOUND
                        ;               STATUS RESULTS, ROBIN MEDIA IS ASSUMED. ON A SUCCESSFUL
                        ;               READ OF PHYSICAL SECTOR 10 OR IF ANY OTHER ERROR OCCURS
                        ;               (SUCH AS NOT READY), RAINBOW MEDIA IS ASSUMED.
                        ; ENTRY:        PACKET ADDRESS IN IX.
                        ; EXIT:         MEDIA TYPE IN THE STATUS FIELD OF THE PACKET:
                        ;               0 IF RAINBOW, 2 IF ROBIN.
```

```
                        ;
                        ;
                        ;
                        DKCHECK:
 0186 3EFF                      MVI     A,0FFH          ; SET "CURRENT TRACK" TO FF TO INDICATE
 0188 CDB004                    CALL    SETRAK          ;  FIRST ACCESS AFTER RESET
 018B CDC603                    CALL    DPSELDR
                                JRC     DKCHECK1        ; DRIVE IS NOT READY
 018E+3805                      DB      38H,DKCHECK1-$-1
 0190 CD3503                    CALL    DPRECAL
                                JRZ     DKCHECK2        ; FALL THROUGH = ERROR ON RESTORE
 0193+2803                      DB      28H,DKCHECK2-$-1
                        DKCHECK1:
 0195 AF                        XRA     A               ; ERROR - ASSUME RAINBOW MEDIA
                                JR      DKCHECK3
 0196+1823                      DB      18H,DKCHECK3-$-1
                        DKCHECK2:
 0198 AF                        XRA     A
                                STX     A,TRACKN
 0199+DD7703                    DB      0DDH,70H+A,TRACKN
 019C CDB004                    CALL    SETRAK          ; TRACK 0 INTO TABLE
                                LDX     A,DRIVEN        ; SETUP PACKET FOR THE READ ROUTINE
 019F+DD7E02    DB              0DDH,A*8+46H,DRIVEN
 01A2 E660                      ANI     DRVNUM
 01A4 47                        MOV     B,A
 01A5 3E0A                      MVI     A,QKMXSECT      ; NOW TRY TO RAED SECTOR 10
 01A7 B7                        ORA     A
                                STX     A,SECTN
 01A8+DD7702                    DB      0DDH,70H+A,SECTN
 01AB 21DB63                    LXI     H,ININ          ;  NO DATA TRANSFER INTO MEMORY
 01AE CD6502                    CALL    DPREADZ         ;    AND NO ERROR RECOVERY
                                JRZ     DKCHECK3        ; GOOD READ
 01B1+2808                      DB      28H,DKCHECK3-$-1
 01B3 FE10                      CPI     QMRNF           ; IS RECORD NOT FOUND THE ONLY ERROR?
 01B5 3E00                      MVI     A,0
                                JRNZ    DKCHECK3        ;  NO  - SET RAINBOW MEDIA
 01B7+2002                      DB      20H,DKCHECK3-$-1
 01B9 3E02                      MVI     A,QKROBIN       ;  YES - ROBIN MEDIA
                        DKCHECK3:
 01BB 47                        MOV     B,A
CP/M MACRO ASSEM 2.0    #020    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION

 01BC 3AC204                    LDA     BCURDRV         ; PUT MEDIA TYPE INTO FORMAT TABLE,
 01BF 5F                        MOV     E,A             ;   CURRENT FORMAT INDICATOR AND
 01C0 1600                      MVI     D,0             ;     INTO THE PACKET
 01C2 21F43C                    LXI     H,TFORMAT
 01C5 19                        DAD     D
 01C6 70                        MOV     M,B
 01C7 78                        MOV     A,B
 01C8 32C304                    STA     BFORMAT
                                STX     A,STATUS
 01CB+DD7701                    DB      0DDH,70H+A,STATUS
 01CE C9                        RET
                                PAGE
CP/M MACRO ASSEM 2.0    #021    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION
```

```
                ; DKREAD/DKWRITE  -  READ OR WRITE SECTOR(S)
                ;
                ; FUNCTION: READS OR WRITES 1 TO 10 SECTORS ON
                ;  SPECIFIED DRIVE AND TRACK.
                ;
                DKREAD:
                DKWRITE:
01CF CD3C02             CALL    DKRWSET         ; SET UP FOR READ/WRITE
01D2 D8                 RC                      ;  EXIT - DRIVE NOT READY
01D3 CAE001             JZ      DKRW20          ; NO SEEK NEEDED
01D6 CD5B03             CALL    DPSEEK          ; PERFORM SEEK
                        JRNZ    DKRW90          ; SEEK ERROR
01D9+2021               DB      20H,DKRW90-$-1
01DB DB61               IN      QPTRKRG         ; SET NEW TRACK IN TABLE
01DD CDB004             CALL    SETRAK
                DKRW20:
01E0 3E13               MVI     A,QKRDCOM       ; READ FUNCTION?
                        CMPX    FNCCOD
01E2+DDBE00             DB      0DDH,0BEH,FNCCOD
                        JRNZ    DKRW30          ;  NO - WRITE
01E5+2005               DB      20H,DKRW30-$-1
01E7 CD5F02             CALL    DPREAD          ;  YES - READ SECTOR
                        JR      DKRW40
01EA+1803               DB      18H,DKRW40-$-1
                DKRW30:
01EC CD7B02             CALL    DPWRITE         ;  NO - WRITE SECTOR
                DKRW40:
01EF CD0702             CALL    NEXTSEC         ; CHECK FOR MULT. SECTOR OPERATION
                        JRNC    DKRW20          ; LOOP UNTIL ALL SECTORS DONE
01F2+30EC               DB      30H,DKRW20-$-1
                        STX     A,STATUS        ; STORE STATUS
01F4+DD7701             DB      0DDH,70H+A,STATUS
01F7 E601               ANI     QMBUSY          ; SEEK ERROR INDICATED?
01F9 C8                 RZ                      ; NO - ALL DONE
                        JR      DKRW95          ; YES - RESET TRACK TABLE RNTRY
01FA+1805               DB      18H,DKRW95-$-1
                ;
                ; SEEK ERROR
                DKRW90:
01FC F601               ORI     QMBUSY          ; MARK AS SEEK ERROR
                        STX     A,STATUS        ; STORE STATUS
01FE+DD7701             DB      0DDH,70H+A,STATUS
                DKRW95:
0201 3EFF               MVI     A,0FFH          ; FORCE RECAL ON NEXT R/W
0203 CDB004             CALL    SETRAK
0206 C9                 RET
                ;
                ;
                ; SUBROUTINE TO CHECK FOR MULTIPLE SECTOR OPERATION
                ; DECREMENTS SECTOR COUNT AND INCREMENTS SECTOR NUMBER.
                ; SECTOR INTERLEAVE IS USED TO READ/WRITE SECTORS IN
                ; LOGICAL ORDER.
                ;
```

```
                ; ENTRY: STATUS OF READ/WRITE IS IN 'A'
                ;
CP/M MACRO ASSEM 2.0    #022    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION

                ; EXIT: CARRY IS SET IF NO FURTHER READ/WRITE OPERATIONS
                ;  ARE NECESSARY (COUNT EXHAUSTED OR ERROR OCCURRED)
                ;  PACKET COUNT AND SECTOR NUMBER ARE UPDATED.
                ;
                NEXTSEC:
 0207 FE01              CPI     1               ; STATUS OK?
 0209 3F                CMC                     ; (REVERSE SENSE OF CARRY)
 020A D8                RC                      ;  EXIT - ERROR OCCURRED
                        DCRX    NSECT           ; DECREMENT SECTOR COUNT
 020B+DD3506            DB      0DDH,035H,NSECT
 020E 37                STC                     ; SET CARRY IN CASE COUNT = 0
 020F C8                RZ                      ;  EXIT ON ZERO COUNT
                ;
                ; INCREMENT DMA ADDRESS IN PACKET
                ;
                        LDX     A,DMAHI         ; TAKE CARE OF POSSIBLE CARRY
 0210+DD7E05            DB      0DDH,A*8+46H,DMAHI
 0213 C602              ADI     2
                        STX     A,DMAHI
 0215+DD7705            DB      0DDH,70H+A,DMAHI
                        LDX     A,SECTN         ; GET SECTOR NUMBER
 0218+DD7E02            DB      0DDH,A*8+46H,SECTN
 021B C601              ADI     1               ; NEXT SECTOR
 021D 47                MOV     B,A
 021E E61F              ANI     SECNUM          ; ISOLATE IT
 0220 FE0A              CPI     QKMXSECT        ; OVER THE LIMIT ON THE TRACK?
                        JRC     NEXTSC15        ; NO
 0222+3813              DB      38H,NEXTSC15-$-1
                        JRNZ    NEXTSC10        ;   DEFINITELY OVER THE LIMIT.
 0224+2006              DB      20H,NEXTSC10-$-1
 0226 3AC304            LDA     BFORMAT         ;   AT SECTOR 10. OVER THE LIMIT ONLY IF
 0229 A7                ANA     A               ;     ROBIN DISK;
                        JRZ     NEXTSC20
 022A+280C              DB      28H,NEXTSC20-$-1
                NEXTSC10:
                        LDX     A,SECTN
 022C+DD7E02            DB      0DDH,A*8+46H,SECTN
 022F E6E0              ANI     NOT SECNUM
 0231 F601              ORI     1
                        STX     A,SECTN         ; STORE IT IN PACKET
 0233+DD7702            DB      0DDH,70H+A,SECTN
 0236 C9                RET
                ;
 0237 3F        NEXTSC15: CMC
                NEXTSC20:
                        STX     B,SECNUM
 0238+DD701F            DB      0DDH,70H+B,SECNUM
 023B C9                RET
                ;
                        PAGE
CP/M MACRO ASSEM 2.0    #023    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION
```

```
                ; DKRWSET  -  SET UP FOR READ OR WRITE
                ;
                ; FUNCTION: SELECTS DRIVE AND CHECKS READY STATUS.  HOMES
                ;  DRIVE IF IT HAS NOT BEEN ACCESSED PREVIOUSLY. COMPARES
                ;  DESIRED TRACK WITH CURRENT TRACK.
                ;
                ; ENTRY: IX = PACKET ADDRESS
                ;
                ; EXIT: HL = ADDRESS OF TRACK TABLE ENTRY FOR DRIVE
                ;       Z FLAG IS SET IF CURRENT TRACK = DESIRED TRACK
                ;       CARRY FLAG IS SET IF DRIVE IS NOT READY
                ;
                DKRWSET:
 023C CDC603            CALL    DPSELDR         ; SELECT DRIVE AND CHECK READY STATUS
 023F D8                RC                      ;  EXIT - NOT READY
 0240 3AC504            LDA     BTRACK          ; GET CURRENT TRACK
 0243 FEFF              CPI     0FFH            ; FIRST ACCESS?
                        JRNZ    DKRWS10         ;  NO
 0245+2008              DB      20H,DKRWS10-$-1
 0247 CD3503            CALL    DPRECAL         ; HOME DRIVE
 024A C0                RNZ                     ;  EXIT ON HOME ERROR
 024B AF                XRA     A               ; SET TRACK ZERO IN TABLE
 024C CDB004            CALL    SETRAK
                DKRWS10:
 024F 3AC304            LDA     BFORMAT         ; COMPARE DESIRED TRACK WITH
 0252 A7                ANA     A               ;  CURRENT TRACK. SINCE CURRENT
                        LDX     A,TRACKN        ;   TRACK IS ALWAYS KEPT IN TRUE FORM,
 0253+DD7E03            DB      0DDH,A*8+46H,TRACKN
                        JRZ     DKRWS20         ;    MUST DOUBLE DESIRED TRACK IF
 0256+2801              DB      28H,DKRWS20-$-1
 0258 87                ADD     A               ;     ROBIN MEDIA
                DKRWS20:
 0259 47                MOV     B,A
 025A 3AC504            LDA     BTRACK
 025D A8                XRA     B               ; COMPARE WITH CURRENT TRACK (RESET CARRY)
 025E C9                RET
                ;
                ;
                        PAGE
CP/M MACRO ASSEM 2.0    #024    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION


                ; DPREAD  -  READ A SECTOR
                ;
                ; FUNCTION:  READS ONE SECTOR FROM DISK.  RETRIES WHEN ERROR IS
                ; ENCOUNTERED.
                ;
                ; ENTRY: SECTOR NUMBER AND DMA ADDRESS ARE IN PACKET
                ;
                ; EXIT:  A = FDC STATUS
                ;
                ;
                DPREAD:
```

```
 025F CDDB02            CALL    DPRWEX          ; (RETURNS TO CALLER OF DPREAD)
                ;
                DPREADX:
 0262 21EDA2            LXI     H,INII          ; SET UP INI INSTR.
                DPREADZ:
 0265 224600            SHLD    INOROUT
                        LDX     A,TRACKN        ; SET TRACK FOR SECTOR READ OR WRITE
 0268+DD7E03            DB      0DDH,A*8+46H,TRACKN
 026B D361              OUT     QPTRKRG         ;  WILL DIFFER FROM PHYS. TRACK IF ROBIN
 026D 1E80              MVI     E,QCREADS       ; SEND COMMAND TO FDC
 026F CD7904            CALL    DPRWSET
 0272 CD4000            CALL    RDORWR          ; TRANSFER DATA
 0275 FB                EI                      ; REENABLE
 0276 DB60              IN      QPCOMD          ; GET STATUS
 0278 E6BD              ANI     QMREAD          ; MASK ERRORS
 027A C9                RET                     ; RETURN WITH STATUS
                ;
                ;
                ;
                ;
                ; DPWRITE  -  WRITE A SECTOR
                ;
                ; FUNCTION:  WRITES ONE SECTOR TO DISK.  RETRIES WHEN ERROR IS
                ; ENCOUNTERED.
                ;
                ; ENTRY: SECTOR NUMBER AND DMA ADDRESS ARE IN PACKET
                ;
                ; EXIT:  A = FDC STATUS
                ;
                DPWRITE:
 027B CDDB02            CALL    DPRWEX          ; (RETURNS TO CALLER OF DPWRITE)
                ;
                DPWRTX:
 027E 21EDA3            LXI     H,OUTII         ; SET UP OUTI INSTR.
 0281 224600            SHLD    INOROUT
                        LDX     A,TRACKN        ; SET TRACK FOR SECTOR READ OR WRITE
 0284+DD7E03            DB      0DDH,A*8+46H,TRACKN
 0287 D361              OUT     QPTRKRG         ;  WILL DIFFER FROM PHYS. TRACK IF ROBIN
                ;
 0289 CDAD02            CALL    GSODRV          ; GET "NOT READY" STATUS OF OTHER DRIVE
 028C D5                PUSH    D               ; SAVE FOR COMPARISON AFTER WRITE OF SECTOR
 028D 1EA0              MVI     E,QCWRTS        ; SEND THIS COMMAND TO FDC
 028F CD7904            CALL    DPRWSET
 0292 CD4000            CALL    RDORWR          ; TRANSFER DATA
CP/M MACRO ASSEM 2.0    #025    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION

 0295 FB                EI                      ; REENABLE
 0296 DB60              IN      QPCOMD          ; GET STATUS
 0298 E6FD              ANI     QMWRITE         ; MASK ERRORS
 029A 47                MOV     B,A             ; SAVE RESULT OF WRITE OPERATION
 029B 0E03              MVI     C,3             ; 1.5  MSEC DELAY
 029D CD4A00            CALL    UDELAY          ; FOR CONTROLLER TO FINISH WRT CLEANUP
 02A0 CDAD02            CALL    GSODRV          ; GET "NOT READY" STATUS OF OTHER DRIVE
 02A3 D1                POP     D               ; RESTORE OLD STATUS OF OTHER DRIVE
 02A4 BA                CMP     D               ; COMPARE WITH OLD STATUS
```

```
02A5 3E00               MVI     A,0             ; ASSUME DOOR OF OTHER DRIVE WAS NOT TOUCHED
                        JRZ     DPWRT3          ; IT WAS'NT
02A7+2802               DB      28H,DPWRT3-$-1
02A9 3E08               MVI     A,QMCRC         ; IT WAS - REPORT AS CRC ERROR
02AB B0         DPWRT3: ORA     B               ; GET STATUS RESULT OF WRITE OPERATION
02AC C9                 RET                     ;  RETURN WITH STATUS
                ;
                ;       GET "NOT READY" STATUS OF OTHER DRIVE
                ;
                GSODRV:
02AD 3AC204             LDA     BCURDRV         ; GET CURRENT DRIVE SELECTED
02B0 EE01               XRI     1               ; SWITCH TO OTHER DRIVE
02B2 D340               OUT     QPSTAT          ; SELECT OTHER DRIVE
02B4 CDC202             CALL    DELAY7          ; DELAY 7 USECS
02B7 DB60               IN      QPCOMD          ; GET STATUS OF OTHER DRIVE
02B9 E680               ANI     QMNRDY          ; THIS BIT TELLS US IF DOOR WAS OPENED OR CLOSED
02BB 57                 MOV     D,A             ; SAVE FOR COMPARISON AFTER WRITE OF SECTOR
02BC 3AC404             LDA     TCURDRV         ; CURRENT DRIVE WITH PRE COMP BIT SET IF NECESSARY
02BF D340               OUT     QPSTAT          ; RESELECT CURRENT DRIVE
02C1 7A                 MOV     A,D             ; RESTORE FOR COMPARISON AFTER WRITE OF SECTOR
02C2 C9         DELAY7: RET                     ; PROVIDES ~ 7 USEC DELAY
                ;
                ;
                ; DPRDADR  -  READ ADDRESS
                ;
                ; FUNCTION: READS ADDRESS OF A DISK SECTOR
                ;       (USED TO DETERMINE TRACK NUMBER)
                ;
                ; ENTRY: N/A
                ;
                ; EXIT: TRACK NUMBER IS IN SECTOR REGISTER OF FDC
                ;  (NO DATA IS TRANSFERRED)
                ;
                ;
                DPRDADR:
02C3 CDDB02             CALL    DPRWEX          ; (RETURNS TO CALLER OF DPRDADR)
                ;
                DPRDADRX:
02C6 21DB63             LXI     H,ININ          ; SET UP IN QPDATA INSTR.
02C9 224600             SHLD    INOROUT
02CC 1EC0               MVI     E,QCRDADR       ; SEND COMMAND TO FDC
02CE CD7904             CALL    DPRWSET
02D1 CD4000             CALL    RDORWR          ; TRANSFER DATA
02D4 FB                 EI                      ; REENABLE
02D5 DB60               IN      QPCOMD          ; GET STATUS
02D7 E69D               ANI     QMRDADR         ; MASK ERRORS
02D9 37                 STC                     ; INHIBIT ADVANCED ERROR RECOVERY
CP/M MACRO ASSEM 2.0    #026    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION

02DA C9                 RET                     ; RETURN WITH STATUS
                ;
                ;
                ;
                ; DPRWEX  -  EXECUTES A READ OR WRITE OPERATION
                ;
```

```
                ; FUNCTION: EXECUTES A READ OR WRITE OPERATION AND PERFORMS
                ;  ERROR RECOVERY AS NECESSARY.
                ;
                ; ENTRY: ADDRESS OF PHYSICAL I/O ROUTINE IS ON TOP OF STACK
                ;
                ; EXIT:  A = FDC STATUS
                ;
                DPRWEX:
 02DB E1                POP     H               ; GET I/O ROUTINE ADDRESS
 02DC E5                PUSH    H               ; SAVE FOR ERROR RECOVERY
 02DD CD3403            CALL    DPRWE90         ; EXECUTE THE ROUTINE
 02E0 E1                POP     H               ; RECOVER I/O ROUTINE ADDRESS
 02E1 C8                RZ                      ; EXIT IF NO ERRORS
                ; ERROR RECOVERY PROCESSING
 02E2 5F                MOV     E,A             ; SAVE ERROR CODES
                        JRC     DPRWE10         ; JUST A FEW RETRIES IF CARRY SET
 02E3+3804              DB      38H,DPRWE10-$-1
 02E5 E618              ANI     QMCRC+QMRNF     ; CRC ERROR OR RNF?
                        JRNZ    DPRWE20         ;  YES - PROCESS IT
 02E7+2014              DB      20H,DPRWE20-$-1
                ;
                ; REPEAT OPERATION UNTIL RETRY COUNT IS EXHAUSTED
                ;
                DPRWE10:
 02E9 1604              MVI     D,QKDRETRY      ; INITIALIZE RETRY COUNTER
                DPRWE15:
 02EB D5                PUSH    D               ; STASH RETRY COUNT AND ORIGINAL ERROR CODES
 02EC 3ED0              MVI     A,QCTERM
 02EE D360              OUT     QPCOMD          ; TERMINATE COMMAND
 02F0 E5                PUSH    H               ; PRESERVE I/O ROUTINE ADDRESS
 02F1 CD3403            CALL    DPRWE90         ; EXECUTE IT
 02F4 E1                POP     H               ; GET BACK STUFF THAT WAS STASHED
 02F5 D1                POP     D
 02F6 C8                RZ                      ; EXIT IF SUCCESSFUL
 02F7 15                DCR     D               ; ANY MORE RETRIES LEFT?
                        JRNZ    DPRWE15         ; TRY AGAIN
 02F8+20F1              DB      20H,DPRWE15-$-1
 02FA 7B                MOV     A,E             ; GET FDC STATUS
 02FB A7                ANA     A               ; SET TO NON-0 TO INDICATE ERROR
 02FC C9                RET
                ;
                ; REPEAT OPERATION, THEN PERFORM ADVANCED RECOVERY
                ;
                DPRWE20:
 02FD CDE902            CALL    DPRWE10         ; REPEAT OPERATION
 0300 C8                RZ                      ; EXIT IF SUCCESSFULL
                ; RESTORE DRIVE, SEEK AGAIN AND DO OPERATION ONE MORE
 0301 D5                PUSH    D               ; SAVE I/O ROTINA ADDRESS, RETRY COUNT AND
 0302 E5                PUSH    H               ;  ORIGINAL ERROR STATUS
 0303 CD3503            CALL    DPRECAL         ; RESTORE
CP/M MACRO ASSEM 2.0    #027    Z80 INTERFCE AND PRIMITIVE ROUTINES - PRIVATE VERSION

                        JRNZ    DPRWE80         ; HARD RESTORE ERROR
 0306+2027              DB      20H,DPRWE80-$-1
 0308 CD6003            CALL    DPSEEK1         ; SEEK TO TRACK
```

```
                        JRNZ    DPRWE80         ; HARD SEEK ERROR
030B+2022               DB      20H,DPRWE80-$-1
030D E1                 POP     H               ; GET ALL THAT STUFF BACK
030E D1                 POP     D
030F CDE902             CALL    DPRWE10         ;  AND DO SOME MORE RETRIES
0312 C8                 RZ                      ; SUCCESS
                        LDX     D,TRACKN        ; NOW WE TRY TO SNEAK UP ON IT FROM
0313+DD5603             DB      0DDH,D*8+46H,TRACKN
0316 E5                 PUSH    H               ;  THE OTHER SIDE: SEEK TO TRACK 79
0317 D5                 PUSH    D               ;   AND THEN BACK TO THE TARGET TRACK.
                        MVIX    QKMXTRK,TRACKN  ;    TO DO THIS, WE MUST SVE TARGET TRACK NO.
0318+DD36034F           DB      0DDH,36H,TRACKN,QKMXTRK
031C CD7303             CALL    DPSEEKX         ;     AND SUBSTITUTE 79 IN PACKET
031F D1                 POP     D               ; PUT BACK CORRECT TRACK INTO PACKET
                        STX     D,TRACKN
0320+DD7203             DB      0DDH,70H+D,TRACKN
0323 D5                 PUSH    D               ; REMEMBER: E HAS ORIGINAL ERROR STATUS
                        JRNZ    DPRWE80         ; OOPS: COLDN'T GET TO TRACK 79
0324+2009               DB      20H,DPRWE80-$-1
0326 CD6003             CALL    DPSEEK1         ; BACK TO DESIRED TRACK
                        JRNZ    DPRWE80         ; SEEK ERROR
0329+2004               DB      20H,DPRWE80-$-1
032B D1                 POP     D
032C E1                 POP     H
                        JR      DPRWE10         ; FINAL ATTEMPT. RETURN TO CALLER.
032D+18BA               DB      18H,DPRWE10-$-1
                ;
                ; RESTORE/SEEK ERROR EXIT
                ;
                DPRWE80:
032F E1                 POP     H               ; FIXUP STACK
0330 E1                 POP     H
0331 F601               ORI     QMBUSY          ; MARK AS SEEK ERROR
0333 C9                 RET
                ;
                ; INDIRECT CALL
                ;
                DPRWE90:
0334 E9                 PCHL
                ;
                ;
                ;
                ;
                ; DPRECAL  -  RESTORE DRIVE
                ;
                ; FUNCTION - RETURNS THE HEAD TO TRACK ZERO
                ;   CHECKS FORMAT TO SEE IF VT-180 DISK IS IN DRIVE
                ;
                ; ENTRY: N/A
                ;
                ; EXIT:
                ;       A = DISK CONTROLLER STATUS
                ;       ZERO FLAG IS SET IF RESTORE WAS SUCCESSFUL
CP/M MACRO ASSEM 2.0    #028    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION
```

```
                        ;
                        DPRECAL:
0335 CD4603                     CALL    DPRECALX        ; PERFORM RESTORE
0338 C8                         RZ                      ;  OK
                        ; ERROR RECOVERY - STEP IN FIVE TIMES AND REPEAT
0339 0605                       MVI     B,5             ; SET NR OF STEPS
                        DPRECAL1:
033B 3E48                       MVI     A,QCSTEPIN
033D CD7104                     CALL    DPCOMD          ; STEP IN ONCE
                                DJNZ    DPRECAL1        ; REPEAT
0340+10F9                       DB      10H,DPRECAL1-$-1
0342 CD4603                     CALL    DPRECALX        ; RESTORE AGAIN
0345 C9                         RET                     ; EXIT COME WHAT MAY
                        ;
                        ;
                        ;
                        ; DPRECALX - DO RESTORE OPERATION
                        ;
                        DPRECALX:
0346 3E08                       MVI     A,QCREST
0348 CD7104                     CALL    DPCOMD          ; DO RESTORE
034B DB60                       IN      QPCOMD          ; GET FDC STATUS
034D 4F                         MOV     C,A             ; SAVE STATUS
034E E604                       ANI     QMTZERO         ; AT TRACK ZERO?
                                JRNZ    DPRECALX1       ;  YES
0350+2004                       DB      20H,DPRECALX1-$-1
0352 2F                         CMA                     ; SET NONZERO STATUS
0353 B7                         ORA     A               ; SET FLAGS
0354 79                         MOV     A,C             ; GET STATUS BACK
0355 C9                         RET                     ; EXIT - NOT AT TRACK ZERO
                        DPRECALX1:
0356 79                         MOV     A,C             ; GET STATUS BACK
0357 E691                       ANI     QMREST          ; MASK ERRORS
0359 79                         MOV     A,C             ; RETURN FULL STATUS
035A C9                         RET                     ; RETURN WITH STATUS
                        ;
                        ;
                        ; DPSEEK  -  SEEK TO TRACK
                        ;
                        ; FUNCTION: SEEKS TO THE DESIRED TRACK.  TESTS FOR ERRORS
                        ; AND RETRIES AS NECESSARY.  SETS PRECOMPENSATION BITS FOR
                        ; RAINBOW DISKETTE.  PERFORMS SPECIAL SEEK FOR VT180 FORMAT
                        ; DISKETTE.

                        ;
                        ; ENTRY: DESIRED TRACK NUMBER IS IN PACKET
                        ;  POINTER TO PACKET IS IN IX.
                        ;
                        ; EXIT: A = FDC STATUS
                        ;       ZERO FLAG IS SET IF SEEK WAS SUCCESSFUL
                        ;
                        DPSEEK:
035B 3AC504                     LDA     BTRACK          ; PUT CURRENT TRACK NO INTO REG
035E D361                       OUT     QPTRKRG
                        DPSEEK1:
```

```
 0360 3AC304            LDA     BFORMAT         ; GET CURRENT FORMAT
CP/M MACRO ASSEM 2.0    #029    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION

 0363 A7                ANA     A               ; IS IT RAINBOW?
 0364 C28C03            JNZ     DPSEEKV         ;  NO - DO VT-180 SEEK
 0367 CD7303            CALL    DPSEEKX         ; DO SEEK OPERATION
 036A C8                RZ                      ; OK
                ; ERROR RECOVERY - RESTORE AND SEEK AGAIN
 036B CD3503            CALL    DPRECAL         ; RESTORE DRIVE
 036E C0                RNZ                     ; HARD RESTORE ERROR
 036F CD7303            CALL    DPSEEKX         ; TRY SEEK AGAIN
 0372 C9                RET
                ;
                ;
                ; DPSEEKX  -  PERFORM SEEK OPERATION
                ;
                DPSEEKX:
                        LDX     A,TRACKN        ; GET TRACK NR FROM PACKET
 0373+DD7E03            DB      0DDH,A*8+46H,TRACKN
 0376 FE50              CPI     QKMXTRK+1       ; CATCH ANY BAD TRACK REQUESTS SO AS NOT
                        JRNC    DPSEEKE         ;  TO HANG IN AN IMPOSSIBLE SEEK
 0378+300E              DB      30H,DPSEEKE-$-1
 037A D363              OUT     QPDATA          ; SEND IT TO FDC
 037C E3                XTHL                    ; DELAY A BIT BEFORE COMMAND REG WRITE
 037D E3                XTHL
 037E 3E1C              MVI     A,QCSEEK
 0380 CD7104            CALL    DPCOMD          ; EXECUTE SEEK
                ;       MVI     C,40            ; DELAY FOR HEAD SETTLING
                ;       CALL    UDELAY
 0383 DB60              IN      QPCOMD          ; GET STATUS
 0385 E699              ANI     QMSEEK          ; MASK ERRORS
 0387 C9                RET                     ; RETURN WITH STATUS

                DPSEEKE:
 0388 3E01              MVI     A,QMBUSY        ; ILLEGAL TRACK REQUESTED. INDICATE SEEK
 038A A7                ANA     A               ;  ERROR WITH NO OTHER STATUS
 038B C9                RET
                ;
                ;
                ;
                ; DPSEEKV  -  VT-180 FORMAT SEEK ROUTINE
                ;
                ; FUNCTION: PERFORMS SEEK ON A VT-180 FORMAT (40 TRACK)
                ; DISKETTE.  DOES A SEEK WITH NO VERIFICATION TO THE
                ; SPECIFIED VT-180 TRACK MULTIPLIED BY TWO, THEN READS
                ; AN ADDRESS FROM THE DISKETTE TO VERIFY THAT THE
                ; SEEK WAS SUCCESSFUL.
                ;
                ; ENTRY: IX = PACKET ADDRESS
                ;
                ; EXIT:  A = STATUS
                ;
                DPSEEKV:
 038C CD9D03            CALL    DPSEEKVX        ; PERFORM SEEK
 038F CD9D03            CALL    DPSEEKVX        ; SEEK ERROR - TRY AGAIN
```

```
                        ;
                        ;       SEEK ERROR AGAIN - HOME DISK AND RETRY
                        ;
 0392 CD3503            CALL    DPRECAL         ; HOME DISK
CP/M MACRO ASSEM 2.0    #030    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION

 0395 C0                RNZ                     ; EXIT ON RECAL ERROR
 0396 CD9D03            CALL    DPSEEKVX        ; TRY SEEK AGAIN
 0399 CD9D03            CALL    DPSEEKVX        ; AND AGAIN
 039C C9                RET
                        ;
                        ;
                        ; DPSEEKVX - PERFORM VT-180 SEEK OPERATION
                        ;
                        DPSEEKVX:
                                LDX     A,TRACKN        ; GET DESIRED TRACK
 039D+DD7E03            DB      0DDH,A*8+46H,TRACKN
 03A0 87                ADD     A               ; MULTIPLY TRACK BY 2
 03A1 FE50              CPI     QKMXTRK+1       ; CATCH ANY BAD TRACK REQUESTS SO AS NOT
                        JRNC    DPSEEKE         ;  TO HANG IN AN IMPOSSIBLE SEEK
 03A3+30E3              DB      30H,DPSEEKE-$-1
 03A5 CDBB03            CALL    DPSEEKN         ; SEEK WITH NO VERIFY
 03A8 CDC302            CALL    DPRDADR         ; READ ADDRESS
 03AB 4F                MOV     C,A             ; SAVE STATUS
 03AC A7                ANA     A               ; READ ERROR?
 03AD C0                RNZ                     ;  YES - EXIT
 03AE DB62              IN      QPSECRG         ; GET TRACK NUMBER
                        CMPX    TRACKN          ; COMPARE WITH DESIRED TRACK
 03B0+DDBE03            DB      0DDH,0BEH,TRACKN
 03B3 79                MOV     A,C             ; GET STATUS BACK
 03B4 C0                RNZ                     ; ERROR - NOT AT REQUESTED TRACK
 03B5 E1                POP     H               ; NO ERROR - EMPTY STK FOR 'GOOD' RET
 03B6 C9                RET
                        ;
                        ; DPSEEKN  -  SEEK WITH NO VERIFY
                        ;
                        ; FUNCTION: SEEKS A SPECIFIED TRACK BUT DOES NO VERIFY THE
                        ; SUCCESS OF THE SEEK.  USED FOR STARTING MOTOR AND
                        ; THE VT-180 SEEK.
                        ;
                        ; ENTRY: A = DESIRED TRACK NUMBER
                        ;
                        ; EXIT: N/A
                        ;
                        DPSEKH0:
 03B7 1610              MVI     D,QCSEKH0       ;SEEK WITH NO HEAD LOAD
                        JR      DPSEKN3
 03B9+1802              DB      18H,DPSEKN3-$-1
                        ;
                        DPSEEKN:
 03BB 1618              MVI     D,QCSEEKN       ;SEEK WITH HEAD LOAD
 03BD D363              DPSEKN3: OUT    QPDATA          ; SEND TRACK NR TO FDC
 03BF 7A                MOV     A,D             ;SEEK COMMAND
 03C0 E3                XTHL
 03C1 E3                XTHL
```

```
03C2 CD7104             CALL    DPCOMD          ; DO SEEK
03C5 C9                 RET
                ;
                ;
                ;
                ;
                ; DPSELDR  -  SELECT DRIVE AND CHECK IF IT'S READY
CP/M MACRO ASSEM 2.0    #031    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION

                ;
                ; ENTRY: DRIVE NUMBER IS IN PACKET
                ;
                ; EXIT: DRIVE NUMBER IN E
                ;       IF DRIVE WAS NOT READY, CARRY FLAG IS SET
                ;       AND 'NOT READY' STATUS IS STORED IN PACKET
                ;
                ;
                DPSELDR:
03C6 CDA704             CALL    GETDRV          ; GET DRIVE NUMBER
03C9 5F                 MOV     E,A
                        LDX     A,TRACKN        ; NOW SEE IF WRITE PRECOMP BITS WILL BE
03CA+DD7E03             DB      0DDH,A*8+46H,TRACKN
03CD FE3D               CPI     QKPCTRK         ;   NEEDED: I.E., WHETHER WE ARE TO BE
03CF 3E00               MVI     A,0             ;     BEYOND TRACK 60
                        JRC     DPSEL1
03D1+3802               DB      38H,DPSEL1-$-1
03D3 3E40               MVI     A,QKPCBIT
                DPSEL1:
03D5 B3                 ORA     E
03D6 32C404             STA     TCURDRV         ; DRIVE SELECTED WITH PRE COMP
03D9 D340               OUT     QPSTAT          ; SELECT DRIVE
                DPSEL2:
03DB DB60               IN      QPCOMD          ; GET FDC STATUS
03DD E681               ANI     QMBUSY+QMNRDY   ; READY AND NOT BUSY?
                        JRZ     DPSEL3          ;  YES
03DF+2811               DB      28H,DPSEL3-$-1
03E1 E601               ANI     QMBUSY          ; BUSY?
                        JRNZ    DPSEL2          ;  YES - WAIT IT OUT
03E3+20F6               DB      20H,DPSEL2-$-1
03E5 DB61               IN      QPTRKRG         ;  NO: NOT READY. DUMMY SEEK TO CURRENT TRACK
03E7 CDBB03             CALL    DPSEEKN         ;   TO READY DRIVE IF POSSIBLE
03EA DB60               IN      QPCOMD          ; GET FDC STATUS
03EC E680               ANI     QMNRDY          ; STILL NOT READY?
                        JRNZ    DPSELS2         ;  YES - RETURN NOT READY STATUS
03EE+206B               DB      20H,DPSELS2-$-1
                        JR      DPSEL3C         ; IGNORE CHECK FOR ISSUANCE OF 'HLT' TO 1793
03F0+182A               DB      18H,DPSEL3C-$-1
                ;
                DPSEL3:
03F2 3AC204             LDA     BCURDRV         ; GET CURRENT DRIVE NR
03F5 AB                 XRA     E               ; SAME AS DESIRED DRIVE?
                        JRZ     DPSEL4          ;  YES - GO SEE IF FIRST ACCESS AFTER RESET
03F6+2838               DB      28H,DPSEL4-$-1
                        LDX     A,FNCCOD        ; GET FUNCTION CODE
03F8+DD7E00             DB      0DDH,A*8+46H,FNCCOD
```

```
03FB FE14               CPI     QKWTCOM         ; WRITE TO BE DONE?
                        JRNZ    DPSEL3C         ; NO
03FD+201D               DB      20H,DPSEL3C-$-1
                ;
03FF 3AC204             LDA     BCURDRV         ; YES - THEN GET CURRENT DRIVE
0402 E602               ANI     2               ; CHECK FOR SELECT OF A OR B
0404 57                 MOV     D,A             ; TO C OR D; C OR D TO A OR B
0405 7B                 MOV     A,E             ; NEW DRIVE TO SELECT
0406 E602               ANI     2
0408 BA                 CMP     D
CP/M MACRO ASSEM 2.0    #032    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION

                        JRNZ    DPSEL3A
0409+2007               DB      20H,DPSEL3A-$-1
                ;
                ;       A TO B, B TO A, C TO D, D TO C
                ;
040B 0E3C               MVI     C,60            ; DELAY FOR 30 MSEC TO ALLOW HEADS TO
040D CD4A00             CALL    UDELAY          ; SETTLE DURING HEAD LOAD
                        JR      DPSEL3C
0410+180A               DB      18H,DPSEL3C-$-1
                ;
                ;       THE SELECT FROM A OR B TO C OR D; C OR D TO A OR B REQUIRES
                ;       THE ISSUANCE OF A SEEK WITH QMHLT = 0, THEN QMHLT = 1, WITH NO VERIFY
                ;       THIS INHIBITS THE "HLT" INPUT TO THE 1793 FOR ~ 500 MSECS
                ;       WHICH ALLOWS THE MOTOR TO GET UP TO SPEED.
                ;
                DPSEL3A:
0412 DB61               IN      QPTRKRG         ;CURRENT TRACK
0414 CDB703             CALL    DPSEKH0         ;SEEK WITH NO HEAD LOAD
0417 DB61               IN      QPTRKRG
0419 CDBB03             CALL    DPSEEKN         ;SEEK WITH HEAD LOAD
                ;
041C 7B         DPSEL3C: MOV    A,E             ; SET NEW DRIVE AS CURRENT
041D 32C204             STA     BCURDRV
0420 CD6604             CALL    GETRAK          ; GET CURR TRACK FROM TABLE
0423 32C504             STA     BTRACK          ;  AND STORE IT
                ;       MVI     C,140           ; DELAY FOR HEAD SETTLING
                ;       CALL    UDELAY
                ; SET FORMAT AND TRACK REGISTER
0426 21F43C             LXI     H,TFORMAT       ; BASE OF FORMAT TABLE
0429 1600               MVI     D,0
042B 19                 DAD     D               ; OFFSET INTO TABLE
042C 7E                 MOV     A,M             ; GET FORMAT INFO
042D 32C304             STA     BFORMAT         ; STORE IN CURRENT FORMAT
                DPSEL4:
0430 3AC504             LDA     BTRACK
0433 3C                 INR     A               ; CURR TRACK IS FF ON FIRST ACCESS AFTER DISK
0434 C0                 RNZ                     ;  SYSTEM RESET. IF NOT, CONSIDER DRIVE READY.
                DPSEL5:
0435 DB61               IN      QPTRKRG         ; FORCE TYPE 1 STATUS
0437 CDBB03             CALL    DPSEEKN
                                                ; ON FIRST ACCESS TO A DRIVE AFTER A DISK
                                                ;  SYSTEM RESET, MAKE SURE DISK IS IN
                                                ;  DRIVE RIGHT SIDE UP BY LOOKING FOR A
```

```
                                        ;  CHANGE IN THE INDEX PULSE STATUS. ON A
                                        ;   TIMEOUT, DRIVE IS CONSIDERED NOT READY.
043A 21409C             LXI     H,QKDRCNT
                DPSEL6:
043D DB60               IN      QPCOMD          ; READ STATUS ONCE TO GET RID OF SPURIOUS
043F DB60               IN      QPCOMD          ;  INDEX STATUS
0441 E602               ANI     QMINDEX         ; SAVE STARTING INDEX BIT STATE
0443 47                 MOV     B,A
0444 CD4C04             CALL    DPSELS          ; WAIT FOR 1 CHANGE IN INDEX
0447 CD4C04             CALL    DPSELS          ; WAIT FOR 2ND CHANGE
044A A7                 ANA     A               ; CLEAR CARRY
044B C9                 RET                     ; DISK IS OK
                DPSELS:                         ; SUBROUTINE: CHECKS FOR CHANGE IN INDEX BIT
CP/M MACRO ASSEM 2.0    #033    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION

044C 2B                 DCX     H
044D 7C                 MOV     A,H
044E B5                 ORA     L
                        JRZ     DPSELS1         ; TIMEOUT: DISK NOT READY
044F+2809               DB      28H,DPSELS1-$-1
0451 DB60               IN      QPCOMD
0453 E602               ANI     QMINDEX
0455 B8                 CMP     B
0456 47                 MOV     B,A             ; SAVE NEW INDEX STATUS
0457 C0                 RNZ                     ; AHA-- CHANGE HAS OCCURRED.
                        JR      DPSELS
0458+18F2               DB      18H,DPSELS-$-1
                DPSELS1:
045A C1                 POP     B
                DPSELS2:
045B DB61               IN      QPTRKRG         ; GET CURRENT TRACK #
045D CDB703             CALL    DPSEKH0         ; UNLOAD HEADS
                        MVIX    QMNRDY,STATUS   ; SET NOT READY STATUS
0460+DD360180           DB      0DDH,36H,STATUS,QMNRDY
0464 37                 STC                     ; CARRY MEANS NOT READY
0465 C9                 RET
                ;
                ;
                ; GETRAK -        GET CURRENT TRACK NUMBER FROM TRACK TABLE
                ;
                ; ENTRY:          A = DRIVE NUMBER
                ;
                ; EXIT:           A = TRACK NUMBER
                ;                 HL= TRACK TABLE PINTER FOR DRIVE
                ;
                GETRAK:
0466 E602               ANI     2               ; ** FOR RX-50 COUPLED DRIVES
0468 4F                 MOV     C,A
0469 0600               MVI     B,0
046B 21F03C             LXI     H,TTRACK        ; BASE OF TRACK TABLE
046E 09                 DAD     B               ; OFFSET INTO TRACK TABLE
046F 7E                 MOV     A,M             ; GET TRACK FOR THIS DRIVE
0470 C9                 RET
                ;
                ;
```

```
                        ;
                        ;
                        ; DPCOMD  -  EXECUTE A TYPE 1 COMMAND
                        ;
                        ; WAITS UNTIL COMMAND IS COMPLETED BEFORE RETURNING
                        ;
                        ; ENTRY: COMMAND CODE IS IN 'A' REG.
                        ;
                        DPCOMD:
 0471 D360                      OUT     QPCOMD          ; SEND COMMAND TO FDC
                        DPCOMD1:
 0473 DB40                      IN      QPSTAT          ; GET GENERAL STATUS
 0475 87                        ADD     A               ; SHIFT INT. BIT TO SIGN
 0476 F8                        RM                      ; EXIT ON INTERRUPT
                                JR      DPCOMD1         ; LOOP UNTIL DONE
 0477+18FA                      DB      18H,DPCOMD1-$-1
CP/M MACRO ASSEM 2.0    #034    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION

                        ;
                        ;
                        ; DPRWSET  - SET UP FDC AND SEND TYPE 2 OR 3 (READ/WRITE) COMMAND
                        ; FUNCTION:  SENDS SECTOR TO FDC.  ADDS HEAD LOAD BIT TO COMMAND
                        ; IF HEAD IS NOT ALREADY LOADED.  SETS UP REGISTERS FOR INI AND
                        ; OUTI INSTRUCTIONS.  SENDS COMMAND TO FDC.
                        ;
                        ;
                        ; ENTRY: FDC COMMAND IS IN "E" REG.  SECTOR IS IN PACKET.
                        ;
                        ; EXIT: N/A
                        ;
                        DPRWSET:
 0479 DB60                      IN      QPCOMD          ; GET CONTROLLER STATUS
 047B E601                      ANI     QMBUSY          ; BUSY?
                                JRNZ    DPRWSET         ;  YES - WAIT UNTIL NOT BUSY
 047D+20FA                      DB      20H,DPRWSET-$-1
 047F E3                        XTHL                    ; NEED SOME DELAY. WE'LL UNDO IT SOON
 0480 3ED0                      MVI     A,QCTERM        ; SEND TERMINATE COMMAND
 0482 D360                      OUT     QPCOMD          ;  TO FORCE TYPE 1 STATUS
 0484 E3                        XTHL                    ; DELAY - MATCHES THE ONE ABOVE
 0485 0E63                      MVI     C,QPDATA        ; SET UP DATA PORT NUMBER
                                LDX     L,DMALOW        ; SET UP DMA ADDRESS
 0487+DD6E04                    DB      0DDH,L*8+46H,DMALOW
                                LDX     H,DMAHI
 048A+DD6605                    DB      0DDH,H*8+46H,DMAHI
                                LDX     A,SECTN         ; GET SECTOR NUMBER
 048D+DD7E02                    DB      0DDH,A*8+46H,SECTN
 0490 E61F                      ANI     SECNUM          ; ISOLATE SECTOR NUMBER
 0492 D362                      OUT     QPSECRG         ; SEND IT TO FDC
 0494 E3                        XTHL                    ; DELAY BEFORE STATUS READ
 0495 E3                        XTHL
 0496 E3                        XTHL
 0497 E3                        XTHL
 0498 E3                        XTHL
 0499 E3                        XTHL
 049A F3                        DI                      ; DISABLE FOR DISK I/O
```

```
 049B DB60              IN      QPCOMD          ; GET FDC STATUS
 049D E620              ANI     QMHLT           ; HEAD ALREADY LOADED?
 049F 7B                MOV     A,E             ; GET COMMAND
                        JRNZ    DPRWSET4        ;;  SKIP IF HEAD ALREADY LOADED
 04A0+2002              DB      20H,DPRWSET4-$-1
 04A2 F604              ORI     QMEFLG          ; ADD HEAD LOAD FLAG
                DPRWSET4:
 04A4 D360              OUT     QPCOMD          ; SEND TO CONTROLLER
 04A6 C9                RET
                ;
                ;
                ; GETDRV  -  GET DRIVE NUMBER FROM PACKET
                ;
                ; ENTRY: IX = PACKET ADDRESS
                ;
                ; EXIT: A = DRIVE NUMBER
                ;
                GETDRV:
                        LDX     A,DRIVEN        ; GET DRIVE NUMBER
CP/M MACRO ASSEM 2.0    #035    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION

 04A7+DD7E02            DB      0DDH,A*8+46H,DRIVEN
 04AA E660              ANI     DRVNUM          ; ISOLATE IT
 04AC 07                RLC                     ; MOVE INTO BITS 0-2
 04AD 07                RLC
 04AE 07                RLC
 04AF C9                RET
                ;
                ;
                ; SETRAK  -  STORE TRACK NUMBER IN TRACK TABLE
                ;
                ; ENTRY: A = TRACK NUMBER
                ;        IX = PACKET ADDRESS
                ;
                ; EXIT: TTRACK CONTAINS NEW TRACK VALUE
                ;
                SETRAK:
 04B0 32C504            STA     BTRACK          ; CURRENT TRACK FOR CURRENT DRIVE
 04B3 47                MOV     B,A             ; SAVE TRACK
 04B4 CDA704            CALL    GETDRV          ; GET DRIVE NUMBER
 04B7 E602              ANI     2               ; ** FOR RX-50 COUPLING
 04B9 5F                MOV     E,A             ; MAKE 16-BIT DRIVE NR
 04BA 1600              MVI     D,0
 04BC 21F03C            LXI     H,TTRACK        ; BASE OF TRACK TABLE
 04BF 19                DAD     D               ; OFFSET INTO TABLE
 04C0 70                MOV     M,B             ; STORE TRACK
 04C1 C9                RET
                ;
                ;
                        PAGE
CP/M MACRO ASSEM 2.0    #036    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION


                ;
                ;
```

```
                ; DATA AREA
                ;
04C2 FF         BCURDRV DB      0FFH    ; CURRENT DRIVE NUMBER
04C3 00         BFORMAT DB      0       ; FORMAT FOR CURRENT DRIVE (NONZERO = VT-180)
04C4 FF         TCURDRV DB      0FFH    ;CURRENT DRIVE # WITH PRE COMP SET IF NECESSARY
04C5 FF         BTRACK  DB      0FFH    ; CURRENT TRACK OF SELECTED DRIVE ("TRUE" TRACK)
                ;
                ;TTRACK DB      0FFH,0FFH,0FFH,0FFH     ; TRACK TABLE
                ;TFORMAT        DB      0,0,0,0         ; FORMAT TABLE (NONZERO = VT180)
                ;** NOTE: TTRACK AND TFORMAT ARE IN THE DATA BLOCK. THEY ARE DEFINED
                ;         AT THE END OF THIS MODULE (Z80CODE.ASM), NEXT TO THE PACKET
                ;         POINTER DEFINITIONS (Z80PKT AND I88PKT)
                ;
                ;
                        PAGE
CP/M MACRO ASSEM 2.0    #037    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION


                ; ********************************************************
                ;
                ; NAME: ZSTART
                ;
                ; FUNCTION: THIS ROUTINE WILL SET UP THE STACK AND ALLOW THE Z80
                ;           TO EXECUTE AN APPLICATION.
                ;
                ; ENTRY: IX = PACKET ADDRESS
                ;
                ; EXIT:
                ;
                ZSTART:
04C6 F3                 DI                              ; DISABLE INTERRUPTS WHILE
04C7 313005             LXI     SP,STACK                ;   SWAPPING STACKS

04CA 210000             LXI     H,0                     ; PUSH 00 ONTO STACK FOR CP/M-80
                                                        ;   TRANSIENTS THAT END WITH 'RET'
                                                        ;   SO IT WILL BE SAME AS ENDING
                                                        ;   WITH 'JUMP 0'

04CD E5                 PUSH    H

04CE 3EFF               MVI     A,TRUE                  ; SET DONE FLAG
04D0 323105             STA     DONEFL

                        LDX     L,STADRL                ; GET START ADDRESS FROM PACKET
04D3+DD6E02             DB      0DDH,L*8+46H,STADRL
                        LDX     H,STADRH
04D6+DD6603             DB      0DDH,H*8+46H,STADRH

04D9 FB                 EI                              ; ENABLE INTERRUPTS

04DA E9                 PCHL                            ; JUMP TO START ADDRESS
                        PAGE
CP/M MACRO ASSEM 2.0    #038    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION
```

```
                ; ***************************************************************
                ;
                ; NAME: ZMOVE
                ;
                ; FUNCTION: THIS ROUTINE WILL MOVE A BLOCK OF DATA FROM ANYWHERE IN
                ;           Z80 MEMORY TO ANYWHERE ELSE IN Z80 MEMORY.
                ;
                ; ENTRY: IX = PACKET ADDRESS
                ;
                ; EXIT: IX = PACKET ADDRESS
                ;
                ZMOVE:
                        LDX     L,SCADRL                ; GET SOURCE ADDRESS INTO HL
 04DB+DD6E02            DB      0DDH,L*8+46H,SCADRL
                        LDX     H,SCADRH
 04DE+DD6603            DB      0DDH,H*8+46H,SCADRH

                        LDX     E,DSADRL                ; GET DESTINATION ADDRESS INTO DE
 04E1+DD5E04            DB      0DDH,E*8+46H,DSADRL
                        LDX     D,DSADRH
 04E4+DD5605            DB      0DDH,D*8+46H,DSADRH

                        LDX     C,BYCNTL                ; GET BYTE COUNT INTO BC
 04E7+DD4E06            DB      0DDH,C*8+46H,BYCNTL
                        LDX     B,BYCNTH
 04EA+DD4607            DB      0DDH,B*8+46H,BYCNTH

                        LDIR                            ; MOVE THE BLOCK OF DATA
 04ED+EDB0              DB      0EDH,0B0H
 04EF C9                RET                             ;   AND RETURN
                        PAGE
CP/M MACRO ASSEM 2.0    #039    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION


                ; ***************************************************************
                ; ***************************************************************

 0530 =         STACK   EQU     $+40H                   ; SERVICE ROUTINE STACK POINTER

 0531 =         DONEFL  EQU     STACK+1                 ; DONE FLAG

 3CFC =         Z80PKT  EQU     INTFPTR                 ; PACKET POINTER FOR PACKET FROM Z80

 3CFE =         I88PKT  EQU     INTFPTR+2               ; PACKET POINTER FOR PACKET FROM 8088

                ; FOLLOWING TWO DEFINITIONS ARE FOR THE DISK I/O ROUTINES.
                ; THE TABLES ARE LOCATED IN THE DATA BLOCK, INITIALIZED BY THE BIOS
                ; COLD BOOT ROUTINE AND MOVED BY Z80CCP WHEN A CONFIGURATION CHANGE OCCURS.
 3CF0 =         TTRACK  EQU     INTFPTR-12              ; TRACK TABLE
 3CF4 =         TFORMAT EQU     INTFPTR-8               ; FORMAT TABLE

 0532                   ORG     DONEFL+1

 0532                   END
CP/M MACRO ASSEM 2.0    #040    Z80 INTERFACE AND PRIMITIVE ROUTINES - PRIVATE VERSION
```

```
0000 BC          04C2 BCURDRV     04C3 BFORMAT     04C5 BTRACK      0007 BYCNTH
0006 BYCNTL      0002 DE          02C2 DELAY7      0186 DKCHECK     0195 DKCHECK1
0198 DKCHECK2    01BB DKCHECK3    01CF DKREAD      01E0 DKRW20      01EC DKRW30
01EF DKRW40      01FC DKRW90      0201 DKRW95      024F DKRWS10     0259 DKRWS20
023C DKRWSET     0014 DKWRIT      01CF DKWRITE     0005 DMAHI       0004 DMALOW
0531 DONEFL      0471 DPCOMD      0473 DPCOMD1     02C3 DPRDADR     02C6 DPRDADRX
025F DPREAD      0262 DPREADX     0265 DPREADZ     0335 DPRECAL     033B DPRECAL1
0346 DPRECALX    0356 DPRECALX1   02E9 DPRWE10     02EB DPRWE15     02FD DPRWE20
032F DPRWE80     0334 DPRWE90     02DB DPRWEX      0479 DPRWSET     04A4 DPRWSET4
035B DPSEEK      0360 DPSEEK1     0388 DPSEEKE     03BB DPSEEKN     038C DPSEEKV
039D DPSEEKVX    0373 DPSEEKX     03B7 DPSEKH0     03BD DPSEKN3     03D5 DPSEL1
03DB DPSEL2      03F2 DPSEL3      0412 DPSEL3A     041C DPSEL3C     0430 DPSEL4
0435 DPSEL5      043D DPSEL6      03C6 DPSELDR     045A DPSELS1     045B DPSELS2
044C DPSELS      027B DPWRITE     02AB DPWRT3      027E DPWRTX      0002 DRIVEN
0060 DRVNUM      0005 DSADRH      0004 DSADRL      0010 DSKFNC      014D ERROR
0000 FALSE       0000 FNCCOD      00FF FNCNG       00F0 FRANGE      04A7 GETDRV
0466 GETRAK      02AD GSODRV      0171 HIFNC       0179 HIFXIT      0004 HL
0000 I88INT      3CFE I88PKT      A2ED INII        63DB ININ        0046 INOROUT
0004 INTBIT      3CFC INTFPTR     0020 INTSTA      0100 INTSTR      0184 IPKT
0004 IX          0004 IY          00C3 JUMP        0007 LEGFUN      022C NEXTSC10
0237 NEXTSC15    0238 NEXTSC20    0207 NEXTSEC     0006 NSECT       000A NUMSEC
0020 OTHFNC      A3ED OUTII       0112 PKTPRO      015E PKTR1       0152 PKTRET
FFFF PRIVATE     00C0 QCRDADR     0080 QCREADS     0008 QCREST      001C QCSEEK
0018 QCSEEKN     0010 QCSEKH0     0048 QCSTEPIN    0068 QCSTEPOT    00D0 QCTERM
00F0 QCWRTRK     00A0 QCWRTS      0015 QKCMCOM     9C40 QKDRCNT     0004 QKDRETRY
000A QKMXSECT    004F QKMXTRK     0040 QKPCBIT     003D QKPCTRK     0000 QKRATE
0013 QKRDCOM     0002 QKROBIN     0014 QKWTCOM     0001 QMBUSY      0008 QMCRC
0020 QMDELDM     0003 QMDRNR      0002 QMDRQ       0004 QMEFLG      0008 QMHLD
0020 QMHLT       0002 QMINDEX     0004 QMLDATA     0080 QMNRDY      0008 QMON0
0018 QMON        0010 QMON1       00C0 QMPRECMP    0020 QMPSIDE     009D QMRDADR
00BD QMREAD      0091 QMREST      0010 QMRNF       0002 QMSCOM      0099 QMSEEK
0010 QMSKERR     0008 QMSSEL      0004 QMTG42      0004 QMTZERO     0010 QMUPDT
0004 QMVERF      0040 QMWPROT     0020 QMWRFLT     00FD QMWRITE     0060 QPCOMD
0063 QPDATA      0062 QPSECRG     0040 QPSTAT      0061 QPTRKRG     0040 RDORWR
0018 RST3        0020 RST4        0030 RST6        0003 SCADRH      0002 SCADRL
001F SECNUM      0200 SECSIZ      0002 SECTN       04B0 SETRAK      0000 SHARE
0530 STACK       0003 STADRH      0002 STADRL      0100 START       0001 STATUS
04C4 TCURDRV     3CF4 TFORMAT     0003 TRACKN      FFFF TRUE        3CF0 TTRACK
004A UDELAY      0040 USRFNC      0021 Z80BGN      0022 Z80MVE      3CFC Z80PKT
04DB ZMOVE       04C6 ZSTART
```

```
                        TITLE   'DRI PATCHES FOR CP/M 86'
                        ; ************************************************************
                        ;
                        ; CP/M 86 V1.1, PATCH 01, BDOS, 3/3/82
                        ;
                        ; ************************************************************
  21E6                  PATCH_1         EQU     21E6H           ; LOCATION OF FIRST PATCH
  21F1                  PATCH_2         EQU     21F1H           ; LOCATION OF SECOND PATCH
  21ED                  F472            EQU     21EDH
  000A                  CCP_BUFF_LEN    EQU     BYTE PTR .0AH
                                CSEG    $
                                ORG     F472
                        FUNC472:
                                ORG     PATCH_1
21E6 AA                         STOSB
21E7 0AC0                       OR      AL,AL
21E9 7402         21ED          JZ      FUNC472
                                ORG     PATCH_2
21F1 1F                         POP     DS
21F2 A20A00                     MOV     CCP_BUFF_LEN,AL
```

```
                        EJECT
                        ; ****************************************************************
                        ;
                        ; CP/M 86 V1.1, PATCH 06, BDOS, 4/23/82
                        ;
                        ; ****************************************************************
   24A2                 DMABASE         EQU     WORD PTR 24A2H
   1896                 PATCH_AREA      EQU     1896H
                                CSEG
                                ORG     PATCH_AREA
 1896 8E06A224                  MOV     ES,.DMABASE
```

```
                        EJECT
                ; ***************************************************************
                ;
                ; CP/M 86 V1.1, PATCH 07, BDOS, 4/21/82
                ;
                ; ***************************************************************
  0B77          PATCH_AREA2     EQU     0B77H
                        CSEG
                        ORG     PATCH_AREA2
0B77 33C0               XOR     AX,AX
```

```
                        EJECT
                    ; ************************************************************
                    ;
                    ; CP/M 86 V1.1, PATCH 08, BDOS, 5/12/82
                    ;
                    ; ************************************************************
                        CSEG
                        ORG     0E23H
0E23 06                 PUSH    ES
0E24 90                 NOP
0E25 8916DC22           MOV     .22DCH,DX
                        ORG     0F1AH
0F1A E903FB     0A20    JMP     PATCH_AREA3
0F1D 90                 NOP
                        ORG     0A20H
                    PATCH_AREA3:
0A20 A39624             MOV     .2496H,AX
0A23 07                 POP     ES
0A24 C3                 RET
```

```
                        EJECT
                        ; ****************************************************************
                        ;
                        ; CP/M 86 V1.1, PATCH 10, BDOS, 5/20/82
                        ;
                        ; ****************************************************************
                                CSEG
                                ORG     0A50H
                        LABEL_0:
0A50 7207       0A59            JB      OK
0A52 C7069B24FFFF               MOV     WORD PTR .249BH,0FFFFH
0A58 C3                         RET
                        OK:
0A59 E93B01     0B97            JMP     LABEL_1
                                ORG     0B94H
0B94 E9B9FE     0A50            JMP     LABEL_0
                                ORG     0B97H
                        LABEL_1:
```

```
                        EJECT
                    ; ****************************************************************
                    ;
                    ; CP/M 86 V1.1, PATCH 11, BDOS, 5/20/82
                    ;
                    ; ****************************************************************
  0F1E              LOADERR_LOC     EQU     0F1EH           ; ADDR OF LOAD ERROR ROUTINE
  0F06              LOAD59_RET      EQU     0F06H           ; EXIT FROM LOAD ERROR ROUTINE
  0A30              PAT_OFFSET      EQU     0A30H           ; PATCH AREA
  2323              USER_PARM_SEG   EQU     02323H          ; USER'S DATA SEGMENT
  249B              ARET            EQU     0249BH          ; RETURN VALUE
  249D              PARAM_SEG       EQU     0249DH          ; USER'S DATA SEGMENT AT ENTRY
                            CSEG
                            ORG     PAT_OFFSET
                    PATCH:
0A30 A12323                 MOV     AX,.USER_PARM_SEG
0A33 A39D24                 MOV     .PARAM_SEG,AX
0A36 C7069B24FFFF           MOV     WORD PTR .ARET,0FFFFH
0A3C E9C704         0F06    JMP     LOADRET
                            ORG     LOAD59_RET
                    LOADRET:
                            ORG     LOADERR_LOC
0F1E E90FFB         0A30    JMP     PATCH
```

```
                        EJECT
                        ; ***************************************************************
                        ;
                        ; CP/M 86 V1.1, PATCH 13, BDOS, 6/08/82
                        ;
                        ; ***************************************************************
  0A00                  PATCH_AREA4     EQU     0A00H
  00E0                  BDOS            EQU     00E0H
                                CSEG
                                ORG     PATCH_AREA4
                        START_PATCH:
 0A00 2EC606982401              MOV     CS:BYTE PTR .2498H,01
 0A06 50                        PUSH    AX
 0A07 2E8A262723                MOV     AH,CS:.2327H
 0A0C 2EA01323                  MOV     AL,CS:.2313H
 0A10 50                        PUSH    AX
 0A11 CDE0                      INT     BDOS
 0A13 58                        POP     AX
 0A14 2EA21323                  MOV     CS:.2313H,AL
 0A18 2E88262723                MOV     CS:.2327H,AH
 0A1D 58                        POP     AX
 0A1E C3                        RET
 0A1F 90                        NOP
                                ORG     0C42H
 0C42 E9BBFD       0A00         JMP     START_PATCH
 0C45 90                        NOP
 0C46 90                        NOP
 0C47 90                        NOP
 0C48 90                        NOP
 0C49 90                        NOP
 0C4A 90                        NOP
```

```
                                EJECT
                                END
```

END OF ASSEMBLY. NUMBER OF ERRORS: 0. USE FACTOR: 2%

```
                        title   'Customized Basic I/O System'
                        ;***********************************************
                        ;*                                             *
                        ;* This Customized BIOS adapts CP/M-86 to      *
                        ;* the following hardware configuration        *
                        ;*      Processor: PC-100 Rainbow              *
                        ;*      Brand:     DEC                         *
                        ;*      Controller:                            *
                        ;*      System: CP/M 86/80                     *
                        ;*                                             *
                        ;*                                             *
                        ;*      Programmer:rdk/CPL                     *
                        ;*      Revisions :                            *
                        ;*                                             *
                        ;*                                             *
                        ;*      Release 1.0.1                          *
                        ;*                                             *
                        ;*      8/25/82 BIOS modified to operate with  *
                        ;*              PC-100 Rainbow hardware, and to*
                        ;*              operate with Z80 second CPU.   *
                        ;*              ROM date: 8/17/82              *
                        ;*                                             *
                        ;*      9/28/82 Mod number 2 (release 1.0.2)   *
                        ;*              Firmware vector initializing   *
                        ;*              added.  Interrupts initialized *
                        ;*              for printer and com port.      *
                        ;*              XON/XOFF support included for  *
                        ;*              input routines as well as      *
                        ;*              output.  Ability to run submit *
                        ;*              files with scratch other than  *
                        ;*              A added.                       *
                        ;*              ROM date: 9/20/82              *
                        ;*                                             *
                        ;***********************************************
0001                    version_number  equ     1
0000                    rev_number      equ     0
0006                    mod_number      equ     6
FFFF                    true            equ -1
0000                    false           equ not true
000D                    cr              equ 0dh ;carriage return
000A                    lf              equ 0ah ;line feed
0043                    lsts            equ     43h
0041                    ldata           equ     41h
0042                    csts            equ     42h
0040                    cdata           equ     40h
0013                    ctrl_s          equ     13h
0011                    ctrl_q          equ     11h
                        ;***********************************************
```

```
                        ;*                                          *
                        ;* Loader_bios is true if assembling the     *
                        ;* LOADER BIOS, otherwise BIOS is for the    *
                        ;* CPM.SYS file.                             *
                        ;*                                          *
                        ;**********************************************
0000                    loader_bios     equ false
00E0                    bdos_int        equ 224 ;reserved BDOS interrupt
                        ;DEBLOCK               EQU TRUE        ;do deblocking
                                IF      not loader_bios
                        ;-----------------------------------------------
                        ;|                                             |
2500                    bios_code       equ 2500h
0000                    ccp_offset      equ 0000h
0B06                    bdos ofst       equ 0B06h       ;BDOS entry point
24B7                    curdrvs         equ 24B7h       ;BDOS current drive address
22E6                    kbchar          equ 22e6h       ;BDOS keyboard character
                        ;|                                             |
                        ;-----------------------------------------------
                                ENDIF   ;not loader bios
FD00                    pb2_adr         equ 0fd00h      ;
                                IF      loader_bios
                        ;-----------------------------------------------
                        ;|                                             |
                        bios code       equ 1200h ;start of LDBIOS
                        ccp_offset      equ 0003h ;base of CPMLOADER
                        bdos ofst       equ 0406h ;stripped BDOS entry
                        ;|                                             |
                        ;-----------------------------------------------
                                ENDIF   ;loader_bios
                        ;
=                               INCLUDE DEFBUF.LIB
=                       ; ************************************************************************
=                       ; OFFSETS FROM START OF POINTERS/BUFFERS DATA BLOCK
=
=  FFA0                 XDPCX           EQU     -60H            ; DISK PARAMETER STORAGE (60H)
=  0000                 XDEFBUF         EQU     000H            ; MISC. BUFFER (LENGTH=80H)
=  0086                 XPACKET         EQU     086H            ; BIOS MESSAGE PACKET (LENGTH=0EH)
=  0086                 XSTPKT          EQU     086H            ; START PACKET BUFFER (LENGTH=0EH)
=  0094                 XADCPKT         EQU     094H            ; DATA PACKET (LENGTH=0EH)
=  0094                 XMVPKT          EQU     094H            ; MOVE PACKET BUFFER (LENGTH=0EH)
=  00A2                 XSHRBUF         EQU     0A2H            ; SEGMENT BUFFER (LENGTH=200H)
=  02F8                 XMEMSIZE        EQU     2F8H            ; MEMORY SIZE (WORD)
=  02FA                 XPCPMADR        EQU     2FAH            ; PSEUDO CP/M ADDRESS (WORD)
=  02FC                 XZ80PKT         EQU     2FCH            ; PACKET POINTER FROM Z80 (WORD)
=  02FE                 XI88PKT         EQU     2FEH            ; PACKET POINTER FROM 8088 (WORD)
=  02F0                 XTTRACK         EQU     2F0H            ;TRACK TABLE
=  02F4                 XTFORMAT        EQU     2F4H            ; FORMAT TABLE
=  02E7                 XCSFLAG         EQU     2E7H            ; CONSOLE STATUS FLAG
```

```
=
=                       ; OFFSETS FROM ZOT FOR CONVENIENCE
=  0000                 ZOTP            EQU     0               ; Z80 FLAG
=  FFFE                 Z80FLAGPT       EQU     -2              ; Z80-RUNNING FLAG
=  FFFB                 CICCK           EQU     -5              ; CONSOLE STATUS FLAG CHECK
=
=                       ; OTHER USEFUL EQUATES
=  0002                 BDCS            EQU     2               ; BDOS CHARACTER READY BIT
=  0001                 BIOCS           EQU     1               ; BIOS CONSOLE STATUS BIT
=  0017                 BIOS_JMPS       EQU     23              ; NUMBER OF FUNCTIONS IN JUMP TABLE
=
=                       ; ************************************************************************
=                               INCLUDE CPLBIOS1.A86
=                       ;
=                               cseg
=                               org     ccpoffset
=                       ccp:
=                               org     bios_code
=
=                       ;*****************************************
=                       ;*                                       *
=                       ;* BIOS Jump Vector for Individual Routines  *
=                       ;*                                       *
=                       ;*****************************************
=
=2500 E92101        2624  jmp INIT        ;Enter from BOOT ROM or LOADER
=2503 E9C201        26C8  jmp WBOOT       ;Arrive here from BDOS call 0
=2506 E92205        2A2B  jmp CONST       ;return console keyboard status
=2509 E92405        2A30  jmp CONIN       ;return console keyboard char
=250C E92605        2A35  jmp CONOUT      ;write char to console device
=250F E92D05        2A3F  jmp LISTOUT     ;write character to list device
=2512 E92F05        2A44  jmp PUNCH       ;write character to punch device
=2515 E93105        2A49  jmp READER      ;return char from reader device
=2518 E98E06        2BA9  jmp HOME        ;move to trk 00 on cur sel drive
=251B E9EE05        2B0C  jmp SELDSK      ;select disk for next rd/write
=251E E99906        2BBA  jmp SETTRK      ;set track for next rd/write
=2521 E99C06        2BC0  jmp SETSEC      ;set sector for next rd/write
=2524 E99F06        2BC6  jmp SETDMA      ;set offset for user buff (DMA)
=2527 E9B806        2BE2  jmp READ        ;read a 128 byte sector
=252A E9D006        2BFD  jmp WRITE       ;write a 128 byte sector
=252D E90A05        2A3A  jmp LISTST      ;return list status
=2530 E99F06        2BD2  jmp SECTRAN     ;xlate logical->physical sector
=2533 E99606        2BCC  jmp SETDMAB     ;set seg base for buff (DMA)
=2536 E9A705        2AE0  jmp GETSEGT     ;return offset of Mem Desc Table
=2539 E92005        2A5C  jmp GETIOBF     ;return I/O map byte (IOBYTE)
=253C E92205        2A61  jmp SETIOBF     ;set I/O map byte (IOBYTE)
=253F E95108        2D93  jmp RWMOVE      ;move block of data (* added for 86/80 *)
=2542 E99F05        2AE4  jmp VIDEO       ;direct video output (* added for 86/80 *)
=
=                               IF      not loader_bios
=                       ;
```

```
=                       ;
=                       ; Segment Table address is placed here immediately after
=                       ; the BIOS jumps to help the loader find the segment table
=                       ; and set it up.
=                       ;
=2545 5E33                      DW      OFFSET SEGTABLE
=                       ;
=                       ;
=2547 0000              DBPTR   DW      0               ; POINTER TO DATA BLOCK
=                       ;                               (FILLED BY LOADER OR MOVE ROUTINE)
=                               ENDIF
=                               IF      loader_bios
=                       DBPTR   EQU     3A00H           ;(Filled by loader)
=                               ENDIF
=                       ;
=                       ;*****************************************************************************
=                       ;*
=                       ;*              INTERPROCESSOR COMMUNICATION ROUTINES
=                       ;*
=                       ;*****************************************************************************
=                       ;
=                       ; EQUATES
=                       ;
=                       ;
=  0000                 INTZ80  EQU     0               ; PORT TO INTERRUPT Z80
=  0002                 GSCR    EQU     2               ; INTERRUPT STATUS PORT
=  0080                 BIT7    EQU     80H             ; Z80 INTERRUPT STATUS BIT (0 = PENDING)
=                       ;
=2549 9D27              SIOINIT         DW      TPRTISQ         ; LOCATION TO FURNISH INIT TABLE START
=254B 00                CONINCHECK      DB      0               ; CONSOLE STATUS CHECK
=254C 0000              Z80PKT          DW      0               ; TEMP STORAGE FOR PACKET ADDRESS
=                       ;
=                       ; Z80FLAG AND ZOT MUST BE POSITIONED IMMEDIATELY BEFORE THE
=                       ; TYPE 39 INTERRUPT HANDLER SO THAT Z80CCP.CMD CAN REFERENCE
=                       ; THEM
=                       ;
=254E 0000              Z80FLAG dw      0       ;Z80 running flag (non-zero if Z80 running)
=2550 0000              ZOT     dw      0       ;z80 flag
=                       ;
=                       ; INTERRUPT HANDLER FOR TYPE 39 INTERRUPT (Z80 COMMUNICATION)
=                       ;
=                       TYPE_39_SERV:
=2552 50                        PUSH    AX              ; SAVE REGISTERS
=2553 53                        PUSH    BX
=2554 2E8B1E4725                MOV     BX,DBPTR        ; GET POINTER TO DATA BLOCK
=2559 1E                        PUSH    DS
=255A 33C0                      XOR     AX,AX
=255C 8ED8                      MOV     DS,AX           ; SET ZERO DS
=255E 8B9FFC02                  MOV     BX,XZ80PKT[BX]  ; GET PACKET POINTER
=2562 85DB                      TEST    BX,BX
=2564 740C          2572        JZ      TYPE_39_EXIT    ; IGNORE ZERO FOR PACKET ADDRESS
=2566 2E891E4C25                MOV     Z80PKT,BX       ; STORE PACKET ADDRESS
```

```
=256B 2EC7065025FF              MOV     ZOT,TRUE        ; SET THE FLAG
      FF
=                       TYPE_39_EXIT:
=2572 E400                      IN      AL,INTZ80       ; CLEAR THE INTERRUPT
=2574 1F                        POP     DS              ; RESTORE DS
=2575 5B                        POP     BX              ; RESTORE REGISTERS
=2576 58                        POP     AX
=2577 CF                        IRET
=                       if not loader_bios
=                       ;---------------------------------------------------------------
=                       ;
=                       ;       interrupt handler for type 44 (line frequency clock)
=                       ;
=                       type_44_serv:
=2578 2EC6064B25FF              mov     CONINCHECK,0ffh ;set flag
=257E CD64                      int 100                 ;see below
=2580 CF                        iret
=                       ;
=                       ;
=                       ;       software interrupt for line frequency clock (type 100)
=                       ;
=                       type_100_serv:
=2581 CF                        iret                    ;user may intercept for use
=                       ;---------------------------------------------------------------
=                       endif   ;not loader_bios
=                       ;
=                       ;
=                       ; SENDPKT  -  SEMD A PACKET TO THE Z80
=                       ;
=                       ; ENTRY:  BX = POINTER TO PACKET (ABSOLUTE)
=                       ;
=                       ; EXIT:  N/A
=                       ;
=                       SENDPKT:
=2582 1E                        PUSH    DS
=2583 53                        PUSH    BX              ; SAVE POINTER
=2584 2E8B1E4725                MOV     BX,DBPTR        ; POINT TO DATA BLOCK
=2589 33C0                      XOR     AX,AX
=258B 8ED8                      MOV     DS,AX           ; SET ZERO DS
=258D 8F87FE02                  POP     WORD PTR XI88PKT[BX]    ; STORE PACKET POINTER
=                       ; SIGNAL Z80 AND WAIT FOR ACKNOWLEDGEMENT
=2591 E600                      OUT     INTZ80,AL       ; INTERRUPT THE Z80
=                       SENDPK10:
=2593 E402                      IN      AL,GSCR         ; GET Z80 STATUS
=2595 A880                      TEST    AL,BIT7         ; INTERRUPT STILL PENDING?
=2597 74FA          2593        JZ      SENDPK10        ;  YES - CHECK AGAIN
=2599 C787FE020000              MOV     WORD PTR XI88PKT[BX],0  ; ZERO THE PACKET POINTER
=259F 1F                        POP     DS              ; RESTORE DS
=25A0 C3                        RET
=                       ;
=                       ;
=                       ;***********************************************
```

```
=                       ;*                                          *
=                       ;*      Routine to wait for an interrupt    *
=                       ;*      (type 39) from Z80                  *
=                       ;*                                          *
=                       ;**********************************************
=
=                       WAITZ80:
=25A1 FA                        cli             ;no interrupts please
=25A2 2E833E5025FF              cmp ZOT,TRUE
=25A8 7404          25AE        je  WAITRET     ;
=25AA FB                        sti             ;must allow interrupts now
=25AB F4                        hlt             ;wait quietly until interrupt
=25AC EBF3          25A1        jmps WAITZ80    ;loop back until non-zero
=25AE FB                WAITRET: sti            ;allow interrupts
=25AF 2E8B1E4C25                mov bx,z80pkt   ; get packet address
=25B4 C3                        ret             ;bye
=                       ;
=                       ;
=                       PACKER:         ;take items off the stack and put them
=25B5 1E                        push ds         ;into the message packet
=25B6 2EA35025                  mov ZOT,ax      ;Z80 flag set
=25BA 2E891E0332                mov BXHLD,bx    ;save bx
=25BF 33DB                      xor bx,bx       ;set zero data segment
=25C1 8EDB                      mov ds,bx       ;
=25C3 2E8F065833                pop SEGHLD      ;save data segment
=25C8 2E8F065A33                pop RTNHLD      ;return address
=25CD 2E8B1E4725                mov bx,DBPTR    ; POINT TO DATA BLOCK
=25D2 81C38600                  add bx,xpacket  ; point to 88 packet
=25D6 03D9                      add bx,cx       ;end of packet
=25D8 03D9                      add bx,cx       ;=bx+2*cx
=25DA 2E890E5C33                mov COUNT,cx    ;save count for later
=                       pklp:                   ;loop to do packing
=25DF 4B                        dec bx          ;back up bx
=25E0 4B                        dec bx          ;twice
=25E1 58                        pop ax          ;get t.o.s.
=25E2 8907                      mov [bx],ax     ;pack it
=25E4 E2F9          25DF        loop pklp       ;loop until done
=25E6 2EC706502500              mov ZOT,false   ; clear "done" flag
      00
=25ED E892FF        2?92        call sendpkt    ; send packet to z80
=25F0 E8AEFF        25A1        call waitz80    ;wait if z80 is working
=25F3 2E8B0E5C33                mov cx,COUNT    ;get the count again
=                       repak:                  ;take stuff out of packet
=25F8 8B07                      mov ax,[bx]     ;and
=25FA 50                        push ax         ;push it on the stack
=25FB 43                        inc bx          ;bump the pointer
=25FC 43                        inc bx          ;twice
=25FD E2F9          25F8        loop repak      ;loop until done
=
=25FF 2EFF365A33                push RTNHLD
=2604 2E8E1E5833                mov ds,SEGHLD   ;restore return and segment
=2609 2E8B1E0332                mov bx,BXHLD    ;restore bx
```

```
=260E C3                        ret
=
=
=                       pmsg:
=260F 8A07                      mov al,[BX]     ;get next char from message
=2611 84C0                      test al,al
=2613 740A          261F        jz pmretn       ;if zero return
=2615 8AC8                      mov CL,AL
=2617 53                        push bx         ; preserve pointer
=2618 E81A04        2A35        call CONOUT     ;print it
=261B 5B                        pop bx
=261C 43                        inc BX
=261D EBF0          260F        jmps pmsg       ;next character and loop
=261F B083              pmretn: mov al,83h      ;make sure that crt is initiallized
=2621 E60A                      out 0ah,al
=2623 C3                        ret
=
=
=                       ;*****************************************
=                       ;*                                       *
=                       ;* INIT Entry Point, Differs for LDBIOS and  *
=                       ;* BIOS, according to "Loader_Bios" value    *
=                       ;*                                       *
=                       ;*****************************************
=
=                       INIT:   ;print signon message and initialize hardware
=                               if loader bios                  ; set up a stack
=                       ;---------------------------------------;
=                               cli                             ; no interrupts, please
=                               mov ax,cs                       ; use cs for stack too
=                               mov ss,ax                       ;
=                               mov sp,offset stkbase           ; set up a local stack
=                               sti                             ; interrupts ok now
=                               push cx
=                       ;---------------------------------------;
=                               endif
=
=2624 8CC8                      mov ax,cs       ;we entered with a JMPF so use
=2626 8ED8                      mov ds,ax       ;CS: as the initial value for DS:,
=2628 8EC0                      mov es,ax       ;and ES:
=
=                               IF      not loader bios
=                       ;---------------------------------------
=                       ;|                                      |
=                               ; This is a BIOS for the CPM.SYS file.
=                               ; Setup all interrupt vectors in low
=                               ; memory to address trap
=                               ;use local stack during initialization
=262A FA                        cli             ;no interrupts while doing the stack
=262B 8ED0                      mov ss,ax       ;CS: as the initial value of SS:,
=262D BC3039                    mov sp,offset stkbase
=2630 FB                        sti             ;interrupts ok now
```

```
=2631 FC                        cld             ;set forward direction
=2632 51                        push cx
=2633 B195                      mov cl,95H      ;set IOBYTE to lst=lpt,con=crt
=2635 E82904        2A61        call SETIOBF    ;
=2638 2EC7064E2500              mov Z80FLAG,0   ;set z80 not running (yet)
      00
=263F E82D00        266F        call REVECTOR   ;initialize interrupts
=
=                           ;       (additional CP/M-86 initialization)
=                           ;|                                          |
=                           ;-------------------------------------------
=                               ENDIF   ;not loader bios
=
=                               IF      loader_bios
=                           ;-------------------------------------------
=                           ;|                                          |
=                               ;This is a BIOS for the LOADER
=                               cld             ;set forward direction
=                               call REVECTOR   ;set up interrupts
=                           ;|                                          |
=                           ;-------------------------------------------
=                               ENDIF   ;loader bios
=                               if      not loader_bios
=2642 FB                        STI
=                               endif
=2643 BB6733                    mov bx,offset signon
=2646 E8C6FF        260F        call pmsg       ;print signon message
=
=                           ;set up track and format tables
=2649 2E8B1E4725                mov bx,dbptr    ;get address of table area
=264E 1E                        push ds         ;
=264F 33C0                      xor ax,ax       ;zero for data seg
=2651 8ED8                      mov ds,ax       ;
=2653 B90400                    mov cx,nrdisks  ;how many disks?
=2656 F6D4                      not ah          ;need ff in ah
=2658 88A7F002          init10: mov xttrack[bx],ah      ;tracks to ff
=265C 8887F402                  mov xtformat[bx],al     ;formats to 00
=2660 43                        inc bx
=2661 E2F5          2658        loop init10     ;loop back until done
=2663 1F                        pop ds          ;restore data seg
=
=2664 59                        pop cx          ;restore drive etc
=                           if not loader_bios
=                           ;-----------------------------------------------------
=2665 8AC1                      mov al,cl       ;let's take a look at that drive
=2667 240F                      and al,0fh      ;make sure it's valid, then ...
=2669 A2B724                    mov byte ptr .curdrvs,al        ;store drive for submit files
=                           ;-----------------------------------------------------
=                           endif   ;not loader_bios
=266C E991D9        0000        jmp ccp         ;jump to cold start entry of CCP
=                           IF loader_bios
=                           ;-------------------------------------------
```

```
=                       REVECTOR:
=                               push ds         ;save data segment
=                               mov ax,0
=                               mov ds,ax       ;point to segment zero
=                               ;BDOS interrupt offset
=                               mov bdos_offset,bdos ofst
=                               mov bdos_segment,CS ;bdos interrupt segment
=                       ;       (additional LOADER initialization)
=                               MOV     Z80_OFFSET,OFFSET TYPE_39_SERV
=                               MOV     Z80_SEG,cs      ;use current code segment (why not?
=                               mov sio offset,offset I232RX
=                               mov sio seg,cs
=                               mov sio2_offset,offset I232RX2
=                               mov sio2_seg,cs
=                               pop ds          ;restore data segment
=                               ret
=                       ;-----------------------------------------
=                       ENDIF ;loader_bios
=
=                       ;-------------------------------------------------
=                               IF not loader bios
=                       ; SET UP INTERRUPT VECTORS
=                       REVECTOR:
=266F 1E                        push ds         ;save the DS register
=                       ;
=                       ;*********************************************************
=                       ;**     Firmware initializing                           ***
=2670 E84801       27BB         call P232CHEK                                   ;**
=2673 32D2                      xor dl,dl                                       ;**
=2675 BF1600                    mov di,16h                                      ;**
=2678 CD28                      int 40                                          ;**
=267A BF0C00                    mov di,0ch                                      ;**
=267D CD28                      int 40                                          ;**
=267F FA                        cli                                             ;**
=                       ;*********************************************************
=2680 B80000                    mov ax,0
=2683 8ED8                      mov ds,ax
=2685 8EC0                      mov es,ax       ;set ES and DS to zero
=                               ;BDOS offset to proper interrupt
=2687 C7068003060B              mov bdos_offset,bdos_ofst
=268D 8C0E8203                  MOV     BDOS_SEGMENT,CS
=2691 C7069C005225              MOV     Z80_OFFSET,OFFSET TYPE_39_SERV
=2697 8C0E9E00                  MOV     Z80_SEG,cs      ;use current code segment (why not?)
=269B C706B0007825              mov tp_44_offset,offset type_44_serv
=26A1 8C0EB200                  mov tp_44_seg,cs
=26A5 C70690018125              mov tp_100_offset,offset type_100_serv
=26AB 8C0E9201                  mov tp_100_seg,cs
=26AF C7069000CC28              mov sio_offset,offset I232RX
=26B5 8C0E9200                  mov sio_seg,cs
=26B9 C7069400EA28              mov sio2_offset,offset I232RX2
=26BF 8C0E9600                  mov sio2_seg,cs
=
```

```
=26C3 FB                        STI             ; RE-ENABLE INTERRUPTS
=                       ;
=26C4 1F                        pop ds          ;restore the DS register
=26C5 E91D01        27E5        jmp P232INIT    ;initialize sio
=                       ENDIF   ;not loader_bios
=                       ;-----------------------------------------------
=
=26C8 2EC7064E2500      WBOOT:  mov Z80FLAG,false       ;set z80 not running
      00
=26CF E89DFF        266F        call revector
=26D2 E931D9        0006        jmp ccp+6       ;direct entry to CCP at command level
=
=
=                       ;*********************************************
=                       ;*                                           *
=                       ;*   CP/M Character I/O Interface Routines   *
=                       ;*                                           *
=                       ;*********************************************
=                       ;
=                       ; TERMINAL DEVICE DRIVERS
=                       ;
=                       CRTOUT:
=26D5 8AC1                      MOV     AL,CL           ; Move character for output
=26D7 BF0000                    MOV     DI,0            ; Function code
=26DA 06                        PUSH    ES
=26DB CD28                      INT     40
=26DD 07                        POP     ES
=26DE C3                        RET
=                       ;
=                       CRTIN:
=26DF 06                        PUSH    ES
=26E0 BF0200            CRTIN1: MOV     DI,2            ; Function code
=26E3 CD28                      INT     40
=26E5 84C9                      TEST    CL,CL           ; Character available?
=26E7 74F7          26E0        JZ      CRTIN1          ;  No - retry
=26E9 2EF7064E25FF              test z80flag,true       ; Is Z80 running?
      FF
=26F0 741E          2710        jz CRTIN2               ; go away if not
=26F2 50                        push ax
=26F3 BF0400                    mov di,4
=26F6 CD28                      int 40
=26F8 33DB                      xor bx,bx
=26FA 8EC3                      mov es,bx
=26FC 2E8B1E4725                mov bx,DBPTR
=2701 80E101                    and cl,BIOCS
=2704 2680A7E702FE              and es:byte ptr xcsflag[bx],not BIOCS   ;clear status flag
=270A 26088FE702                or es:byte ptr xcsflag[bx],cl
=270F 58                        pop ax
=2710 07                CRTIN2: pop es
=2711 C3                        RET                     ; Return character in AL
=                       ;
=                       CRTSTI:
```

```
=2712 BF0400                    MOV     DI,4            ; Function code
=2715 06                        PUSH    ES
=2716 CD28                      INT     40
=2718 07                        POP     ES
=2719 8AC1                      MOV     AL,CL           ; Move status for return
=271B C3                        RET
=                       ;
=                       CRTSTO:
=271C B0FF                      MOV     AL,0FFH         ; Always ready
=271E C3                        RET
=                       ;
=                       ;
=                       ; SERIAL I/O ROUTINES FOR DEC RAINBOW 100
=                       ;
=                       ;
=  0080                 HIPAR   EQU     80H             ; HIGH PARITY BIT
=                       ;
=                       ; CONTROL BLOCK OFFSETS
=                       ;
=  0000                 QTPORT  EQU     BYTE PTR 0      ; CONTROL PORT ADDRESS
=  0001                 QTFLAGS EQU     BYTE PTR 1      ; DEVICE FLAGS (SEE MEANINGS BELOW)
=  0002                 QTNRCHR EQU     BYTE PTR 2      ; NR. OF CHARACTERS CURRENTLY IN BUFFER
=  0003                 QTCAP   EQU     BYTE PTR 3      ; BUFFER CAPACITY IN BYTES (CONSTANT)
=  0004                 QTINPTR EQU     BYTE PTR 4      ; BUFFER INPUT POINTER
=  0005                 QTOTPTR EQU     BYTE PTR 5      ; BUFFER OUTPUT POINTER
=  0006                 QTDEND  EQU     BYTE PTR 6      ; OFFSET OF LAST DATA BYTE (CONSTANT)
=  0007                 QTDEVID EQU     7               ; OFFSET OF PHYS DEVICE ID FOR ERROR MESSAGE
=  000A                 QTDATA  EQU     10              ; BUFFER DATA AREA
=                       ;
=                       ; BIT ASSIGNMENTS FOR "QTFLAGS"
=                       ;
=  0001                 QMSUSP  EQU     1       ; 1 = OUTPUT SUSPENDED
=  0002                 QMTYPE  EQU     2       ; 0 = XON/XOFF, PARITY ERROR CHECKING
=  0004                 QMINIT  EQU     4       ; 1 = DEVICE REQUIRES INITIALIZATION
=  0008                 QMISUSP EQU     8       ; 1 = INPUT SUSPENDED
=  0010                 QMBREAK EQU     10H     ; 1 = BREAK DETECTED
=                       ;
=                       ; BUFFER LENGTHS IN CONTROL BLOCKS
=                       ;
=  0020                 QKPRTBL EQU     32      ; PRINTER CONTROL BLOCK
=  0020                 QKCOMBL EQU     32      ; COMM PORT CONTROL BLOCK
=  0020                 QKCOM2BL EQU    32      ; OPTIONAL COMM PORT CTL BLOCK
=                       ;
=                       ; CONTROL PORT ADDRESSES
=                       ;
=  0043                 QPPRT   EQU     43H     ; PRINTER PORT
=  0042                 QPCOM   EQU     42H     ; COMM PORT
=  0022                 QPCOM2  EQU     22H     ; OPTIONAL COMM PORT
=                       ;
=                       ; ASCII CONTROL CHARACTERS
=                       ;
=  0007                 QKBEL   EQU     7       ; BEL
```

```
=  0011                 QKXON   EQU     17      ; XON (CTL-Q)
=  0013                 QKXOFF  EQU     19      ; XOFF (CTL-S)
=  0091                 QKXONP  EQU     17+HIPAR        ; XON+ (CTL-Q)
=  0093                 QKXOFFP EQU     19+HIPAR        ; XOFF+ (CTL-S)
=  001A                 QKSUB   EQU     26      ; SUB
=                       ;
=                       ; SIO STATUS BITS - RR0
=                       ;
=  0001                 QMRXR   EQU     1       ; RECEIVED CHAR. READY
=  0004                 QMTXR   EQU     4       ; TRANSMIT READY
=  0080                 QMBRK   EQU     80H     ; BREAK
=                       ;
=                       ; SIO STATUS BITS - RR1
=                       ;
=  0010                 QMPARE  EQU     10H     ; PARITY ERROR
=  0020                 QMOVRE  EQU     20H     ; OVERRUN ERROR
=                       ;
=                       ; SIO COMMANDS - WR0
=                       ;
=  0038                 QKEOI EQU       38H     ; END OF INTERRUPT
=  0018                 QKCHRST EQU     18H     ; CHANNEL RESET
=  0030                 QKRESERR EQU    30H     ; RESET ERRORS
=  0010                 QKRESI  EQU     10H     ; RESET EXT/STATUS INT.
=                       ;
=                       ; SIO COMMANDS - WR1
=                       ;
=  0018                 QMMRXI  EQU     18H     ; INTERRUPT ON ALL RX CHAR.
=                       ;
=                       ; SIO COMMANDS - WR3
=                       ;
=  0040                 QMR7BIT EQU     40H     ; RX 7 BITS/CHAR
=  00C0                 QMR8BIT EQU     0C0H    ; RX 8 BITS/CHAR
=  0001                 QMMRXE  EQU     1       ; RX ENABLE
=                       ;
=                       ; SIO COMMANDS - WR4
=                       ;
=  0040                 QMMX16  EQU     40H     ; X16 CLOCK
=  000C                 QMMST2  EQU     0CH     ; 2 STOP BITS
=  0004                 QMMST1  EQU     4       ; 1 STOP BIT
=                       ;
=                       ; SIO COMMANDS - WR5
=                       ;
=  0080                 QMMDTR  EQU     80H     ; DTR ON
=  0002                 QMMRTS  EQU     2       ; RTS ON
=  0008                 QMMTXE  EQU     8       ; TX ENABLE
=  0020                 QMT7BIT EQU     20H     ; TX 7 BITS/CHAR
=  0060                 QMT8BIT EQU     60H     ; TX 8 BITS/CHAR
=                       ;
=                       ;
=                       ;
=                       ;
=                       ; CONTROL BLOCKS FOR EACH DEVICE
```

```
=                       ;
=                       ; PRINTER CONTROL BLOCK
=                       ;
=271F                   TPRTCB  RS      0
=271F 43                        DB      QPPRT   ; PORT ADDRESS
=2720 04                        DB      QMINIT  ; USES XON/XOFF
=2721 00                        DB      0       ; NUMBER OF CHARS.
=2722 20                        DB      QKPRTBL ; CAPACITY
=2723 0A                        DB      QTDATA  ; INPUT POINTER
=2724 0A                        DB      QTDATA  ; OUTPUT POINTER
=2725 29                        DB      QTDATA-1+QKPRTBL        ; OFFSET OF LAST DATA BYTE
=2726 545459                    DB      'TTY'   ; DEVICE ID
=2729                           RS      QKPRTBL ; DATA BUFFER
=                       ;
=                       ; COMM PORT CONTROL BLOCK
=                       ;
=2749                   TCOMCB  RS      0
=2749 42                        DB      QPCOM   ; PORT ADDRESS
=274A 06                        DB      QMTYPE+QMINIT   ; NO XON/XOFF
=274B 00                        DB      0       ; NUMBER OF CHARACTERS
=274C 20                        DB      QKCOMBL ; CAPACITY
=274D 0A                        DB      QTDATA  ; INPUT POINTER
=274E 0A                        DB      QTDATA  ; OUTPUT POINTER
=274F 29                        DB      QTDATA-1+QKCOMBL        ; OFFSET OF LAST DATA BYTE
=2750 505450                    DB      'PTP'   ; DEVICE ID
=2753                           RS      QKCOMBL ; DATA BUFFER
=                       ;
=                       ;OPTIONAL COMM PORT CONTROL BLOCK
=                       ;
=2773                   TCOM2CB RS      0
=2773 22                        DB      QPCOM2  ; PORT ADDRESS
=2774 02                        DB      QMTYPE  ; NO XON/XOFF
=2775 00                        DB      0       ; NUMBER OF CHARACTERS
=2776 20                        DB      QKCOM2BL        ; CAPACITY
=2777 0A                        DB      QTDATA  ; INPUT POINTER
=2778 0A                        DB      QTDATA  ; OUTPUT POINTER
=2779 29                        DB      QTDATA-1+QKCOM2BL       ; OFFSET OF LAST DATA BYTE
=277A 554331                    DB      'UC1'           ; DEVICE ID
=277D                           RS      QKCOM2BL        ; DATA BUFFER
=                       ;
=                       ;
=                       ; INITIALIZATION SEQUENCES FOR SIO
=                       ;
=                       ; PRINTER PORT
=                       ;
=279D                   TPRTISQ RS      0
=279D 18                        DB      QKCHRST                 ; CHANNEL RESET
=279E 14                        DB      4+QKRESI                ; WR4
=279F 4C                        DB      QMMX16+QMMST2           ; X16 CLOCK, 2 STOP BITS
=27A0 13                        DB      3+QKRESI                ; WR3
=27A1 41                        DB      QMR7BIT+QMMRXE          ; RX ENABLE, 7 BITS/CH
=27A2 15                        DB      5+QKRESI                ; WR5
```

```
=27A3 AA                        DB      QMMDTR+QMMRTS+QMT7BIT+QMMTXE  ; TX ENABLE, 7 BITS, RTS, DTR
=27A4 11                TPRTISI DB      1+QKRESI                      ; WR1
=27A5 18                        DB      QMMRXI                        ; INTERRUPT ON ALL RX CHAR
=27A6 00                        DB      0                             ; END OF SEQUENCE
=                       ;
=                       ; COMM PORT
=                       ;
=27A7                   TCOMISQ RS      0
=27A7 18                        DB      QKCHRST                       ; CHANNEL RESET
=27A8 14                        DB      4+QKRESI                      ; WR4
=27A9 44                        DB      QMMX16+QMMST1                 ; X16 CLOCK, 1 STOP BIT
=27AA 13                        DB      3+QKRESI                      ; WR3
=27AB C1                        DB      QMR8BIT+QMMRXE                ; RX ENABLE, 8 BITS/CH
=27AC 15                        DB      5+QKRESI                      ; WR5
=27AD EA                        DB      QMMDTR+QMMRTS+QMT8BIT+QMMTXE  ; TX ENABLE, 8 BITS, RTS, DTR
=27AE 11                TCOMISI DB      1+QKRESI                      ; WR1
=27AF 18                        DB      QMMRXI                        ; INTERRUPT ON ALL RX CHAR
=27B0 00                        DB      0                             ; END OF SEQUENCE
=                       ;
=                       ; OPTIONAL COMM PORT
=                       ;
=27B1                   TCOM2ISQ        RS      0
=27B1 18                        DB      QKCHRST                       ; CHANNEL RESET
=27B2 14                        DB      4+QKRESI                      ; WR4
=27B3 44                        DB      QMMX16+QMMST1                 ; X16 CLOCK, 1 STOP BIT
=27B4 13                        DB      3+QKRESI                      ; WR3
=27B5 C1                        DB      QMR8BIT+QMMRXE                ; RX ENABLE, 8 BITS/CH
=27B6 15                        DB      5+QKRESI                      ; WR5
=27B7 EA                        DB      QMMDTR+QMMRTS+QMT8BIT+QMMTXE  ; TX ENABLE, 8 BITS, RTS, DTR
=27B8 11                        DB      1+QKRESI                      ; WR1
=27B9 18                        DB      QMMRXI                        ; INTERRUPT ON ALL RX CHAR
=27BA 00                        DB      0                             ; END OF SEQUENCE
=                       ;
=                       ;
=                       ;
=                       ; P232CHEK - CHECK RS232 (SIO) PORTS
=                       ;
=                       ; FUNCTION: CHECK TO SEE IF LAST CHARACTER HAS BEEN SENT FROM PORT IF
=                       ;  INITIALIZATION IS REQUIRED.
=                       ;
=                       ; ENTRY AND EXIT PARAMETERS: N/A
=                       ;
=                       P232CHEK:
=27BB BB1F27                    MOV     BX,OFFSET TPRTCB              ; PRINTER CONTROL BLOCK
=27BE E80C00        27CD        CALL    P232CHK             ; CHECK
=27C1 BB4927                    MOV     BX,OFFSET TCOMCB              ; COMM CONTROL BLOCK
=27C4 E80600        27CD        CALL    P232CHK             ; CHECK
=27C7 BB7327                    MOV     BX,OFFSET TCOM2CB             ; OPT COMM CONTROL BLOCK
=27CA E90000        27CD        JMP     P232CHK             ; CHECK
=                       ;
=                       ;
=                       ; P232CHK - CHECK AN SIO PORT
```

```
=                       ;
=                       ; FUNCTION: CHECKS AN SIO PORT ASSOCIATED WITH A CONTROL BLOCK.
=                       ;  TO SEE IF THE LAST CHARACTER HAS BEEN SENT BEFORE DOING INITILAZATION.
=                       ;  CHECKS CONTROL BLOCK FIRST TO SEE IF INITIALIZATION IS REQUIRED.
=                       ;
=                       ; ENTRY:
=                       ;       BX = POINTER TO CONTROL BLOCK
=                       ;
=                       ; EXIT : N/A
=                       ;
=                       P232CHK:
=27CD F6470104                  TEST    QTFLAGS[BX],QMINIT      ; INITIALIZATION REQUIRED?
=27D1 7501          27D4        JNZ     P232CH10                ;  YES
=27D3 C3                        RET                             ; ELSE RETURN
=                       P232CH10:
=27D4 8A17                      MOV     DL,QTPORT[BX]           ; GET PORT ADDRESS
=27D6 32F6                      XOR     DH,DH                   ; MAKE 16 BIT ADDR
=27D8 33C9                      XOR     CX,CX                   ; ZERO THE COUNTER
=                       P232CH30:
=27DA B001                      MOV     AL,1                    ; CONTROL REGISTER 1
=27DC EE                        OUT     DX,AL                   ; SEND IT TO PORT
=27DD EC                        IN      AL,DX                   ; READ IT
=27DE 2401                      AND     AL,1                    ; CHECK FOR LAST
=27E0 7502          27E4        JNZ     P232CH20                ; GO AWAY IF OK
=27E2 E2F6          27DA        LOOP    P232CH30                ; LOOP UNTIL TIMOUT
=                       P232CH20:
=27E4 C3                        RET
=                       ;
=                       ;
=                       ;
=                       ;
=                       ;
=                       ; P232INIT - INITIALIZE RS232 (SIO) PORTS
=                       ;
=                       ; FUNCTION: INITIALIZES ALL RS232 PORTS FOR WHICH INITIALIZATION
=                       ;  IS REQUIRED.
=                       ;
=                       ; ENTRY AND EXIT PARAMETERS: N/A
=                       ;
=                       P232INIT:
=27E5 BB1F27                    MOV     BX,OFFSET TPRTCB        ; PRINTER CONTROL BLOCK
=27E8 BEA427                    MOV     SI,OFFSET TPRTISI       ; PRINTER INIT SEQUENCE
=27EB E81200        2800        CALL    P232IPR         ; INITIALIZE
=27EE BB4927                    MOV     BX,OFFSET TCOMCB        ; COMM CONTROL BLOCK
=27F1 BEAE27                    MOV     SI,OFFSET TCOMISI       ; COMM INIT SEQUENCE
=27F4 E80900        2800        CALL    P232IPR         ; INITIALIZE
=27F7 BB7327                    MOV     BX,OFFSET TCOM2CB       ; OPT COMM CONTROL BLOCK
=27FA BEB127                    MOV     SI,OFFSET TCOM2ISQ      ; OPT COMM INIT SEQUENCE
=27FD E90000        2800        JMP     P232IPR         ; INITIALIZE
=                       ;
=                       ;
=                       ; P232IPR - INITIALIZE AN SIO PORT
```

```
=                       ;
=                       ; FUNCTION: INITIALIZES AN SIO PORT ASSOCIATED WITH A CONTROL BLOCK.
=                       ;  CHECKS CONTROL BLOCK FIRST TO SEE IF INITIALIZATION IS REQUIRED.
=                       ;  (DOES NOT INITIALIZE BAUD RATES.)
=                       ;
=                       ; ENTRY:
=                       ;       BX = POINTER TO CONTROL BLOCK
=                       ;       SI = POINTER TO INITIALIZATION SEQUENCE
=                       ;            (SEQUENCE ENDS WITH A ZERO BYTE)
=                       ;
=                       ; EXIT : N/A
=                       ;
=                       P232IPR:
=2800 F6470104                  TEST    QTFLAGS[BX],QMINIT      ; INITIALIZATION REQUIRED?
=2804 7501          2807        JNZ     P232IP10                ;  YES
=2806 C3                        RET                             ; ELSE RETURN
=                       P232IP10:
=2807 8A17                      MOV     DL,QTPORT[BX]           ; GET PORT ADDRESS
=2809 32F6                      XOR     DH,DH                   ; MAKE 16 BIT ADDR
=280B FC                        CLD                             ; SET FORWARD DIRECTION
=                       P232IP20:
=280C AC                        LODSB                           ; GET A BYTE
=280D 84C0                      TEST    AL,AL                   ; END OF SEQUENCE?
=280F 7501          2812        JNZ     P232IP30                ;  NO
=2811 C3                        RET                             ; EXIT IF END
=                       P232IP30:
=2812 EE                        OUT     DX,AL                   ; SEND IT TO PORT
=2813 EBF7          280C        JMPS    P232IP20                ; GET NEXT BYTE
=                       ;
=                       ;
=                       ;
=                       ; P232IN - READ A CHARACTER
=                       ;
=                       ; FUNCTION: RETURNS ONE BYTE FROM CIRCULAR BUFFER FOR A SPECIFIED
=                       ;  DEVICE.  WAITS IF THE BUFFER IS EMPTY.  IF XOFF HAS SUSPENDED INPUT,
=                       ;  AND THE BUFFER IS EMPTY, SEND AN XON.
=                       ;
=                       ; ENTRY:
=                       ;       BX = POINTER TO CONTROL BLOCK
=                       ;
=                       ; EXIT:
=                       ;       AL = CHARACTER
=                       ;
=                       P232IN:
=2815 8A4702                    MOV     AL,QTNRCHR[BX]          ; GET NUMBER OF CHARS. IN BUFFER
=2818 84C0                      TEST    AL,AL                   ; IS IT ZERO?
=281A 74F9          2815        JZ      P232IN                  ; LOOP BACK IF SO
=281C FA                        CLI                             ; DISABLE WHILE MANIPULATING BUFFER
=281D FE4F02                    DEC     QTNRCHR[BX]             ; DECREMENT CHARACTER COUNT
=2820 7526          2848        JNZ     P232IN1                 ; JUMP AHEAD IF NOT ZERO        **
=2822 FB                        STI                             ; TURN ON INTERRUPTS            **
=2823 2EF687010002              TEST    CS:QTFLAGS[BX],QMTYPE   ; DO WE SUPPORT XON?            **
```

```
=2829 751D          2848          JNZ     P232IN1                 ; NO. SKIP AHEAD                **
=282B 2EF687010008                EST     CS:QTFLAGS[BX],QMISUSP  ; YES. IS INPUT SUSPENDED?      **
=2831 7415          2848          JZ      P232IN1                 ; NO, BUFFER IS JUST EMPTY      **
=                           P232IN2:                              ;                               **
=2833 2EF687010001                TEST    CS:QTFLAGS[BX],QMSUSP   ; IS OUTPUT SUSPENDED?          **
=2839 75F8          2833          JNZ     P232IN2                 ; YES, WAIT FOR IT              **
=283B 2E80A70100F7                AND     CS:QTFLAGS[BX],NOT QMISUSP      ;INPUT NOT SUSPENDED    **
=2841 51                          PUSH    CX                      ; (JIC)                         **
=2842 B111                        MOV     CL,QKXON                ; OUTPUT AN ...                 **
=2844 E81900        2860          CALL    P232OUT                 ; XON.                          **
=2847 59                          POP     CX                      ;                               **
=                           P232IN1:                              ;                               **
=2848 FA                          CLI                             ; NO MORE INTERRUPTS            **
=2849 8A4705                      MOV     AL,QTOTPTR[BX]          ; GET OUTPUT POINTER
=284C 32E4                        XOR     AH,AH                   ; MAKE 16 BIT OFFSET
=284E 8BF0                        MOV     SI,AX                   ; MOVE TO INDEX REG.
=2850 FEC0                        INC     AL                      ; INCREMENT OUTPUT POINTER
=2852 3A4706                      CMP     AL,QTDEND[BX]           ; PAST END OF BUFFER?
=2855 7602          2859          JBE     P232IN4                 ;  NO
=2857 B00A                        MOV     AL,QTDATA               ; SET TO START OF BUFFER
=                           P232IN4:
=2859 884705                      MOV     QTOTPTR[BX],AL          ; STORE UPDATED POINTER
=285C 8A00                        MOV     AL,[BX+SI]              ; GET CHARACTER FROM BUFFER
=285E FB                          STI                             ; RE-ENABLE
=285F C3                          RET                             ; EXIT
=                           ;
=                           ;
=                           ; P232OUT - WRITE A CHARACTER
=                           ;
=                           ; FUNCTION: WRITES A CHARACTER TO A SPECIFIED DEVICE.
=                           ;
=                           ; ENTRY:
=                           ;       BX = POINTER TO CONTROL BLOCK
=                           ;       CL = CHARACTER TO BE WRITTEN
=                           ;
=                           ; EXIT: N/A
=                           ;
=                           P232OUT:
=2860 8AE1                        MOV     AH,CL                   ; SAVE THE CHARACTER
=                           P232OUT0:
=2862 8A17                        MOV     DL,QTPORT[BX]           ; GET PORT ADDRESS
=2864 32F6                        XOR     DH,DH                   ; MAKE 16 BIT ADDRESS
=2866 F6470102                    TEST    QTFLAGS[BX],QMTYPE      ; OK TO SEND XOFF?
=286A 7509          2875          JNZ     P232OUT1                ;  YES
=286C 8AC4                        MOV     AL,AH                   ; GET CHARACTER FOR TESTS
=286E 247F                        AND     AL,07FH                 ; STRIP PARITY
=2870 3C13                        CMP     AL,QKXOFF               ; IS IT XOFF?
=2872 7501          2875          JNE     P232OUT1                ;  NO - SEND IT
=2874 C3                          RET                             ; EXIT WITHOUT SENDING
=                           P232OUT1:
=2875 33C9                        XOR     CX,CX                   ; INIT TIMEOUT LOOP COUNTER
=                           P232OUT2:
```

```
=2877 EC                        IN      AL,DX                   ; GET PORT STATUS
=2878 A804                      TEST    AL,QMTXR                ; TX READY?
=287A 7522          289E        JNZ     P232OUT3                ;  YES - DO IT
=287C 51                        PUSH    CX                      ; THESE 2 INSTRUCTIONS ARE FILLERS
=287D 59                        POP     CX                      ;  TO MAKE A VALID TIMEOUT
=287E E2F7          2877        LOOP    P232OUT2                ; TRY IT AGAIN IF NO TIMEOUT YET
=2880 50                        PUSH    AX                      ; TIMEOUT - SOMETHING'S WRONG WITH
=2881 53                        PUSH    BX                      ;  UART
=2882 8B4707                    MOV     AX,QTDEVID[BX]          ; PUT DEVICE ID INTO MESSAGE
=2885 2EA35233                  MOV     P232TO1,AX
=2889 8A4709                    MOV     AL,QTDEVID+2[BX]
=288C 2EA25433                  MOV     P232TO2,AL
=2890 BB4533                    MOV     BX,OFFSET P232TO
=2893 E879FD        260F        CALL    PMSG                    ; WRITE TIMEOUT MESSAGE
=2896 E87306        2F0C        CALL    KQERY                   ; GET USER OPTION
=2899 5B                        POP     BX                      ;  SUBROUTINE WILL NOT RETURN IF CTL-C
=289A 58                        POP     AX
=289B 74C5          2862        JZ      P232OUT0                ; TRY AGAIN IF SPACE BAR
=289D C3                        RET                             ; PRETEND IT'S OK IF ANY OTHER KEY
=                           P232OUT3:
=289E F6470101                  TEST    QTFLAGS[BX],QMSUSP      ; IS OUTPUT SUSPENDED?
=28A2 75FA          289E        JNZ     P232OUT3                ;  YES - WAIT UNTIL ENABLED
=28A4 80EA02                    SUB     DL,2                    ; POINT TO DATA PORT
=28A7 8AC4                      MOV     AL,AH                   ; GET CHARACTER
=28A9 EE                        OUT     DX,AL                   ; SEND IT
=28AA C3                        RET                             ; EXIT
=                       ;
=                       ;
=                       ;
=                       ; P232STI  -  RETURN INPUT STATUS
=                       ;
=                       ; FUNCTION:  RETURNS A VALUE INDICATING WHETHER AN INPUT CHARACTER
=                       ;  IS AVAILABLE FROM A SPECIFIED DEVICE.
=                       ;
=                       ; ENTRY:
=                       ;       BX = POINTER TO CONTROL BLOCK
=                       ;
=                       ; EXIT:
=                       ;       AL = 0     IF NO CHARACTER IS READY (BUFFER EMPTY)
=                       ;       AL = X'FF' IF ONE OR MORE CHAPACTERS ARE READY
=                       ;
=                           P232STI:
=28AB 8A4702                    MOV     AL,QTNRCHR[BX]          ; GET CURRENT NR OF CHARS.
=28AE 84C0                      TEST    AL,AL                   ; IS IT ZERO?
=28B0 7501          28B3        JNZ     P232STI1                ;  NO
=28B2 C3                        RET                             ; RETURN "NOT READY"
=                           P232STI1:
=28B3 B0FF                      MOV     AL,0FFH                 ; SET "READY"
=28B5 C3                        RET
=                       ;
=                       ;
=                       ;
```

```
=                       ; P232STO  -  RETURN OUTPUT STATUS
=                       ;
=                       ; FUNCTION: RETURNS A VALUE INDICATING WHETHER THE SPECIFIED DEVICE IS
=                       ;  READY TO ACCEPT AN OUTPUT CHARACTER.
=                       ;
=                       ; ENTRY:
=                       ;       BX = POINTER TO CONTROL BLOCK
=                       ;
=                       ; EXIT:
=                       ;       AL = 0      IF DEVICE IS NOT READY TO ACCEPT A CHARACTER
=                       ;       AL = X'FF'  IF DEVICE IS READY TO ACCEPT A CHARACTER
=                       ;
=                       P232STO:
=28B6 8A17                      MOV     DL,QTPORT[BX]           ; GET PORT ADDRESS
=28B8 32F6                      XOR     DH,DH                   ; MAKE 16 BIT ADDRESS
=28BA EC                        IN      AL,DX                   ; GET PORT STATUS
=28BB 2404                      AND     AL,QMTXR                ; IS TX READY?
=28BD 7501              28C0    JNZ     P232STO1                ;  YES
=28BF C3                        RET                             ; RETURN "NOT READY"
=                       P232STO1:
=28C0 32C0                      XOR     AL,AL                   ; SET "NOT READY" VALUE
=28C2 F6470101                  TEST    QTFLAGS[BX],QMSUSP      ; IS OUTPUT SUSPENDED?
=28C6 7401              28C9    JZ      P232STO3                ;  NO
=28C8 C3                        RET                             ; RETURN "NOT READY"
=                       P232STO3:
=28C9 F6D0                      NOT     AL                      ; SET "READY"
=28CB C3                        RET
=                       ;
=                       ;
=                       ;
=                       ; I232RX  -  HANDLE SIO RECEIVE INTERRUPTS
=                       ;
=                       ; FUNCTION:  PROCESS RECEIVE INTERRUPTS FROM ONE OR MORE SIO PORTS.
=                       ;  SET CONTROL BLOCK POINTERS AND CALL PORT INTERRUPT PROCESSOR.
=                       ;
=                       ; ENTRY AND EXIT PARAMETERS: N/A
=                       ;
=                       I232RX:
=28CC 50                        PUSH    AX                      ; SAVE REGISTERS
=28CD 53                        PUSH    BX
=28CE 52                        PUSH    DX
=                       ; POINT TO EACH CONTROL BLOCK AND CHECK RECEIVER
=28CF BB1F27                    MOV     BX,OFFSET TPRTCB        ; PRINTER CONTROL BLOCK
=28D2 E81D00            28F2    CALL    I232RPT
=28D5 BB4927                    MOV     BX,OFFSET TCOMCB        ; COMM PORT CONTROL BLOCK
=                       I232RXX:
=28D8 E81700            28F2    CALL    I232RPT
=                       ; SEND END-OF-INTERRUPT
=28DB 2E8A970000                MOV     DL,CS:QTPORT[BX]
=28E0 80E2FE                    AND     DL,0FEH
=28E3 B038                      MOV     AL,QKEOI
=28E5 EE                        OUT     DX,AL
```

```
=                       ; RESTORE REGISTERS AND EXIT
=28E6 5A                        POP     DX
=28E7 5B                        POP     BX
=28E8 58                        POP     AX
=28E9 CF                        IRET
=                       ;
=                       ;
=                       ;
=                       ; I232RX2  -  HANDLE OPTIONAL SIO RECEIVE INTERRUPTS
=                       ;
=                       ; FUNCTION:  PROCESS RECEIVE INTERRUPTS FROM ONE OR MORE SIO PORTS.
=                       ;  SET CONTROL BLOCK POINTERS AND CALL PORT INTERRUPT PROCESSOR.
=                       ;
=                       ; ENTRY AND EXIT PARAMETERS: N/A
=                       ;
=                       I232RX2:
=28EA 50                        PUSH    AX                      ; SAVE REGISTERS
=28EB 53                        PUSH    BX
=28EC 52                        PUSH    DX
=                       ; POINT TO EACH CONTROL BLOCK AND CHECK RECEIVER
=28ED BB7327                    MOV     BX,OFFSET TCOM2CB       ; COMM PORT CONTROL BLOCK
=28F0 EBE6          28D8        JMPS    I232RXX                 ;SAVE REGISTERS AND EXIT
=                       ;
=                       ;
=                       ;
=                       ; I232RPT  -  PROCESS RECEIVED CHARACTER
=                       ;
=                       ; FUNCTION:  IF A CHARACTER HAS BEEN RECEIVED AT A PORT, READS IT
=                       ;  AND PLACES IT IN A CIRCULAR BUFFER.  IF REQUIRED, PROCESSES
=                       ;  XON/XOFF PROTOCOL AND SUBSTITUTES A 'SUB' CHARACTER WHEN A
=                       ;  PARITY ERROR OCCURS.
=                       ;
=                       ; ENTRY:
=                       ;       BX = POINTER TO CONTROL BLOCK
=                       ;
=                       ; EXIT:  N/A
=                       ;
=                       ; REGISTER USE:
=                       ;       AL      PORT INPUT/OUTPUT
=                       ;       AH      ERROR STATUS STORAGE
=                       ;       BX      CONTROL BLOCK POINTER
=                       ;       DX      PORT ADDRESS AND INPUT POINTER WORKING REG.
=                       ;
=                       I232RPT:
=28F2 2E8A970000                MOV     DL,CS:QTPORT[BX]        ; GET PORT ADDRESS
=28F7 32F6                      XOR     DH,DH                   ; MAKE 16 BIT ADDRESS
=28F9 EC                        IN      AL,DX                   ; GET PORT STATUS
=28FA A801                      TEST    AL,QMRXR                ; RECEIVE READY?
=28FC 7501          28FF        JNZ     I232RP05                ;  YES
=                       I232RPTX:
=28FE C3                        RET                             ; EXIT - NO PROCESSING NECESSARY
=                       I232RP05:
```

```
=28FF A880                      TEST    AL,QMBRK                ; BREAK?
=2901 7409          290C        JZ      I232RP10                ;
=2903 2E808F010010              OR      CS: QTFLAGS[BX],QMBREAK ; SIGNAL IT
=2909 E9D100        29DD        JMP     I232RP40                ;
=                           ; PROCESS A RECEIVED CHARACTER
=                           I232RP10:
=290C 2EF687010010              TEST    CS:QTFLAGS[BX],QMBREAK  ; DO WE HAVE A BREAK?
=2912 7409          291D        JZ      I232RP11                ; SKIP IF NOT
=2914 2E80A70100EF              AND     CS:QTFLAGS[BX],NOT QMBREAK      ;ELSE CLEAR FLAG
=291A E9C000        29DD        JMP     I232RP40                ; AND GO AWAY
=                           I232RP11:
=291D B001                      MOV     AL,1                    ; POINT TO RR1
=291F EE                        OUT     DX,AL                   ; WRITE TO SIO
=2920 EC                        IN      AL,DX                   ; READ RR1
=2921 8AE0                      MOV     AH,AL                   ; SAVE IT
=2923 B030                      MOV     AL,QKRESERR             ; RESET SIO ERRORS (IF ANY)
=2925 EE                        OUT     DX,AL
=2926 80EA02                    SUB     DL,2                    ; POINT TO DATA PORT
=2929 EC                        IN      AL,DX                   ; READ RECEIVED CHARACTER
=                           ; CHECK FOR RECEIVE ERRORS
=292A F6C420                    TEST    AH,QMOVRE               ; OVERRUN ERROR?
=292D 7406          2935        JZ      I232RP12                ;  NO
=292F 2E80A70100FE              AND     CS:QTFLAGS[BX],NOT QMSUSP ; ENABLE OUTPUT (ASSUME XON LOST)
=                           I232RP12:
=2935 F6C410                    TEST    AH,QMPARE               ; PARITY ERROR?
=2938 7402          293C        JZ      I232RP14                ;  NO
=293A B01A                      MOV     AL,QKSUB                ; CHANGE RX CHAR. TO 'SUB'
=                           I232RP14:
=293C 2EF687010002              TEST    CS:QTFLAGS[BX],QMTYPE   ; XON/XOFF APPLICABLE?
=2942 7530          2974        JNZ     I232RP20                ;  NO - STORE CHARACTER
=                           ; PROCESS XON/XOFF PROTOCOL
=2944 84C0                      TEST    AL,AL                   ; CHECK FOR NULL
=2946 7501          2949        JNZ     I232RP15                ; NOT A NULL
=2948 C3                        RET                             ; IGNORE NULL CHARACTERS
=                           I232RP15:
=2949 3C93                      CMP     AL,QKXOFFP              ; IS IT XOFF+?
=294B 7404          2951        JZ      I232RP15A               ;  YES
=294D 3C13                      CMP     AL,QKXOFF               ; IS IT XOFF?
=294F 7507          2958        JNE     I232RP16                ;  NO
=                           I232RP15A:
=2951 2E808F010001              OR      CS:QTFLAGS[BX],QMSUSP   ; SUSPEND OUTPUT
=2957 C3                        RET                             ;  AND EXIT
=                           I232RP16:
=2958 3C91                      CMP     AL,QKXONP               ; IS IT XON?
=295A 7404          2960        JZ      I232RP16A               ;  NO - STORE IT
=295C 3C11                      CMP     AL,QKXON                ; IS IT XON?
=295E 7514          2974        JNE     I232RP20                ;  NO - STORE IT
=                           I232RP16A:
=2960 F6C420                    TEST    AH,QMOVRE               ; WAS THERE AN OVERRUN ERROR?
=2963 7508          296D        JNZ     I232RP18                ;  YES - ASSUME AN XOFF WAS LOST
=2965 2EF687010001              TEST    CS:QTFLAGS[BX],QMSUSP   ; IS OUTPUT SUSPENDED?
=296B 7407          2974        JZ      I232RP20                ;  NO - STORE XON IN BUFFER
```

```
=                       I232RP18:
=296D 2E80A70100FE              AND     CS:QTFLAGS[BX],NOT QMSUSP       ; ENABLE OUTPUT
=2973 C3                        RET                                     ;  AND EXIT
=                       ; STORE CHARACTER IN BUFFER
=                       I232RP20:
=2974 2E8A970200                MOV     DL,CS:QTNRCHR[BX]       ; GET NR. OF CHARS. IN BUFFER
=2979 2E3A970300                CMP     DL,CS:QTCAP[BX]         ; IS BUFFER FULL?
=297E 744C          29CC        JE      I232RP30                ;  YES
=2980 2EFE870200                INC     CS:QTNRCHR[BX]          ; INCREMENT CHARACTER COUNT
=2985 2E8A970400                MOV     DL,CS:QTINPTR[BX]       ; GET INPUT POINTER
=298A 2EFE870400                INC     CS:QTINPTR[BX]          ; INCREMENT POINTER
=298F 2E3A970600                CMP     DL,CS:QTDEND[BX]        ; PAST END OF BUFFER?
=2994 7206          299C        JB      I232RP26                ;  NO
=2996 2EC68704000A              MOV     CS:QTINPTR[BX],QTDATA   ; SET TO START OF BUFFER
=                       I232RP26:
=299C 53                        PUSH    BX                      ; NEED TO SAVE POINTER          **
=299D 03DA                      ADD     BX,DX                   ; STORE UPDATED POINTER
=299F 2E88870000                MOV     CS:[BX],AL              ; STORE CHARACTER
=                       ; NOW SEE IF BUFFER IS MORE THAN HALF FULL AND IF XOFF/XON              **
=                       ; IS SUPPORTED.  IF BOTH, SEND AN XOFF.                                 **
=29A4 5B                        POP     BX                      ;                               **
=29A5 2EF687010002              TEST    CS:QTFLAGS[BX],QMTYPE   ; XOFF SUPPORTED?               **
=29AB 751E          29CB        JNZ     I232RP28                ; GO AWAY IF NOT                **
=29AD 2E8A970200                MOV     DL,CS:QTNRCHR[BX]       ; GET NO. OF CHARACTERS         **
=29B2 02D2                      ADD     DL,DL                   ; DOUBLE IT                     **
=29B4 2E3A970300                CMP     DL,CS:QTCAP[BX]         ; MORE THAN HALF-FULL?          **
=29B9 7610          29CB        JBE     I232RP28                ; GO AWAY IF NOT                **
=                       I232RP27:                               ; OUTPUT A XOFF                 **
=29BB 2EF687010008              TEST    CS:QTFLAGS[BX],QMISUSP  ; UNLESS INPUT IS ALREADY SUSP- **
=29C1 7508          29CB        JNZ     I232RP28                ; ENDED.                        **
=29C3 2E808F010008              OR      CS:QTFLAGS[BX],QMISUSP  ; SUSPEND INPUT                 **
=29C9 EB01          29CC        JMPS    I232RP32                ; OUTPUT IT                     **
=                       I232RP28:                               ;                               **
=29CB C3                        RET                             ;  AND EXIT
=                       ;
=                       ; ERROR CONDITION ROUTINES
=                       ;
=                       ; BUFFER IS FULL, OR HALF-FULL, OR SOMETHING
=                       ;
=                       I232RP30:
=                       I232RP32:                               ; ENTER HERE FOR XOFF           **
=29CC 2E8A970000                MOV     DL,CS:QTPORT[BX]
=                       I232RP33:
=29D1 EC                        IN      AL,DX                   ;
=29D2 A804                      TEST    AL,QMTXR                ;
=29D4 74FB          29D1        JZ      I232RP33                ;
=29D6 B013                      MOV     AL,QKXOFF               ;
=29D8 80EA02                    SUB     DL,2
=29DB EE                        OUT     DX,AL
=29DC C3                        RET                             ;  AND EXIT
=                       ;
=                       ; BREAK CONDITION
```

```
=                       ;
=                       I232RP40:
=29DD B010                      MOV     AL,QKRESI               ; RESET BREAK CONDITION
=29DF EE                        OUT     DX,AL
=29E0 B030                      MOV     AL,QKRESERR             ; RESET ANY OTHER ERRORS
=29E2 EE                        OUT     DX,AL
=29E3 80EA02                    SUB     DL,2                    ; POINT TO DATA PORT
=29E6 EC                        IN      AL,DX                   ; CLEAR INPUT
=29E7 C3                        RET
=                       ;
=                       ; ************************************************************************
=                       ;
=                       ;               PHYSICAL DEVICE DRIVERS
=                       ;
=                       ; ************************************************************************;
=                       ;
=                       ;
=                       ; PTR: AND PTP: DRIVERS
=                       ;
=                       PTRIN:
=29E8 BB4927                    MOV     BX,OFFSET TCOMCB        ; POINT TO CONTROL BLOCK
=29EB E927FE       2815         JMP     P232IN
=                       ;
=                       PTPOUT:
=29EE BB4927                    MOV     BX,OFFSET TCOMCB        ; POINT TO CONTROL BLOCK
=29F1 E96CFE       2860         JMP     P232OUT
=                       ;
=                       PTRST:
=29F4 BB4927                    MOV     BX,OFFSET TCOMCB        ; POINT TO CONTROL BLOCK
=29F7 E9B1FE       28AB         JMP     P232STI
=                       ;
=                       ;
=                       ; TTY: DRIVERS
=                       ;
=                       TTYIN:
=29FA BB1F27                    MOV     BX,OFFSET TPRTCB        ; POINT TO CONTROL BLOCK
=29FD E915FE       2815         JMP     P232IN
=                       ;
=                       TTYOUT:
=2A00 BB1F27                    MOV     BX,OFFSET TPRTCB        ; POINT TO CONTROL BLOCK
=2A03 E95AFE       2860         JMP     P232OUT
=                       ;
=                       TTYSTI:
=2A06 BB1F27                    MOV     BX,OFFSET TPRTCB        ; POINT TO CONTROL BLOCK
=2A09 E99FFE       28AB         JMP     P232STI
=                       ;
=                       TTYSTO:
=2A0C BB1F27                    MOV     BX,OFFSET TPRTCB        ; POINT TO CONTROL BLOCK
=2A0F E9A4FE       28B6         JMP     P232STO
=                       ;
=                       ;
=                       ;
```

```
=                       ; NULL DEVICE DRIVERS
=                       ;
=  001A                 QKCTLZ  EQU     26              ; END OF FILE (CTL-Z)
=                       ;
=                       ;
=                       PNULIN:
=2A12 B01A                      MOV     AL,QKCTLZ       ; INDICATE END OF FILE
=2A14 C3                        RET
=                       ;
=                       PNULOUT:
=2A15 C3                        RET                     ; ACCEPT ALL CHARACTERS
=                       ;
=                       PNULSTI:
=2A16 B0FF                      MOV     AL,0FFH         ; INDICATE DEVICE READY
=2A18 C3                        RET
=                       ;
=  2A16                 PNULSTO EQU     PNULSTI
=                       ;
=                       ;
=                       ; BATCH drivers
=                       ;
=  29F4                 BATST   EQU     PTRST
=  29E8                 BATIN   EQU     PTRIN
=  26D5                 BATOUT  EQU     CRTOUT
=                       ;
=                       ;
=                       ; LPT: DRIVERS
=                       ;
=  2A00                 LPTOUT  EQU     TTYOUT
=  2A0C                 LPTSTO  EQU     TTYSTO
=                       ;
=                       ;
=                       ; UC1: DRIVERS
=                       ;
=                       UC1IN:
=2A19 BB7327                    MOV     BX,OFFSET TCOM2CB       ; POINT TO CONTROL BLOCK
=2A1C E9F6FD       2815         JMP     P232IN
=                       ;
=                       UC1OUT:
=2A1F BB7327                    MOV     BX,OFFSET TCOM2CB       ; POINT TO CONTROL BLOCK
=2A22 E93BFE       2860         JMP     P232OUT
=                       ;
=                       UC1ST:
=2A25 BB7327                    MOV     BX,OFFSET TCOM2CB       ; POINT TO CONTROL BLOCK
=2A28 E980FE       28AB         JMP     P232STI
=                       ;
=                       ;
=                       ;
=                       ; UR1: DRIVERS
=                       ;
=  2A19                 UR1IN   EQU     UC1IN
=  2A25                 UR1ST   EQU     UC1ST
```

```
=                       ;
=                       ; UR2: DRIVERS
=                       ;
=  2A12                 UR2IN   EQU     PNULIN
=  2A16                 UR2ST   EQU     PNULSTI
=                       ;
=                       ; UP1: AND UP2: DRIVERS
=                       ;
=  2A1F                 UP1OUT  EQU     UC1OUT
=  2A15                 UP2OUT  EQU     PNULOUT
=                       ;
=                       ; UL1: DRIVERS
=                       ;
=  2A15                 UL1OUT  EQU     PNULOUT
=  2A16                 UL1ST   EQU     PNULSTO
=                       ;
=                       ;
=
=2A2B 2EFF264E2A        CONST:  jmp word ptr constjmp           ;console status
=
=                       CONIN:                  ;console input
=2A30 2EFF26502A                jmp word ptr coninjmp
=
=2A35 2EFF26522A        CONOUT: jmp word ptr conoutjmp          ;console output
=
=2A3A 2EFF26582A        LISTST: jmp word ptr liststjmp          ;list device status
=
=2A3F 2EFF265A2A        LISTOUT:jmp word ptr listoutjmp         ;output to list device
=
=2A44 2EFF26562A        PUNCH:  jmp word ptr punchjmp           ;output to punch
=
=2A49 2EFF26542A        READER: jmp word ptr readerjmp          ;input from reader
=
=                       ;Indirect jump table for I/O
=2A4E 1227              constjmp        dw      CRTSTI
=2A50 DF26              coninjmp        dw      CRTIN
=2A52 D526              conoutjmp       dw      CRTOUT
=2A54 E829              readerjmp       dw      PTRIN
=2A56 EE29              punchjmp        dw      PTPOUT
=2A58 0C2A              liststjmp       dw      LPTSTO
=2A5A 002A              listoutjmp      dw      LPTOUT
=
=
=
=
=
=                       GETIOBF:
=2A5C 2EA00232                  mov al,IOBYTE
=2A60 C3                        ret
=
=                       SETIOBF:
=2A61 2E880E0232                mov IOBYTE,cl   ;set iobyte
```

```
=                       IF      NOT LOADER_BIOS
=                       ;-----------------------------------------------;
=                       ;|                                              |
=2A66 BF4E2A                    mov di,offset constjmp
=2A69 BE8E2A                    mov si,offset iojtbl
=2A6C E85700        2AC6        call cioset                     ;low two bits
=2A6F E86000        2AD2        call iojset                     ;const
=2A72 E85D00        2AD2        call iojset                     ;conin
=2A75 E85A00        2AD2        call iojset                     ;conout
=2A78 E84B00        2AC6        call cioset                     ;second two bits
=2A7B E85400        2AD2        call iojset                     ;reader
=2A7E E84500        2AC6        call cioset                     ;third two bits
=2A81 E84E00        2AD2        call iojset                     ;punch
=2A84 E83F00        2AC6        call cioset                     ;high two bits
=2A87 E84800        2AD2        call iojset                     ;listst
=2A8A E84500        2AD2        call iojset                     ;listout
=2A8D C3                        ret
=
=2A8E 062A              iojtbl  dw      TTYSTI  ;CONST
=2A90 1227                      dw      CRTSTI
=2A92 F429                      dw      BATST
=2A94 252A                      dw      UC1ST
=
=2A96 FA29                      dw      TTYIN   ;CONIN
=2A98 DF26                      dw      CRTIN
=2A9A E829                      dw      BATIN
=2A9C 192A                      dw      UC1IN
=
=2A9E 002A                      dw      TTYOUT  ;CONOUT
=2AA0 D526                      dw      CRTOUT
=2AA2 D526                      dw      BATOUT
=2AA4 1F2A                      dw      UC1OUT
=
=2AA6 FA29                      dw      TTYIN   ;READER
=2AA8 E829                      dw      PTRIN
=2AAA 192A                      dw      UR1IN
=2AAC 122A                      dw      UR2IN
=
=2AAE 002A                      dw      TTYOUT  ;PUNCH
=2AB0 EE29                      dw      PTPOUT
=2AB2 1F2A                      dw      UP1OUT
=2AB4 152A                      dw      UP2OUT
=
=2AB6 0C2A                      dw      TTYSTO  ;LISTST
=2AB8 1C27                      dw      CRTSTO
=2ABA 0C2A                      dw      LPTSTO
=2ABC 162A                      dw      UL1ST
=
=2ABE 002A                      dw      TTYOUT  ;LPTOUT
=2AC0 D526                      dw      CRTOUT
=2AC2 002A                      dw      LPTOUT
=2AC4 152A                      dw      UL1OUT
```

```
=
=
=
=                       cioset: ;Set the offsets in bx according to the low two bits in cl
=2AC6 BB0300                    mov bx,3        ;and then rotate cl twice
=2AC9 22D9                      and bl,cl       ;
=2ACB 03DB                      add bx,bx       ;twice the number for offset
=2ACD D0E9                      shr cl,1        ;shift cl right ...
=2ACF D0E9                      shr cl,1        ;twice
=2AD1 C3                        ret
=
=                       iojset: ;Move the appropriate entry from the jump list to the indirect
=2AD2 8B00                      mov cs:ax,[bx+si]       ;jump table and then increment the pointers
=2AD4 2E89850000                mov cs:[di],ax          ;for the next call
=2AD9 83C608                    add si,8                ;
=2ADC 83C702                    add di,2                ;
=                       ;|                                                      |
=                       ;|------------------------------------------------------|
=                       ENDIF   ;not loader_bios
=2ADF C3                        ret
=
=                       GETSEGT:  ;return address of physical memory table
=2AE0 BB5E33                    mov bx,offset seg_table
=2AE3 C3                        ret
=
=                       ;
=                       VIDEO:  ;Output video directly to PC-100 via int 40
=2AE4 1E                        push ds
=2AE5 06                        push es
=2AE6 51                        push cx
=2AE7 52                        push dx
=2AE8 BF0800                    mov di,8
=2AEB CD28                      int 40          ; turn cursor off
=2AED 5D                        pop bp
=2AEE 8EDD                      mov ds,bp
=2AF0 5E                        pop si
=2AF1 8B04                      mov ax,[si]
=2AF3 8B5C02                    mov bx,2[si]
=2AF6 8B4C04                    mov cx,4[si]
=2AF9 8B5406                    mov dx,6[si]
=2AFC 8B7408                    mov si,8[si]
=2AFF BF1400                    mov di,14h
=2B02 CD28                      int 40          ; move in the video
=2B04 BF0A00                    mov di,0ah
=2B07 CD28                      int 40          ; turn cursor on
=2B09 07                        pop es
=2B0A 1F                        pop ds
=2B0B C3                        ret
=                       ;
=                               INCLUDE CPLBLOK.LIB
=                       ;       SECTOR BLOCKING/DEBLOCKING
```

```
=
=                       ;       Modified for CP/M 86/80
=                       ;       May 1982  by CPL (RK)
=
=                       ;*****************************************************
=                       ;*                                                   *
=                       ;*         CP/M to host disk constants               *
=                       ;*                                                   *
=                       ;*****************************************************
=  0000                 una     equ     byte ptr [BX]   ;name for byte at BX
=                       ;
=  0004                 nrdisks equ     4               ;four disks allowed
=  0800                 blksiz  equ     2048            ;CP/M allocation size
=  0200                 hstsiz  equ     512             ;host disk sector size
=  000A                 hstspt  equ     10              ;host disk sectors/trk
=  0004                 hstblk  equ     hstsiz/128      ;CP/M sects/host buff
=  FD00                 xfrbuf  equ     pb2_adr+xdefbuf ;buffer in shared memory
=                       ;
=                       ;*****************************************************
=                       ;*                                                   *
=                       ;* secshf is log2(hstblk), and is listed below for   *
=                       ;* values of hstsiz up to 2048.                      *
=                       ;*                                                   *
=                       ;*           hstsiz    hstblk    secshf              *
=                       ;*              256         2         1              *
=                       ;*              512         4         2              *
=                       ;*             1024         8         3              *
=                       ;*             2048        16         4              *
=                       ;*                                                   *
=                       ;*****************************************************
=  0002                 secshf  equ     2               ;log2(hstblk)
=  0028                 cpmspt  equ     hstblk * hstspt ;CP/M sectors/track
=  0003                 secmsk  equ     hstblk-1        ;sector mask
=                       ;
=                       ;*****************************************************
=                       ;*                                                   *
=                       ;*        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
=                       ;
=                       ;*************************************************
=                       ;*                                               *
=                       ;*      BDOS function table beginning            *
=                       ;*                                               *
=                       ;*************************************************
=                               if not loader_bios
=                       ;_____________________________________________________;
=  0A80                 bdos_ftbl       equ     0a80h
=  22EA                 bdos_dlog       equ     22eah
```

```
=                       ;_____________________________________________________;
=                               endif   ;not loader_bios
=                       ;
=                       ;*****************************************************
=                       ;*                                                   *
=                       ;*      The BIOS entry points given below show the   *
=                       ;*      code which is relevant to deblocking only.   *
=                       ;*                                                   *
=                       ;*****************************************************
=                       seldsk:
=2B0C 80F904                    cmp cl,nrdisks          ;valid disk number?
=2B0F 7204          2B15        jb seldsk1              ;go ahead if ok
=2B11 BB0000                    mov bx,0                ;else zero bx
=2B14 C3                        ret                     ;and let bdos take care of it
=                               ;select disk
=                               ;is this the first activation of the drive?
=                       seldsk1:
=2B15 2EA0FE2F                  mov al,hstdsk           ;which disk?
=2B19 3CFF                      cmp al,0ffh             ;absolutely first activation?
=2B1B 7409          2B26        je seldsk2              ;go clear host buffer
=2B1D F6C201                    test DL,1               ;lsb = 0?
=2B20 7510          2B32        jnz selset
=                               ;if this is the first activation, clear host buff
=2B22 3AC1                      cmp al,cl               ;but is this the same disk?
=2B24 750C          2B32        jne selset              ;if not, don't
=                       seldsk2:
=2B26 2EC606033000              mov hstact,0
=2B2C 2EC606053000              mov unacnt,0
=                       selset:
=2B32 8AC198                    mov al,cl ! cbw         ;put in AX
=2B35 2EA2FA2F                  mov sekdsk,al           ;seek disk number
=2B39 B104D2E0                  mov cl,4 ! shl al,cl    ;times 16
=2B3D 051330                    add ax,offset dpbase
=2B40 8BD8                      mov bx,ax
=                       if not loader bios
=                       ;_____________________________________________________;
=                       ;check for Robin media on any unallocated disk called
=2B42 52                        push dx                 ;save dx and bx
=2B43 53                        push bx
=2B44 8B1EEA22                  mov bx,word ptr .bdos_dlog      ;get allocation vector
=2B48 0BDB                      or bx,bx                ;any allocated?
=2B4A 7420          2B6C        jz seld1                ;skip ahead if none
=2B4C 5B                        pop bx                  ;else find out ...
=2B4D 53                        push bx                 ;
=2B4E 83C30A                    add bx,0ah              ;if this disk has been checked
=2B51 8B1F                      mov bx,[bx]             ;get dpb offset
=2B53 0BDB                      or bx,bx                ;zero? not checked yet
=2B55 7424          2B7B        jz seld2                ;go check it if so
=2B57 2EC606123000              mov mediatype,0         ;assume Rainbow
=2B5D 81FB5330                  cmp bx,dpb0             ;is it really?
=2B61 7406          2B69        jz seld0                ;move on if so
=2B63 2EC606123002              mov mediatype,2         ;else mark as Robin
```

```
=2B69 5B                seld0:  pop bx                  ;then ...
=2B6A 5A                        pop dx                  ;
=2B6B C3                        ret                     ;return
=
=2B6C 33C0              seldl:  xor ax,ax               ;zero all dpb's
=2B6E BB1D30                    mov bx,offset dpbase+0ah        ;dpb offset from base
=2B71 B90400                    mov cx,nrdisks          ;loop for all disks
=
=2B74 8907              seldll: mov [bx],ax             ;zero a dpb
=2B76 83C310                    add bx,10h              ;get the next one
=2B79 E2F9          2B74        loop seldll             ;loop till done
=
=2B7B B81500            seld2:  mov ax,15h              ;get media type
=2B7E 50                        push ax                 ;via packer
=2B7F 2EA0FA2F                  mov al,sekdsk           ;get disk number back again
=2B83 B105                      mov cl,5                ; and put in proper
=2B85 D3C0                      rol ax,cl               ;  position for packet
=2B87 50                        push ax                 ;
=2B88 B90200                    mov cx,2                ;move 2 words
=2B8B E827FA        25B5        call packer
=2B8E 58                        pop ax                  ;discard drive number
=2B8F 58                        pop ax                  ;ah has status/media type
=2B90 33DB                      xor bx,bx               ;clean out a register
=2B92 2E88261230                mov mediatype,ah        ;mark the media type
=2B97 8ADC                      mov bl,ah               ;get the type (Rainbow = 0, robin = 2
=2B99 81C39930                  add bx,mediatbl         ;add the table address
=2B9D 8B17                      mov dx,[bx]             ;get contents of table
=2B9F 5B                        pop bx                  ;
=2BA0 53                        push bx                 ;get dpb pointer
=2BA1 83C30A                    add bx,0ah              ;
=2BA4 8917                      mov [bx],dx             ;put address ilnto pointer
=2BA6 5B                        pop bx                  ;get return data
=2BA7 5A                        pop dx                  ;
=                       ;_______________________________________________________;
=                               endif   ;not loader_bios
=2BA8 C3                        ret
=                       ;
=                       home:
=                               ;home the selected disk
=2BA9 2EA00430                  mov al,hstwrt           ;check for pending write
=2BAD 84C0                      test al,al
=2BAF 7506          2BB7        jnz homed
=2BB1 2EC606033000              mov hstact,0            ;clear host active flag
=                       homed:
=2BB7 B90000                    mov cx,0                ;now, set track zero
=                       ;       (continue HOME routine)
=                       ;
=                       settrk:
=                               ;set track given by registers CX
=2BBA 2E890EFB2F                mov sektrk,CX           ;track to seek
=2BBF C3                        ret
=                       ;
```

```
=                       setsec:
=                               ;set sector given by register cl
=2BC0 2E880EFD2F                mov seksec,cl           ;sector to seek
=2BC5 C3                        ret
=                       ;
=                       setdma:
=                               ;set dma address given by CX
=2BC6 2E890E1030                mov dma_off,CX
=2BCB C3                        ret
=                       ;
=                       setdmab:
=                               ;set segment address given by CX
=2BCC 2E890E0E30                mov dma_seg,CX
=2BD1 C3                        ret
=                       ;
=                       sectran:
=                               ;translate sector number CX with table at [DX]
=2BD2 85D2                      test DX,DX      ;test for hard skewed
=2BD4 7409          2BDF        jz notran       ;(blocked must be hard skewed)
=2BD6 8BD9                      mov BX,CX
=2BD8 03DA                      add BX,DX
=2BDA 8A1F                      mov BL,[BX]
=2BDC B700                      mov BH,0        ;*** be compatible with 86/80  (fix 5/21/82) ***
=2BDE C3                        ret
=                       no_tran:
=                               ;hard skewed disk, physical = logical sector
=2BDF 8BD9                      mov BX,CX
=2BE1 C3                        ret
=                       ;
=                       read:
=                               ;read the selected CP/M sector
=2BE2 2EC606053000              mov unacnt,0            ;clear unallocated counter
=2BE8 2EC6060C3001              mov readop,1            ;read operation
=2BEE 2EC6060B3001              mov rsflag,1            ;must read data
=2BF4 2EC6060D3002              mov wrtype,wrual        ;treat as unalloc
=2BFA E9A800        2CA5        jmp rwoper              ;to perform the read
=                       ;
=                       write:
=                               ;write the selected CP/M sector
=                               if not loader_bios
=                       ;-----------------------------------------------;
=2BFD 2EF6061230FF              test mediatype,0ffh     ;is this Rainbow media?
=2C03 7413          2C18        jz okwrite              ;go write if so
=2C05 BB0B33                    mov bx,offset unabl     ;'unable to write...'
=2C08 E804FA        260F        call pmsg               ;print it
=2C0B E89B03        2FA9        call prthst             ;print drive no.
=2C0E BB5432                    mov bx,offset endlin    ;
=2C11 E8FBF9        260F        call pmsg               ;print cr,lf.
=2C14 B80100                    mov ax,1                ;mark for error
=2C17 C3                        ret                     ;return
=                       okwrite:
=                       ;                                               ;
```

```
=                               endif   ;not loader_bios
=2C18 2EC6060C3000              mov readop,0            ;write operation
=2C1E 2E880E0D30                mov wrtype,cl
=2C23 80F902                    cmp cl,wrual            ;write unallocated?
=2C26 751E          2C46        jnz chkuna              ;check for unalloc
=                           ;
=                           ;       write to unallocated, set parameters
=                           ;
=2C28 2EC606053010              mov unacnt,(blksiz/128) ;next unalloc recs
=2C2E 2EA0FA2F                  mov al,sekdsk           ;disk to seek
=2C32 2EA20630                  mov unadsk,al           ;unadsk = sekdsk
=2C36 2EA1FB2F                  mov ax,sektrk
=2C3A 2EA30730                  mov unatrk,ax           ;unatrk = sektrk
=2C3E 2EA0FD2F                  mov al,seksec
=2C42 2EA20930                  mov unasec,al           ;unasec = seksec
=                           ;
=                           chkuna:
=                               ;check for write to unallocated sector
=                           ;
=2C46 BB0530                    mov bx,offset unacnt    ;point "UNA" at UNACNT
=2C49 8A0784C0                  mov al,una ! test al,al ;any unalloc remain?
=2C4D 744A          2C99        jz alloc                ;skip if not
=                           ;
=                           ;       more unallocated records remain
=2C4F FEC8                      dec al                  ;unacnt = unacnt-1
=2C51 8807                      mov una,al
=2C53 2EA0FA2F                  mov al,sekdsk           ;same disk?
=2C57 BB0630                    mov BX,offset unadsk
=2C5A 3A07                      cmp al,una              ;sekdsk = unadsk?
=2C5C 753B          2C99        jnz alloc               ;skip if not
=                           ;
=                           ;       disks are the same
=2C5E 2EA10730                  mov AX, unatrk
=2C62 2E3B06FB2F                cmp AX, sektrk
=2C67 7530          2C99        jnz alloc               ;skip if not
=                           ;
=                           ;       tracks are the same
=2C69 2EA0FD2F                  mov al,seksec           ;same sector?
=                           ;
=2C6D BB0930                    mov BX,offset unasec    ;point una at unasec
=                           ;
=2C70 3A07                      cmp al,una              ;seksec = unasec?
=2C72 7525          2C99        jnz alloc               ;skip if not
=                           ;
=                           ;       match, move to next sector for future ref
=                           ;        (Code modified for skewed sectors)
=                           ;
=2C74 8CD9                      mov cx,ds               ; set up ES
=2C76 8EC1                      mov es,cx
=2C78 FC                        cld                     ; scan forward
=2C79 B92700                    mov cx,cpmspt-1         ; set count for scan
=2C7C BF6230                    mov di,xlt0             ; point to translate table
```

```
=2C7F F2AE                      repne scasb             ; scan for sector number
=2C81 E306          2C89        jcxz ovf                ;  didn't find it
=2C83 8A05                      mov al,[di]             ; get nr of next sector
=2C85 8807                      mov una,al
=2C87 EB08          2C91        jmps noovf
=                           ;
=                           ;       overflow to next track
=                           ovf:
=2C89 C60700                    mov una,0               ;unasec = 0
=2C8C 2EFF060730                inc unatrk              ;unatrk=unatrk+1
=                           ;
=                           noovf:
=                               ;match found, mark as unnecessary read
=2C91 2EC6060B3000              mov rsflag,0            ;rsflag = 0
=2C97 EB0C          2CA5        jmps rwoper             ;to perform the write
=                           ;
=                           alloc:
=                               ;not an unallocated record, requires pre-read
=2C99 2EC606053000              mov unacnt,0            ;unacnt = 0
=2C9F 2EC6060B3001              mov rsflag,1            ;rsflag = 1
=                                                       ;drop through to rwoper
=                           ;
=                           ;*****************************************************
=                           ;*                                                   *
=                           ;*       Common code for READ and WRITE follows      *
=                           ;*                                                   *
=                           ;*****************************************************
=                           rwoper:
=                               ;enter here to perform the read/write
=2CA5 2EC6060A3000              mov erflag,0            ;no errors (yet)
=2CAB 2EA0FD2F                  mov al, seksec          ;compute host sector
=2CAF B102                      mov cl, secshf
=2CB1 D2E8                      shr al,cl
=2CB3 FEC0                      inc al                  ;*** added for 1-based sectors ***
=2CB5 2EA20230                  mov sekhst,al           ;host sector to seek
=                           ;
=                           ;       active host sector?
=2CB9 B001                      mov al,1
=2CBB 2E86060330                xchg al,hstact          ;always becomes 1
=2CC0 84C0                      test al,al              ;was it already?
=2CC2 7437          2CFB        jz filhst               ;fill host if not
=                           ;
=                           ;       host buffer active, same as seek buffer?
=2CC4 2EA0FA2F                  mov al,sekdsk
=2CC8 2E3A06FE2F                cmp al,hstdsk           ;sekdsk = hstdsk?
=2CCD 7516          2CE5        jnz nomatch
=                           ;
=                           ;       same disk, same track?
=2CCF 2EA1FF2F                  mov ax,hsttrk
=2CD3 2E3B06FB2F                cmp ax,sektrk           ;host track same as seek track
=2CD8 750B          2CE5        jnz nomatch
=                           ;
```

```
=                       ;       same disk, same track, same buffer?
=2CDA 2EA00230                  mov al,sekhst
=2CDE 2E3A060130                cmp al,hstsec           ;sekhst = hstsec?
=2CE3 744A      2D2F            jz match                ;skip if match
=                       nomatch:
=                               ;proper disk, but not correct sector
=2CE5 2EA00430                  mov al, hstwrt
=2CE9 84C0                      test al,al              ;"dirty" buffer ?
=2CEB 740E      2CFB            jz filhst               ;no, don't need to write
=2CED E87B01    2E6B            call writehst           ;yes, clear host buff
=                       ;       (check errors here)
=2CF0 2EA00A30                  mov al,erflag
=2CF4 0AC0                      or al,al                ;any?
=2CF6 7403      2CFB            jz filhst               ;skip if none
=2CF8 E99700    2D92            jmp return_rw           ;exit if so
=                       ;
=                       filhst:
=                               ;may have to fill the host buffer
=2CFB 2EA0FA2F2EA2              mov al,sekdsk ! mov hstdsk,al
      FE2F
=2D03 2EA1FB2F2EA3              mov ax,sektrk ! mov hsttrk,ax
      FF2F
=2D0B 2EA002302EA2              mov al,sekhst ! mov hstsec,al
      0130
=2D13 2EA00B30                  mov al,rsflag
=2D17 84C0                      test al,al              ;need to read?
=2D19 740E      2D29            jz filhstl
=                       ;
=2D1B E8A301    2EC1            call readhst            ;yes, if 1
=                       ;       (check errors here)
=2D1E 2EA00A30                  mov al,erflag
=2D22 0AC0                      or al,al                ;any?
=2D24 7403      2D29            jz filhstl              ;skip if none
=2D26 E96900    2D92            jmp return_rw           ;exit if so
=                       ;
=                       filhstl:
=2D29 2EC606043000              mov hstwrt,0            ;no pending write
=                       ;
=                       match:
=                               ;copy data to or from buffer depending on "readop"
=2D2F 2EA0FD2F                  mov al,seksec           ;mask buffer number
=2D33 250300                    and ax,secmsk           ;least signif bits are masked
=2D36 B107D3E0                  mov cl, 7 ! shl ax,cl   ;shift left 7 (* 128 = 2**7)
=                       ;
=                       ;       ax has relative host buffer offset
=                       ;
=2D3A 2E03064725                add ax,dbptr            ; address of data block
=2D3F 05A200                    add ax, xshrbuf         ;*** new buffer for CP/M 86/80 ***
=2D42 8BF0                      mov si,ax               ;put in source index register
=2D44 2E8B3E1030                mov di,dma_off          ;user buffer is dest if readop
=                       ;
=2D49 1E06                      push DS ! push ES       ;save segment registers
```

```
=                       ;
=2D4B 2E8E060E30                mov ES,dma seg          ;set destseg to the users seg
=                                                       ;SI/DI and DS/ES is swapped
=                                                       ;if write op
=2D50 2BC0                      sub ax,ax               ;*** added for CP/M 86/80 ***
=2D52 8ED8                      mov ds,ax               ;which needs ds=0 (*** end add ***)
=2D54 B98000                    mov cx,128              ;length of move in bytes
=2D57 2EA00C30                  mov al,readop
=2D5B 84C0                      test al,al              ;which way?
=2D5D 7511              2D70    jnz     rwmovx          ;skip if read
=                       ;
=                       ;       write operation, mark and switch direction
=2D5F 2EC606043001              mov hstwrt,1            ;hstwrt = 1 (dirty buffer now)
=2D65 87F7                      xchg si,di              ;source/dest index swap
=2D67 8CD8                      mov ax,DS
=2D69 8EC0                      mov ES,ax
=2D6B 2E8E1E0E30                mov DS,dma_seg          ;setup DS,ES for write
=2D70 E82000            2D93 rwmovx: call rwmove
=                       ;
=
=2D73 071F                      pop ES ! pop DS         ;restore segment registers
=                       ;
=                       movdone:
=                       ;       data has been moved to/from host buffer
=2D75 2E803E0D3001              cmp wrtype,wrdir        ;write type to directory?
=2D7B 2EA00A30                  mov al,erflag           ;in case of errors
=2D7F 7511              2D92    jnz return_rw           ;no further processing
=                       ;
=                       ;       clear host buffer for directory write
=2D81 84C0                      test al,al              ;errors?
=2D83 750D              2D92    jnz return_rw           ;skip if so
=2D85 2EC606043000              mov hstwrt,0            ;buffer written
=2D8B E8DD00            2E6B    call writehst
=2D8E 2EA00A30                  mov al,erflag
=                       return_rw:
=2D92 C3                        ret
=                       ;
=                       ;***********************************************
=                       ;*      this subroutine is made available      *
=                       ;*      for other parts of CP/M.               *
=                       ;***********************************************
=
=                       rwmove:
=                               if not loader_bios
=                       ;---------------------------------------
=                       ;       Added for PC100 -- RK/CPL -- 4/13/82
=2D93 2EF7064E25FF              test Z80FLAG,true       ; is z80 running?
      FF
=2D9A 7503              2D9F    jnz rwml                ;
=2D9C E9C800            2E67    jmp move88              ;normal move if z80 not running
=
=2D9F 51                rwml:   push cx                 ;preserve the count
```

```
=2DA0 8BC1                      mov ax,cx               ;get the count into accum
=2DA2 8CDB                      mov bx,ds               ;check the source segment
=2DA4 8CC2                      mov dx,es               ;and destination segment
=2DA6 F7C380FF                  test bx,0ff80h          ;out of Z80 private?
=2DAA 750C          2DB8        jnz nsz80p              ;jump ahead if it is
=2DAC B104                      mov cl,4                ;set for shl
=2DAE D3E3                      shl bx,cl               ;mult by 16
=2DB0 03DE                      add bx,si               ;get absolute address for Z80
=2DB2 81FB0008                  cmp bx,800h             ;in z80 pvt?
=2DB6 7251          2E09        jb sz80p                ;jump ahead so
=
=2DB8 F7C280FF          nsz80p: test dx,0ff80h          ;source is not z80 private.
=2DBC 7403          2DC1        jz nszl                 ;
=2DBE E9A500        2E66        jmp notz80              ;jump ahead if destination isn't, either
=2DC1 B104              nszl:   mov cl,4                ;set for shl
=2DC3 D3E2                      shl dx,cl               ;mul dest seg by 16
=2DC5 03D7                      add dx,di               ;create absolute address
=2DC7 81FA0008                  cmp dx,800h             ;dest in z80 pvt?
=2DCB 7203          2DD0        jb dz80p                ;
=2DCD E99600        2E66        jmp notz80              ;jump ahead if not
=
=2DD0 8CDB              dz80p:  mov bx,ds               ;destination is in z80 pvt, source isn't
=2DD2 F7C300F0                  test bx,0f000h          ;check for source in 8088 pvt
=2DD6 750C          2DE4        jnz dzs88p              ;dest z80, source 8088.
=2DD8 B104                      mov cl,4                ;
=2DDA D3E3                      shl bx,cl               ;
=2DDC 03DE                      add bx,si               ;absolute source
=2DDE 7204          2DE4        jc dzs88p               ;dest z80, source 8088.
=2DE0 03D8                      add bx,ax               ;how about the end of the source?
=2DE2 735D          2E41        jnc not88               ;z80 move if low enough
=
=2DE4 59                dzs88p: pop cx                  ;dest in z80 pvt, source in 8088 pvt
=2DE5 81F98000                  cmp cx,128              ;is count too big?
=2DE9 7603          2DEE        jbe dzsl                ;
=2DEB B98000                    mov cx,128              ;make it maximum if so
=2DEE 1E                dzsl:   push ds                 ;save ds for later
=2DEF 51                        push cx                 ;save count for later
=2DF0 06                        push es                 ;
=2DF1 57                        push di                 ;
=2DF2 33C0                      xor ax,ax               ;zero ...
=2DF4 8EC0                      mov es,ax               ;the destination segment
=2DF6 BF00FD                    mov di,xfrbuf           ;destination is buffer
=2DF9 E86B00        2E67        call move88             ;move source to buffer
=2DFC 5F                        pop di                  ;
=2DFD 07                        pop es                  ;
=2DFE 8ED8                      mov ds,ax               ;zero the source segment
=2E00 BE00FD                    mov si,xfrbuf           ;source is buffer
=2E03 59                        pop cx                  ;
=2E04 E83B00        2E42        call movz80             ;move buffer to destination after restoring cx
=2E07 1F                        pop ds                  ;restore ds
=2E08 C3                        ret                     ;
=
```

```
=2E09 8CC2              sz80p:  mov dx,es               ;source is in z80 pvt, destination isn't
=2E0B F7C200F0                  test dx,0f000h          ;check for destination in 8088 pvt
=2E0F 750C        2E1D          jnz d88szp              ;dest 8088, source z80.
=2E11 B104                      mov cl,4                ;
=2E13 D3E2                      shl dx,cl               ;
=2E15 03D7                      add dx,di               ;absolute destination
=2E17 7204        2E1D          jc d88szp               ;source z80, destination 8088.
=2E19 03D0                      add dx,ax               ;how about the end of the destination?
=2E1B 7324        2E41          jnc not88               ;z80 move if low enough
=
=2E1D 59                d88szp: pop cx                  ;dest in z80 pvt, source in 8088 pvt
=2E1E 81F98000                  cmp cx,128              ;is count too big?
=2E22 7603        2E27          jbe d88s1               ;
=2E24 B98000                    mov cx,128              ;make it maximum if so
=2E27 1E                d88s1:  push ds                 ;save ds for later
=2E28 51                        push cx                 ;save count for later
=2E29 06                        push es                 ;
=2E2A 57                        push di                 ;
=2E2B 33C0                      xor ax,ax               ;zero ...
=2E2D 8EC0                      mov es,ax               ;the destination segment
=2E2F BF00FD                    mov di,xfrbuf           ;destination is buffer
=2E32 E80D00      2E42          call movz80             ;move source to buffer
=2E35 5F                        pop di                  ;
=2E36 07                        pop es                  ;
=2E37 8ED8                      mov ds,ax               ;zero the source segment
=2E39 BE00FD                    mov si,xfrbuf           ;source is buffer
=2E3C E82800      2E67          call move88             ;move buffer to destination after restoring cx
=2E3F 1F                        pop ds                  ;restore ds
=2E40 C3                        ret                     ;
=
=2E41 59                not88:  pop cx                  ;do a z80 move and exit
=
=2E42 8CDB              movz80: mov bx,ds               ;Z80 move to emulate 8088 move
=2E44 8CC2                      mov dx,es               ;
=2E46 51                        push cx                 ;
=2E47 B104                      mov cl,4                ;
=2E49 D3E3                      shl bx,cl               ;
=2E4B D3E2                      shl dx,cl               ;
=2E4D 03DE                      add bx,si               ;bx has source absolute
=2E4F 03D7                      add dx,di               ;dx has destination absolute
=2E51 B82200                    mov ax,22h              ;function: transfer ...
=2E54 59                        pop cx                  ;to Z80 move routine
=2E55 50                        push ax                 ;via packer
=2E56 53                        push bx                 ;
=2E57 52                        push dx                 ;
=2E58 51                        push cx                 ;
=2E59 B90400                    mov cx,0004             ;4 words stacked
=2E5C 32C0                      xor al,al               ;zero for Z80 wait
=2E5E E854F7      25B5          call packer             ;call interface layer
=2E61 58                        pop ax                  ;after return,
=2E62 58                        pop ax                  ;level the stack
=2E63 58                        pop ax                  ;
```

```
=2E64 58                        pop ax                  ;
=2E65 C3                        ret                     ;go past 8088 move
=
=2E66 59                notz80: pop cx                  ;don't use z80 move
=
=                       move88:                         ;continue to normal -- end of additiion
=                       ;-------------------------------------------------
=                               endif                   ;not loader_bios
=
=2E67 FCF3A4                    cld ! rep movs AL,AL    ;move as bytes
=2E6A C3                        ret                     ;end of move subroutine
=
=                       ;*****************************************************
=                       ;*                                                   *
=                       ;* WRITEHST performs the physical write to the host  *
=                       ;* disk, while READHST reads the physical disk.      *
=                       ;*                                                   *
=                       ;*****************************************************
=
=                       ;|-----------------------------------------------------|
=                       ;| Code added for pc100, CP/M 86/80 4/13/82            |
=
=                       writehst:
=2E6B B81400            rewhst: mov ax,0014h            ;try once (or retry)
=2E6E E84901       2FBA         call rwhst              ;
=2E71 F6C4FC                    test ah,0fch            ;check for error
=2E74 7501         2E77         jnz wragain             ;try if any error
=2E76 C3                        ret                     ;else it's done
=
=                       wragain:                        ;If at first...
=2E77 F6C480                    test ah,80h             ;is the drive ready?
=2E7A 7514         2E90         jnz wnrdy               ;error if not
=2E7C F6C440                    test ah,40h             ;is drive write-protected?
=2E7F 7531         2EB2         jnz writeprot           ;error if so
=2E81 F6C401                    test ah,1               ;seek error
=2E84 7510         2E96         jnz wrsker              ;
=2E86 BBC632                    mov bx,offset biwmsg    ;then give up.
=2E89 E883F7       260F         call pmsg               ;no sense making a fool of yourself
=2E8C E8CD00       2F5C         call rwerr              ;print messages
=2E8F C3                        ret
=
=2E90 E87000       2F03 wnrdy:  call notready           ;disk not ready, print message
=2E93 74D6         2E6B         jz rewhst               ;try again if so directed
=2E95 C3                        ret                     ;else return with error
=
=2E96 BBDE32            wrsker: mov bx,offset skmsg
=2E99 E873F7       260F         call pmsg
=2E9C E80A01       2FA9         call prthst             ;print drive number
=2E9F BB3233                    mov bx,offset trkmsg    ;print 'track'
=2EA2 E86AF7       260F         call pmsg
=2EA5 2EA1FF2F                  mov ax,hsttrk           ;get track number
=2EA9 E8E000       2F8C         call decprt             ;print decimal
```

```
=2EAC E85D00       2F0C          call kqery
=2EAF 74BA         2E6B          jz rewhst               ;try again?
=2EB1 C3                         ret                     ;or return with error
=
=                          writeprot:                    ;disk is write-protected
=2EB2 BB9332                     mov bx,offset wpmsg
=2EB5 E857F7       260F          call pmsg               ;print error message
=2EB8 E8EE00       2FA9          call prthst             ;print which one
=2EBB E84E00       2F0C          call kqery              ;check keys
=2EBE 74AB         2E6B          jz rewhst               ;try again if directed
=2EC0 C3                         ret                     ;else return with bios error
=
=                          ;
=                          readhst:
=2EC1 B81300               rerhst: mov ax,0013h          ;ask interface layer
=2EC4 E8F300       2FBA          call rwhst              ;
=2EC7 F6C4FC                     test ah,0fch            ;check for error
=2ECA 7501         2ECD          jnz rdagain             ;jump ahead if error
=2ECC C3                         ret                     ;else you're done
=
=                          rdagain:
=2ECD F6C480                     test ah,80h             ;drive ready?
=2ED0 750F         2EE1          jnz rdnrdy              ;go away if not
=2ED2 F6C401                     test ah,1               ;seek error?
=2ED5 7510         2EE7          jnz rdsker              ;take care of it
=2ED7 BBAF32                     mov bx,offset birmsg    ;then give up.
=2EDA E832F7       260F          call pmsg               ;print an error message
=2EDD E87C00       2F5C          call rwerr              ;print messages
=2EE0 C3                         ret
=
=2EE1 E81F00       2F03 rdnrdy:  call notready           ;not ready -- print message, etc.
=2EE4 74DB         2EC1          jz readhst              ;try again if directed
=2EE6 C3                         ret                     ;else return with error
=
=2EE7 BBDE32               rdsker: mov bx,offset skmsg
=2EEA E822F7       260F          call pmsg
=2EED E8B900       2FA9          call prthst             ;print drive number
=2EF0 BB3233                     mov bx,offset trkmsg    ;print 'track'
=2EF3 E819F7       260F          call pmsg
=2EF6 2EA1FF2F                   mov ax,hsttrk           ;get track number
=2EFA E88F00       2F8C          call decprt             ;print decimal
=2EFD E80C00       2F0C          call kqery
=2F00 74BF         2EC1          jz rerhst               ;try again?
=2F02 C3                         ret                     ;or return with error
=
=                          notready:                     ;routine to print "not ready"
=2F03 BBF532                     mov bx,offset nrmsg     ;get message address
=2F06 E806F7       260F          call pmsg               ;print it then fall through ...
=2F09 E89D00       2FA9          call prthst             ;print which one
=
=                          kqery:                        ;routine to check keyboard for ^C
=2F0C BB0632                     mov bx,offset kqmsg     ;standard keyboard message
```

```
=2F0F E8FDF6       260F          call pmsg               ;print it
=2F12 E81BFB       2A30          call CONIN              ;get character
=2F15 3C03                       cmp al,03               ;ctrl c?
=2F17 7523         2F3C          jnz kqnex               ;skip ahead if not
=2F19 2EC606053000               mov unacnt,0            ;clear write flags
=2F1F 2EC606033000               mov hstact,0            ;
=2F25 2EC6060D3002               mov wrtype,wrual        ;
=                           if not loader_bios
=                           ;-------------------------------------------------------
=2F2B 2EC7064E2500               mov Z80FLAG,false       ;clear z80 flag
      00
=2F32 E83AF7       266F          call revector           ;start over if so
=2F35 8A0EB724                   mov cl,byte ptr .curdrvs
=2F39 E9C7D0       0003          jmp ccp+3               ;go back to ccp
=                           ;-------------------------------------------------------
=                           endif ;not loader bios
=                           if loader_bios
=                           ;-------------------------------------------------------
=                                jmp WBOOT
=                           ;-------------------------------------------------------
=                           endif ;loader bios
=2F3C 3C20                  kqnex:  cmp al,' '              ;space bar for retry
=2F3E 7503         2F43          jnz kqret               ;other key for error return
=2F40 32C0                       xor al,al               ;zero it
=2F42 C3                         ret                     ;
=2F43 2EC6060A3001          kqret:  mov erflag,1            ;error code
=2F49 2EC606053000               mov unacnt,0            ;clear write flags
=2F4F 2EC606033000               mov hstact,0            ;
=2F55 2EC6060D3002               mov wrtype,wrual        ;
=2F5B C3                         ret                     ;else return
=
=                           ;****************************************
=                           ;*
=                           ;*      print disk, track, sector
=                           ;*
=                           ;****************************************
=
=2F5C E84A00       2FA9 rwerr:  call prthst             ;print disk":"
=2F5F BB3233                    mov bx,offset trkmsg    ;print" TRACK "
=2F62 E8AAF6       260F         call pmsg
=2F65 2EA1FF2F                  mov ax,hsttrk           ;print track no.
=2F69 E82000       2F8C         call decprt
=2F6C BB3B33                    mov bx,offset secmsg    ;print " SECTOR "
=2F6F E89DF6       260F         call pmsg
=2F72 2EA00130                  mov al,hstsec           ;print sector no.
=2F76 32E4                      xor ah,ah
=2F78 E81100       2F8C         call decprt
=2F7B BB5732                    mov bx,offset kqxmsg    ;ask for options
=2F7E E88EF6       260F         call pmsg
=2F81 E8ACFA       2A30         call CONIN              ;
=2F84 3C0D                      cmp al,cr               ;carriage return?
=2F86 7501         2F89         jne rwmker              ;mark error if not
```

```
=2F88 C3                        ret                     ;ignore if return
=2F89 E9B7FF        2F43 rwmker: jmp kqret              ;return with marked error
=
=                        decprt: ;enter with number to be printed in ax
=2F8C 33C9                      xor cx,cx               ;count digits in cx
=2F8E 33D2                 rediv: xor dx,dx             ;clear the remainder
=2F90 BB0A00                    mov bx,10
=2F93 F7F3                      div bx                  ;divide by 10
=2F95 41                        inc cx                  ;bump the count
=2F96 52                        push dx                 ;store digit
=2F97 0BC0                      or ax,ax                ;any more?
=2F99 75F3          2F8E        jnz rediv               ;loop if so
=2F9B 5A                   redout: pop dx               ;the last shall be first
=2F9C 51                        push cx                 ;store the count
=2F9D 8ACA                      mov cl,dl               ;remainder in dl to cl
=2F9F 80C130                    add cl,'0'              ;ascii number
=2FA2 E890FA        2A35        call CONOUT             ;print digit
=2FA5 59                        pop cx                  ;get the count
=2FA6 E2F3          2F9B        loop redout             ;loop back if more
=2FA8 C3                        ret                     ;else return
=
=                        prthst: ;print disk, A:, B:, etc.
=2FA9 2E8A0EFE2F                mov cl,hstdsk           ;get the number
=2FAE 80C141                    add cl,'A'
=2FB1 E881FA        2A35        call CONOUT             ;Print it
=2FB4 B13A                      mov cl,':'
=2FB6 E87CFA        2A35        call CONOUT             ;print a colon
=2FB9 C3                        ret
=
=                        ;****************************************
=                        ;*
=                        ;*      read/write from host
=                        ;*
=                        ;****************************************
=
=2FBA 50                 rwhst: push ax                 ;function set for packer
=2FBB B400                      mov ah,0                ;clear high
=2FBD B104                      mov cl,4                ;set for rol
=2FBF 2EA0FE2F                  mov al,hstdsk           ;disk number
=2FC3 D3C0                      rol ax,cl               ;times 32
=2FC5 BB1330                    mov bx,offset dpbase    ;get disk data address
=2FC8 03D8                      add bx,ax               ;for this disk
=2FCA 8B17                      mov dx,[bx]             ;get pointer in dx
=2FCC 2E8A1E0130                mov bl,hstsec           ;host sector
=2FD1 D1C0                      rol ax,1                ;rotate drive once more
=2FD3 80E31F                    and bl,1fh              ;mask sector number (jic)
=2FD6 0AC3                      or al,bl                ;combine sector
=2FD8 2E8B1EFF2F                mov bx,hsttrk           ;and track
=2FDD 8AE3                      mov ah,bl               ;in one byte
=2FDF 50                        push ax                 ;stack for packer (2)
=2FE0 2EA14725                  mov ax,dbptr            ; point to data block
=2FE4 05A200                    add ax,xshrbuf          ;point to buffer
```

```
=2FE7 50                        push ax                 ;(3)
=2FE8 B80100                    mov ax,0001             ;
=2FEB 50                        push ax                 ;(4)
=2FEC B80000                    mov ax,0                ;wait for z80
=2FEF B90400                    mov cx,4                ;4 words for packer
=2FF2 E8C0F5        25B5        call packer             ;
=2FF5 58                        pop ax                  ;level the stack
=2FF6 58                        pop ax                  ;
=2FF7 58                        pop ax                  ;
=2FF8 58                        pop ax                  ;
=2FF9 C3                        ret                     ;go away when done
=                       ;|                                                      |
=                       ;| End of added code                                    |
=                       ;|------------------------------------------------------|
=
=                       ;
=                       ;*******************************************************
=                       ;*                                                     *
=                       ;* Use the GENDEF utility to create disk def tables    *
=                       ;*                                                     *
=                       ;*******************************************************
=                       ;dpbase equ     offset $
=                       ;       disk parameter tables go here
=                       ;
=
=                       ;*******************************************************
=                       ;*                                                     *
=                       ;* Uninitialized RAM areas follow, including the      *
=                       ;* areas created by the GENDEF utility listed above.  *
=                       ;*                                                     *
=                       ;*******************************************************
=2FFA                   sek_dsk rb      1               ;seek disk number
=2FFB                   sek_trk rw      1               ;seek track number
=2FFD                   sek_sec rb      1               ;seek sector number
=                       ;
=2FFE FF                hst_dsk db      0ffh            ;host disk number
=2FFF                   hst_trk rw      1               ;host track number
=3001                   hst_sec rb      1               ;host sector number
=                       ;
=3002                   sek_hst rb      1               ;seek shr secshf
=3003                   hst_act rb      1               ;host active flag
=3004                   hst_wrt rb      1               ;host written flag
=                       ;
=3005                   una_cnt rb      1               ;unalloc rec cnt
=3006                   una_dsk rb      1               ;last unalloc disk
=3007                   una_trk rw      1               ;last unalloc track
=3009                   una_sec rb      1               ;last unalloc sector
=                       ;
=300A                   erflag  rb      1               ;error reporting
=300B                   rsflag  rb      1               ;read sector flag
=300C                   readop  rb      1               ;1 if read operation
=300D                   wrtype  rb      1               ;write operation type
```

```
=300E                   dma_seg rw      1               ;last dma segment
=3010                   dma_off rw      1               ;last dma offset
=3012                   mediatype rb    1               ;Rainbow or Robin media
=                       ;hstbuf rb      hstsiz          ;host buffer (not in CP/M 86/80)
                                end
=                               INCLUDE CAT.LIB
=                       ;       DISKS 5
=  3013                 dpbase  equ     $               ;Base of Disk Parameter Blocks
=3013 62300000          dpe0    dw      xlt0,0000h      ;Translate Table
=3017 00000000                  dw      0000h,0000h     ;Scratch Area
=301B 9D305330                  dw      dirbuf,dpb0     ;Dir Buff, Parm Block
=301F 36311D31                  dw      csv0,alv0       ;Check, Alloc Vectors
=3023 62300000          dpe1    dw      xlt1,0000h      ;Translate Table
=3027 00000000                  dw      0000h,0000h     ;Scratch Area
=302B 9D305330                  dw      dirbuf,dpb1     ;Dir Buff, Parm Block
=302F 6F315631                  dw      csv1,alv1       ;Check, Alloc Vectors
=3033 62300000          dpe2    dw      xlt2,0000h      ;Translate Table
=3037 00000000                  dw      0000h,0000h     ;Scratch Area
=303B 9D305330                  dw      dirbuf,dpb2     ;Dir Buff, Parm Block
=303F A8318F31                  dw      csv2,alv2       ;Check, Alloc Vectors
=3043 62300000          dpe3    dw      xlt3,0000h      ;Translate Table
=3047 00000000                  dw      0000h,0000h     ;Scratch Area
=304B 9D305330                  dw      dirbuf,dpb3     ;Dir Buff, Parm Block
=304F E131C831                  dw      csv3,alv3       ;Check, Alloc Vectors
=                       ;-- DUMMY DISK FOR ROBIN MEDIA REFERENCE
=                       ;       DISKDEF 0,0,39,1,2048,195,128,128,2
=  3053                 dpb0    equ     offset $        ;Disk Parameter Block
=3053 2800                      dw      40              ;Sectors Per Track
=3055 04                        db      4               ;Block Shift
=3056 0F                        db      15              ;Block Mask
=3057 01                        db      1               ;Extnt Mask
=3058 C200                      dw      194             ;Disk Size - 1
=305A 7F00                      dw      127             ;Directory Max
=305C C0                        db      192             ;Alloc0
=305D 00                        db      0               ;Alloc1
=305E 2000                      dw      32              ;Check Size
=3060 0200                      dw      2               ;Offset
=  3062                 xlt0    equ     offset $        ;Translate Table
=                       ;** Modified for Rainbow media ...
=3062 00010203                  db      0,1,2,3
=3066 08090A0B                  db      8,9,10,11
=306A 10111213                  db      16,17,18,19
=306E 18191A1B                  db      24,25,26,27
=3072 20212223                  db      32,33,34,35
=3076 04050607                  db      4,5,6,7
=307A 0C0D0E0F                  db      12,13,14,15
=307E 14151617                  db      20,21,22,23
=3082 1C1D1E1F                  db      28,29,30,31
=3086 24252627                  db      36,37,38,39
=  0019                 als0    equ     25              ;Allocation Vector Size
=  0020                 css0    equ     32              ;Check Vector Size
=                       ;       DISKDEF 1,0
```

```
=  3053                 dpbl    equ     dpb0            ;Equivalent Parameters
=  0019                 alsl    equ     als0            ;Same Allocation Vector Size
=  0020                 cssl    equ     css0            ;Same Checksum Vector Size
=  3062                 xltl    equ     xlt0            ;Same Translate Table
=                       ;       DISKDEF 2,0
=  3053                 dpb2    equ     dpb0            ;Equivalent Parameters
=  0019                 als2    equ     als0            ;Same Allocation Vector Size
=  0020                 css2    equ     css0            ;Same Checksum Vector Size
=  3062                 xlt2    equ     xlt0            ;Same Translate Table
=                       ;       DISKDEF 3,0
=  3053                 dpb3    equ     dpb0            ;Equivalent Parameters
=  0019                 als3    equ     als0            ;Same Allocation Vector Size
=  0020                 css3    equ     css0            ;Same Checksum Vector Size
=  3062                 xlt3    equ     xlt0            ;Same Translate Table
=                       ;       DISKDEF 4,0,35,1,1024,171,64,64,2
=  308A                 dpb4    equ     offset $        ;Disk Parameter Block
=308A 2400                       dw      36              ;Sectors Per Track
=308C 03                         db      3               ;Block Shift
=308D 07                         db      7               ;Block Mask
=308E 00                         db      0               ;Extnt Mask
=308F AA00                       dw      170             ;Disk Size - 1
=3091 3F00                       dw      63              ;Directory Max
=3093 C0                         db      192             ;Alloc0
=3094 00                         db      0               ;Allocl
=3095 1000                       dw      16              ;Check Size
=3097 0200                       dw      2               ;Offset
=  3062                 xlt4    equ     xlt0            ;Translate Table
=  000C                 als4    equ     12              ;Allocation Vector Size
=  0010                 css4    equ     16              ;Check Vector Size
=                       ;       ENDEF
=                       ;
=                       ;       Table for transfering dpb's between rainbow and robin media:
=                       ;
=  3099                 mediatbl        equ offset $
=3099 5330                       dw      dpb0    ;Rainbow PC-100
=309B 8A30                       dw      dpb4    ;Robin
=                       ;
=                       ;       Uninitialized Scratch Memory Follows:
=                       ;
=  309D                 begdat  equ     offset $        ;Start of Scratch Area
=309D                   dirbuf  rs      128             ;Directory Buffer
=311D                   alv0    rs      als0            ;Alloc Vector
=3136                   csv0    rs      css0            ;Check Vector
=3156                   alv1    rs      als1            ;Alloc Vector
=316F                   csv1    rs      css1            ;Check Vector
=318F                   alv2    rs      als2            ;Alloc Vector
=31A8                   csv2    rs      css2            ;Check Vector
=31C8                   alv3    rs      als3            ;Alloc Vector
=31E1                   csv3    rs      css3            ;Check Vector
=  3201                 enddat  equ     offset $        ;End of Scratch Area
=  0164                 datsiz  equ     offset $-begdat ;Size of Scratch Area
=3201 00                         db      0               ;Marks End of Module
```

```
=                       INCLUDE CPLBIOS2.A86
=
=
=
=                       ;*****************************************
=                       ;*                                       *
=                       ;*               Data Areas              *
=                       ;*                                       *
=                       ;*****************************************
=
=  3010                 dma adr equ dma_off
=
=3202 00                IOBYTE  db      0
=3203 0000              BXHLD   dw      0               ;store bx here
=3205 00                xoff_flg db     0
=
=                       IF      loader_bios
=                       ;---------------------------------------
=                       ;|                                     |
=                       signon  db      27,'[2J',27,'[3;1H'
=                       db      cr,lf,cr,lf
=                       DB      'CP/M-86/80 Loading ...',CR,LF,0
=                       ;|                                     |
=                       ;---------------------------------------
=                       ENDIF   ;loader_bios
=
=                       ;error messages:
=
=3206 0D0A50726573      kqmsg   db      cr,lf,'Press CTRL-C to restart, ',cr,lf
      73204354524C
      2D4320746F20
      726573746172
      742C200D0A
=3223 737061636520              db      'space bar to retry, or any other key to continue.'
      62617220746F
      207265747279
      2C206F722061
      6E79206F7468
      6572206B6579
      20746F20636F
      6E74696E7565
      2E
=3254 0D0A00            endlin  db      cr,lf,0
=
=3257 0D0A              kqxmsg  db      cr,lf
=3259 507265737320              db      'Press Return to ignore error, any other key to continue'
      52657475726E
      20746F206967
      6E6F72652065
      72726F722C20
      616E79206F74
```

```
      686572206B65
      7920746F2063
      6F6E74696E75
      65
=3290 0D0A00                  db      cr,lf,0
=
=3293 0D0A44726976      wpmsg   db      cr,lf,'Drive write-protected -- ',0
      652077726974
      652D70726F74
      656374656420
      2D2D2000
=
=32AF 0D0A52656164      er1     db      cr,lf,'Read error on drive ',0
      206572726F72
      206F6E206472
      6976652000
=32C6 0D0A57726974      er2     db      cr,lf,'Write error on drive ',0
      65206572726F
      72206F6E2064
      726976652000
=32DE 0D0A5365656B      skmsg   db      cr,lf,'Seek error on drive ',0
      206572726F72
      206F6E206472
      6976652000
=32F5 0D0A44726976      er5     db      cr,lf,'Drive not ready -- ',0
      65206E6F7420
      726561647920
      2D2D2000
=330B 0D0A43616E6E      unabl   db      cr,lf,'Cannot write on VT180 disk on drive ',0
      6F7420777269
      7465206F6E20
      565431383020
      6469736B206F
      6E2064726976
      652000
=3332 2C2074726163      trkmsg  db      ', track ',0
      6B2000
=333B 2C2073656374      secmsg  db      ', sector ',0
      6F722000
=
=  32AF                 birmsg  equ erl
=  32C6                 biwmsg  equ er2
=  32F5                 nrmsg   equ er5
=
=3345 0D0A54696D65      p232to  db      cr,lf,'Timeout on '
      6F7574206F6E
      20
=3352 4544              p232tol dw      'DE'
=3354 563A00            p232to2 db      'V:',0
=
=
=                               if      not loader_bios
```

```
=3357 00                dispst  db      0
=                               endif
=                       ;
=3358                   SEGHLD  rw 1            ;save segment
=335A                   RTNHLD  rw 1            ;save return address
=335C                   COUNT   rw 1            ;
=                       ;
=                       ;       System Memory Segment Table
=
=335E                   segtable rb 1   ;2 segments
=335F                           rw 4    ;room for two segs
=
=
=                               IF      not loader bios
=                       ;---------------------------------------------
=                       ;|                                           |
=3367 1B5B324A1B5B      signon  db      27,'[2J',27,'[3;1H'
      333B3148
=3371 202020202043              DB      '     CP/M-86/80 Version '
      502F4D2D3836
      2F3830205665
      7273696F6E20
=3389 31                        db      version_number+'0'
=338A 2E                        db      '.'
=338B 30                        db      rev number+'0'
=                       ;       db      '.'
=                       ;       db      mod_number+'0'
=338C 2028312E3129              db      ' (1.1)',CR,LF
      0D0A
=3394 28632920436F              DB      '(c) Copyright 1981 Digital Research Inc.',CR,LF
      707972696768
      742031393831
      204469676974
      616C20526573
      656172636820
      496E632E0D0A
=33BE 28632920436F              DB      '(c) Copyright 1982 Digital Equipment Corporation',cr,lf
      707972696768
      742031393832
      204469676974
      616C20457175
      69706D656E74
      20436F72706F
      726174696F6E
      0D0A
=33F0 0D0A00                    db      cr,lf,0
=
=                       ;|                                           |
=                       ;---------------------------------------------
=                               ENDIF   ;not loader_bios
=
=  33F3                 lastoff equ offset $
```

```
=                       if not loader_bios
=                       ;-----------------------------------------
=                       org     3900h
=                       ;-----------------------------------------
=                       endif ;not loader_bios
=
=3900                   loc_stk rw  24  ;local stack for initialization
=  3930                 stkbase equ offset $
=
=  0380                 tpa_seg equ (lastoff+0400h+15) / 16
=  0C77                 tpa_len equ 0FF7h - tpa_seg
=3930 00                        db 0    ;fill last address for GENCMD
=
=                       ;*****************************************
=                       ;*                                       *
=                       ;*          Dummy Data Section           *
=                       ;*                                       *
=                       ;*****************************************
=  0000                         dseg    0       ;absolute low memory
=                               org     0       ;(interrupt vectors)
=0000                   int0_offset     rw      1
=0002                   int0_segment    rw      1
=                       ;       pad to system call vector
=0004                           rw      2*(bdos int-1)
=
=0380                   bdos_offset     rw      1
=0382                   bdos_segment    rw      1
=                               org 36*4                        ;type 36 service
=0090                   sio_offset      rw      1
=0092                   sio_seg         rw      1
=0094                   sio2_offset     rw      1
=0096                   sio2_seg        rw      1
=                               org 39*4                        ;type 39 service
=009C                   Z80_OFFSET      RW      1
=009E                   Z80_SEG         RW      1
=                               org 44*4
=00B0                   tp44_offset     rw      1
=00B2                   tp44_seg        rw      1
=                               org 100*4
=0190                   tp100_offset    rw      1
=0192                   tp100_seg       rw      1
=
=                       ;*************************************************************************
=                       ;
=                       ;       DUMMY CODE SECTION (FOR FAR CALL TO Z80CCP SERVICES)
=                       ;
=                       ;*************************************************************************       END
=                       ;
=  0040                         CSEG    40h     ; ABSOLUTE LOCATION
=                               ORG     6h
=  0006                 CCPSERV EQU     $
=                       ;
```

END OF ASSEMBLY. NUMBER OF ERRORS: 0. USE FACTOR: 43%

```
                        TITLE   'CPL PATCHES FOR CP/M 8?/80'

000A                COMLEN_OFFSET   EQU     000AH           ; VARIABLE LOCATION
008B                BDOS_LOC        EQU     008BH           ; LABEL LOCATION
00AF                PRINT_LOC       EQU     00AFH           ; LABEL LOCATION
00D2                INITMEM_LOC     EQU     00D2H           ; LABEL LOCATION
00E9                OPEN_LOC        EQU     00E9H           ; LABEL LOCATION
00ED                OPENC_LOC       EQU     00EDH           ; LABEL LOCATION
034D                BOOT_PATCH_LOC  EQU     034DH           ; PATCH 3 BYTES
035E                CCP_LOC         EQU     035EH           ; LABEL LOCATION
046C                RESETDISK_LOC   EQU     046CH           ; LABEL LOCATION
0727                COM_PATCH_LOC   EQU     0727H           ; PATCH 5 BYTES
0732                LOADUSER_LOC    EQU     0732H           ; LABEL LOCATION
0738                GOUSER_LOC      EQU     0738H           ; LABEL LOCATION
0762                GOUSERI_LOC     EQU     0762H           ; NEW LABEL LOCATION
07BE                ZRUN_PATCH_LOC  EQU     07BEH           ; PATCH 6 BYTES
07F9                IP0_OFFSET      EQU     07F9H           ; VARIABLE LOCATION
0803                PAG0_OFFSET     EQU     0803H           ; VARIABLE LOCATION
0827                COMFCB_OFFSET   EQU     0827H           ; VARIABLE LOCATION
0939                CDISK_OFFSET    EQU     0939H           ; VARIABLE LOCATION
093A                SDISK_OFFSET    EQU     093AH           ; VARIABLE LOCATION
097B                LOADMSG_OFFSET  EQU     097BH           ; VARIABLE LOCATION

14E0                DOBACKSP_LOC    EQU     14E0H           ; NEW LABEL LOCATION
14F2                DELETE_PATCH_LOC EQU    14F2H           ; PATCH 12 BYTES
1622                MOVE_PATCH_LOC  EQU     1622H           ; PATCH 3 BYTES

253F                ZMOVEF_LOC      EQU     2500H+(3*21)    ; LABEL LOCATION

3400                CODE8680        EQU     03400H          ; OFFSET FOR ADDITIONAL CCP CODE
                                                            ;   FOR CP/M-8680
                                                            ;   (LOCATED AFTER BIOS)

                    .****************************************************************
                    ;**     EQUATES FOR PATCH TO ALLOW SUBMIT FILES FROM DISKS      ;**
                    ;**     OTHER THAN A.                                           ;**
                    :****************************************************************

0147                READCP          EQU     147H
24B7                CURDRVS         EQU     24B7H
0939                CDISK           EQU     939H
01FF                DELSUBP         EQU     1FFH
0153                READCA_LOC      EQU     153H
00DA                SELECT_LOC      EQU     0DAH
0202                DELSUBR_LOC     EQU     202H

                    ;****************************************************************
```

```
                        ;*****************************************************************
                        ;**     EQUATES FOR PATCH TO MAINTAIN BDOS CONSOLE IN STATUS    ;**
                        ;**     FOR Z80 PROGRAMS IN THE DATA BLOCK                      ;**
CP/M ASM86 1.1  SOURCE: CPLPATCH.A86  CPL PATCHES FOR CP/M 86/80


                        ;*****************************************************************

   138B                 CONIN_PATCH_LOC EQU     138BH           ; PATCH 3 BYTES

   1407                 CONB0_PATCH_LOC EQU     1407H           ; PATCH 3 BYTES

   22E6                 KBCHAR_OFFSET   EQU     22E6H           ; VARIABLE LOCATION
CP/M ASM86 1.1  SOURCE: CPLPATCH.A86  CPL PATCHES FOR CP/M 86/80


                                EJECT

                                CSEG


                                ORG     BDOS_LOC
                        BDOS:


                                ORG     PRINT_LOC
                        PRINT:


                                ORG     INITMEM_LOC
                        INITMEM:


                                ORG     OPEN_LOC
                        OPEN:


                                ORG     OPENC_LOC
                        OPENC:


                                ORG     BOOT_PATCH_LOC
 034D E9B030       3400         JMP     BOOT_PATCH              ; SAVE BOOT DRIVE#
                        BOOT_RET:


                                ORG     CCP_LOC
                        CCP:


                                ORG     RESETDISK_LOC
                        RESETDISK:
```

```
                                ORG     COM_PATCH_LOC
0727 E9DF2C             3409    JMP     COM_PATCH               ; LOAD Z80CCP.SYS IF NECESSARY
072A 90                         NOP
072B 90                         NOP
                        COM_RET:

                                ORG     LOADUSER_LOC
                        LOADUSER:

                                ORG     GOUSER_LOC
                        GOUSER:
```



```
                                ORG     GOUSER1_LOC
                        GOUSER1:

                                ORG     ZRUN_PATCH_LOC
07BE E9EE2C             34AF    JMP     ZRUN_PATCH              ; RUN Z80CCP.SYS IF LOADED
07C1 90                         NOP
07C2 90                         NOP
07C3 90                         NOP
                        ZRUN_RET:

                                ORG     DOBACKSP_LOC
                        DOBACKSP:

                                ORG     DELETE_PATCH_LOC
14F2 EBEC               14E0    JMPS    DOBACKSP                ; MAKE DELETE BEHAVE AS BACKSPACE
14F4 90                         NOP
14F5 90                         NOP
14F6 90                         NOP
14F7 90                         NOP
14F8 90                         NOP
14F9 90                         NOP
14FA 90                         NOP
14FB 90                         NOP
14FC 90                         NOP
14FD 90                         NOP
                        DELETE_RET:

                                ORG     MOVE_PATCH_LOC
1622 E81A0F             253F    CALL    ZMOVEF                  ; ALLOW MOVE TO Z80 PRIVATE SPACE
                        MOVE_RET:
```

```
                        ORG     ZMOVEF_LOC
                    ZMOVEF:


CP/M ASM86 1.1  SOURCE: CPLPATCH.A86  CPL PATCHES FOR CP/M 86/80


                        EJECT
                    ;*****************************************************************
                    ;**     PATCH TO ALLOW SUBMIT FILES TO USE OTHER DISKS THAN     **
                    ;**     A FOR SCRATCH FILE (SUBMIT.$$$)                         **
                    ;*****************************************************************

                        ORG READCA_LOC
                    READCA:

                        ORG SELECT_LOC
                    SELECT:

                        ORG DELSUBR_LOC
                    DELSUBR:

                        ORG     READCP
 0147 A0B724                    MOV     AL,BYTE PTR .CURDRVS
 014A 3A063909                  CMP     AL,BYTE PTR .CDISK
 014E 7403          0153        JE      READCA

                        ORG DELSUBP
 01FF E90933        350B        JMP     DELSUBPA



                    ;*****************************************************************;**
                    ;**     PATCHES FOR  BDOS CONIN STATUS IN DATA BLOCK            **
                    ;*****************************************************************

                        ORG     CONIN_PATCH_LOC
 138B E98621        3514        JMP     CONIN_PATCH     ; RESET BDOS CONIN FLAG
                    CONIN_RET:

                        ORG     CONB0_PATCH_LOC
 1407 E92221        352C        JMP     CONB0_PATCH     ; SET BDOS CONIN FLAG
                    CONB0_RET:
CP/M ASM86 1.1  SOURCE: CPLPATCH.A86  CPL PATCHES FOR CP/M 86/80


                        EJECT

=                       INCLUDE DEFBUF.LIB
=                   ; ************************************************************************
=                   ; OFFSETS FROM START OF POINTERS/BUFFERS DATA BLOCK
=
=  FFA0             XDPCX           EQU     -60H            ; DISK PARAMETER STORAGE (60H)
=  0000             XDEFBUF         EQU     000H            ; MISC. BUFFER (LENGTH=80H)
```

```
=  0086                 XPACKET         EQU     086H            ; BIOS MESSAGE PACKET (LENGTH=0EH)
=  0086                 XSTPKT          EQU     086H            ; START PACKET BUFFER (LENGTH=0EH)
=  0094                 XADCPKT         EQU     094H            ; DATA PACKET (LENGTH=0EH)
=  0094                 XMVPKT          EQU     094H            ; MOVE PACKET BUFFER (LENGTH=0EH)
=  00A2                 XSHRBUF         EQU     0A2H            ; SEGMENT BUFFER (LENGTH=200H)
=  02F8                 XMEMSIZE        EQU     2F8H            ; MEMORY SIZE (WORD)
=  02FA                 XPCPMADR        EQU     2FAH            ; PSEUDO CP/M ADDRESS (WORD)
=  02FC                 XZ80PKT         EQU     2FCH            ; PACKET POINTER FROM Z80 (WORD)
=  02FE                 XI88PKT         EQU     2FEH            ; PACKET POINTER FROM 8088 (WORD)
=  02F0                 XTTRACK         EQU     2F0H            ;TRACK TABLE
=  02F4                 XTFORMAT        EQU     2F4H            ; FORMAT TABLE
=  02E7                 XCSFLAG         EQU     2E7H            ; CONSOLE STATUS FLAG
=
=                       ; OFFSETS FROM ZOT FOR CONVENIENCE
=  0000                 ZOTP            EQU     0               ; Z80 FLAG
=  FFFE                 Z80FLAGPT       EQU     -2              ; Z80-RUNNING FLAG
=  FFFB                 CICCK           EQU     -5              ; CONSOLE STATUS FLAG CHECK
=
=                       ; OTHER USEFUL EQUATES
=  0002                 BDCS            EQU     2               ; BDOS CHARACTER READY BIT
=  0001                 BIOCS           EQU     1               ; BIOS CONSOLE STATUS BIT
=  0017                 BIOS_JMPS       EQU     23              ; NUMBER OF FUNCTIONS IN JUMP TABLE
=
=                       ; ************************************************************************



   2500                 BIOS_OFFSET     EQU     2500H           ; OFFSET OF BIOS

   0047                 PBADR_OFFSET    EQU     BIOS_JMPS*3+2   ; OFFSET (FROM START OF BIOS)
                                                                ;   OF WORD THAT CONTAINS
                                                                ;   ADDRESS OF POINTERS/BUFFERS
                                                                ;   DATA BLOCK

   02FA                 PCPMADR_OFFSET  EQU     XPCPMADR        ; OFFSET (FROM START OF PTRSBUFS)
                                                                ;   OF WORD CONTAINING ADDRESS
                                                                ;   OF PSEUDO CP/M
                                                                ;   0=PSEUDO CP/M NOT LOADED

   003B                 LOADF           EQU     59              ; BDOS FUNCTION - LOAD FILE

   000A                 LF              EQU     0AH
   000D                 CR              EQU     0DH

CP/M ASM86 1.1  SOURCE: CPLPATCH.A86  CPL PATCHES FOR CP/M 86/80


                                EJECT

                                ORG     CODE8680

                        BOOT_PATCH:
 3400 A23909                            MOV     .CDISK_OFFSET,AL
 3403 A24735                            MOV     BOOTDRV,AL
 3406 E947CF        0350                JMP     BOOT_RET
```

```
CP/M ASM86 1.1  SOURCE: CPLPATCH.A86  CPL PATCHES FOR CP/M 86/80


                              EJECT

                         COM_PATCH:
3409 E8E1CC       00ED          CALL    OPENC
340C 7403         3411          JZ      USERCOM
340E E91700       3428          JMP     CMDTEST
                         USERCOM:                                ; TRY .COM FILE
3411 BF3008                     MOV     DI,COMFCB_OFFSET+9
3414 BE4435                     MOV     SI,OFFSET COMTYPE
3417 B90300                     MOV     CX,3
341A FC                         CLD
341B F3A4                  REP  MOVS    AL,AL
341D BA2708                     MOV     DX,COMFCB_OFFSET
3420 E8CACC       00ED          CALL    OPENC
3423 752B         3450          JNZ     COMTEST
3425 E904D3       072C          JMP     COM_RET

                         CMDTEST:
3428 BE2808                     MOV     SI,COMFCB_OFFSET+1      ; FILE = SAVE.CMD?
342B BF4D35                     MOV     DI,OFFSET SAVEFCB
342E B90800                     MOV     CX,8
3431 FC                         CLD
3432 F3A6                  REPE CMPS    AL,AL
3434 E317         344D          JCXZ    CMDTESTX                ;   YES - GO EXECUTE

3436 BB4725                     MOV     BX,BIOS_OFFSET+PBADR_OFFSET     ; IS PSEUDO CP/M LOADED?
3439 8B1F                       MOV     BX,[BX]
343B 81C3FA02                   ADD     BX,PCPMADR_OFFSET
343F 06                         PUSH    ES
3440 33C0                       XOR     AX,AX
3442 8EC0                       MOV     ES,AX
3444 26833F00                   CMP     ES:WORD PTR [BX],0
3448 07                         POP     ES
3449 7402         344D          JZ      CMDTESTX                        ;   NO - GO EXECUTE
344B EB03         3450          JMPS    LOADZSYS

                         CMDTESTX:
344D E9E2D2       0732          JMP     LOADUSER

                         COMTEST:

                         LOADZSYS:
3450 E87FCC       00D2          CALL    INITMEM                 ; RELEASE ALL MEMORY

3453 BE2708                     MOV     SI,COMFCB_OFFSET        ; SAVE THIS FCB
3456 BF7635                     MOV     DI,OFFSET PARMFCB       ; ... TO USE AS PARAMETER
3459 B92100                     MOV     CX,33
345C FC                         CLD
345D F3A4                  REP  MOVS    AL,AL
345F A03A09                     MOV     AL,.SDISK_OFFSET
3462 A27635                     MOV     PARMFCB,AL              ; DO NOT USE DEFAULT DISK
```

```
 3465 C606963500                MOV     BYTE PTR PARMFCB+32,0   ; REC CNT = 0
CP/M ASM86 1.1  SOURCE: CPLPATCH.A86  CPL PATCHES FOR CP/M 86/80


                        ;
                                ; OPEN & LOAD Z80CCP.SYS
 346A A04735                    MOV     AL,BOOTDRV
 346D FEC0                      INC     AL
 346F A25535                    MOV     ZSYSFCB,AL              ; SET FOR BOOT DRIVE
 3472 C606753500                MOV     BYTE PTR ZSYSFCB+32,0   ; REC CNT = 0
 3477 BA5535                    MOV     DX,OFFSET ZSYSFCB
 347A E86CCC        00E9        CALL    OPEN
 347D 7503          3482        JNZ     LZ80A
 347F E91600        3498        JMP     ERRZ80                  ; ERROR
                        LZ80A:
 3482 BA5535                    MOV     DX,OFFSET ZSYSFCB
 3485 B13B                      MOV     CL,LOADF
 3487 E801CC        008B        CALL    BDOS                    ; LOAD Z80CCP.SYS
 348A 40                        INC     AX
 348B 7503          3490        JNZ     LZ80B
 348D E90D00        349D        JMP     ERRLOAD                 ; ERROR
                        LZ80B:
 3490 C606483501                MOV     ZSYSFLAG,1
 3495 E9A0D2        0738        JMP     GOUSER

                        ERRZ80:
 3498 B99735                    MOV     CX,OFFSET NOZ80CCP
 349B EB03          34A0        JMPS    ZERR

                        ERRLOAD:
 349D B9C435                    MOV     CX,OFFSET LOADZ80ERR

                        ZERR:
 34A0 E80CCC        00AF        CALL    PRINT
 34A3 E8C6CF        046C        CALL    RESETDISK
 34A6 2EC6060A0000              MOV     CS:BYTE PTR .COMLEN_OFFSET,0
 34AC E9AFCE        035E        JMP     CCP

CP/M ASM86 1.1  SOURCE: CPLPATCH.A86  CPL PATCHES FOR CP/M 86/80


                                EJECT

                        ZRUN_PATCH:
 34AF 2EC6060A0000              MOV     CS:BYTE PTR .COMLEN_OFFSET,00H
 34B5 803E483501                CMP     ZSYSFLAG,1              ; Z80CCP.SYS LOADED?
 34BA 7403          34BF        JZ      RUNZSYS                 ;   YES
 34BC E905D3        07C4        JMP     ZRUN_RET

                        RUNZSYS:
 34BF C606483500                MOV     ZSYSFLAG,0

                                ; SETUP PARAMETER BLOCK
 34C4 BB4935                    MOV     BX,OFFSET ZSYSPARM
 34C7 C7077635                  MOV     WORD PTR [BX],OFFSET PARMFCB
```

```
 34CB A03909                    MOV     AL,.CDISK_OFFSET
 34CE 8A264735                  MOV     AH,BOOTDRV
 34D2 894702                    MOV     2[BX],AX

                                ; SETUP REGISTERS FOR CALL
 34D5 8CD8                      MOV     AX,DS
 34D7 8EC0                      MOV     ES,AX                   ; SET ES = CCP'S DS; BX = PARM OFFSET
 34D9 8E1E0308                  MOV     DS,.PAG0_OFFSET         ; SET DS = Z80CCP'S DS
 34DD 2EFF1EF907                CALLF   CS:DWORD PTR .IP0_OFFSET        ; RUN Z80CCP.SYS
 34E2 8CCB                      MOV     BX,CS
 34E4 8EDB                      MOV     DS,BX                   ; GET NEW DS VALUE

 34E6 0AC0                      OR      AL,AL                   ; Z80CCP.SYS ERROR?
 34E8 7403          34ED        JZ      CMDRUN
 34EA E971CE        035E        JMP     CCP                     ;   YES - MSG ALREADY PRINTED

                    ; EXECUTE ORIGINAL .CMD FILE
                    CMDRUN:
 34ED E8E2CB        00D2        CALL    INITMEM                 ; RESET BDOS' SEG TABLE

 34F0 BA7635                    MOV     DX,OFFSET PARMFCB
 34F3 B13B                      MOV     CL,LOADF
 34F5 E893CB        008B        CALL    BDOS                    ; LOAD .CMD FILE
 34F8 40                        INC     AX
 34F9 7509          3504        JNZ     CMDRUN1                 ; LOAD ERROR?
 34FB B97B09                    MOV     CX,LOADMSG_OFFSET       ;   YES
 34FE E8AECB        00AF        CALL    PRINT
 3501 E95ACE        035E        JMP     CCP

                    CMDRUN1:
 3504 891E0308                  MOV     .PAG0_OFFSET,BX
 3508 E957D2        0762        JMP     GOUSER1
```



```
                                EJECT
                    ;*****************************************************************
                    ;**     PATCH TO ALLOW SUBMIT FILES TO USE OTHER DISKS THAN     **
                    ;**     A FOR SCRATCH FILE (SUBMIT.$$$)                         **
                    ;*****************************************************************
                    DELSUBPA:
 350B A0B724                    MOV     AL,BYTE PTR .CURDRVS
 350E E8C9CB        00DA        CALL    SELECT
 3511 E9EECC        0202        JMP     DELSUBR
```



```
                                EJECT
                    ;*****************************************************************
                    ;**     PATCHES FOR BDOS CONIN STATUS MAINTENANCE IN DATA BLOCK **
                    ;*****************************************************************

                    CONIN_PATCH:
```

```
3514 C60700                     MOV     BYTE PTR 0[BX],0        ; PATCHED BDOS INSTRUCTION
3517 1E                         PUSH    DS                      ; SET DS TO ADDRESS SHARED RAM
3518 53                         PUSH    BX
3519 33DB                       XOR     BX,BX
351B 8EDB                       MOV     DS,BX
351D 2E8B1E4725                 MOV     BX,CS: .BIOS_OFFSET+PBADR_OFFSET  ; DATA BLOCK ADDRESS
3522 80A7E702FD                 AND     BYTE PTR .XCSFLAG[BX]',NOT BDCS   ; RESET BDOS IN BIT
3527 5B                         POP     BX
3528 1F                         POP     DS
3529 E962DE       138E          JMP     CONIN_RET


                        CONB0_PATCH:
352C A2E622                     MOV     .KBCHAR_OFFSET,AL       ; PATCHED BDOS INSTRUCTION
352F 1E                         PUSH    DS                      ; SAME COMMENTS AS ABOVE, EXCEPT
3530 53                         PUSH    BX                      ;  THAT BIT IS SET
3531 33DB                       XOR     BX,BX
3533 8EDB                       MOV     DS,BX
3535 2E8B1E4725                 MOV     BX,CS: .BIOS_OFFSET+PBADR_OFFSET
353A 808FE70202                 OR      BYTE PTR .XCSFLAG[BX]',BDCS
353F 5B                         POP     BX
3540 1F                         POP     DS
3541 E9C6DE       140A          JMP     CONB0_RET
```



```
                                EJECT
  3544                  DATA_OFFSET     EQU     OFFSET $
                                        DSEG
                                        ORG     DATA_OFFSET

3544 434F4D             COMTYPE         DB      'COM'
3547                    BOOTDRV         RB      1
3548 00                 ZSYSFLAG        DB      0
3549                    ZSYSPARM        RB      4               ; PARAMETER BLOCK FOR Z80CCP.SYS

354D 534156452020       SAVEFCB         DB      'SAVE    '
     2020
3555 005A38304343       ZSYSFCB         DB      0,'Z80CCP  ','SYS'
     502020535953
3561 000000000000                       DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
     000000000000
     000000000000
     000000
3576                    PARMFCB         RB      33

3597 0D0A54686520       NOZ80CCP        DB      CR,LF,'The file Z80CCP.SYS Not Found on Boot Disk",0
     66696C65205A
     38304343502E
     535953204E6F
     7420466F756E
     64206F6E2042
     6F6F74204469
     736B00
```

```
35C4 0D0A54686520       LOADZ80ERR      DB      CR,LF,'The file Z80CCP.SYS Cannot be Loaded',0
     66696C65205A
     38304343502E
     535953204361
     6E6E6F742062
     65204C6F6164
     656400

                                        END

END OF ASSEMBLY.  NUMBER OF ERRORS:   0.  USE FACTOR: 12%
```

```
        TITLE   'CONFIG SYSTEM; INIT FOR .COM'

; THIS MODULE WRITTEN FOR DEC RAINBOW 100
;                       BY CPL
;                        JULY 1982

; THIS MODULE IS LOADED BY CP/M-86/80 CCP TO
;       A) RECONFIGURE THE SYSTEM PRIOR TO EXECUTION OF A .COM FILE,
;       B) EXECUTE A .COM FILE,
;    OR C) RECONFIGURE THE SYSTEM PRIOR TO RESUMED EXECUTION OF .CMD FILES.

; UPON ENTRY:
;       ES = SEGMENT BASE OF CP/M
;       BX = OFFSET TO PARAMETER BLOCK

; PARAMETER BLOCK:
;    BYTES 0-1  OFFSET TO .COM FILE FCB
;              2  CURRENT DISK DRIVE
;              3  BOOT DRIVE

; PARSED COMMAND LINE IS IN FCB LOCATION FOR THIS MODULE
```

```
FFFF                TRUE    EQU     -1
0000                FALSE   EQU     NOT TRUE

CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM


                        EJECT

                    ; MEMORY CONFIGURATION #1 (FOR 8088 .CMD FILES)
                    ;
0100                PRMTV1_ADR      EQU     0100H           ; STARTING ADDRESS FOR PRMTVPVT.SYS
                    ;
0040                CPM1_SEG        EQU     0040H           ; STARTING SEGMENT OF CP/M-86/80
3600                CPM1_LEN        EQU     3600H           ; MAX LENGTH (IN BYTES) OF CP/M-86/80
3A00                PB1_ADR         EQU     (CPM1_SEG*16)+CPM1_LEN
                                                            ; STARTING ADDRESS OF POINTERS/
                                                            ;   BUFFERS DATA BLOCK
0300                PB1_LEN         EQU     0300H           ; LENGTH (IN BYTES) OF POINTERS/
                                                            ;   BUFFERS DATA BLOCK
                    ;
03D0                TPABEG1_SEG     EQU     (PB1_ADR+PB1_LEN)/16
                                                            ; STARTING SEGMENT OF TPA
1000                TPAEND1_SEG     EQU     1000H           ; ENDING SEGMENT OF TPA + 1
0C30                TPA1_LEN        EQU     TPAEND1_SEG-TPABEG1_SEG
                                                            ; LENGTH (IN PARAGRAPHS) OF TPA


                    ; MEMORY CONFIGURATION #2 (FOR Z80 .COM FILES)
```

```
                        ;
0300                    PB2_LEN         EQU     PB1_LEN         ; LENGTH (IN BYTES) OF POINTERS/
                                                                ;   BUFFERS DATA BLOCK
FD00                    PB2_ADR         EQU     (1000H-(PB2_LEN/16))*16
                                                                ; STARTING ADDRESS OF POINTERS/
                                                                ;   BUFFERS DATA BLOCK
F600                    PCPM_ADR        EQU     0F600H          ; STARTING ADDRESS OF PSEUDO CP/M
F603                    PRMTV2_ADR      EQU     PCPM_ADR+3      ; STARTING ADDRESS FOR PRMTVSHR.SYS
                        ;
3600                    CPM2_LEN        EQU     CPM1_LEN        ; MAX LENGTH (IN BYTES) OF CP/M-86/80
0C00                    CPM2A_SEG       EQU     (PCPM_ADR-CPM2_LEN)/16
                                                                ; STARTING SEGMENT OF CP/M-86/80 --
                                                                ;   FOR SMALL SYSTEM
1000                    CPM2B_SEG       EQU     1000H           ; STARTING SEGMENT OF CP/M-86/80 --
                                                                ;   FOR LARGE SYSTEM
                        ;
0040                    TPABEG2_SEG     EQU     0040H           ; STARTING SEGMENT OF TPA
0C00                    TPAEND2A_SEG    EQU     CPM2A_SEG       ; ENDING SEGMENT OF TPA + 1 --
                                                                ;   FOR SMALL SYSTEM
0BC0                    TPA2A_LEN       EQU     TPAEND2A_SEG-TPABEG2_SEG
                                                                ; LENGTH (IN PARAGRAPHS) OF TPA --
                                                                ;   FOR SMALL SYSTEM
0F60                    TPAEND2B_SEG    EQU     PCPM_ADR/16     ; ENDING SEGMENT OF TPA + 1 --
                                                                ;   FOR LARGE SYSTEM
0F20                    TPA2B_LEN       EQU     TPAEND2B_SEG-TPABEG2_SEG
                                                               ; LENGTH (IN PARAGRAPHS) OF TPA --
                                                                ;   FOR LARGE SYSTEM
```

```
                        ; ************************************************************
CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM



   2202                 BIOSCS_OFFSET   EQU     2202H           ; OFFSET FROM START OF CPM WHERE
                                                                ;  WHERE BDOS STASHES BIOS' CS:
   2500                 BIOS_OFFSET     EQU     2500H           ; OFFSET (FROM START OF CP/M-86/80)
                                                                ;   OF BIOS


=                                   INCLUDE DEFBUF.LIB
=                       ; ************************************************************************
=                       ; OFFSETS FROM START OF POINTERS/BUFFERS DATA BLOCK
=
=  FFA0                 XDPCX           EQU     -60H            ; DISK PARAMETER STORAGE (60H)
=  0000                 XDEFBUF         EQU     000H            ; MISC. BUFFER (LENGTH=80H)
=  0086                 XPACKET         EQU     086H            ; BIOS MESSAGE PACKET (LENGTH=0EH)
=  0086                 XSTPKT          EQU     086H            ; START PACKET BUFFER (LENGTH=0EH)
=  0094                 XADCPKT         EQU     094H            ; DATA PACKET (LENGTH=0EH)
=  0094                 XMVPKT          EQU     094H            ; MOVE PACKET BUFFER (LENGTH=0EH)
=  00A2                 XSHRBUF         EQU     0A2H            ; SEGMENT BUFFER (LENGTH=200H)
=  02F8                 XMEMSIZE        EQU     2F8H            ; MEMORY SIZE (WORD)
=  02FA                 XPCPMADR        EQU     2FAH            ; PSEUDO CP/M ADDRESS (WORD)
=  02FC                 XZ80PKT         EQU     2FCH            ; PACKET POINTER FROM Z80 (WORD)
=  02FE                 XI88PKT         EQU     2FEH            ; PACKET POINTER FROM 8088 (WORD)
```

```
=  02F0                 XTTRACK         EQU     2F0H            ;TRACK TABLE
=  02F4                 XTFORMAT        EQU     2F4H            ; FORMAT TABLE
=  02E7                 XCSFLAG         EQU     2E7H            ; CONSOLE STATUS FLAG
=
=                       ; OFFSETS FROM ZOT FOR CONVENIENCE
=  0000                 ZOTP            EQU     0               ; Z80 FLAG
=  FFFE                 Z80FLAGPT       EQU     -2              ; Z80-RUNNING FLAG
=  FFFB                 CICCK           EQU     -5              ; CONSOLE STATUS FLAG CHECK
=
=                       ; OTHER USEFUL EQUATES
=  0002                 BDCS            EQU     2               ; BDOS CHARACTER READY BIT
=  0001                 BIOCS           EQU     1               ; BIOS CONSOLE STATUS BIT
=  0017                 BIOS_JMPS       EQU     23              ; NUMBER OF FUNCTIONS IN JUMP TABLE
=
=                       ; *************************************************************************


   0045                 SEGTBL_OFFSET   EQU     BIOS_JMPS*3     ; OFFSET (FROM START OF BIOS)
                                                                ;   OF WORD WITH OFFSET
                                                                ;   (FROM START OF CP/M) TO
                                                                ;   MEMORY SEGMENT TABLE
   0047                 PBADR_OFFSET    EQU     SEGTBL_OFFSET+2 ; OFFSET (FROM START OF BIOS)
                                                                ;   OF WORD THAT CONTAINS
                                                                ;   ADDRESS OF POINTERS/BUFFERS
                                                                ;   DATA BLOCK


   02FE                 I88PKT_OFFSET   EQU     XI88PKT         ; OFFSET (FROM START OF PTRSBUFS)
```

```
                                                        ;   OF PACKET POINTER FROM 8088
02FC                Z80PKT_OFFSET   EQU     XZ80PKT     ; OFFSET (FROM START OF PTRSBUFS)
                                                        ;   OF PACKET POINTER FROM Z80
CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM


02FA                PCPMADR_OFFSET  EQU     XPCPMADR    ; OFFSET (FROM START OF PTRSBUFS)
                                                        ;   OF WORD CONTAINING ADDRESS
                                                        ;   OF PSEUDO CP/M
                                                        ;   0=PSEUDO CP/M NOT LOADED
02F8                MEMSIZE_OFFSET  EQU     XMEMSIZE    ; OFFSET (FROM START OF PTRSBUFS)
                                                        ;   OF WORD INDICATING
                                                        ;   #PARAGRAPHS ADDITIONAL MEMORY
0094                MOVEPKT_OFFSET  EQU     XMVPKT      ; OFFSET (FROM START OF PTRSBUFS)
                                                        ;   OF PACKET BUFFER TO BE USED FOR
                                                        ;   Z80 MOVE FUNCTION
0022                MOVEFC          EQU     022H        ; MOVE FUNCTION CODE
0086                STARTPKT_OFFSET EQU     XSTPKT      ; OFFSET (FROM START OF PTRSBUFS)
                                                        ;   OF PACKET BUFFER TO BE USED FOR
                                                        ;   Z80 START FUNCTION
0021                STARTFC         EQU     021H        ; START Z80 FUNCTION CODE
0100                STARTCOM        EQU     100H        ; .COM FILE STARTING ADDRESS


0100                PBIOS_OFFSET    EQU     0100H       ; OFFSET (FROM START OF PSEUDO CP/M)
                                                        ;   TO PSEUDO BIOS JUMP TABLE
0000                PBDOS_OFFSET    EQU     0           ; OFFSET (FROM START OF PSEUDO CP/M)
                                                        ;   TO PSEUDO BDOS ENTRY POINT
```

```
F700            PBIOS_ADR       EQU     PCPM_ADR+PBIOS_OFFSET
                                                ; ADDRESS OF PSEUDO BIOS JUMP TABLE
F600            PBDOS_ADR       EQU     PCPM_ADR+PBDOS_OFFSET
                                                ; ADDRESS OF PSEUDO BDOS ENTRY POINT


00C3            JUMPINST        EQU     0C3H            ; Z80 JUMP INSTRUCTION
005C            FCB_OFFSET      EQU     05CH            ; FCB OFFSET


0000            RELO_OFFSET     EQU     0               ; LOCATION IN 8088 PRIVATE MEMORY
0040            RELO_SEG        EQU     40H             ;   TO RELOCATE .COM-FILE-LOAD CODE


                ; INTERRUPTS
                ;
00E0            BDOS            EQU     224
                ;
0092            TYPE36SEG_ADR           EQU     36*4+2
0096            TYPE37SEG_ADR           EQU     37*4+2
009C            TYPE39OFFSET_ADR        EQU     39*4
009E            TYPE39SEG_ADR           EQU     39*4+2
00B0            TYPE44OFFSET_ADR        EQU     44*4
00B2            TYPE44SEG_ADR           EQU     44*4+2
0190            TYPE100OFFSET_ADR       EQU     100*4
0192            TYPE100SEG_ADR          EQU     100*4+2
0382            TYPE224SEG_ADR          EQU     224*4+2
```

```
   000D                 CR              EQU     0DH
CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM


   000A                 LF              EQU     0AH


                        ; BDOS FUNCTION NUMBERS
                        ;
   0007                 FIOB            EQU     7               ; GET I/O BYTE
   0009                 FPRINT          EQU     9               ; PRINT STRING
   000F                 FOPEN           EQU     15              ; OPEN FILE
   0014                 FREAD           EQU     20              ; READ SEQUENTIAL
   001A                 FDMA            EQU     26              ; SET DMA ADDRESS
   0033                 FDMAB           EQU     51              ; SET DMA SEGMENT BASE


CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM


                                EJECT

                        ; ****************************************************************
                        ; ****************************************************************

                                CSEG
```

```
                        ORG     0

                ; THE FOLLOWING CODE IS RELOCATED TO PRIVATE 8088 MEMORY BEFORE
                ; BEING EXECUTED

                RELOBEG:                                ; MUST BE AT OFFSET 0
0000 E9DE03     03E1            JMP     START
0003 90                         NOP
0004 90                         NOP
0005 90                         NOP

=                       INCLUDE Z80SVC.LIB
=               ;************************************
=               ;*                                  *
=               ;*        Z80 SERVICE ROUTINE       *
=               ;*        IN 8088                   *
=               ;*        FOR PC-100                *
=               ;*                                  *
=               ;************************************
=
= 009E          BDOS_SEG        EQU     39*4+2
= 009C          PZOT    EQU     39*4
= FD00          DEFBUF  EQU     PB2_ADR + XDEFBUF
= FD86          PACKET  EQU     PB2_ADR + XPACKET
= FD94          ADCPAC  EQU     PB2_ADR + XADCPKT
= 0090          FDOSM   EQU     90H
= 0028          MAXBDOS EQU     28H                     ;Highest BDOS function serviced
```

```
=  000F                 DPBLEN  EQU     15                      ;Length of disk parameter block
=  0019                 ALVLEN  EQU     25                      ;Length of allocation vector
=  0028                 TRNLEN  EQU     40                      ;Length of maximum translation table
=  FCA0                 PSDPH   EQU     PB2_ADR+XDPCX           ;Pseudo-DPH
=  FCB0                 PSTRN   EQU     PSDPH+16                ;Pseudo translate table
=  FCD8                 PSDPB   EQU     PSTRN+TRNLEN            ;Pseudo-DPB
=  FCE7                 PSALV   EQU     PSDPB+DPBLEN            ;Pseudo-ALV
=
=                       ;***************************************
=                       ;*                                     *
=                       ;* Wait for a call from Z80 to bios or *
=                       ;* bdos, then set bx to point to the   *
=                       ;* packet.                             *
=                       ;*                                     *
=                       ;***************************************
=                       WAIT_FOR_CALL:
=0006 E84203       034B         CALL WAIT39                     ;Actual wait loop
=                       SERVICE_CALL:
=0009 83EB04                    SUB BX,4                        ;BX was set to packet pointer+4
CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM


=000C 268B1F                    MOV BX,ES:[BX]                  ;Put packet address in BX
=                                                               ;Then fall through
=
=
=                       ;***************************************
=                       ;*                                     *
```

```
=                       ;* subroutine to unpack data packet,    *
=                       ;* call appropriate BIOS or BDOS routine*
=                       ;* then repack data to return to Z80    *
=                       ;* operation.                           *
=                       ;*                                      *
=                       ;****************************************
=
=                       PACKIN:
=000F 1E                        PUSH DS                 ;SAVE CURRENT DS
=0010 33C0                      XOR AX,AX
=0012 8ED8                      MOV DS,AX               ;Zero data segment
=0014 2E891E9702                MOV MAILBOX,BX          ;save packet address
=0019 8BF3                      MOV SI,BX               ;Packet address is in MAILBOX
=001B 8A5408                    MOV DL,08[SI]           ;IOBYTE in packet +8
=001E 2E3A169602                CMP DL,IOBLCL           ;check for changed iobyte
=0023 740E          0033        JE PACK1                ;skip if it's ok
=0025 2E88169602                MOV IOBLCL,DL           ;else save it as updated
=002A B108                      MOV CL,8                ;BDOS SET IOBYTE
=002C CDE0                      INT 224                 ;call BDOS
=002E 2E8B369702                MOV SI,MAILBOX          ;restore SI (jic)
=0033 8A24              PACK1:  MOV AH,[SI]             ;function code in AH, by the way
=0035 8B4C02                    MOV CX,02[SI]           ;same as Z80 BC
=0038 8B5404                    MOV DX,04[SI]           ;same as Z80 DE
=
=003B 80FC90                    CMP AH,FDOSM            ;is this a BDOS call?
=003E 7445          0085        JE BDOS80               ;go do BDOS  if so
=                       BIOS80:                         ;else fall through to BIOS
```

```
=0040 80EC40                    SUB AH,40H              ;remove function offset
=0043 B500                      MOV CH,0                ;possible word needed
=0045 80FC01                    CMP AH,1                ;WARM BOOT?
=0048 7503          004D        JNE BIOXX
=004A E9D100        011E        JMP XINI                ;go away if so
=004D 80FC10            BIOXX:  CMP AH,10H              ;SECTRAN?
=0050 7502          0054        JNE BIOS80A             ;SKIP IF NOT
=0052 EB18          006C        JMPS YOK                ;Z80 handles sectran.
=
=                       BIOS80A:
=0054 80FC09                    CMP AH,9                ;SELDSK?
=0057 7503          005C        JNE BIOS80B             ;skip if not
=0059 E93001        018C        JMP BSELD               ;go do table move if so
=
=                       BIOS80B:
=005C 886401                    MOV 01[SI],AH           ;BIOS function number.  We are going
=005F 8BD6                      MOV DX,SI               ;to cheat and use BDOS call 50
=0061 42                        INC DX                  ;and the original packet for the
=0062 B93200                    MOV CX,50               ;BIOS descripter block.
CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM


=0065 CDE0                      INT 224                 ;DO BDOS direct BIOS call.
=0067 2E891E9402        PACKUP: MOV BHLD,BX             ;SAVE BX REG. (Z80 HL)
=
=                       YOK:                            ;RETURN FROM BDOS
=                                                       ;OR BIOS
=006C 2E8B1E9402                MOV BX,BHLD             ;RESTORE BX REG. (Z80 HL)
```

```
=0071 2E8B369702              MOV SI,MAILBOX          ;PACKET ADDRESS
=0076 884401                  MOV 01[SI],AL
=0079 895C06                  MOV 06[SI],BX
=007C 8BCE                    MOV CX,SI               ;PACKET ADDRESS IN CX
=007E 1F                      POP DS                  ;
=007F E89102       0313       CALL XFERPKT            ;SEND THE PACKET
=0082 E984FF       0009       JMP SERVICE_CALL        ;GO WAIT FOR ANOTHER ONE
=
=
=                     BDOS80:                         ;DO A BDOS FUNCTION
=0085 51              BDLP1:  PUSH CX                 ;SAVE THE FUNCTION
=0086 B500                    MOV CH,0                ;MAKE A JUMP INSTRUCTION
=0088 03C9                    ADD CX,CX               ;
=008A BBF001                  MOV BX,OFFSET BDJMPS    ;BEGINNING OF TABLE
=008D 03D9                    ADD BX,CX               ;DIRECTOR
=008F 59                      POP CX                  ;GET THE FUNCTION BACK
=0090 2EFFA70000              JMP CS: WORD PTR [BX]   ;GO DO PRE-CONDITIONING
=
=
=                     XADCS:                          ;WE HAVE A STRING
=0095 2EC706990280            MOV ADCLEN,128          ;OF 128 BYTES OR LESS
      00
=009C EB07         00A5       JMPS XADC0
=
=                     XADC:                           ;PRECOMPENSATION
=009E 2EC706990224            MOV ADCLEN,36           ;FOR FCB (36 BYTES)
      00
```

```
=
=                       XADC0:
=00A5 2E89169B02                MOV ADCFLG,DX           ;SAVE THE ABSOLUTE ADDRESS
=00AA 50                        PUSH AX                 ;
=00AB 33C0                      XOR AX,AX               ;
=00AD 8ED8                      MOV DS,AX
=00AF 81FA0008                  CMP DX,800H
=00B3 732A          00DF        JAE XADC1               ;JUMP IF ALL IS WELL
=00B5 2EC7069D0200              MOV CS: WORD PTR ADCDES,DEFBUF  ;DEST ADDRESS
      FD
=00BC E83A00        00F9        CALL ZADC
=00BF 80F90A                    CMP CL,10               ;LIMIT STRINGS IN AND OUT TO 128 BYTES
=00C2 7510          00D4        JNZ XADC01              ;-NOT READ CONSOLE STRING
=00C4 26803E00FD7E              CMP ES: BYTE PTR .DEFBUF,126    ;MAX .GT. 126?
=00CA 7213          00DF        JB XADC1                ;NO, SKIP AHEAD
=00CC 26C60600FD7E              MOV ES: BYTE PTR .DEFBUF,126    ;YES, MAKE IT 126.
=00D2 EB0B          00DF        JMPS XADC1              ;THEN SKIP AHEAD
=00D4 80F909            XADC01: CMP CL,9                ;PRINT STRING?
=00D7 7506          00DF        JNZ XADC1               ;NO, SKIP AHEAD
CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM


=00D9 26C6067FFD24              MOV ES: BYTE PTR .DEFBUF+127,'$'        ;YES, IMPLANT END-STRING (JIC)
=
=00DF 58                XADC1:  POP AX
=
=                       XOK:
=00E0 51                        PUSH CX                 ;SAVE FUNCTION CODE
```

```
=00E1 52                        PUSH DX                 ;SAVE ADDRESS FOR ADC
=00E2 CDE0                      INT 224                 ;DO BDOS FUNCTION
=00E4 5A                        POP DX                  ;GET ADDRESS BACK
=
=                                                       ;CONTINUE AFTER BDOS
=00E5 2E891E9402                MOV BHLD,BX             ;SAVE BX REG (Z80 HL)
=00EA 59                        POP CX                  ;RESTORE FUNCTION CODE
=00EB B500                      MOV CH,0                ;MAKE 16 BIT NR
=00ED 03C9                      ADD CX,CX               ;DIRECTOR
=00EF BB4202                    MOV BX,OFFSET BDYJMPS   ;TO ROUTINE
=00F2 03D9                      ADD BX,CX               ;OFFSET TO ROUTINE
=00F4 2EFFA70000                JMP CS: WORD PTR [BX]   ;DO POST-CONDITIONING
=
=                       ZADC:   ;COMMON ROUTINE TO MOVE A BLOCK.  SOURCE IS IN DX, DEST AND LENGTH
=                               ;IN ADCDES AND ADCLEN RESPECTIVELY
=00F9 53                        PUSH BX                 ;MOVING TO DO
=00FA 51                        PUSH CX
=00FB BB94FD                    MOV BX,ADCPAC   ;GET PACKET ADDRESS
=00FE C60722                    MOV BYTE PTR [BX],22H   ;MOVE FUNCTION
=0101 895702                    MOV 02[BX],DX           ;SOURCE IN DX ON ENTRY
=0104 2E8B169D02                MOV DX,ADCDES           ;DESTINATION
=0109 895704                    MOV 04[BX],DX           ;ADDRESS
=010C 52                        PUSH DX                 ;WE WANT THIS BACK
=010D 2E8B169902                MOV DX,ADCLEN           ;LENGTH
=0112 895706                    MOV 06[BX],DX           ;FROM ENTRY VARIABLE
=0115 8BCB                      MOV CX,BX               ;INSTRUCTION TO INTERFACE
=0117 E8F901       0313         CALL XFERPKT            ;CALL INTERFACE
```

```
=011A 5A                    POP DX              ;RESTORE REGS
=011B 59                    POP CX
=011C 5B                    POP BX
=011D C3                    RET
=
=011E E8E501      0306 XINI:  CALL RESETZFLAG     ;
=0121 33C0                  XOR AX,AX           ;
=0123 B200                  MOV DL,0            ;CODE TO RELEASE MEMORY
=0125 B100                  MOV CL,0            ;RESET SYSTEM
=0127 EBB7        00E0      JMPS XOK
=
=0129 80FAFF          XDCI:   CMP DL,0FFH         ;CPM-80 USES FF FOR STATUS AND INPUT
=012C 7402        0130      JZ XDCI1            ;SKIP AHEAD IF FF
=012E EBB0        00E0      JMPS XOK            ;ELSE EXIT
=0130 51          XDCI1:  PUSH CX             ;SAVE THE CX REG
=0131 FECA                  DEC DL              ;CL=0FEH FOR STATUS
=0133 CDE0                  INT 224             ;BDOS CALL
=0135 59                    POP CX              ;RESTORE THE CALL NO.
=0136 0AC0                  OR AL,AL            ;CHECK FOR AL=0
CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM


=0138 7503        013D      JNZ XDCI2           ;SKIP IF NON-ZERO
=013A E92FFF      006C      JMP YOK             ;ELSE ALL DONE, AL=0
=013D B2FF          XDCI2:  MOV DL,0FFH         ;CHARACTER IS READY, GO GET IT
=013F EB9F        00E0      JMPS XOK            ;FROM BDOS
=
=                   YADC:                       ;POST-COMPENSATION
```

```
=0141 2E3B169B02              CMP DX,ADCFLG          ;DID WE HAVE TO DO PRECOMPENSATION?
=0146 7503         014B       JNE YADC1              ;
=0148 E921FF       006C       JMP YOK                ;EXIT IF NOT
=014B 50           YADC1:     PUSH AX                ;ELSE SAVE REGISTERS
=014C 52                      PUSH DX
=014D 2E8B169B02              MOV DX,ADCFLG          ;GET OLD DESTINATION
=0152 2E89169D02              MOV ADCDES,DX          ;
=0157 5A                      POP DX
=0158 E89EFF       00F9       CALL ZADC
=015B 58           YADC2:     POP AX
=015C E90DFF       006C       JMP YOK                ;BYE.
=
=015F 0AC0         YGCS:      OR AL,AL               ;WE NEED ZERO OR
=0161 7402         0165       JZ YCG1                ;...
=0163 B0FF                    MOV AL,0FFH            ;FF.
=0165 E904FF       006C YCG1: JMP YOK
=
=0168 2EA19402     YLIV:      MOV AX,BHLD            ;CPM-80 EXPECTS A TO DUPLICATE H.
=016C E9FDFE       006C       JMP YOK
=
=016F B90F00       YDPB:      MOV CX,DPBLEN          ;Move DPB to pseudo
=0172 B8D8FC                  MOV AX,PSDPB
=0175 EB06         017D       JMPS YALVX
=
=0177 B91900       YALV:      MOV CX,ALVLEN          ;Move ALV to pseudo
=017A B8E7FC                  MOV AX,PSALV
=017D 2E8B1E9402   YALVX:     MOV BX,BHLD            ;Restore BX
```

```
=0182 2EA39402                  MOV BHLD,AX             ;Give user pseudo-ALV address
=0186 E85100        01DA        CALL YDPM
=0189 E9E0FE        006C        JMP YOK
=
=018C 886401              BSELD:  MOV 01[SI],AH         ;BIOS function number.  We are going
=018F 8BD6                        MOV DX,SI             ;to cheat and use BDOS call 50
=0191 42                          INC DX                ;and the original packet for the
=0192 B93200                      MOV CX,50             ;BIOS descripter block.
=0195 CDE0                        INT 224               ;DO BDOS direct BIOS call.
=0197 0BDB                        OR BX,BX              ;Check for zero here
=0199 7503          019E          JNZ BSELD1            ;Skip if non-zero
=019B E9C9FE        0067          JMP PACKUP            ;Else exit with zero to mark error
=019E 1E                  BSELD1: PUSH DS               ;Need it for later
=019F 52                          PUSH DX               ;Save for reference
=01A0 53                          PUSH BX               ;BX has DPH offset
=01A1 268B1F                      MOV BX,ES:[BX]        ;Get translate table address
=01A4 B92800                      MOV CX,TRNLEN
=01A7 B8B0FC                      MOV AX,PSTRN          ;Pseudo-translate address
=01AA E82D00        01DA          CALL YDPM             ;Do move
CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM


=01AD 5B                          POP BX                ;DPH address
=01AE 83C30A                      ADD BX,10             ;Get DPB address
=01B1 53                          PUSH BX
=01B2 268B1F                      MOV BX,ES:[BX]
=01B5 B90F00                      MOV CX,DPBLEN
=01B8 B8D8FC                      MOV AX,PSDPB
```

```
=01BB E81C00       01DA         CALL YDPM
=01BE 5B                        POP BX
=01BF 83C304                    ADD BX,4                ;Get ALV address
=01C2 268B1F                    MOV BX,ES:[BX]
=01C5 B91900                    MOV CX,ALVLEN
=01C8 B8E7FC                    MOV AX,PSALV
=01CB E80C00       01DA         CALL YDPM
=01CE 2EC7069402A0              MOV BHLD,PSDPH          ;Furnish user with pseudo-DPH
     FC
=01D5 5A                        POP DX                  ;Reference
=01D6 1F                        POP DS                  ;DS=0
=01D7 E992FE       006C         JMP YOK          ;go back to z80
=
=01DA 8BF3             YDPM:    MOV SI,BX               ;Routine to move a block
=01DC 1E                        PUSH DS                 ;       ENTRY - CX has byte count
=01DD 06                        PUSH ES                 ;               BX has source offset
=01DE 8BF8                      MOV DI,AX               ;               AX has destination (absolute)
=01E0 33C0                      XOR AX,AX               ;               ES has source segment
=01E2 2E893E9402                MOV BHLD,DI             ;
=01E7 1F                        POP DS                  ;(shift former es to ds)
=01E8 1E                        PUSH DS                 ;save to restore es
=01E9 8EC0                      MOV ES,AX
=01EB F3A4                      REP MOVS AL,AL          ;move bytes
=01ED 07                        POP ES                  ;Restore es and ds
=01EE 1F                        POP DS
=01EF C3                        RET
=
```

```
=01F0 1E01              BDJMPS  DW      XINI    ;0
=01F2 E000                      DW      XOK     ;1
=01F4 E000                      DW      XOK     ;2
=01F6 E000                      DW      XOK     ;3
=01F8 E000                      DW      XOK     ;4
=01FA E000                      DW      XOK     ;5
=01FC 2901                      DW      XDCI    ;6
=01FE E000                      DW      XOK     ;7
=0200 E000                      DW      XOK     ;8
=0202 9500                      DW      XADCS   ;9
=0204 9500                      DW      XADCS   ;10
=0206 E000                      DW      XOK     ;11
=0208 E000                      DW      XOK     ;12
=020A E000                      DW      XOK     ;13
=020C E000                      DW      XOK     ;14
=020E 9E00                      DW      XADC    ;15
=0210 9E00                      DW      XADC    ;16
=0212 9E00                      DW      XADC    ;17
=0214 E000                      DW      XOK     ;18
CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM


=0216 9E00                      DW      XADC    ;19
=0218 9E00                      DW      XADC    ;20
=021A 9E00                      DW      XADC    ;21
=021C 9E00                      DW      XADC    ;22
=021E 9E00                      DW      XADC    ;23
=0220 E000                      DW      XOK     ;24
```

```
=0222 E000                      DW      XOK     ;25
=0224 E000                      DW      XOK     ;26
=0226 E000                      DW      XOK     ;27
=0228 E000                      DW      XOK     ;28
=022A E000                      DW      XOK     ;29
=022C 9E00                      DW      XADC    ;30
=022E E000                      DW      XOK     ;31
=0230 E000                      DW      XOK     ;32
=0232 9E00                      DW      XADC    ;33
=0234 9E00                      DW      XADC    ;34
=0236 9E00                      DW      XADC    ;35
=0238 9E00                      DW      XADC    ;36
=023A E000                      DW      XOK     ;37
=023C E000                      DW      XOK     ;38     ?
=023E E000                      DW      XOK     ;39     ?
=0240 9E00                      DW      XADC    ;40
=
=0242 6C00              BDYJMPS DW      YOK     ;0
=0244 6C00                      DW      YOK     ;1
=0246 6C00                      DW      YOK     ;2
=0248 6C00                      DW      YOK     ;3
=024A 6C00                      DW      YOK     ;4
=024C 6C00                      DW      YOK     ;5
=024E 6C00                      DW      YOK     ;6
=0250 6C00                      DW      YOK     ;7
=0252 6C00                      DW      YOK     ;8
=0254 4101                      DW      YADC    ;9
```

```
=0256 4101                      DW      YADC    ;10
=0258 5F01                      DW      YGCS    ;11
=025A 6C00                      DW      YOK     ;12
=025C 6C00                      DW      YOK     ;13
=025E 6C00                      DW      YOK     ;14
=0260 4101                      DW      YADC    ;15
=0262 4101                      DW      YADC    ;16
=0264 4101                      DW      YADC    ;17
=0266 6C00                      DW      YOK     ;18
=0268 4101                      DW      YADC    ;19
=026A 4101                      DW      YADC    ;20
=026C 4101                      DW      YADC    ;21
=026E 4101                      DW      YADC    ;22
=0270 4101                      DW      YADC    ;23
=0272 6801                      DW      YLIV    ;24
=0274 6C00                      DW      YOK     ;25
=0276 6C00                      DW      YOK     ;26
=0278 7701                      DW      YALV    ;27
=027A 6C00                      DW      YOK     ;28
CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM


=027C 6C00                      DW      YOK     ;29
=027E 4101                      DW      YADC    ;30
=0280 6F01                      DW      YDPB    ;31
=0282 6C00                      DW      YOK     ;32
=0284 4101                      DW      YADC    ;33
=0286 4101                      DW      YADC    ;34
```

```
=0288 4101                      DW      YADC    ;35
=028A 4101                      DW      YADC    ;36
=028C 6C00                      DW      YOK     ;37
=028E 6C00                      DW      YOK     ;38     ?
=0290 6C00                      DW      YOK     ;39     ?
=0292 4101                      DW      YADC    ;40
=
=0294 0000              BHLD    DW      0               ; SAVE LOC FOR BX REG
=0296 00                IOBLCL  DB      0               ;LOCAL COPY OF IOBYTE
=0297 0000              MAILBOX DW      0
=0299 0000              ADCLEN  DW      0
=029B 0000              ADCFLG  DW      0
=029D 0000              ADCDES  DW      0
=029F 09                TMP50   DB      9
=02A0 00000100                  DW      0,01


CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM


                                EJECT

                        RELOSTART:
 02A4 8CC8                      MOV     AX,CS
 02A6 8ED8                      MOV     DS,AX

 02A8 BA0001                    MOV     DX,STARTCOM
```

```
                        LOADCOM:
02AB E84300       02F1          CALL    DMASET                  ; SET DMA ADR
02AE 52                         PUSH    DX
02AF E84700       02F9          CALL    READCOM                 ; READ
02B2 5A                         POP     DX
02B3 3C01                       CMP     AL,01H                  ; EOF?
02B5 741B         02D2          JE      LOADEND                 ;   YES
02B7 81C28000                   ADD     DX,80H
02BB 2E3B16A403                 CMP     DX,MAXADR               ; TPA OVERFLOW?
02C0 7703         02C5          JA      LOADERR                 ;   YES
02C2 E9E6FF       02AB          JMP     LOADCOM

                        LOADERR:
02C5 BAC903                     MOV     DX,OFFSET COMERR
02C8 E82A00       02F5          CALL    SPRINT
02CB E83800       0306          CALL    RESETZFLAG
02CE B8FF00                     MOV     AX,255
02D1 CB                         RETF                            ; RETURN TO CCP WITH ERROR

                        LOADEND:
02D2 BA8000                     MOV     DX,80H                  ; SET DEFAULT BUFFER
02D5 E81900       02F1          CALL    DMASET

02D8 33C0                       XOR     AX,AX
02DA 8EC0                      MOV     ES,AX
02DC BB86FD                     MOV     BX,PB2_ADR+STARTPKT_OFFSET
02DF 26C60721                   MOV     ES:BYTE PTR[BX],STARTFC
02E3 26C747020001               MOV     ES:WORD PTR 2[BX],STARTCOM
```

```
02E9 8BCB                       MOV     CX,BX                   ; CX = PACKET POINTER
02EB E82500         0313        CALL    XFERPKT                 ; EXECUTE .COM FILE
                            LOADSTOP:
02EE E915FD         0006        JMP     WAIT_FOR_CALL           ; GO WAIT FOR BDOS OR BIOS CALL

                            DMASET:
02F1 B11A                       MOV     CL,FDMA
02F3 EB09           02FE        JMPS    INTBDOS

                            SPRINT:
02F5 B109                       MOV     CL,FPRINT
02F7 EB05           02FE        JMPS    INTBDOS

                            READCOM:
02F9 B114                       MOV     CL,FREAD
02FB BAA803                     MOV     DX,OFFSET COMFCB
CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM


                            INTBDOS:
02FE CDE0                       INT     BDOS
0300 C3                         RET

CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM
```

```
                        EJECT

                    ; ***************************************************************
                    ;
                    ; SET Z80-RUNNING-FLAG (THIS IS TO INDICATE TO BDOS THAT THE
                    ; DMA ADDRES IS IN Z80 SPACE)
                    ;
                    ; NOTE: THESE ROUTINES DO NOT CHANGE THE SETTING OF CARRY FLAG
                    ;
                    SETZFLAG:
0301 B8FFFF                 MOV     AX,TRUE
0304 EB02       0308        JMPS    SETZF1

                    RESETZFLAG:
0306 33C0                   XOR     AX,AX

                    SETZF1:
0308 50                     PUSH    AX
0309 E87F00     038B        CALL    FINDZOT
030C 4B                     DEC     BX
030D 4B                     DEC     BX
030E 58                     POP     AX
030F 268907                 MOV     ES:[BX],AX
0312 C3                     RET
```

```
                        EJECT

                ; ***************************************************************
                ;
                ; SEND PACKET AND WAIT FOR Z80 TO FINISH
                ;
                ; ENTRY: CX = PACKET ADDRESS
                ;
                XFERPKT:
0313 2E8E06A203         MOV     ES,ZOTSEG
0318 2E8B1EA003         MOV     BX,ZOTADR
031D 26C7070000         MOV     ES:WORD PTR[BX],FALSE   ; RESET DONE FLAG
0322 E80200     0327    CALL    SENDPKT
0325 EB24       034B    JMPS    WAIT39



                ; ***************************************************************
                ;
                ; SEND PACKET TO Z80
                ;
                ; ENTRY: CX = PACKET ADDRESS
                ;
  0002          GSCR    EQU     2
  0000          INTZ80  EQU     0
  0080          BIT7    EQU     80H
```

```
                        SENDPKT:
0327 2E8E06A603                 MOV     ES,PARM_SEG
032C BB4725                     MOV     BX,BIOS_OFFSET+PBADR_OFFSET
032F 268B1F                     MOV     BX,ES:[BX]
0332 81C3FE02                   ADD     BX,I88PKT_OFFSET

0336 33C0                       XOR     AX,AX
0338 8EC0                       MOV     ES,AX
033A 26890F                     MOV     ES:[BX],CX              ; STORE POINTER

                        ; SIGNAL Z80 THEN WAIT FOR Z80 TO ACKNOWLEDGE
033D E600                       OUT     INTZ80,AL               ; INTERRUPT Z80
                        SENDPK10:
033F E402                       IN      AL,GSCR                 ; GET Z80 STATUS
0341 A880                       TEST    AL,BIT7                 ; INTERRUPT STILL PENDING?
0343 74FA       033F            JZ      SENDPK10                ;   YES - CHECK AGAIN
0345 26C7070000                 MOV     ES: WORD PTR [BX],0     ; THEN ZERO OUT THE PACKET POINTER
034A C3                         RET



                        ; ***************************************************************
                        ;
                        ; WAIT FOR Z80 TO FINISH
                        ;
```

```
                        WAIT39:
034B 2E8E06A203                 MOV     ES,ZOTSEG
0350 2E8B1EA003                 MOV     BX,ZOTADR
                        WAIT39_LOOP:
0355 FA                         CLI
0356 26F707FFFF                 TEST    ES:WORD PTR[BX],TRUE
035B 752C          0389         JNZ     WAIT39_DONE
035D 26F647FBFF                 TEST    ES:BYTE PTR .CICCK[BX]',TRUE
0362 7421          0385         JZ      WAIT39_HALT
0364 06                         PUSH    ES
0365 53                         PUSH    BX
0366 BF0400                     MOV     DI,4                    ;ROM ROUTINE TO GET CRT STATUS
0369 CD28                       INT     40                      ;
036B 84C9                       TEST    CL,CL
036D 740D          037C         JZ      WAIT39_NEXT
036F 33C0                       XOR     AX,AX
0371 8EC0                       MOV     ES,AX
0373 BB00FD                     MOV     BX,PB2_ADR
0376 26808FE70201               OR      ES:BYTE PTR .XCSFLAG[BX]',BIOCS
                        WAIT39_NEXT:
037C 5B                         POP     BX
037D 07                         POP     ES
037E 26C647FB00                 MOV     ES:BYTE PTR .CICCK[BX],0
0383 EBD0          0355         JMPS    WAIT39_LOOP
                        WAIT39_HALT:
0385 FB                         STI
0386 F4                         HLT
```

```
0387 EBCC          0355          JMPS    WAIT39_LOOP
                           WAIT39_DONE:
0389 FB                          STI
038A C3                          RET


                           FINDZOT:
038B 33C0                        XOR     AX,AX
038D 8EC0                        MOV     ES,AX
038F BB9E00                      MOV     BX,TYPE39SEG_ADR
0392 268B07                      MOV     AX,ES:[BX]
0395 BB9C00                      MOV     BX,TYPE39OFFSET_ADR
0398 268B1F                      MOV     BX,ES:[BX]
039B 4B                          DEC     BX
039C 4B                          DEC     BX
039D 8EC0                        MOV     ES,AX
039F C3                          RET                             ; RETURN WITH ZOT = ES:[BX]


CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM


                                 EJECT

                           ; ***************************************************************

03A0                       ZOTADR  RW      1                     ;ADDRESS OF ZOT
03A2                       ZOTSEG  RW      1                     ;SEGMENT OF ZOT
```

```
03A4                    MAXADR  RW      1                       ; LAST DMA ADDRESS BEFORE TPA OVERFLOW
03A6                    PARM_SEG RW     1                       ; SEGMENT FOR CURRENT CPM
03A8                    COMFCB  RB      33                      ; .COM FILE FCB

03C9 0D0A43414E4E       COMERR  DB      CR,LF,'CANNOT LOAD .COM FILE$'
     4F54204C4F41
     44202E434F4D
     2046494C4524


  03E1                  RELOEND EQU     $                       ; END OF CODE TO BE RELOCATED
```



```
                        EJECT

                        ; ***************************************************************
                        ; ***************************************************************

                        START:
03E1 2E8C06A603                 MOV     PARM_SEG,ES
03E6 891E0401                   MOV     PARM_OFFSET,BX

03EA 268B1F                     MOV     BX,ES:[BX]              ; GET FCB OFFSET
03ED 268A470A                   MOV     AL,ES:BYTE PTR 10[BX]   ; GET MIDDLE LETTER OF TYPE
03F1 247F                       AND     AL,07FH                 ; KILL UPPER BYTE
```

```
03F3 3C4D                  CMP     AL,'M'                  ; .CMD FILE?
03F5 7477       046E       JE      CONFGCMD                ;   YES

                      ; .COM FILE WAS REQUESTED

03F7 BB4725                MOV     BX,BIOS_OFFSET+PBADR_OFFSET
03FA 268B1F                MOV     BX,ES:[BX]              ; GET ADR OF DATA BLOCK
03FD 81C3FA02              ADD     BX,PCPMADR_OFFSET       ; GET ADR OF PSEUDO CPM INDICATOR
0401 33C0                  XOR     AX,AX
0403 8EC0                  MOV     ES,AX
0405 268B07                MOV     AX,ES:[BX]
0408 0BC0                  OR      AX,AX                   ; IS PSEUDO CP/M LOADED ALREADY?
040A 7520       042C       JNZ     INITCOM                 ;   YES

                      ; CHANGE FROM CONFIGURATION #1 TO CONFIGURATION #2

040C E8CB00     04DA       CALL    READPCPM                ; READ PSEUDO CP/M AND ITS
                                                           ;  PRIMITIVE ROUTINES
040F 727F       0490       JC      RETERR                  ;   ERROR

0411 E80C01     0520       CALL    MOVEBLOCK1              ; MOVE POINTERS/BUFFERS DATA BLOCK
0414 E82A01     0541       CALL    MOVECPM1                ; MOVE CP/M-86/80

                      ; SET ALL POINTERS/VALUES VALID
0417 E8AA01     05C4       CALL    SETPBADR                ; SET POINTER TO DATA BLOCK IN CP/M
041A E8B501     05D2       CALL    SETPCPM                 ; INDICATE PSEUDO CP/M IS LOADED

                      ; NOW TRANSFER CONTROL TO NEW MODULES
```

```
041D E8D001       05F0          CALL    SETZ80SHR               ; ADJUST Z80 VECTORS AND STACK POINTER
0420 E8F601       0619          CALL    SET8088                 ; ADJUST 8088 VECTORS AND
                                                                ;   CCP RETURN ADDRESS
0423 8ED0                       MOV     SS,AX                   ; ADJUST STACK SEGMENT
0425 A10001                     MOV     AX,NEWCPM_SEG
0428 2EA3A603                   MOV     PARM_SEG,AX

                        ; INITIALIZE Z80 MEMORY AND EXECUTE .COM FILE

                        INITCOM:
042C E85CFF       038B          CALL    FINDZOT                 ; ### VER 1.0.4
042F 2E8C06A203                 MOV     ZOTSEG,ES               ; ### VER 1.0.4
CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM


0434 2E891EA003                 MOV     ZOTADR,BX               ; ### VER 1.0.4
0439 E81302       064F          CALL    INITZL07                ; INITIALIZE Z80 LOCATIONS 0-7
043C E88502       06C4          CALL    INITFCBBUF              ; INITIALIZE FCB AND DEFAULT BUFFER
043F E8B802       06FA          CALL    INITVECTOR              ; INITIALIZE PBDOS VECTOR
0442 E8C902       070E          CALL    SAVEFCB                 ; SAVE .COM FILE FCB

                        ; READY .COM FILE FOR LOADING
0445 BAA803                     MOV     DX,OFFSET COMFCB        ; OPEN .COM FILE
0448 1E                         PUSH    DS
0449 8CC8                       MOV     AX,CS                   ; COMFCB IS IN CSEG
044B 8ED8                       MOV     DS,AX
044D E80D03       075D          CALL    OPENFILE
```

```
0450 1F                     POP     DS

0451 E8ADFE       0301      CALL    SETZFLAG
0454 BA0000                 MOV     DX,0
0457 E80703       0761      CALL    SETDMAB

045A A10E01                 MOV     AX,PCPMVECTOR
045D 2D8000                 SUB     AX,80H
0460 2EA3A403               MOV     MAXADR,AX           ; SAVE MAX LOAD ADR FOR .COM FILE

0464 E8E602       074D      CALL    INIDPH              ;INITIALIZE PSEUDO-DPH

0467 E8CB02       0735      CALL    RELOCATE            ; RELOCATE LOAD CODE TO
                                                        ;   PRIVATE 8088 MEMORY

046A FF2E1001               JMPF    DWORD PTR JMPF_OFFSET   ; LOAD .COM FILE AND EXECUTE IT

                    ; CHANGE FROM CONFIGURATION #2 TO CONFIGURATION #1

                    CONFGCMD:
046E E890FE       0301      CALL    SETZFLAG
0471 E82000       0494      CALL    READPVT             ; READ PRMTVPVT.SYS
0474 E88FFE       0306      CALL    RESETZFLAG
0477 7217         0490      JC      RETERR              ;   ERROR

0479 E8AC00       0528      CALL    MOVEBLOCK2          ; MOVE POINTERS/BUFFERS DATA BLOCK
047C E8F500       0574      CALL    MOVECPM2            ; MOVE CP/M-86/80
```

```
                        ; SET ALL POINTERS/VALUES VALID
 047F E84201    05C4    CALL    SETPBADR            ; SET POINTER TO DATA BLOCK IN CP/M
 0482 E85201    05D7    CALL    RESETPCPM           ; INDICATE PSEUDO CP/M NOT LOADED

                        ; NOW TRANSFTER CONTROL TO NEW MODULES
 0485 E86D01    05F5    CALL    SETZ80PVT           ; ADJUST Z80 VECTORS AND STACK POINTER
 0488 E88E01    0619    CALL    SET8088             ; ADJUST 8088 VECTORS AND
                                                    ;   CCP RETURN ADDRESS
 048B 8ED0              MOV     SS,AX               ; ADJUST STACK SEGMENT

 048D 33C0              XOR     AX,AX
 048F CB                RETF                        ; RETURN TO CCP TO RUN .CMD FILE
CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM



                RETERR:
 0490 B8FF00            MOV     AX,255
 0493 CB                RETF                        ; RETURN TO CCP WITH ERROR

CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM


                        EJECT

                ; ****************************************************************
                ;
```

```
                        ; READ PRMTVPVT.SYS INTO PRIVATE Z80 MEMORY
                        ; (Z80 RUNNING FLAG MUST BE SET BEFORE ENTERING ROUTINE AND MUST
                        ; BE RESET AFTER EXITING)
                        ;
                        ; ENTRY: NONE
                        ;
                        ; EXIT: CF=1 = ERROR (MSG ALREADY PRINTED)
                        ;
                        READPVT:
0494 2E8E06A603                 MOV     ES,PARM_SEG
0499 8B1E0401                   MOV     BX,PARM_OFFSET
049D 268A4703                   MOV     AL,ES:3[BX]             ; GET BOOT DRIVE
04A1 FEC0                       INC     AL
04A3 A22401                     MOV     PRMTVFCB,AL             ; SET FCB DISK DRIVE

04A6 BA2401                     MOV     DX,OFFSET PRMTVFCB
04A9 E8B102         075D        CALL    OPENFILE
04AC FEC0                       INC     AL                      ; GOOD FILE?
04AE 7422           04D2        JZ      READPERR                ;   NO

04B0 BA0000                     MOV     DX,0
04B3 E8AB02         0761        CALL    SETDMAB

04B6 BA0001                     MOV     DX,PRMTV1_ADR
                        READP2:
04B9 E8A902         0765        CALL    SETDMA                  ; SET DMA OFFSET FOR ...
04BC 52                         PUSH    DX
04BD BA2401                     MOV     DX,OFFSET PRMTVFCB
```

```
04C0 E8A602       0769          CALL    READFILE                ; ... NEXT SECTOR READ
04C3 5A                         POP     DX
04C4 3C01                       CMP     AL,01H                  ; EOF?
04C6 7407         04CF          JZ      READPEND                ;   YES
04C8 81C28000                   ADD     DX,80H                  ; ADDRESS FOR NEXT RECORD
04CC E9EAFF       04B9          JMP     READP2

                        READPEND:
04CF 33C0                       XOR     AX,AX                   ; RESET CARRY
04D1 C3                         RET

                        READPERR:
04D2 BA6601                     MOV     DX,OFFSET PRMTVERR
04D5 E89902       0771          CALL    PRTSTRING

04D8 F9                         STC                             ; SET CARRY
04D9 C3                         RET
```



```
                        EJECT

                ; ***************************************************************
                ;
                ; READ PSEUDO CP/M AND ITS PRIMITIVE ROUTINES (Z80.SYS)
                ;
```

```
                        ; ENTRY: NONE
                        ;
                        ; EXIT: CF=1 = ERROR (MSG ALREADY PRINTED)
                        ;
                        ;       CF=0 = NORMAL
                        ;         AX = PSEUDO CP/M LOAD ADDRESS
                        ;
                        READPCPM:
04DA 2E8E06A603                 MOV     ES,PARM_SEG
04DF 8B1E0401                   MOV     BX,PARM_OFFSET
04E3 268A4703                   MOV     AL,ES:3[BX]             ; GET BOOT DRIVE
04E7 FEC0                       INC     AL
04E9 A24501                     MOV     Z80FCB,AL               ; SET FCB DISK DRIVE

04EC BA4501                     MOV     DX,OFFSET Z80FCB
04EF E86B02         075D        CALL    OPENFILE
04F2 FEC0                       INC     AL                      ; GOOD FILE?
04F4 7422           0518        JZ      READCERR                ;   NO

04F6 BA0000                     MOV     DX,0
04F9 E86502         0761        CALL    SETDMAB

04FC BA00F6                     MOV     DX,PCPM_ADR
                        READC1:
04FF E86302         0765        CALL    SETDMA                  ; SET DMA OFFSET FOR ...
0502 52                         PUSH    DX
0503 BA4501                     MOV     DX,OFFSET Z80FCB
0506 E86002         0769        CALL    READFILE                ; ... NEXT SECTOR READ
```

```
0509 5A                     POP     DX
050A 3C01                   CMP     AL,01H              ; EOF?
050C 7407         0515      JZ      READCEND            ;   YES
050E 81C28000               ADD     DX,80H              ; ADDRESS FOR NEXT RECORD
0512 E9EAFF       04FF      JMP     READC1

                  READCEND:
0515 33C0                   XOR     AX,AX               ; RESET CARRY
0517 C3                     RET

                  READCERR:
0518 BA9501                 MOV     DX,OFFSET PCPMERR
051B E85302       0771      CALL    PRTSTRING

051E F9                     STC                         ; SET CARRY
051F C3                     RET
```



```
                            EJECT

                  ; ****************************************************************
                  ;
                  ; MOVE POINTERS/BUFFERS DATA BLOCK
                  ;
                  MOVEBLOCK1:
```

```
0520 BE003A                     MOV     SI,PB1_ADR
0523 BF00FD                     MOV     DI,PB2_ADR
0526 EB06          052E         JMPS    MOVEB1

                        MOVEBLOCK2:
0528 BE00FD                     MOV     SI,PB2_ADR
052B BF003A                     MOV     DI,PB1_ADR

                        MOVEB1:
052E 893E0201                   MOV     NEWPB_ADR,DI            ; SAVE POINTER TO NEW BLOCK

0532 1E                         PUSH    DS
0533 33C0                       XOR     AX,AX
0535 8ED8                       MOV     DS,AX
0537 8EC0                       MOV     ES,AX
0539 B90003                     MOV     CX,PB1_LEN
053C FC                         CLD
053D F3A4                   REP MOVS    AL,AL                   ; MOVE DATA BLOCK
053F 1F                         POP     DS

0540 C3                         RET
```



```
                                EJECT

                        ; ***************************************************************
```

```
                        ;
                        ; MOVE CP/M-86/80 AND ADJUST MEMORY REGION TABLE
                        ;
                        MOVECPM1:
0541 2E8E06A603                 MOV     ES,PARM_SEG
0546 BB4725                     MOV     BX,BIOS_OFFSET+PBADR_OFFSET
0549 268B1F                     MOV     BX,ES:[BX]              ; GET ADR OF DATA BLOCK
054C 81C3F802                   ADD     BX,MEMSIZE_OFFSET       ; GET ADR OF MEMORY SIZE
0550 33C0                       XOR     AX,AX
0552 8EC0                       MOV     ES,AX
0554 268B07                     MOV     AX,ES:[BX]              ; GET MEMORY SIZE
0557 0BC0                       OR      AX,AX                   ; LARGE SYSTEM?
0559 750A       0565            JNZ     MCPM1A                  ;   YES

055B B8000C                     MOV     AX,CPM2A_SEG
055E 8EC0                       MOV     ES,AX                   ; SET DESTINATION SEGMENT
0560 B8C00B                     MOV     AX,TPA2A_LEN            ; SET TPA LENGTH
0563 EB08       056D            JMPS    MCPM1B
                        MCPM1A:
0565 B80010                     MOV     AX,CPM2B_SEG
0568 8EC0                       MOV     ES,AX                   ; SET DESTINATION SEGMENT
056A B8200F                     MOV     AX,TPA2B_LEN            ; SET TPA LENGTH
                        MCPM1B:
056D 50                         PUSH    AX
056E B84000                     MOV     AX,TPABEG2_SEG          ; SET TPA START
0571 50                         PUSH    AX
0572 EB23       0597            JMPS    MCPM2
```

```
                        MOVECPM2:
0574 2E8E06A603                 MOV     ES,PARM_SEG
0579 BB4725                     MOV     BX,BIOS_OFFSET+PBADR_OFFSET
057C 268B1F                     MOV     BX,ES:[BX]              ; GET ADR OF DATA BLOCK
057F 81C3F802                   ADD     BX,MEMSIZE_OFFSET       ; GET ADR OF MEMORY SIZE
0583 33C0                       XOR     AX,AX
0585 8EC0                       MOV     ES,AX
0587 268B07                     MOV     AX,ES:[BX]              ; GET #ADDITIONAL PARAGRAPHS
058A 05300C                     ADD     AX,TPA1_LEN             ; SET TPA LENGTH
058D 50                         PUSH    AX
058E B8D003                     MOV     AX,TPABEG1_SEG          ; SET TPA START
0591 50                         PUSH    AX
0592 B84000                     MOV     AX,CPM1_SEG
0595 8EC0                       MOV     ES,AX                   ; SET DESTINATION SEGMENT

                        MCPM2:
0597 1E                         PUSH    DS
0598 2E8E1EA603                 MOV     DS,PARM_SEG             ; GET CURRENT CP/M SEGMENT
059D BE0000                     MOV     SI,0
05A0 BF0000                     MOV     DI,0
05A3 B90036                     MOV     CX,CPM1_LEN
CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM


05A6 FC                         CLD
05A7 F3A4                   REP MOVS    AL,AL
05A9 1F                         POP     DS
```

```
05AA 8C060001                   MOV     NEWCPM_SEG,ES           ; SAVE SEGMENT OF NEW CP/M

05AE BB4525                     MOV     BX,BIOS_OFFSET+SEGTBL_OFFSET
05B1 268B1F                     MOV     BX,ES:[BX]              ; GET SEG TBL OFFSET
05B4 58                         POP     AX
05B5 26894701                   MOV     ES:1[BX],AX             ; SET START OF 1ST SEGMENT
05B9 58                         POP     AX
05BA 26894703                   MOV     ES:3[BX],AX             ; SET LENGTH OF 1ST SEGMENT
                                                                ; IGNORE 2ND SEGMENT FOR
                                                                ;   CONFIGURATION #2 -- LARGE SYSTEM
05BE 268C060222                 MOV     ES: WORD PTR .BIOSCS_OFFSET,ES  ; STASH NEW BIOS CS


05C3 C3                         RET

CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM

                                EJECT

                        ; ************************************************************
                        ;
                        ; SET CP/M'S POINTER TO POINTERS/BUFFERS DATA BLOCK
                        ;
                        SETPBADR:
05C4 8E060001                   MOV     ES,NEWCPM_SEG
```

```
05C8 BB4725                     MOV     BX,BIOS_OFFSET+PBADR_OFFSET
05CB A10201                     MOV     AX,NEWPB_ADR
05CE 268907                     MOV     ES:[BX],AX

05D1 C3                         RET



                        ; ****************************************************************
                        ;
                        ; INDICATE THAT PSEUDO CP/M IS LOADED
                        ;
                        SETPCPM:
05D2 BA00F6                     MOV     DX,PCPM_ADR
05D5 EB03       05DA            JMPS    PCPM1

                        ;
                        ; INDICATE THAT PSEUDO CP/M IS NOT LOADED
                        ;
                        RESETPCPM:
05D7 BA0000                     MOV     DX,0

                        PCPM1:
05DA 8E060001                   MOV     ES,NEWCPM_SEG
05DE BB4725                     MOV     BX,BIOS_OFFSET+PBADR_OFFSET
05E1 268B1F                     MOV     BX,ES:[BX]              ; GET ADR OF DATA BLOCK
05E4 81C3FA02                   ADD     BX,PCPMADR_OFFSET       ; GET ADR OF PCPM LOAD ADR
05E8 33C0                       XOR     AX,AX
05EA 8EC0                       MOV     ES,AX
```

```
05EC 268917                     MOV     ES:[BX],DX

05EF C3                         RET

CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM


                                EJECT

                        ; ****************************************************************
                        ;
                        ; ADJUST Z80 INTERRUPT VECTORS, STACK POINTER
                        ;
                        SETZ80SHR:
05F0 BA03F6                     MOV     DX,PRMTV2_ADR
05F3 EB03       05F8            JMPS    RESETP1

                        SETZ80PVT:
05F5 BA0001                     MOV     DX,PRMTV1_ADR

                        RESETP1:
05F8 2E8E06A603                 MOV     ES,PARM_SEG
05FD BB4725                     MOV     BX,BIOS_OFFSET+PBADR_OFFSET
0600 268B1F                     MOV     BX,ES:[BX]              ; GET ADR OF DATA BLOCK
0603 81C38600                   ADD     BX,STARTPKT_OFFSET      ; GET ADR OF PACKET BUFFER TO USE
0607 33C0                       XOR     AX,AX
0609 8EC0                       MOV     ES,AX
```

```
060B 26C60721                   MOV     ES:BYTE PTR [BX],STARTFC
060F 26895702                   MOV     ES:2[BX],DX             ; SET Z80 START ADR

0613 8BCB                       MOV     CX,BX                   ; CX = PACKET POINTER
0615 E80FFD       0327          CALL    SENDPKT                 ; ADJUST VECTORS, STACK POINTER
                                                                ; (Z80 WILL NOT SEND INTERRUPT BACK)

0618 C3                         RET


CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM


                        EJECT

                ; ****************************************************************
                ;
                ; ADJUST 8088 INTERRUPT VECTORS AND CP/M RETURN
                ; (AFTER EXITING, SS MUST BE SET TO VALUE IN AX)
                ;
                ; THIS ROUTINE ASSUMES THAT CP/M'S CS=DS=SS
                ; AND THAT SP+4 CONTAINS THE CSEG VALUE FOR A RETURN FAR TO CCP
                ;
                SET8088:
0619 33C0               XOR     AX,AX
061B 8EC0               MOV     ES,AX
```

```
061D A10001                     MOV     AX,NEWCPM_SEG           ; GET SEGMENT VALUE
0620 FA                         CLI
0621 BB9200                     MOV     BX,TYPE36SEG_ADR
0624 268907                     MOV     ES:[BX],AX
0627 BB9600                     MOV     BX,TYPE37SEG_ADR
062A 268907                     MOV     ES:[BX],AX
062D BB9E00                     MOV     BX,TYPE39SEG_ADR
0630 268907                     MOV     ES:[BX],AX
0633 BBB200                     MOV     BX,TYPE44SEG_ADR
0636 268907                     MOV     ES:[BX],AX
0639 BB9201                     MOV     BX,TYPE100SEG_ADR
063C 268907                     MOV     ES:[BX],AX
063F BB8203                     MOV     BX,TYPE224SEG_ADR
0642 268907                     MOV     ES:[BX],AX
0645 FB                         STI

0646 8EC0                       MOV     ES,AX
0648 8BDC                       MOV     BX,SP
064A 26894704                   MOV     ES:4[BX],AX             ; SET RETURN TO NEW CP/M SEGMENT

064E C3                         RET
```



```
                                EJECT
```

```
                        ; ***************************************************************
                        ;
                        ; INTIALIZE Z80 LOCATION 0 THRU 7
                        ;
                        INITZL07:
064F B803F7                     MOV     AX,PBIOS_ADR+3          ; GET ADDRESS OF PSEUDO BIOS
0652 BB0601                     MOV     BX,OFFSET BUFFER
0655 C607C3                     MOV     BYTE PTR [BX],JUMPINST  ; SET 'JMP ...
0658 894701                     MOV     1[BX],AX                ; ... PBIOS+3'

065B E80F01     076D            CALL    GETIOB                  ; GET IOBYTE IN AL
065E 2E8E06A603                 MOV     ES,PARM_SEG
0663 8B1E0401                   MOV     BX,PARM_OFFSET
0667 268A6702                   MOV     AH,ES:2[BX]             ; GET CURRENT DISK
066B BB0601                     MOV     BX,OFFSET BUFFER
066E 894703                     MOV     3[BX],AX                ; SET LOCATIONS 3-4

                                                                ; COMPUTE LOCATION OF PBDOS VECTOR
0671 BB4525                     MOV     BX,BIOS_OFFSET+SEGTBL_OFFSET
0674 268B1F                     MOV     BX,ES:[BX]
0677 268B4701                   MOV     AX,ES:1[BX]             ; GET START OF 1ST SEGMENT
067B 26034703                   ADD     AX,ES:3[BX]             ; ADD LENGTH OF 1ST SEGMENT
067F B104                       MOV     CL,4
0681 D3E0                       SHL     AX,CL
0683 2DFA00                     SUB     AX,100H-6H
0686 A30E01                     MOV     PCPMVECTOR,AX           ; SAVE LOCATION OF PBDOS VECTOR

0689 BB0601                     MOV     BX,OFFSET BUFFER
```

```
068C C64705C3                   MOV     BYTE PTR 5[BX],JUMPINST ; SET 'JMP ...
0690 894706                     MOV     6[BX],AX                ; ... PBDOS-VECTOR'

0693 BB4725                     MOV     BX,BIOS_OFFSET+PBADR_OFFSET
0696 268B1F                     MOV     BX,ES:[BX]              ; GET ADR OF DATA BLOCK
0699 81C39400                   ADD     BX,MOVEPKT_OFFSET       ; GET ADR OF PACKET BUFFER TO USE
069D 33C0                       XOR     AX,AX
069F 8EC0                       MOV     ES,AX

06A1 26C60722                   MOV     ES:BYTE PTR [BX],MOVEFC
06A5 8CD8                       MOV     AX,DS
06A7 B104                       MOV     CL,4
06A9 D3E0                       SHL     AX,CL
06AB 050601                     ADD     AX,OFFSET BUFFER
06AE 26894702                   MOV     ES:2[BX],AX             ; SET FROM ADDRESS
06B2 26C747040000               MOV     ES:WORD PTR 4[BX],0     ; SET TO ADDRESS
06B8 26C747060800               MOV     ES:WORD PTR 6[BX],8     ; SET #BYTES

06BE 8BCB                       MOV     CX,BX                   ; CX = PACKET POINTER
06C0 E850FC       0313          CALL    XFERPKT                 ; SEND AND WAIT FOR Z80 TO FINISH

06C3 C3                         RET
CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM
```

CP/M ASM86 1.1 SOURCE: Z80CCP.A86   CONFIG SYSTEM; INIT FOR .COM

```
                        EJECT

                ; ************************************************************
                ;
                ; INITIALIZE Z80 FCB AND DEFAULT BUFFER AREAS
                ;
                INITFCBBUF:
06C4 2E8E06A603                 MOV     ES,PARM_SEG
06C9 BB4725                     MOV     BX,BIOS_OFFSET+PBADR_OFFSET
06CC 268B1F                     MOV     BX,ES:[BX]              ; GET ADR OF DATA BLOCK
06CF 81C39400                   ADD     BX,MOVEPKT_OFFSET       ; GET ADR OF PACKET BUFFER TO USE
06D3 33C0                       XOR     AX,AX
06D5 8EC0                       MOV     ES,AX

06D7 26C60722                   MOV     ES:BYTE PTR [BX],MOVEFC
06DB 8CD8                       MOV     AX,DS
06DD B104                       MOV     CL,4
06DF D3E0                       SHL     AX,CL
06E1 055C00                     ADD     AX,FCB_OFFSET
06E4 26894702                   MOV     ES:2[BX],AX                             ; SET FROM ADDRESS
06E8 26C747045C00               MOV     ES:WORD PTR 4[BX],FCB_OFFSET            ; SET TO ADDRESS
06EE 26C74706A400               MOV     ES:WORD PTR 6[BX],100H-FCB_OFFSET       ; SET #BYTES

06F4 8BCB                       MOV     CX,BX                   ; CX = PACKET POINTER
06F6 E81AFC       0313          CALL    XFERPKT                 ; SEND AND WAIT FOR Z80 TO FINISH
```

```
 06F9 C3                        RET

CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM


                               EJECT

                        ; ****************************************************************
                        ;
                        ; INITIALIZE PBDOS VECTOR
                        ;
                        INITVECTOR:
 06FA B800F6                    MOV     AX,PBDOS_ADR            ; GET ADDRESS OF PSEUDO BDOS
 06FD 8B1E0E01                  MOV     BX,PCPMVECTOR
 0701 33D2                      XOR     DX,DX
 0703 8EC2                      MOV     ES,DX
 0705 26C607C3                  MOV     ES:BYTE PTR [BX],JUMPINST       ; SET 'JMP ...
 0709 26894701                  MOV     ES:1[BX],AX                     ; ... PBDOS'

 070D C3                        RET

CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM


                               EJECT

                        ; ****************************************************************
```

```
                        ;
                        ; SAVE .COM FILE FCB
                        ;
                        SAVEFCB:
070E 2E8E06A603                 MOV     ES,PARM_SEG
0713 8B1E0401                   MOV     BX,PARM_OFFSET
0717 268B37                     MOV     SI,ES:[BX]              ; GET FCB OFFSET
071A 8CC8                       MOV     AX,CS                   ; COMFCB IS IN CSEG
071C 8EC0                       MOV     ES,AX
071E BFA803                     MOV     DI,OFFSET COMFCB
0721 1E                         PUSH    DS
0722 2E8E1EA603                 MOV     DS,PARM_SEG
0727 B92000                     MOV     CX,32
072A FC                         CLD
072B F3A4                   REP MOVS    AL,AL
072D 1F                         POP     DS
072E 2EC606C80300               MOV     BYTE PTR COMFCB+32,0

0734 C3                         RET

CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM


                        EJECT

                        ; ****************************************************************
                        ;
                        ; RELOCATE .COM-FILE-LOAD-CODE TO PRIVATE 8088 MEMORY
```

```
                        ;
                        RELOCATE:
0735 B84000                     MOV     AX,RELO_SEG
0738 8EC0                       MOV     ES,AX
073A BF0000                     MOV     DI,RELO_OFFSET
073D BE0000                     MOV     SI,OFFSET RELOBEG
0740 1E                         PUSH    DS
0741 8CC8                       MOV     AX,CS
0743 8ED8                       MOV     DS,AX
0745 B9E103                     MOV     CX,OFFSET RELOEND
0748 FC                         CLD
0749 F3A4                   REP MOVS    AL,AL
074B 1F                         POP     DS

074C C3                         RET

CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM


                                EJECT

                        ; ************************************
                        ;
                        ;       INITIALIZE PSEUDO-DPH
                        ;
                        INIDPH:
074D 33C0                       XOR     AX,AX           ;WE NEED ES TO BE 0
```

```
074F 8EC0                       MOV     ES,AX
0751 B91000                     MOV     CX,16           ;MOVING 16 BYTES
0754 BE1401                     MOV     SI,OFFSET DMYDPH        ;TABLE TO COPY
0757 BFA0FC                     MOV     DI,PSDPH        ;PSEUDO ADDRESS (ABSOLUTE)
075A F3A4                       REP MOVS AL,AL
075C C3                         RET

CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM


                                EJECT

                        ; ***************************************************************
                        ;
                        ; BDOS ROUTINES
                        ;
                        OPENFILE:
075D B10F                       MOV     CL,FOPEN
075F EB12       0773            JMPS    BDOSCALL

                        SETDMAB:
0761 B133                       MOV     CL,FDMAB
0763 EB0E       0773            JMPS    BDOSCALL

                        SETDMA:
0765 B11A                       MOV     CL,FDMA
0767 EB0A       0773            JMPS    BDOSCALL
```

```
                        READFILE:
0769 B114                       MOV     CL,FREAD
076B EB06           0773        JMPS    BDOSCALL

                        GETIOB:
076D B107                       MOV     CL,FIOB
076F EB02           0773        JMPS    BDOSCALL

                        PRTSTRING:
0771 B109                       MOV     CL,FPRINT

                        BDOSCALL:
0773 CDE0                       INT     BDOS
0775 C3                         RET

CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM


                                EJECT

                        ; ***************************************************************
                        ; ***************************************************************

                                        DSEG
                                        ORG     100H
```

```
0100                    NEWCPM_SEG      RW      1               ; SEGMENT FOR NEW CP/M-86/80
0102                    NEWPB_ADR       RW      1               ; ADDRESS OF NEW POINTERS/BUFFERS
                                                                ;   DATA BLOCK
0104                    PARM_OFFSET     RW      1               ; OFFSET TO PARAMETER BLOCK FROM CCP

0106                    BUFFER          RB      8

010E                    PCPMVECTOR      RW      1               ; ADDRESS OF VECTOR TO PSEUDO BDOS

0110 A402               JMPF_OFFSET     DW      OFFSET RELOSTART        ; LOCATION TO START EXECUTING
0112 4000               JMPF_SEG        DW      RELO_SEG                ;   .COM-FILE-LOAD CODE

0114 B0FC               DMYDPH          DW      PSTRN                   ;DUMMY DPH TO COPY
0116 000000000000               DW      0,0,0,0
     0000
011E D8FC0000                   DW      PSDPB,0
0122 E7FC                       DW      PSALV

0124 0050524D5456       PRMTVFCB        DB      0,'PRMTVPVT','SYS'
     505654535953
0130 000000000000               DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
     000000000000
     000000000000
     000000
0145 005A38302020       Z80FCB          DB      0,'Z80     ','SYS'
     202020535953
0151 000000000000               DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
     000000000000
```

```
      000000000000
      000000
 0166 0D0A54686520      PRMTVERR        DB      CR,LF,'The file PRMTVPVT.SYS Not Found on Boot Disk$'
      66696C652050
      524D54565056
      542E53595320
      4E6F7420466F
      756E64206F6E
      20426F6F7420
      4469736B24
 0195 0D0A54686520      PCPMERR         DB      CR,LF,'The File Z80.SYS Not Found on Boot Disk$'
      46696C65205A
      38302E535953
      204E6F742046
      6F756E64206F
CP/M ASM86 1.1  SOURCE: Z80CCP.A86    CONFIG SYSTEM; INIT FOR .COM


      6E20426F6F74
      204469736B24

                                      END


END OF ASSEMBLY.  NUMBER OF ERRORS:   0.  USE FACTOR: 25%
```

```
CP/M MACRO ASSEM 2.0    #001    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

     1                  TITLE 'Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION'
     2
     3   0000 =         FALSE   EQU     0
     4   FFFF =         TRUE    EQU     NOT FALSE
     5
     6   0000 =         PRIVATE EQU     FALSE                   ; ASSEMBLE SHARED  VERSION
     7
     8
     9
    10
    11   FFFF =         SHARE   EQU     NOT PRIVATE
    12
    13                          MACLIB  Z80
    14                          PAGE
CP/M MACRO ASSEM 2.0    #002    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

    15
    16                          IF      SHARE
    17                  ;****************************************
    18                  ;*                                      *
    19                  ;*      PSEUDO-BIOS FOR Z80             *
    20                  ;*      CP/M 86/80                      *
    21                  ;*                                      *
    22                  ;*      04/19/82        RDK             *
    23                  ;*      10/19/82 1100   LZ              *
    24                  ;*      11/22/82 1200   LZ              *
    25                  ;*                                      *
    26                  ;****************************************
    27
    28
    29   F600           ORG     0F600H                  ;CHANGE THIS IF NECESSARY TO RE-ORG
    30
    31   0003 =         IOBYTE  EQU     3
    32   0090 =         FDOSM   EQU     90H
    33   FFE7 =         CSFLAG  EQU     0FFE7H
    34   0001 =         BIOCS   EQU     1
    35   0002 =         BDCS    EQU     2
    36
    37                  ; ** JUMP TABLE FOR STARTING:
    38                  PSBDOS:
    39   F600 C306F6            JMP PSBDOS1
    40   F603 C343F8            JMP START               ;INITIALIZE RST 3 AND 6
    41                  ; ** PSEUDO BDOS:
    42
    43                  PSBDOS1:
    44   F606 79                MOV A,C                 ;** VER 1.0.3
    45   F607 FE29              CPI 41                  ;** VER 1.0.4
    46   F609 D0                RNC                     ;** VER 1.0.4 BAD BDOS CALL
    47   F60A FE0B              CPI 11                  ;** VER 1.0.3
    48                          JRZ PCONS               ;** VER 1.0.3 CONSOLE STATUS
    49   F60C+2826              DB      28H,PCONS-$-1
    50                          JRNC PSBD1              ;** VER 1.0.4
    51   F60E+300E              DB      30H,PSBD1-$-1
```

```
    52   F610 FE06              CPI 6                   ;** VER 1.0.3
    53                          JRZ PDRC                ;** VER 1.0.3 DIRECT I/O
    54   F612+2835              DB      28H,PDRC-$-1
    55   F614 FE07              CPI 7                   ;** VER 1.0.3
    56   F616 CAE8F7            JZ PGIOB                ;** VER 1.0.3 GET IOBYTE
    57   F619 FE08              CPI 8                   ;** VER 1.0.3
    58   F61B CAECF7            JZ PSIOB                ;** VER 1.0.3 STORE IOBYTE
    59                  PSBD1:
    60   F61E CD7FF6            CALL PACKIT
    61   F621 21BAF6            LXI H,GTABLE            ;** VER 1.0.4 START CHECK TO SEE
    62   F624 0600              MVI B,0                 ;** VER 1.0.4 IF WE CAN RETURN BEFORE
    63   F626 09                DAD B                   ;** VER 1.0.4 THE 8088 IS FINISHED
    64   F627 7E                MOV A,M                 ;** VER 1.0.4 GET CODE
    65   F628 320EF8            STA GONOW               ;** VER 1.0.4 STORE IT
    66   F62B 3E90              MVI A,FDOSM             ;BDOS FUNCTION CODE
    67   F62D CD58F6    PSEUX:  CALL PSEUD
    68   F630 CDFBF7            CALL UNPSTAK            ;
    69   F633 C9                RET
    70
CP/M MACRO ASSEM 2.0    #003    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

    71   F634 3AE7FF    PCONS:  LDA CSFLAG              ;** VER 1.0.3
    72   F637 E602              ANI BDCS                ;** VER 1.0.3
    73                          JRNZ PCONS1             ;** VER 1.0.4
    74   F639+200A              DB      20H,PCONS1-$-1
    75   F63B CDE0F7            CALL IOBCHK             ;** VER 1.0.4
    76                          JRNZ PSBD1              ;** VER 1.0.4
    77   F63E+20DE              DB      20H,PSBD1-$-1
    78   F640 3AE7FF            LDA CSFLAG              ;** VER 1.0.4
    79   F643 E601              ANI BIOCS               ;** VER 1.0.4
    80   F645 C8        PCONS1: RZ                      ;** VER 1.0.3
    81   F646 3EFF              MVI A,0FFH              ;** VER 1.0.3
    82   F648 C9                RET                     ;** VER 1.0.3
    83
    84   F649 7B        PDRC:   MOV A,E                 ;** VER 1.0.3
    85   F64A FEFF              CPI 0FFH                ;** VER 1.0.3
    86                          JRZ PDRC1               ;** VER 1.0.4
    87   F64C+2804              DB      28H,PDRC1-$-1
    88   F64E 4B                MOV C,E                 ;** VER 1.0.4
    89   F64F C36AF7            JMP CONOUT              ;** VER 1.0.4
    90   F652 CD4DF7    PDRC1:  CALL CONST              ;** VER 1.0.3
    91                          JRNZ PSBD1              ;** VER 1.0.3
    92   F655+20C7              DB      20H,PSBD1-$-1
    93   F657 C9                RET                     ;** VER 1.0.3
    94
    95   F658 3275F6    PSEUD:  STA PACKET
    96   F65B 3A0EF8            LDA GONOW               ;** VER 1.0.4
    97   F65E 3276F6            STA PACKET+1            ;** VER 1.0.4
    98   F661 2175F6            LXI H,PACKET
    99   F664 CDC7F8            CALL    I88SVC          ; REQUEST 8080 PROCESSING ( 1.0.3)
   100   F667 3A0EF8            LDA GONOW               ;** VER 1.0.4
   101   F66A B7                ORA A                   ;** VER 1.0.4
   102   F66B C8                RZ                      ;** VER 1.0.4
   103   F66C CDE3F8            CALL WAIT88             ;** VER 1.0.4
   104   F66F CDA2F6    PSEUD1: CALL UNPACKIT
```

```
   105   F672 C9                 RET
   106
   107
   108
   109
   110                   ;** DATA AREA
   111   F673 0000       IPKT    DW 0
   112   F675 0000000000PACKET   DB 0,0,0,0,0,0,0,0,0,0
   113
   114                   ;*
   115                   ;*      THE MESSAGE PACKET WILL LOOK LIKE THIS:
   116                   ;*      BYTE    USED            Z80     8088    USED
   117                   ;*              FOR             REG     REG     AT
   118                   ;*      ____________________________________________
   119                   ;*      0       FUNCTION        --      AH      ENTRY
   120                   ;*      1       RET VAL         A       AL      RETURN (GONOW ON ENTRY)
   121                   ;*      2       FUNCTION NO.    C       CL      ENTRY
   122                   ;*      3                       B       CH      ENTRY
   123                   ;*      4                       E       DL      ENTRY
   124                   ;*      5                       D       DH      ENTRY
   125                   ;*      6                       L       BL      RETURN
   126                   ;*      7                       H       BH      RETURN
CP/M MACRO ASSEM 2.0     #004    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

   127                   ;*      8       IOBYTE          --      --      BOTH
   128                   ;*
   129
   130
   131                   ; ** SUBROUTINE TO PUT REGISTERS IN PACKET FOR BIOS OR BDOS CALLS
   132
   133                   PACKIT:
   134                   REPSTAK:                ;ROUTINE TO SHIFT TO NEW STACK
   135   F67F 220AF8             SHLD HSAVE      ;HANG ON TO HL
   136   F682 F3                 DI              ;NO INTERRUPTS, PLEASE
   137   F683 E1                 POP H           ;GET RETURN
   138                           SSPD PSTAKSAV   ;STASH THE STACK POINTER
   139   F684+ED73               DB      0EDH,73H
   140   F686+0CF8               DW      PSTAKSAV
   141   F688 3141F8             LXI SP,PSTACK   ;SET STACK TO NEW AREA
   142   F68B E5                 PUSH H          ;SET UP RETURN
   143   F68C FB                 EI              ;INTERUPTS OK
   144   F68D 2A0AF8             LHLD HSAVE      ;RESTORE HL
   145   F690 CDE3F8             CALL WAIT88             ;** VER 1.0.4
   146                           SBCD PACKET+2           ;STORE REGISTERS
   147   F693+ED43               DB      0EDH,43H
   148   F695+77F6               DW      PACKET+2
   149                           SDED PACKET+4
   150   F697+ED53               DB      0EDH,53H
   151   F699+79F6               DW      PACKET+4
   152   F69B 3A0300             LDA IOBYTE
   153   F69E 327DF6             STA PACKET+8
   154   F6A1 C9                 RET
   155
   156                   ; ** SUBROUTINE TO PUT MESSAGE-PACKET DATA INTO REGISTERS
   157
```

```
   158                  UNPACKIT:               ;
   159                          PUSHIX          ;STORE INDEX REG
   160   F6A2+DDE5              DB      0DDH,0E5H
   161                          LIXD IPKT       ;** VER 1.0.4
   162   F6A4+DD2A              DB      0DDH,2AH
   163   F6A6+73F6              DW      IPKT
   164                          LDX A,8         ;IOBYTE
   165   F6A8+DD7E08            DB      0DDH,A*8+46H,8
   166   F6AB 320300            STA IOBYTE      ;(LOC 3)
   167                          LDX A,1
   168   F6AE+DD7E01            DB      0DDH,A*8+46H,1
   169                          LDX L,6
   170   F6B1+DD6E06            DB      0DDH,L*8+46H,6
   171                          LDX H,7
   172   F6B4+DD6607            DB      0DDH,H*8+46H,7
   173                          POPIX
   174   F6B7+DDE1              DB      0DDH,0E1H
   175   F6B9 C9                RET
   176
   177                  GTABLE: ;TABLE OF CRITERIA FOR WAITING AFTER A BDOS CALL. ZERO = DON'T WAIT
   178   F6BA 01                DB      1               ;0
   179   F6BB 01                DB      1               ;1
   180   F6BC 00                DB      0               ;2
   181   F6BD 01                DB      1               ;3
   182   F6BE 00                DB      0               ;4
CP/M MACRO ASSEM 2.0    #005    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

   183   F6BF 00                DB      0               ;5
   184   F6C0 01                DB      1               ;6
   185   F6C1 01                DB      1               ;7
   186   F6C2 01                DB      1               ;8
   187   F6C3 01                DB      1               ;9
   188   F6C4 01                DB      1               ;10
   189   F6C5 01                DB      1               ;11
   190   F6C6 01                DB      1               ;12
   191   F6C7 01                DB      1               ;13
   192   F6C8 01                DB      1               ;14
   193   F6C9 01                DB      1               ;15
   194   F6CA 01                DB      1               ;16
   195   F6CB 01                DB      1               ;17
   196   F6CC 01                DB      1               ;18
   197   F6CD 01                DB      1               ;19
   198   F6CE 01                DB      1               ;20
   199   F6CF 01                DB      1               ;21
   200   F6D0 01                DB      1               ;22
   201   F6D1 01                DB      1               ;23
   202   F6D2 01                DB      1               ;24
   203   F6D3 01                DB      1               ;25
   204   F6D4 00                DB      0               ;26
   205   F6D5 01                DB      1               ;27
   206   F6D6 01                DB      1               ;28
   207   F6D7 01                DB      1               ;29
   208   F6D8 01                DB      1               ;30
   209   F6D9 01                DB      1               ;31
   210   F6DA 00                DB      0               ;32
```

```
   211   F6DB 01                DB      1               ;33
   212   F6DC 01                DB      1               ;34
   213   F6DD 01                DB      1               ;35
   214   F6DE 01                DB      1               ;36
   215   F6DF 01                DB      1               ;37
   216   F6E0 01                DB      1               ;38
   217   F6E1 01                DB      1               ;39
   218   F6E2 01                DB      1               ;40
   219
   220                          PAGE
CP/M MACRO ASSEM 2.0    #006    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

   221
   222   F700                   ORG     PSBDOS+100H     ; PAGE BOUNDARY FOR BIOS JUMPS
   223                  ;
   224                  ; ** BIOS JUMP TABLE            ;FUNCT NO.
   225
   226   F700 C336F7            JMP BOOT                ;0
   227   F703 C345F7            JMP WBOOT               ;1
   228   F706 C34DF7            JMP CONST               ;2
   229   F709 C362F7            JMP CONIN               ;3
   230   F70C C36AF7            JMP CONOUT              ;4
   231   F70F C372F7            JMP LIST                ;5
   232   F712 C37AF7            JMP PUNCH               ;6
   233   F715 C382F7            JMP READER              ;7
   234   F718 C38AF7            JMP HOME                ;8
   235   F71B C392F7            JMP SELDSK              ;9
   236   F71E C39AF7            JMP SETTRK              ;A
   237   F721 C3A4F7            JMP SETSEC              ;B
   238   F724 C3ACF7            JMP SETDMA              ;C
   239   F727 C3B4F7            JMP READ                ;D
   240   F72A C3BCF7            JMP WRITE               ;E
   241   F72D C3C4F7            JMP LISTST              ;F
   242   F730 C3CCF7            JMP SECTRAN             ;10
   243   F733 C3D4F7            JMP VIDEO               ;11
   244
   245
   246   F736 CD7FF6    BOOT:   CALL PACKIT
   247   F739 3E40              MVI A,40H
   248   F73B 3275F6    XBOOT:  STA PACKET              ;0
   249   F73E 2175F6            LXI H,PACKET            ;STORE CODE IN PACKET
   250                  ;       CALL WAIT88             ;** VER 1.0.4
   251   F741 CDC7F8            CALL    I88SVC          ; REQUEST 8088 PROCESSING (1.0.3)
   252   F744 E7                RST 4                   ;BYE.
   253
   254   F745 CD7FF6    WBOOT:  CALL PACKIT             ;1
   255   F748 3E41              MVI A,41H
   256   F74A C33BF7            JMP XBOOT               ;DO IT TO IT
   257
   258                  CONST:                          ;2
   259   F74D CDE0F7            CALL IOBCHK             ;** VER 1.0.3
   260                          JRNZ CONST1             ;** VER 1.0.3
   261   F750+2008              DB      20H,CONST1-$-1
   262   F752 3AE7FF            LDA CSFLAG              ;** VER 1.0.3
   263   F755 E601              ANI BIOCS               ;** VER 1.0.3
```

```
   264   F757 C345F6           JMP PCONS1              ;** VER 1.0.3
   265                 CONST1:
   266   F75A CD7FF6           CALL PACKIT
   267   F75D CDF1F7           CALL PSEUN              ;** VER 1.0.4
   268   F760 0142             DB      1,42H
   269
   270   F762 CD7FF6   CONIN:  CALL PACKIT             ;3
   271   F765 CDF1F7           CALL PSEUN              ;** VER 1.0.4
   272   F768 0143             DB      1,43H
   273
   274   F76A CD7FF6   CONOUT: CALL PACKIT             ;4
   275   F76D CDF1F7           CALL PSEUN              ;** VER 1.0.4
   276   F770 0044             DB      0,44H
CP/M MACRO ASSEM 2.0    #007    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

   277
   278   F772 CD7FF6   LIST:   CALL PACKIT             ;5
   279   F775 CDF1F7           CALL PSEUN              ;** VER 1.0.4
   280   F778 0045             DB      0,45H
   281
   282   F77A CD7FF6   PUNCH:  CALL PACKIT             ;6
   283   F77D CDF1F7           CALL PSEUN              ;** VER 1.0.4
   284   F780 0046             DB      0,46H
   285
   286   F782 CD7FF6   READER: CALL PACKIT             ;7
   287   F785 CDF1F7           CALL PSEUN              ;** VER 1.0.4
   288   F788 0147             DB      1,47H
   289
   290   F78A CD7FF6   HOME:   CALL PACKIT             ;8
   291   F78D CDF1F7           CALL PSEUN              ;** VER 1.0.4
   292   F790 0048             DB      0,48H
   293
   294   F792 CD7FF6   SELDSK: CALL PACKIT             ;9
   295   F795 CDF1F7           CALL PSEUN              ;** VER 1.0.4
   296   F798 0149             DB      1,49H
   297
   298   F79A 0600     SETTRK: MVI B,0                 ;10
   299   F79C CD7FF6           CALL PACKIT
   300   F79F CDF1F7           CALL PSEUN              ;** VER 1.0.4
   301   F7A2 004A             DB      0,4AH
   302
   303   F7A4 CD7FF6   SETSEC: CALL PACKIT             ;11
   304   F7A7 CDF1F7           CALL PSEUN              ;** VER 1.0.4
   305   F7AA 004B             DB      0,4BH
   306
   307   F7AC CD7FF6   SETDMA: CALL PACKIT             ;12
   308   F7AF CDF1F7           CALL PSEUN              ;** VER 1.0.4
   309   F7B2 004C             DB      0,4CH
   310
   311   F7B4 CD7FF6   READ:   CALL PACKIT             ;13
   312   F7B7 CDF1F7           CALL PSEUN              ;** VER 1.0.4
   313   F7BA 014D             DB      1,4DH
   314
   315   F7BC CD7FF6   WRITE:  CALL PACKIT             ;14
   316   F7BF CDF1F7           CALL PSEUN              ;** VER 1.0.4
```

```
   317   F7C2 014E              DB      1,4EH
   318
   319   F7C4 CD7FF6    LISTST: CALL PACKIT             ;15
   320   F7C7 CDF1F7            CALL PSEUN              ;** VER 1.0.4
   321   F7CA 014F              DB      1,4FH
   322
   323                  SECTRAN:                        ;16
   324   F7CC 0600              MVI B,0                 ;DOUBLE PRECISION IN BC
   325   F7CE EB                XCHG                    ;TRANSLATE TABLE ADDRESS TO HL
   326   F7CF 09                DAD B                   ;TRANSLATE SECTOR ADDRESS
   327   F7D0 6E                MOV L,M                 ;RETURN SECTOR IN L
   328   F7D1 2600              MVI H,0                 ;
   329   F7D3 C9                RET                     ;DON'T BOTHER CP/M-86
   330
   331                  VIDEO:                          ;17
   332   F7D4 1600              MVI D,0                 ;FAST VIDEO. NEED 0 SEG
CP/M MACRO ASSEM 2.0    #008    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

   333   F7D6 1E00              MVI E,0                 ;** VER 1.0.4
   334   F7D8 CD7FF6            CALL PACKIT
   335   F7DB CDF1F7            CALL PSEUN
   336   F7DE 0156              DB      1,56H
   337
   338   F7E0 3A0300    IOBCHK: LDA IOBYTE              ;** VER 1.0.3
   339   F7E3 E603              ANI 3                   ;** VER 1.0.3 CRT?
   340   F7E5 FE01              CPI 1                   ;** VER 1.0.3 REALLY?
   341   F7E7 C9                RET                     ;** VER 1.0.3
   342
   343   F7E8 3A0300    PGIOB:  LDA IOBYTE              ;** VER 1.0.3 GET IOBYTE
   344   F7EB C9                RET                     ;** VER 1.0.3
   345
   346   F7EC 7B        PSIOB:  MOV A,E                 ;** VER 1.0.3 STORE IOBYTE
   347   F7ED 320300            STA IOBYTE              ;** VER 1.0.3
   348   F7F0 C9                RET                     ;** VER 1.0.3
   349
   350   F7F1 E1        PSEUN:  POP H                   ;ROUTINE TO CARRY PARAMETERS FORWARD INTO
   351   F7F2 7E                MOV A,M                 ;THE PACKET FOR BIOS CALLS
   352   F7F3 320EF8            STA GONOW               ;STORE WAIT CRITERION
   353   F7F6 23                INX H                   ;
   354   F7F7 7E                MOV A,M                 ;GET FUNCTION NUMBER
   355   F7F8 C32DF6            JMP PSEUX               ;GO TO PSEUDO-BIOS EXIT
   356
   357                  UNPSTAK:                ;ROUTINE TO RESET TO OLD STACK
   358   F7FB 220AF8            SHLD HSAVE      ;HANG ON TO HL
   359   F7FE F3                DI              ;NO INTERRUPTS, PLEASE
   360   F7FF E1                POP H           ;GET RETURN ADDRESS
   361                          LSPD PSTAKSAV   ;RESTORE USER STACK
   362   F800+ED7B              DB      0EDH,07BH
   363   F802+0CF8              DW      PSTAKSAV
   364   F804 E5                PUSH H          ;RESTORE RETURN
   365   F805 FB                EI              ;INTERRUPTS OK
   366   F806 2A0AF8            LHLD HSAVE      ;RESTORE HL
   367   F809 C9                RET             ;
   368
   369   F80A 0000      HSAVE:          DW      0
```

```
   370   F80C 0000      PSTAKSAV:       DW      0
   371   F80E 00        GONOW:          DB      0
   372
   373
   374                  ;SPACE FOR A STACK
   375   F80F                   DS      50
   376   F841 0000      PSTACK: DW      0
   377
   378                          ENDIF
   379                          PAGE
CP/M MACRO ASSEM 2.0    #009    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

   380
   381
   382                  ; Z80 INTERFACE LAYER AND PRIMITIVE ROUTINES
   383
   384                  ; REWRITTEN FOR DEC RAINBOW 100
   385                  ;           BY CPL
   386                  ;           JULY 1982
   387
   388
   389                          IF      PRIVATE
   390                  START   EQU     00100H                  ; STARTING ADDRESS OF CODE
   391                  INTFPTR EQU     03CFCH                  ; LOCATION OF PACKET POINTERS
   392                          ELSE
   393   F843 =         START   EQU     $                       ; STARTING ADDRESS OF CODE
   394   FFFC =         INTFPTR EQU     0FFFCH                  ; LOCATION OF PACKET POINTERS
   395                          ENDIF
   396
   397   0018 =         RST3    EQU     18H                     ; ADDRESS OF RST 18H (RST 3) VECTOR
   398   0020 =         RST4    EQU     20H                     ; ADDRESS OF RST 20H (RST 4) VECTOR
   399   0030 =         RST6    EQU     30H                     ; ADDRESS OF RST 30H (RST 6) VECTOR
   400
   401   0000 =         I88INT  EQU     0                       ; 8088 INTERRUPT PORT
   402   0020 =         INTSTA  EQU     20H                     ; CROSS-CPU INTERRUPT STATUS PORT
   403
   404   00C3 =         JUMP    EQU     0C3H                    ; OPCODE FOR JUMP INSTRUCTION
   405
   406   0004 =         INTBIT  EQU     4H                      ; INTERRUPT BIT
   407                                                          ;   WHEN LOW, SHOWS 8088 INTERRUPT
   408                                                          ;   FLOP IS INTERRUPTING THE 8088;
   409                                                          ;   ORIGINALLY SET BY Z80
   410
   411   00F0 =         FRANGE  EQU     0F0H                    ; FUNCTION RANGE MASK
   412                                                          ;   FOR HIGH NIBBLE OF FUNCTION CODE
   413
   414   0010 =         DSKFNC  EQU     10H                     ; DISK FUNCTION (HIGH NIBBLE VALUE)
   415   0020 =         OTHFNC  EQU     20H                     ; OTHER Z80 FUNCTIONS (HIGH NIBBLE VALUE)
   416   0040 =         USRFNC  EQU     40H                     ; USER-DEFINED FUNCTIONS
   417
   418   0021 =         Z80BGN  EQU     21H                     ; ALLOW Z80 TPA EXECUTION
   419   0022 =         Z80MVE  EQU     22H                     ; MOVE Z80 MEMORY CONTENTS
   420                                                          ;   LOW VALUE LIMIT - 1
   421
   422   0007 =         LEGFUN  EQU     07H                     ; FUNCTION CODE MASK
```

```
                                                        ;   USE LOW 3 BITS
                                                        ;
00FF =          FNCNG   EQU     0FFH                    ; FUNCTION CODE NO GOOD
                                                        ;
0014 =          DKWRIT  EQU     14H                     ; PACKET DISK WRITE FUNCTION CODE
                                                        ;
000A =          NUMSEC  EQU     10                      ; NUMBER OF SECTORS PER TRACK
0200 =          SECSIZ  EQU     512                     ; NUMBER OF BYTES PER DISK SECTOR
                                                        ;
0002 =          STADRL  EQU     2                       ; START ADDRESS LSB PACKET OFFSET
0003 =          STADRH  EQU     3                       ; START ADDRESS MSB PACKET OFFSET
                                                        ;
0002 =          SCADRL  EQU     2                       ; SOURCE ADDRESS LSB PACKET OFFSET
CP/M MACRO ASSEM 2.0    #010    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

0003 =          SCADRH  EQU     3                       ; SOURCE ADDRESS MSB PACKET OFFSET
0004 =          DSADRL  EQU     4                       ; DESTINATION ADDRESS LSB PACKET OFFSET
0005 =          DSADRH  EQU     5                       ; DESTINATION ADDRESS MSB PACKET OFFSET
0006 =          BYCNTL  EQU     6                       ; BYTE COUNT ADDRESS LSB PACKET OFFSET
0007 =          BYCNTH  EQU     7                       ; BYTE COUNT ADDRESS MSB PACKET OFFSET
                                                        ;
                        PAGE
CP/M MACRO ASSEM 2.0    #011    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

                                                        ;
F843                    ORG     START                   ;
                                                        ;
                ; ***************************************************************
                ;
                ; NAME: INTSTR
                ;
                ; FUNCTION: THIS ROUTINE WILL RE-INITIALIZE THE RST 6
                ;           VECTORS, SET UP THE Z80 SERVICE STACK POINTER,
                ;           AND DO A RST 4.
                ;
                ; ENTRY: NONE
                ;
                ; EXIT: NONE
                ;
                INTSTR:
                                                        ;
                ; SET UP RST VECTORS
                                                        ;
F843 2155F8             LXI     H,PKTPRO                ; SET UP RST 6 VECTOR
F846 223100             SHLD    RST6+1                  ;
                                                        ;
                ; SET UP Z80 SERVICE STACK POINTER
                                                        ;
F849 319CFC             LXI     SP,STACK                ;
                                                        ;
                ; INITIALIZE FLAGS
                                                        ;
F84C 3E00               MVI     A,FALSE                 ;
F84E 329DFC             STA     DONEFL                  ;
                                                        ;
```

```
                        ; WAIT UNTIL NEEDED
                        
F851 FB                         EI
F852 C32000                     JMP     RST4                    ; GO TO HALT AND WAIT FOR INTERRUPT
                                PAGE
CP/M MACRO ASSEM 2.0    #012    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

                        ; *************************************************************
                        ;
                        ; NAME: PKTPRO
                        ;
                        ; FUNCTION: THIS ROUTINE IS PASSED A PACKET ADDRESS IN I88PKT.  IT USES
                        ;           THAT TO GET THE FUNCTION NUMBER WHICH IT USES TO JUMP
                        ;           TO THE SPECIFIC FUNCTION HANDLER ROUTINE.
                        ;
                        ;           THIS ROUTINE IS ENTERED VIA AN INTERRUPT FROM THE 8088 AT RST 6 -
                        ;           THEREFORE, INTERRUPTS ARE OFF.
                        ;
                        ;           FOR ALL FUNCTIONS EXCEPT Z80START AND SERIAL SUPPORT FROM 8088,
                        ;           WE WILL INTERRUPT THE 8088 TO INDICATE WE HAVE FINISHED
                        ;           THE FUNCTION
                        ;
                        ; ENTRY: I88PKT = PACKET ADDRESS
                        ;
                        ; EXIT: IX = PACKET ADDRESS
                        ;
                        PKTPRO:
                                PUSHIX                          ; SAVE
F855+DDE5                       DB      0DDH,0E5H
F857 F5                         PUSH    PSW
                        
                                LIXD    I88PKT                  ; GET ADDRESS OF PACKET
F858+DD2A                       DB      0DDH,2AH
F85A+FEFF                       DW      I88PKT
                        
F85C DB00                       IN      I88INT                  ; CLEAR 8088 INTERRUPT FLAG
                        
                                PUSHIY                          ; SAVE REMAINING REGISTERS
F85E+FDE5                       DB      0FDH,0E5H
F860 E5                         PUSH    H
                                PUSHIX                          ; CHECK FOR ZERO ADDRESS
F861+DDE5                       DB      0DDH,0E5H
F863 E1                         POP     H                       ; MOVE IX TO H
F864 7C                         MOV     A,H                     ; PUT IN  ACCUM
F865 B5                         ORA     L                       ; GET LOW-ORDER, TOO.
F866 CABCF8                     JZ      HIFXIT                  ; GO AWAY IF ZERO
F869 D5                         PUSH    D
F86A C5                         PUSH    B
                        
F86B 0195F8                     LXI     B,PKTRET                ; PUSH RETURN ADDRESS ONTO STACK
F86E C5                         PUSH    B
                        
F86F FB                         EI                              ; RE-ENABLE INTERRUPTS
                        
```

```
                                LDX     A,FNCCOD        ; GET FUNCTION CODE
F870+DD7E00                     DB      0DDH,A*8+46H,FNCCOD
                        ;
F873 FE13                       CPI     QKRDCOM
F875 CA3BF9                     JZ      DKREAD          ; READ FUNCTION
                        ;
F878 FE14                       CPI     QKWTCOM
F87A CA3BF9                     JZ      DKWRITE         ; WRITE FUNCTION
CP/M MACRO ASSEM 2.0    #013    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

                        ;
F87D FE42                       CPI     USRFNC+2        ; BIOS 42 - 4F, BDOS 90
                                JRNC    HIFNC           ; YES
F87F+3033                       DB      30H,HIFNC-$-1
                        ;
F881 FE22                       CPI     Z80MVE          ; MOVE Z80 MEMORY
F883 CA47FC                     JZ      ZMOVE
                        ;
F886 FE15                       CPI     QKCMCOM
F888 CAF2F8                     JZ      DKCHECK         ; CHECK MEDIA FUNCTION
                        ;
F88B FE21                       CPI     Z80BGN          ; ALLOW Z80 TPA EXECUTION
F88D CA32FC                     JZ      ZSTART
                        ;
                        ERROR:
F890 C1                         POP     B               ; EMPTY STACK OF RETURN ADR
                                MVIX    FNCNG,STATUS    ; SET ERROR STATUS
F891+DD3601FF                   DB      0DDH,36H,STATUS,FNCNG

                        PKTRET:
F895 FB                         EI                              ; RE-ENABLE INTERRUPTS IN CASE
                                                                ;   THEY WERE OFF (DISK I/O)
F896 C1                         POP     B                       ; RESTORE REGISTERS
F897 D1                         POP     D
F898 E1                         POP     H
                                POPIY
F899+FDE1                       DB      0FDH,0E1H

                                SIXD    Z80PKT                  ; SET UP RETURN PACKET POINTER
F89B+DD22                       DB      0DDH,22H
F89D+FCFF                       DW      Z80PKT
F89F D300                       OUT     I88INT                  ;   AND PASS IT

                        PKTR1:
F8A1 DB20                       IN      INTSTA                  ; LOOP UNTIL 8088 CLEARS THE
F8A3 E604                       ANI     INTBIT                  ;   INTERRUPT
F8A5 CAA1F8                     JZ      PKTR1

                                LXIX    0
F8A8+DD21                       DB      0DDH,21H
F8AA+0000                       DW      0
                                SIXD    Z80PKT                  ;ZERO ADDRESS (8088 HAS IT NOW)
F8AC+DD22                       DB      0DDH,22H
F8AE+FCFF                       DW      Z80PKT

```

```
   580   F8B0 F1                POP     PSW
   581                          POPIX
   582   F8B1+DDE1              DB      0DDH,0E1H
   583
   584   F8B3 C9                RET                             ; AND RETURN
   585
   586                  ; PROCESS FUNCTION CODES  42H TO 4FH, OR 90H
   587
   588                  HIFNC:
   589   F8B4 C1                POP     B                       ; FIX STACK
   590   F8B5 3EFF              MVI     A,TRUE                  ; SET DONE FLAG
CP/M MACRO ASSEM 2.0    #014    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

   591   F8B7 329DFC            STA     DONEFL
   592
   593   F8BA C1                POP     B                       ; RESTORE REGISTERS
   594   F8BB D1                POP     D                       ;   EXCEPT HL,AF, AND IX
   595   F8BC E1        HIFXIT: POP     H
   596                          POPIY
   597   F8BD+FDE1              DB      0FDH,0E1H
   598
   599   F8BF F1                POP     PSW                     ; RESTORE
   600                          SIXD    IPKT                    ; STASH PACKET ADDRESS
   601   F8C0+DD22              DB      0DDH,22H
   602   F8C2+73F6              DW      IPKT
   603                          POPIX
   604   F8C4+DDE1              DB      0DDH,0E1H
   605   F8C6 C9                RET
   606
   607                          PAGE
CP/M MACRO ASSEM 2.0    #015    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

   608
   609                          IF      SHARE
   610                  ; *********************************************************
   611                  ;
   612                  ; NAME: I88SVC
   613                  ;
   614                  ; FUNCTION: THIS ROUTINE REQUESTS SERVICE FROM THE 8088 FOR A
   615                  ;           Z80-RESIDENT ROUTINE.
   616                  ;
   617                  ; ENTRY: HL = PACKET ADDRESS
   618                  ;
   619                  ; EXIT: HL = PACKET ADDRESS
   620                  ;
   621                  I88SVC:
   622
   623   F8C7 F3                DI
   624   F8C8 22FCFF            SHLD    Z80PKT                  ; WRITE ADDRESS OF PACKET
   625
   626   F8CB 3E00              MVI     A,FALSE                 ; CLEAR DONE FLAG
   627   F8CD 329DFC            STA     DONEFL
   628
   629   F8D0 D300              OUT     I88INT                  ; SET 8088 INTERRUPT FLAG
   630
```

```
                        INTCLR:
F8D2 DB20                       IN      INTSTA                  ; LOOP UNTIL 8088 TURNS OFF
F8D4 E604                       ANI     INTBIT                  ;   INTERRUPT FLAG
F8D6 CAD2F8                     JZ      INTCLR
F8D9 E5                         PUSH    H
F8DA 210000                     LXI     H,0
F8DD 22FCFF                     SHLD    Z80PKT                  ; ZERO THE PACKET ADDRESS
F8E0 E1                         POP     H
                        ;
F8E1 FB                         EI                              ; RE-ENABLE INTERRUPTS
F8E2 C9                         RET

                        ;
                        ; ROUTINE TO WAIT FOR THE DONE FLAG
                        ;
                        WAIT88:
F8E3 F3                         DI
F8E4 3A9DFC                     LDA     DONEFL                  ; JUMP IF DONE
F8E7 FEFF                       CPI     TRUE
                                JRZ     WAITDN
F8E9+2805                       DB      28H,WAITDN-$-1
F8EB CD2300                     CALL    23H                     ; WAIT UNTIL INTERRUPTED

                                JR      WAIT88                  ; JUMP TO MAKE SURE DONE FLAG IS TRUE
F8EE+18F3                       DB      18H,WAIT88-$-1

F8F0 FB                 WAITDN: EI
F8F1 C9                         RET

                                PAGE
CP/M MACRO ASSEM 2.0    #016    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION


                                ENDIF
                        ; *****************************************************************
                        ;
                        ; NAME: DSKPRO
                        ;
                        ; FUNCTION: THIS ROUTINE IS THE ENTRY POINT FOR ALL OF THE
                        ;           DISK I/O FUNCTIONS.  IT WILL DETERMINE WHICH
                        ;           PRIMITIVE TO CALL, AND PASS CONTROL TO THAT PRIMITIVE.
                        ;
                        ; ENTRY: IX = PACKET ADDRESS
                        ;
                        ; EXIT: IX = PACKET ADDRESS
                        ;
                        ; *****************************************************************
                        ;
                        ; PACKET OFFSETS
                        ;
0000 =                  FNCCOD  EQU     0       ; FUNCTION CODE
0001 =                  STATUS  EQU     1       ; RETURNED STATUS
0002 =                  DRIVEN  EQU     2       ; DRIVE NUMBER
0002 =                  SECTN   EQU     2       ; SECTOR NUMBER
```

```
   684   0003 =          TRACKN  EQU     3       ; TRACK NUMBER
   685   0004 =          DMALOW  EQU     4       ; DMA ADDRESS (LSB)
   686   0005 =          DMAHI   EQU     5       ; DMA ADDRESS (MSB)
   687   0006 =          NSECT   EQU     6       ; NUMBER OF SECTORS
   688                   ;
   689   0060 =          DRVNUM  EQU     60H     ; MASK FOR DRIVE NUMBER
   690   001F =          SECNUM  EQU     1FH     ; MASK FOR SECTOR NUMBER
   691                   ;
   692                   ;  FUNCTION CODES
   693   0013 =          QKRDCOM EQU     13H     ; READ FUNCTION CODE
   694   0014 =          QKWTCOM EQU     14H     ; WRITE FUNCTION CODE
   695   0015 =          QKCMCOM EQU     15H     ; CHECK MEDIA FUNCTION CODE
   696                   ;
   697                   ; FDC COMMAND FLAGS
   698                   ;
   699   0008 =          QMHLD   EQU     8       ; HEAD LOAD FLAG
   700   0004 =          QMVERF  EQU     4       ; VERIFY FLAG
   701   0004 =          QMEFLG  EQU     4       ; HEAD LOAD FLAG FOR READ/WRITE
   702   0008 =          QMSSEL  EQU     8       ; SIDE SELECT FLAG
   703   0002 =          QMSCOM  EQU     2       ; SIDE COMPARE FLAG
   704   0010 =          QMUPDT  EQU     10H     ; UPDATE TRACK REG FLAG
   705                   ;
   706                   ; FDC STEP RATE
   707                   ;
   708   0000 =          QKRATE  EQU     0       ; 6 MS RATE
   709                   ;
   710                   ; FDC COMMANDS
   711                   ;
   712   0008 =          QCREST  EQU     0+QMHLD+QKRATE          ; RESTORE (RECAL)
   713   001C =          QCSEEK  EQU     10H+QMHLD+QKRATE+QMVERF ; SEEK (WITH VERIFY)
   714   0018 =          QCSEEKN EQU     10H+QMHLD+QKRATE        ; SEEK (NO VERIFY)
   715   0010 =          QCSEKH0 EQU     10H+QKRATE              ; SEEK - NO HEAD LOAD
   716   0048 =          QCSTEPIN EQU    40H+QMHLD+QKRATE        ; STEP IN
CP/M MACRO ASSEM 2.0    #017    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

   717   0068 =          QCSTEPOT EQU    60H+QMHLD+QKRATE        ; STEP OUT
   718   0080 =          QCREADS EQU     80H                     ; READ SECTOR
   719   00A0 =          QCWRTS  EQU     0A0H                    ; WRITE SECTOR
   720   00F0 =          QCWRTRK EQU     0F0H                    ; WRITE TRACK
   721   00C0 =          QCRDADR EQU     0C0H                    ; READ ADDRESS
   722   00D0 =          QCTERM  EQU     0D0H                    ; TERMINATE COMMAND
   723                   ;
   724                   ; DISK CONTROLLER STATUS - TYPE 1 COMMANDS
   725                   ;
   726   0080 =          QMNRDY  EQU     80H     ; NOT READY
   727   0040 =          QMWPROT EQU     40H     ; WRITE PROTECTED
   728   0020 =          QMHLT   EQU     20H     ; HEAD LOADED
   729   0010 =          QMSKERR EQU     10H     ; SEEK ERROR
   730   0008 =          QMCRC   EQU     8       ; CRC ERROR
   731   0004 =          QMTZERO EQU     4       ; TRACK ZERO
   732   0002 =          QMINDEX EQU     2       ; INDEX
   733   0001 =          QMBUSY  EQU     1       ; BUSY
   734                   ;
   735                   ; DISK CONTROLLER STATUS - TYPE 2 AND 3 COMMANDS
   736                   ;
```

```
   737   0020 =          QMWRFLT EQU     20H     ; WRITE FAULT
   738   0010 =          QMRNF   EQU     10H     ; RECORD NOT FOUND
   739   0004 =          QMLDATA EQU     4       ; LOST DATA
   740   0002 =          QMDRQ   EQU     2       ; DATA REQUEST
   741   0020 =          QMDELDM EQU     20H     ; DELETED DATA MARK
   742                   ;
   743                   ; ERROR MASKS FOR OPERATIONS
   744                   ;
   745   0091 =          QMREST  EQU     QMNRDY+QMSKERR+QMBUSY           ; RESTORE
   746   0099 =          QMSEEK  EQU     QMNRDY+QMSKERR+QMCRC+QMBUSY     ; SEEK
   747   00BD =          QMREAD  EQU     QMNRDY+QMRNF+QMCRC+QMLDATA+QMDELDM+QMBUSY       ; READ SECTOR
   748   00FD =          QMWRITE EQU     QMNRDY+QMRNF+QMCRC+QMLDATA+QMWRFLT+QMWPROT+QMBUSY  ; WRITE SECTOR
   749   009D =          QMRDADR EQU     QMNRDY+QMRNF+QMCRC+QMLDATA+QMBUSY       ; READ ADDRESS
   750                   ;
   751                   ; GENERAL STATUS PORT (QPSTAT) EQUATES
   752                   ;
   753   00C0 =          QMPRECMP EQU    0C0H    ; PRECOMPENSATION BITS
   754   0020 =          QMPSIDE EQU     20H     ; SIDE SIGNAL
   755   0010 =          QMON1   EQU     10H     ; MOTOR 1 ON
   756   0008 =          QMON0   EQU     8       ; MOTOR 0 ON
   757   0018 =          QMON    EQU     QMON0+QMON1
   758   0004 =          QMTG42  EQU     4       ; TG 42 SIGNAL
   759   0003 =          QMDRNR  EQU     3       ; DRIVE NUMBER
   760                   ;
   761                   ; MISCELLANEOUS EQUATES
   762                   ;
   763   0004 =          QKDRETRY EQU    4       ; DISK OPERATION RETRIES
   764   9C40 =          QKDRCNT EQU     40000   ; COUNT FOR DISK READY TIMING (.5 SEC)
   765   003D =          QKPCTRK EQU     61      ; PRECOMP REQUIRED STARTING AT THIS TRACK
   766   0040 =          QKPCBIT EQU     40H     ; PRECOMP BIT TO USE AFTER TRACK 60
   767   004F =          QKMXTRK EQU     79      ; MAX TRACK NUMBER ON RX50
   768   000A =          QKMXSECT EQU    10      ; MAX PHYSICAL SECTOR NO. ON RAINBOW DISKS
   769   0002 =          QKROBIN EQU     2       ; STATUS RETURNED FOR ROBIN MEDIA ON CHECK FUNCTION
   770                   ;
   771                   ;
   772                   ;
CP/M MACRO ASSEM 2.0    #018    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

   773                   ;
   774                   ;
   775                   ; EQUATES FOR PHYSICAL DISK CONTROL
   776                   ;
   777   0040 =          QPSTAT  EQU     40H     ; GENERAL DISK CONTROL/STATUS REG.
   778   0060 =          QPCOMD  EQU     60H     ; FDC COMMAND/STATUS REG.
   779   0061 =          QPTRKRG EQU     61H     ; FDC TRACK REG.
   780   0062 =          QPSECRG EQU     62H     ; FDC SECTOR REG.
   781   0063 =          QPDATA  EQU     63H     ; FDC DATA REG.
   782                   ;
   783                   ;
   784                   ; EQUATES FOR PRIVATE RAM I/O ROUTINE
   785                   ;
   786   0040 =          RDORWR  EQU     0040H   ; ADDRESS OF ROUTINE
   787   0046 =          INOROUT EQU     0046H   ; ADDRESS OF INI OR OUTI INSTR.
   788   004A =          UDELAY  EQU     004AH   ; DELAY IN 0.500 MSEC INCREMENTS, C = COUNT
   789                   ;
```

```
                                INII    EQU     0A2EDH  ; INI INSTRUCTION
                                OUTII   EQU     0A3EDH  ; OUTI INSTRUCTION
                                ININ    EQU     QPDATA SHL 8 + IN ; IN QPDATA INSTR.
                                ;
                                        PAGE
CP/M MACRO ASSEM 2.0            #019    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION


                                ; DKCHECK  -  CHECK MEDIA TYPE
                                ;
                                ; FUNCTION:     ATTEMPTS TO DETERMINE WHETHER A RAINBOW OR ROBIN DISKETTE
                                ;               IS MOUNTED IN THE SPECIFIED DRIVE.
                                ;               INVOKED BY THE BIOS IN THE 8088 WHENEVER DRIVE IS
                                ;               SELECTED FOR THE FIRST TIME AFTER A DISK SYSTEM RESET -
                                ;               MOST COMMONLY AFTER A ^C AT THE CCP LEVEL.
                                ; RULES:        NO ERRORS ARE RETURNED. A RESTORE OPERTION IS DONE,
                                ;               FOLLOWED BY A READ OF SECTOR 10. IF A RECORD NOT FOUND
                                ;               STATUS RESULTS, ROBIN MEDIA IS ASSUMED. ON A SUCCESSFUL
                                ;               READ OF PHYSICAL SECTOR 10 OR IF ANY OTHER ERROR OCCURS
                                ;               (SUCH AS NOT READY), RAINBOW MEDIA IS ASSUMED.
                                ; ENTRY:        PACKET ADDRESS IN IX.
                                ; EXIT:         MEDIA TYPE IN THE STATUS FIELD OF THE PACKET:
                                ;               0 IF RAINBOW, 2 IF ROBIN.
                                ;
                                ;
                                ;
                                DKCHECK:
F8F2 3EFF                               MVI     A,0FFH          ; SET "CURRENT TRACK" TO FF TO INDICATE
F8F4 CD1CFC                             CALL    SETRAK          ;  FIRST ACCESS AFTER RESET
F8F7 CD32FB                             CALL    DPSELDR
                                        JRC     DKCHECK1        ; DRIVE IS NOT READY
F8FA+3805                               DB      38H,DKCHECK1-$-1
F8FC CDA1FA                             CALL    DPRECAL
                                        JRZ     DKCHECK2        ; FALL THROUGH = ERROR ON RESTORE
F8FF+2803                               DB      28H,DKCHECK2-$-1
                                DKCHECK1:
F901 AF                                 XRA     A               ; ERROR - ASSUME RAINBOW MEDIA
                                        JR      DKCHECK3
F902+1823                               DB      18H,DKCHECK3-$-1
                                DKCHECK2:
F904 AF                                 XRA     A
                                        STX     A,TRACKN
F905+DD7703                             DB      0DDH,70H+A,TRACKN
F908 CD1CFC                             CALL    SETRAK          ; TRACK 0 INTO TABLE
                                        LDX     A,DRIVEN        ; SETUP PACKET FOR THE READ ROUTINE
F90B+DD7E02                             DB      0DDH,A*8+46H,DRIVEN
F90E E660                               ANI     DRVNUM
F910 47                                 MOV     B,A
F911 3E0A                               MVI     A,QKMXSECT      ; NOW TRY TO RAED SECTOR 10
F913 B7                                 ORA     A
                                        STX     A,SECTN
F914+DD7702                             DB      0DDH,70H+A,SECTN
F917 21DB63                             LXI     H,ININ          ;  NO DATA TRANSFER INTO MEMORY
F91A CDD1F9                             CALL    DPREADZ         ;    AND NO ERROR RECOVERY
                                        JRZ     DKCHECK3        ; GOOD READ
```

```
   843   F91D+2808              DB      28H,DKCHECK3-$-1
   844   F91F FE10              CPI     QMRNF           ; IS RECORD NOT FOUND THE ONLY ERROR?
   845   F921 3E00              MVI     A,0
   846                          JRNZ    DKCHECK3        ;  NO  - SET RAINBOW MEDIA
   847   F923+2002              DB      20H,DKCHECK3-$-1
   848   F925 3E02              MVI     A,QKROBIN       ;  YES - ROBIN MEDIA
   849                  DKCHECK3:
   850   F927 47                MOV     B,A
CP/M MACRO ASSEM 2.0    #020    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

   851   F928 3A2EFC            LDA     BCURDRV         ; PUT MEDIA TYPE INTO FORMAT TABLE,
   852   F92B 5F                MOV     E,A             ;   CURRENT FORMAT INDICATOR AND
   853   F92C 1600              MVI     D,0             ;     INTO THE PACKET
   854   F92E 21F4FF            LXI     H,TFORMAT
   855   F931 19                DAD     D
   856   F932 70                MOV     M,B
   857   F933 78                MOV     A,B
   858   F934 322FFC            STA     BFORMAT
   859                          STX     A,STATUS
   860   F937+DD7701            DB      0DDH,70H+A,STATUS
   861   F93A C9                RET
   862                          PAGE
CP/M MACRO ASSEM 2.0    #021    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

   863
   864                  ; DKREAD/DKWRITE  -  READ OR WRITE SECTOR(S)
   865                  ;
   866                  ; FUNCTION: READS OR WRITES 1 TO 10 SECTORS ON
   867                  ;  SPECIFIED DRIVE AND TRACK.
   868                  ;
   869                  DKREAD:
   870                  DKWRITE:
   871   F93B CDA8F9            CALL    DKRWSET         ; SET UP FOR READ/WRITE
   872   F93E D8                RC                      ;  EXIT - DRIVE NOT READY
   873   F93F CA4CF9            JZ      DKRW20          ; NO SEEK NEEDED
   874   F942 CDC7FA            CALL    DPSEEK          ; PERFORM SEEK
   875                          JRNZ    DKRW90          ; SEEK ERROR
   876   F945+2021              DB      20H,DKRW90-$-1
   877   F947 DB61              IN      QPTRKRG         ; SET NEW TRACK IN TABLE
   878   F949 CD1CFC            CALL    SETRAK
   879                  DKRW20:
   880   F94C 3E13              MVI     A,QKRDCOM       ; READ FUNCTION?
   881                          CMPX    FNCCOD
   882   F94E+DDBE00            DB      0DDH,0BEH,FNCCOD
   883                          JRNZ    DKRW30          ;  NO - WRITE
   884   F951+2005              DB      20H,DKRW30-$-1
   885   F953 CDCBF9            CALL    DPREAD          ;  YES - READ SECTOR
   886                          JR      DKRW40
   887   F956+1803              DB      18H,DKRW40-$-1
   888                  DKRW30:
   889   F958 CDE7F9            CALL    DPWRITE         ;  NO - WRITE SECTOR
   890                  DKRW40:
   891   F95B CD73F9            CALL    NEXTSEC         ; CHECK FOR MULT. SECTOR OPERATION
   892                          JRNC    DKRW20          ; LOOP UNTIL ALL SECTORS DONE
   893   F95E+30EC              DB      30H,DKRW20-$-1
```

```
                                STX     A,STATUS        ; STORE STATUS
F960+DD7701                     DB      0DDH,70H+A,STATUS
F963 E601                       ANI     QMBUSY          ; SEEK ERROR INDICATED?
F965 C8                         RZ                      ; NO - ALL DONE
                                JR      DKRW95          ; YES - RESET TRACK TABLE RNTRY
F966+1805                       DB      18H,DKRW95-$-1
                        ;
                        ; SEEK ERROR
                        DKRW90:
F968 F601                       ORI     QMBUSY          ; MARK AS SEEK ERROR
                                STX     A,STATUS        ; STORE STATUS
F96A+DD7701                     DB      0DDH,70H+A,STATUS
                        DKRW95:
F96D 3EFF                       MVI     A,0FFH          ; FORCE RECAL ON NEXT R/W
F96F CD1CFC                     CALL    SETRAK
F972 C9                         RET
                        ;
                        ;
                        ; SUBROUTINE TO CHECK FOR MULTIPLE SECTOR OPERATION
                        ; DECREMENTS SECTOR COUNT AND INCREMENTS SECTOR NUMBER.
                        ; SECTOR INTERLEAVE IS USED TO READ/WRITE SECTORS IN
                        ; LOGICAL ORDER.
                        ;
                        ; ENTRY: STATUS OF READ/WRITE IS IN 'A'
                        ;
CP/M MACRO ASSEM 2.0    #022    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

                        ; EXIT: CARRY IS SET IF NO FURTHER READ/WRITE OPERATIONS
                        ;  ARE NECESSARY (COUNT EXHAUSTED OR ERROR OCCURRED)
                        ;  PACKET COUNT AND SECTOR NUMBER ARE UPDATED.
                        ;
                        NEXTSEC:
F973 FE01                       CPI     1               ; STATUS OK?
F975 3F                         CMC                     ; (REVERSE SENSE OF CARRY)
F976 D8                         RC                      ;  EXIT - ERROR OCCURRED
                                DCRX    NSECT           ; DECREMENT SECTOR COUNT
F977+DD3506                     DB      0DDH,035H,NSECT
F97A 37                         STC                     ; SET CARRY IN CASE COUNT = 0
F97B C8                         RZ                      ;  EXIT ON ZERO COUNT
                        ;
                        ; INCREMENT DMA ADDRESS IN PACKET
                        ;
                                LDX     A,DMAHI         ; TAKE CARE OF POSSIBLE CARRY
F97C+DD7E05                     DB      0DDH,A*8+46H,DMAHI
F97F C602                       ADI     2
                                STX     A,DMAHI
F981+DD7705                     DB      0DDH,70H+A,DMAHI
                                LDX     A,SECTN         ; GET SECTOR NUMBER
F984+DD7E02                     DB      0DDH,A*8+46H,SECTN
F987 C601                       ADI     1               ; NEXT SECTOR
F989 47                         MOV     B,A
F98A E61F                       ANI     SECNUM          ; ISOLATE IT
F98C FE0A                       CPI     QKMXSECT        ; OVER THE LIMIT ON THE TRACK?
                                JRC     NEXTSC15        ; NO
F98E+3813                       DB      38H,NEXTSC15-$-1
```

```
                                JRNZ    NEXTSC10        ;   DEFINITELY OVER THE LIMIT.
F990+2006                       DB      20H,NEXTSC10-$-1
F992 3A2FFC                     LDA     BFORMAT         ;   AT SECTOR 10. OVER THE LIMIT ONLY IF
F995 A7                         ANA     A               ;     ROBIN DISK;
                                JRZ     NEXTSC20
F996+280C                       DB      28H,NEXTSC20-$-1
                        NEXTSC10:
                                LDX     A,SECTN
F998+DD7E02                     DB      0DDH,A*8+46H,SECTN
F99B E6E0                       ANI     NOT SECNUM
F99D F601                       ORI     1
                                STX     A,SECTN         ; STORE IT IN PACKET
F99F+DD7702                     DB      0DDH,70H+A,SECTN
F9A2 C9                         RET
                        ;
F9A3 3F                 NEXTSC15: CMC
                        NEXTSC20:
                                STX     B,SECNUM
F9A4+DD701F                     DB      0DDH,70H+B,SECNUM
F9A7 C9                         RET
                        ;
                                PAGE
CP/M MACRO ASSEM 2.0    #023    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

                        ; DKRWSET  -  SET UP FOR READ OR WRITE
                        ;
                        ; FUNCTION: SELECTS DRIVE AND CHECKS READY STATUS.  HOMES
                        ;  DRIVE IF IT HAS NOT BEEN ACCESSED PREVIOUSLY. COMPARES
                        ;  DESIRED TRACK WITH CURRENT TRACK.
                        ;
                        ; ENTRY: IX = PACKET ADDRESS
                        ;
                        ; EXIT: HL = ADDRESS OF TRACK TABLE ENTRY FOR DRIVE
                        ;       Z FLAG IS SET IF CURRENT TRACK = DESIRED TRACK
                        ;       CARRY FLAG IS SET IF DRIVE IS NOT READY
                        ;
                        DKRWSET:
F9A8 CD32FB                     CALL    DPSELDR         ; SELECT DRIVE AND CHECK READY STATUS
F9AB D8                         RC                      ;  EXIT - NOT READY
F9AC 3A31FC                     LDA     BTRACK          ; GET CURRENT TRACK
F9AF FEFF                       CPI     0FFH            ; FIRST ACCESS?
                                JRNZ    DKRWS10         ;  NO
F9B1+2008                       DB      20H,DKRWS10-$-1
F9B3 CDA1FA                     CALL    DPRECAL         ; HOME DRIVE
F9B6 C0                         RNZ                     ;  EXIT ON HOME ERROR
F9B7 AF                         XRA     A               ; SET TRACK ZERO IN TABLE
F9B8 CD1CFC                     CALL    SETRAK
                        DKRWS10:
F9BB 3A2FFC                     LDA     BFORMAT         ; COMPARE DESIRED TRACK WITH
F9BE A7                         ANA     A               ;  CURRENT TRACK. SINCE CURRENT
                                LDX     A,TRACKN        ;   TRACK IS ALWAYS KEPT IN TRUE FORM,
F9BF+DD7E03                     DB      0DDH,A*8+46H,TRACKN
                                JRZ     DKRWS20         ;    MUST DOUBLE DESIRED TRACK IF
F9C2+2801                       DB      28H,DKRWS20-$-1
```

```
1000   F9C4 87              ADD     A               ;     ROBIN MEDIA
1001                    DKRWS20:
1002   F9C5 47              MOV     B,A
1003   F9C6 3A31FC          LDA     BTRACK
1004   F9C9 A8              XRA     B               ; COMPARE WITH CURRENT TRACK (RESET CARRY)
1005   F9CA C9              RET
1006                    ;
1007                    ;
1008                         PAGE
CP/M MACRO ASSEM 2.0    #024    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

1009
1010                    ; DPREAD  -  READ A SECTOR
1011                    ;
1012                    ; FUNCTION:  READS ONE SECTOR FROM DISK.  RETRIES WHEN ERROR IS
1013                    ; ENCOUNTERED.
1014                    ;
1015                    ; ENTRY: SECTOR NUMBER AND DMA ADDRESS ARE IN PACKET
1016                    ;
1017                    ; EXIT:  A = FDC STATUS
1018                    ;
1019                    ;
1020                    DPREAD:
1021   F9CB CD47FA          CALL    DPRWEX          ; (RETURNS TO CALLER OF DPREAD)
1022                    ;
1023                    DPREADX:
1024   F9CE 21EDA2          LXI     H,INII          ; SET UP INI INSTR.
1025                    DPREADZ:
1026   F9D1 224600          SHLD    INOROUT
1027                        LDX     A,TRACKN        ; SET TRACK FOR SECTOR READ OR WRITE
1028   F9D4+DD7E03          DB      0DDH,A*8+46H,TRACKN
1029   F9D7 D361            OUT     QPTRKRG         ;  WILL DIFFER FROM PHYS. TRACK IF ROBIN
1030   F9D9 1E80            MVI     E,QCREADS       ; SEND COMMAND TO FDC
1031   F9DB CDE5FB          CALL    DPRWSET
1032   F9DE CD4000          CALL    RDORWR          ; TRANSFER DATA
1033   F9E1 FB              EI                      ; REENABLE
1034   F9E2 DB60            IN      QPCOMD          ; GET STATUS
1035   F9E4 E6BD            ANI     QMREAD          ; MASK ERRORS
1036   F9E6 C9              RET                     ; RETURN WITH STATUS
1037                    ;
1038                    ;
1039                    ;
1040                    ;
1041                    ; DPWRITE  -  WRITE A SECTOR
1042                    ;
1043                    ; FUNCTION:  WRITES ONE SECTOR TO DISK.  RETRIES WHEN ERROR IS
1044                    ; ENCOUNTERED.
1045                    ;
1046                    ; ENTRY: SECTOR NUMBER AND DMA ADDRESS ARE IN PACKET
1047                    ;
1048                    ; EXIT:  A = FDC STATUS
1049                    ;
1050                    DPWRITE:
1051   F9E7 CD47FA          CALL    DPRWEX          ; (RETURNS TO CALLER OF DPWRITE)
1052                    ;
```

```
1053                    DPWRTX:
1054    F9EA 21EDA3             LXI     H,OUTII         ; SET UP OUTI INSTR.
1055    F9ED 224600             SHLD    INOROUT
1056                            LDX     A,TRACKN        ; SET TRACK FOR SECTOR READ OR WRITE
1057    F9F0+DD7E03             DB      0DDH,A*8+46H,TRACKN
1058    F9F3 D361               OUT     QPTRKRG         ;  WILL DIFFER FROM PHYS. TRACK IF ROBIN
1059                    ;
1060    F9F5 CD19FA             CALL    GSODRV          ; GET "NOT READY" STATUS OF OTHER DRIVE
1061    F9F8 D5                 PUSH    D               ; SAVE FOR COMPARISON AFTER WRITE OF SECTOR
1062    F9F9 1EA0               MVI     E,QCWRTS        ; SEND THIS COMMAND TO FDC
1063    F9FB CDE5FB             CALL    DPRWSET
1064    F9FE CD4000             CALL    RDORWR          ; TRANSFER DATA
CP/M MACRO ASSEM 2.0    #025    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

1065    FA01 FB                 EI                      ; REENABLE
1066    FA02 DB60               IN      QPCOMD          ; GET STATUS
1067    FA04 E6FD               ANI     QMWRITE         ; MASK ERRORS
1068    FA06 47                 MOV     B,A             ; SAVE RESULT OF WRITE OPERATION
1069    FA07 0E03               MVI     C,3             ; 1.5  MSEC DELAY
1070    FA09 CD4A00             CALL    UDELAY          ; FOR CONTROLLER TO FINISH WRT CLEANUP
1071    FA0C CD19FA             CALL    GSODRV          ; GET "NOT READY" STATUS OF OTHER DRIVE
1072    FA0F D1                 POP     D               ; RESTORE OLD STATUS OF OTHER DRIVE
1073    FA10 BA                 CMP     D               ; COMPARE WITH OLD STATUS
1074    FA11 3E00               MVI     A,0             ; ASSUME DOOR OF OTHER DRIVE WAS NOT TOUCHED
1075                            JRZ     DPWRT3          ; IT WAS'NT
1076    FA13+2802               DB      28H,DPWRT3-$-1
1077    FA15 3E08               MVI     A,QMCRC         ; IT WAS - REPORT AS CRC ERROR
1078    FA17 B0         DPWRT3: ORA     B               ; GET STATUS RESULT OF WRITE OPERATION
1079    FA18 C9                 RET                     ;  RETURN WITH STATUS
1080                    ;
1081                    ;       GET "NOT READY" STATUS OF OTHER DRIVE
1082                    ;
1083                    GSODRV:
1084    FA19 3A2EFC             LDA     BCURDRV         ; GET CURRENT DRIVE SELECTED
1085    FA1C EE01               XRI     1               ; SWITCH TO OTHER DRIVE
1086    FA1E D340               OUT     QPSTAT          ; SELECT OTHER DRIVE
1087    FA20 CD2EFA             CALL    DELAY7          ; DELAY 7 USECS
1088    FA23 DB60               IN      QPCOMD          ; GET STATUS OF OTHER DRIVE
1089    FA25 E680               ANI     QMNRDY          ; THIS BIT TELLS US IF DOOR WAS OPENED OR CLOSED
1090    FA27 57                 MOV     D,A             ; SAVE FOR COMPARISON AFTER WRITE OF SECTOR
1091    FA28 3A30FC             LDA     TCURDRV         ; CURRENT DRIVE WITH PRE COMP BIT SET IF NECESSARY
1092    FA2B D340               OUT     QPSTAT          ; RESELECT CURRENT DRIVE
1093    FA2D 7A                 MOV     A,D             ; RESTORE FOR COMPARISON AFTER WRITE OF SECTOR
1094    FA2E C9         DELAY7: RET                     ; PROVIDES ~ 7 USEC DELAY
1095                    ;
1096                    ;
1097                    ; DPRDADR  -  READ ADDRESS
1098                    ;
1099                    ; FUNCTION: READS ADDRESS OF A DISK SECTOR
1100                    ;       (USED TO DETERMINE TRACK NUMBER)
1101                    ;
1102                    ; ENTRY: N/A
1103                    ;
1104                    ; EXIT: TRACK NUMBER IS IN SECTOR REGISTER OF FDC
1105                    ;  (NO DATA IS TRANSFERRED)
```

```
1106                    ;
1107                    ;
1108                    DPRDADR:
1109    FA2F CD47FA             CALL    DPRWEX          ; (RETURNS TO CALLER OF DPRDADR)
1110                    ;
1111                    DPRDADRX:
1112    FA32 21DB63             LXI     H,ININ          ; SET UP IN QPDATA INSTR.
1113    FA35 224600             SHLD    INOROUT
1114    FA38 1EC0               MVI     E,QCRDADR       ; SEND COMMAND TO FDC
1115    FA3A CDE5FB             CALL    DPRWSET
1116    FA3D CD4000             CALL    RDORWR          ; TRANSFER DATA
1117    FA40 FB                 EI                      ; REENABLE
1118    FA41 DB60               IN      QPCOMD          ; GET STATUS
1119    FA43 E69D               ANI     QMRDADR         ; MASK ERRORS
1120    FA45 37                 STC                     ; INHIBIT ADVANCED ERROR RECOVERY
CP/M MACRO ASSEM 2.0    #026    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

1121    FA46 C9                 RET                     ; RETURN WITH STATUS
1122                    ;
1123                    ;
1124                    ;
1125                    ; DPRWEX  -  EXECUTES A READ OR WRITE OPERATION
1126                    ;
1127                    ; FUNCTION: EXECUTES A READ OR WRITE OPERATION AND PERFORMS
1128                    ;  ERROR RECOVERY AS NECESSARY.
1129                    ;
1130                    ; ENTRY: ADDRESS OF PHYSICAL I/O ROUTINE IS ON TOP OF STACK
1131                    ;
1132                    ; EXIT:  A = FDC STATUS
1133                    ;
1134                    DPRWEX:
1135    FA47 E1                 POP     H               ; GET I/O ROUTINE ADDRESS
1136    FA48 E5                 PUSH    H               ; SAVE FOR ERROR RECOVERY
1137    FA49 CDA0FA             CALL    DPRWE90         ; EXECUTE THE ROUTINE
1138    FA4C E1                 POP     H               ; RECOVER I/O ROUTINE ADDRESS
1139    FA4D C8                 RZ                      ; EXIT IF NO ERRORS
1140                    ; ERROR RECOVERY PROCESSING
1141    FA4E 5F                 MOV     E,A             ; SAVE ERROR CODES
1142                            JRC     DPRWE10         ; JUST A FEW RETRIES IF CARRY SET
1143    FA4F+3804               DB      38H,DPRWE10-$-1
1144    FA51 E618               ANI     QMCRC+QMRNF     ; CRC ERROR OR RNF?
1145                            JRNZ    DPRWE20         ;  YES - PROCESS IT
1146    FA53+2014               DB      20H,DPRWE20-$-1
1147                    ;
1148                    ; REPEAT OPERATION UNTIL RETRY COUNT IS EXHAUSTED
1149                    ;
1150                    DPRWE10:
1151    FA55 1604               MVI     D,QKDRETRY      ; INITIALIZE RETRY COUNTER
1152                    DPRWE15:
1153    FA57 D5                 PUSH    D               ; STASH RETRY COUNT AND ORIGINAL ERROR CODES
1154    FA58 3ED0               MVI     A,QCTERM
1155    FA5A D360               OUT     QPCOMD          ; TERMINATE COMMAND
1156    FA5C E5                 PUSH    H               ; PRESERVE I/O ROUTINE ADDRESS
1157    FA5D CDA0FA             CALL    DPRWE90         ; EXECUTE IT
1158    FA60 E1                 POP     H               ; GET BACK STUFF THAT WAS STASHED
```

```
 1159   FA61 D1                POP     D
 1160   FA62 C8                RZ                      ; EXIT IF SUCCESSFUL
 1161   FA63 15                DCR     D               ; ANY MORE RETRIES LEFT?
 1162                          JRNZ    DPRWE15         ; TRY AGAIN
 1163   FA64+20F1              DB      20H,DPRWE15-$-1
 1164   FA66 7B                MOV     A,E             ; GET FDC STATUS
 1165   FA67 A7                ANA     A               ; SET TO NON-0 TO INDICATE ERROR
 1166   FA68 C9                RET
 1167                  ;
 1168                  ; REPEAT OPERATION, THEN PERFORM ADVANCED RECOVERY
 1169                  ;
 1170                  DPRWE20:
 1171   FA69 CD55FA            CALL    DPRWE10         ; REPEAT OPERATION
 1172   FA6C C8                RZ                      ; EXIT IF SUCCESSFULL
 1173                  ; RESTORE DRIVE, SEEK AGAIN AND DO OPERATION ONE MORE
 1174   FA6D D5                PUSH    D               ; SAVE I/O ROTINA ADDRESS, RETRY COUNT AND
 1175   FA6E E5                PUSH    H               ;  ORIGINAL ERROR STATUS
 1176   FA6F CDA1FA            CALL    DPRECAL         ; RESTORE
CP/M MACRO ASSEM 2.0    #027    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

 1177                          JRNZ    DPRWE80         ; HARD RESTORE ERROR
 1178   FA72+2027              DB      20H,DPRWE80-$-1
 1179   FA74 CDCCFA            CALL    DPSEEK1         ; SEEK TO TRACK
 1180                          JRNZ    DPRWE80         ; HARD SEEK ERROR
 1181   FA77+2022              DB      20H,DPRWE80-$-1
 1182   FA79 E1                POP     H               ; GET ALL THAT STUFF BACK
 1183   FA7A D1                POP     D
 1184   FA7B CD55FA            CALL    DPRWE10         ;  AND DO SOME MORE RETRIES
 1185   FA7E C8                RZ                      ; SUCCESS
 1186                          LDX     D,TRACKN        ; NOW WE TRY TO SNEAK UP ON IT FROM
 1187   FA7F+DD5603            DB      0DDH,D*8+46H,TRACKN
 1188   FA82 E5                PUSH    H               ;  THE OTHER SIDE: SEEK TO TRACK 79
 1189   FA83 D5                PUSH    D               ;   AND THEN BACK TO THE TARGET TRACK.
 1190                          MVIX    QKMXTRK,TRACKN  ;    TO DO THIS, WE MUST SVE TARGET TRACK NO.
 1191   FA84+DD36034F          DB      0DDH,36H,TRACKN,QKMXTRK
 1192   FA88 CDDFFA            CALL    DPSEEKX         ;     AND SUBSTITUTE 79 IN PACKET
 1193   FA8B D1                POP     D               ; PUT BACK CORRECT TRACK INTO PACKET
 1194                          STX     D,TRACKN
 1195   FA8C+DD7203            DB      0DDH,70H+D,TRACKN
 1196   FA8F D5                PUSH    D               ; REMEMBER: E HAS ORIGINAL ERROR STATUS
 1197                          JRNZ    DPRWE80         ; OOPS: COLDN'T GET TO TRACK 79
 1198   FA90+2009              DB      20H,DPRWE80-$-1
 1199   FA92 CDCCFA            CALL    DPSEEK1         ; BACK TO DESIRED TRACK
 1200                          JRNZ    DPRWE80         ; SEEK ERROR
 1201   FA95+2004              DB      20H,DPRWE80-$-1
 1202   FA97 D1                POP     D
 1203   FA98 E1                POP     H
 1204                          JR      DPRWE10         ; FINAL ATTEMPT. RETURN TO CALLER.
 1205   FA99+18BA              DB      18H,DPRWE10-$-1
 1206                  ;
 1207                  ; RESTORE/SEEK ERROR EXIT
 1208                  ;
 1209                  DPRWE80:
 1210   FA9B E1                POP     H               ; FIXUP STACK
 1211   FA9C E1                POP     H
```

```
1212   FA9D F601              ORI     QMBUSY          ; MARK AS SEEK ERROR
1213   FA9F C9                RET
1214                  ;
1215                  ; INDIRECT CALL
1216                  ;
1217                  DPRWE90:
1218   FAA0 E9                PCHL
1219                  ;
1220                  ;
1221                  ;
1222                  ;
1223                  ; DPRECAL  -  RESTORE DRIVE
1224                  ;
1225                  ; FUNCTION - RETURNS THE HEAD TO TRACK ZERO
1226                  ;   CHECKS FORMAT TO SEE IF VT-180 DISK IS IN DRIVE
1227                  ;
1228                  ; ENTRY: N/A
1229                  ;
1230                  ; EXIT:
1231                  ;       A = DISK CONTROLLER STATUS
1232                  ;       ZERO FLAG IS SET IF RESTORE WAS SUCCESSFUL
CP/M MACRO ASSEM 2.0  #028    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

1233                  ;
1234                  DPRECAL:
1235   FAA1 CDB2FA            CALL    DPRECALX        ; PERFORM RESTORE
1236   FAA4 C8                RZ                      ;  OK
1237                  ; ERROR RECOVERY - STEP IN FIVE TIMES AND REPEAT
1238   FAA5 0605              MVI     B,5             ; SET NR OF STEPS
1239                  DPRECAL1:
1240   FAA7 3E48              MVI     A,QCSTEPIN
1241   FAA9 CDDDFB            CALL    DPCOMD          ; STEP IN ONCE
1242                          DJNZ    DPRECAL1        ; REPEAT
1243   FAAC+10F9              DB      10H,DPRECAL1-$-1
1244   FAAE CDB2FA            CALL    DPRECALX        ; RESTORE AGAIN
1245   FAB1 C9                RET                     ; EXIT COME WHAT MAY
1246                  ;
1247                  ;
1248                  ;
1249                  ; DPRECALX - DO RESTORE OPERATION
1250                  ;
1251                  DPRECALX:
1252   FAB2 3E08              MVI     A,QCREST
1253   FAB4 CDDDFB            CALL    DPCOMD          ; DO RESTORE
1254   FAB7 DB60              IN      QPCOMD          ; GET FDC STATUS
1255   FAB9 4F                MOV     C,A             ; SAVE STATUS
1256   FABA E604              ANI     QMTZERO         ; AT TRACK ZERO?
1257                          JRNZ    DPRECALX1       ;  YES
1258   FABC+2004              DB      20H,DPRECALX1-$-1
1259   FABE 2F                CMA                     ; SET NONZERO STATUS
1260   FABF B7                ORA     A               ; SET FLAGS
1261   FAC0 79                MOV     A,C             ; GET STATUS BACK
1262   FAC1 C9                RET                     ; EXIT - NOT AT TRACK ZERO
1263                  DPRECALX1:
1264   FAC2 79                MOV     A,C             ; GET STATUS BACK
```

```
1265   FAC3 E691              ANI     QMREST          ; MASK ERRORS
1266   FAC5 79                MOV     A,C             ; RETURN FULL STATUS
1267   FAC6 C9                RET                     ; RETURN WITH STATUS
1268                  ;
1269                  ;
1270                  ; DPSEEK  -  SEEK TO TRACK
1271                  ;
1272                  ; FUNCTION: SEEKS TO THE DESIRED TRACK.  TESTS FOR ERRORS
1273                  ; AND RETRIES AS NECESSARY.  SETS PRECOMPENSATION BITS FOR
1274                  ; RAINBOW DISKETTE.  PERFORMS SPECIAL SEEK FOR VT180 FORMAT
1275                  ; DISKETTE.
1276
1277                  ;
1278                  ; ENTRY: DESIRED TRACK NUMBER IS IN PACKET
1279                  ;  POINTER TO PACKET IS IN IX.
1280                  ;
1281                  ; EXIT: A = FDC STATUS
1282                  ;       ZERO FLAG IS SET IF SEEK WAS SUCCESSFUL
1283                  ;
1284                  DPSEEK:
1285   FAC7 3A31FC            LDA     BTRACK          ; PUT CURRENT TRACK NO INTO REG
1286   FACA D361              OUT     QPTRKRG
1287                  DPSEEK1:
1288   FACC 3A2FFC            LDA     BFORMAT         ; GET CURRENT FORMAT
CP/M MACRO ASSEM 2.0    #029    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

1289   FACF A7                ANA     A               ; IS IT RAINBOW?
1290   FAD0 C2F8FA            JNZ     DPSEEKV         ;  NO - DO VT-180 SEEK
1291   FAD3 CDDFFA            CALL    DPSEEKX         ; DO SEEK OPERATION
1292   FAD6 C8                RZ                      ; OK
1293                  ; ERROR RECOVERY - RESTORE AND SEEK AGAIN
1294   FAD7 CDA1FA            CALL    DPRECAL         ; RESTORE DRIVE
1295   FADA C0                RNZ                     ; HARD RESTORE ERROR
1296   FADB CDDFFA            CALL    DPSEEKX         ; TRY SEEK AGAIN
1297   FADE C9                RET
1298                  ;
1299                  ;
1300                  ; DPSEEKX  -  PERFORM SEEK OPERATION
1301                  ;
1302                  DPSEEKX:
1303                          LDX     A,TRACKN        ; GET TRACK NR FROM PACKET
1304   FADF+DD7E03            DB      0DDH,A*8+46H,TRACKN
1305   FAE2 FE50              CPI     QKMXTRK+1       ; CATCH ANY BAD TRACK REQUESTS SO AS NOT
1306                          JRNC    DPSEEKE         ;  TO HANG IN AN IMPOSSIBLE SEEK
1307   FAE4+300E              DB      30H,DPSEEKE-$-1
1308   FAE6 D363              OUT     QPDATA          ; SEND IT TO FDC
1309   FAE8 E3                XTHL                    ; DELAY A BIT BEFORE COMMAND REG WRITE
1310   FAE9 E3                XTHL
1311   FAEA 3E1C              MVI     A,QCSEEK
1312   FAEC CDDDFB            CALL    DPCOMD          ; EXECUTE SEEK
1313                  ;       MVI     C,40            ; DELAY FOR HEAD SETTLING
1314                  ;       CALL    UDELAY
1315   FAEF DB60              IN      QPCOMD          ; GET STATUS
1316   FAF1 E699              ANI     QMSEEK          ; MASK ERRORS
1317   FAF3 C9                RET                     ; RETURN WITH STATUS
```

```
                                
                        DPSEEKE:
FAF4 3E01                       MVI     A,QMBUSY        ; ILLEGAL TRACK REQUESTED. INDICATE SEEK
FAF6 A7                         ANA     A               ;  ERROR WITH NO OTHER STATUS
FAF7 C9                         RET
                        ;
                        ;
                        ;
                        ; DPSEEKV  -  VT-180 FORMAT SEEK ROUTINE
                        ;
                        ; FUNCTION: PERFORMS SEEK ON A VT-180 FORMAT (40 TRACK)
                        ; DISKETTE.  DOES A SEEK WITH NO VERIFICATION TO THE
                        ; SPECIFIED VT-180 TRACK MULTIPLIED BY TWO, THEN READS
                        ; AN ADDRESS FROM THE DISKETTE TO VERIFY THAT THE
                        ; SEEK WAS SUCCESSFUL.
                        ;
                        ; ENTRY: IX = PACKET ADDRESS
                        ;
                        ; EXIT:  A = STATUS
                        ;
                        DPSEEKV:
FAF8 CD09FB                     CALL    DPSEEKVX        ; PERFORM SEEK
FAFB CD09FB                     CALL    DPSEEKVX        ; SEEK ERROR - TRY AGAIN
                        ;
                        ;       SEEK ERROR AGAIN - HOME DISK AND RETRY
                        ;
FAFE CDA1FA                     CALL    DPRECAL         ; HOME DISK
CP/M MACRO ASSEM 2.0    #030    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

FB01 C0                         RNZ                     ; EXIT ON RECAL ERROR
FB02 CD09FB                     CALL    DPSEEKVX        ; TRY SEEK AGAIN
FB05 CD09FB                     CALL    DPSEEKVX        ; AND AGAIN
FB08 C9                         RET
                        ;
                        ;
                        ; DPSEEKVX - PERFORM VT-180 SEEK OPERATION
                        ;
                        DPSEEKVX:
                                LDX     A,TRACKN        ; GET DESIRED TRACK
FB09+DD7E03                     DB      0DDH,A*8+46H,TRACKN
FB0C 87                         ADD     A               ; MULTIPLY TRACK BY 2
FB0D FE50                       CPI     QKMXTRK+1       ; CATCH ANY BAD TRACK REQUESTS SO AS NOT
                                JRNC    DPSEEKE         ;  TO HANG IN AN IMPOSSIBLE SEEK
FB0F+30E3                       DB      30H,DPSEEKE-$-1
FB11 CD27FB                     CALL    DPSEEKN         ; SEEK WITH NO VERIFY
FB14 CD2FFA                     CALL    DPRDADR         ; READ ADDRESS
FB17 4F                         MOV     C,A             ; SAVE STATUS
FB18 A7                         ANA     A               ; READ ERROR?
FB19 C0                         RNZ                     ;  YES - EXIT
FB1A DB62                       IN      QPSECRG         ; GET TRACK NUMBER
                                CMPX    TRACKN          ; COMPARE WITH DESIRED TRACK
FB1C+DDBE03                     DB      0DDH,0BEH,TRACKN
FB1F 79                         MOV     A,C             ; GET STATUS BACK
FB20 C0                         RNZ                     ; ERROR - NOT AT REQUESTED TRACK
FB21 E1                         POP     H               ; NO ERROR - EMPTY STK FOR 'GOOD' RET
```

```
  1371   FB22 C9                 RET
  1372                   ;
  1373                   ; DPSEEKN  -  SEEK WITH NO VERIFY
  1374                   ;
  1375                   ; FUNCTION: SEEKS A SPECIFIED TRACK BUT DOES NO VERIFY THE
  1376                   ; SUCCESS OF THE SEEK.  USED FOR STARTING MOTOR AND
  1377                   ; THE VT-180 SEEK.
  1378                   ;
  1379                   ; ENTRY: A = DESIRED TRACK NUMBER
  1380                   ;
  1381                   ; EXIT: N/A
  1382                   ;
  1383                   DPSEKH0:
  1384   FB23 1610               MVI     D,QCSEKH0       ;SEEK WITH NO HEAD LOAD
  1385                           JR      DPSEKN3
  1386   FB25+1802               DB      18H,DPSEKN3-$-1
  1387                   ;
  1388                   DPSEEKN:
  1389   FB27 1618               MVI     D,QCSEEKN       ;SEEK WITH HEAD LOAD
  1390   FB29 D363       DPSEKN3: OUT    QPDATA          ; SEND TRACK NR TO FDC
  1391   FB2B 7A                 MOV     A,D             ;SEEK COMMAND
  1392   FB2C E3                 XTHL
  1393   FB2D E3                 XTHL
  1394   FB2E CDDDFB             CALL    DPCOMD          ; DO SEEK
  1395   FB31 C9                 RET
  1396                   ;
  1397                   ;
  1398                   ;
  1399                   ;
  1400                   ; DPSELDR  -  SELECT DRIVE AND CHECK IF IT'S READY
CP/M MACRO ASSEM 2.0    #031    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

  1401                   ;
  1402                   ; ENTRY: DRIVE NUMBER IS IN PACKET
  1403                   ;
  1404                   ; EXIT: DRIVE NUMBER IN E
  1405                   ;       IF DRIVE WAS NOT READY, CARRY FLAG IS SET
  1406                   ;       AND 'NOT READY' STATUS IS STORED IN PACKET
  1407                   ;
  1408                   ;
  1409                   DPSELDR:
  1410   FB32 CD13FC             CALL    GETDRV          ; GET DRIVE NUMBER
  1411   FB35 5F                 MOV     E,A
  1412                           LDX     A,TRACKN        ; NOW SEE IF WRITE PRECOMP BITS WILL BE
  1413   FB36+DD7E03             DB      0DDH,A*8+46H,TRACKN
  1414   FB39 FE3D               CPI     QKPCTRK         ;   NEEDED: I.E., WHETHER WE ARE TO BE
  1415   FB3B 3E00               MVI     A,0             ;     BEYOND TRACK 60
  1416                           JRC     DPSEL1
  1417   FB3D+3802               DB      38H,DPSEL1-$-1
  1418   FB3F 3E40               MVI     A,QKPCBIT
  1419                   DPSEL1:
  1420   FB41 B3                 ORA     E
  1421   FB42 3230FC             STA     TCURDRV         ; DRIVE SELECTED WITH PRE COMP
  1422   FB45 D340               OUT     QPSTAT          ; SELECT DRIVE
  1423                   DPSEL2:
```

```
1424   FB47 DB60            IN      QPCOMD          ; GET FDC STATUS
1425   FB49 E681            ANI     QMBUSY+QMNRDY   ; READY AND NOT BUSY?
1426                        JRZ     DPSEL3          ;  YES
1427   FB4B+2811            DB      28H,DPSEL3-$-1
1428   FB4D E601            ANI     QMBUSY          ; BUSY?
1429                        JRNZ    DPSEL2          ;  YES - WAIT IT OUT
1430   FB4F+20F6            DB      20H,DPSEL2-$-1
1431   FB51 DB61            IN      QPTRKRG         ;  NO: NOT READY. DUMMY SEEK TO CURRENT TRACK
1432   FB53 CD27FB          CALL    DPSEEKN         ;   TO READY DRIVE IF POSSIBLE
1433   FB56 DB60            IN      QPCOMD          ; GET FDC STATUS
1434   FB58 E680            ANI     QMNRDY          ; STILL NOT READY?
1435                        JRNZ    DPSELS2         ;  YES - RETURN NOT READY STATUS
1436   FB5A+206B            DB      20H,DPSELS2-$-1
1437                        JR      DPSEL3C         ; IGNORE CHECK FOR ISSUANCE OF 'HLT' TO 1793
1438   FB5C+182A            DB      18H,DPSEL3C-$-1
1439                ;
1440                DPSEL3:
1441   FB5E 3A2EFC          LDA     BCURDRV         ; GET CURRENT DRIVE NR
1442   FB61 AB              XRA     E               ; SAME AS DESIRED DRIVE?
1443                        JRZ     DPSEL4          ;  YES - GO SEE IF FIRST ACCESS AFTER RESET
1444   FB62+2838            DB      28H,DPSEL4-$-1
1445                        LDX     A,FNCCOD        ; GET FUNCTION CODE
1446   FB64+DD7E00          DB      0DDH,A*8+46H,FNCCOD
1447   FB67 FE14            CPI     QKWTCOM         ; WRITE TO BE DONE?
1448                        JRNZ    DPSEL3C         ; NO
1449   FB69+201D            DB      20H,DPSEL3C-$-1
1450                ;
1451   FB6B 3A2EFC          LDA     BCURDRV         ; YES - THEN GET CURRENT DRIVE
1452   FB6E E602            ANI     2               ; CHECK FOR SELECT OF A OR B
1453   FB70 57              MOV     D,A             ; TO C OR D; C OR D TO A OR B
1454   FB71 7B              MOV     A,E             ; NEW DRIVE TO SELECT
1455   FB72 E602            ANI     2
1456   FB74 BA              CMP     D
CP/M MACRO ASSEM 2.0    #032    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

1457                        JRNZ    DPSEL3A
1458   FB75+2007            DB      20H,DPSEL3A-$-1
1459                ;
1460                ;       A TO B, B TO A, C TO D, D TO C
1461                ;
1462   FB77 0E3C            MVI     C,60            ; DELAY FOR 30 MSEC TO ALLOW HEADS TO
1463   FB79 CD4A00          CALL    UDELAY          ; SETTLE DURING HEAD LOAD
1464                        JR      DPSEL3C
1465   FB7C+180A            DB      18H,DPSEL3C-$-1
1466                ;
1467                ;       THE SELECT FROM A OR B TO C OR D; C OR D TO A OR B REQUIRES
1468                ;       THE ISSUANCE OF A SEEK WITH QMHLT = 0, THEN QMHLT = 1, WITH NO VERIFY
1469                ;       THIS INHIBITS THE "HLT" INPUT TO THE 1793 FOR ~ 500 MSECS
1470                ;       WHICH ALLOWS THE MOTOR TO GET UP TO SPEED.
1471                ;
1472                DPSEL3A:
1473   FB7E DB61            IN      QPTRKRG         ;CURRENT TRACK
1474   FB80 CD23FB          CALL    DPSEKH0         ;SEEK WITH NO HEAD LOAD
1475   FB83 DB61            IN      QPTRKRG
1476   FB85 CD27FB          CALL    DPSEEKN         ;SEEK WITH HEAD LOAD
```

```
1477                    ;
1478   FB88 7B          DPSEL3C: MOV    A,E             ; SET NEW DRIVE AS CURRENT
1479   FB89 322EFC              STA    BCURDRV
1480   FB8C CDD2FB              CALL   GETRAK          ; GET CURR TRACK FROM TABLE
1481   FB8F 3231FC              STA    BTRACK          ;  AND STORE IT
1482                    ;        MVI    C,140           ; DELAY FOR HEAD SETTLING
1483                    ;        CALL   UDELAY
1484                    ; SET FORMAT AND TRACK REGISTER
1485   FB92 21F4FF              LXI    H,TFORMAT       ; BASE OF FORMAT TABLE
1486   FB95 1600                MVI    D,0
1487   FB97 19                  DAD    D               ; OFFSET INTO TABLE
1488   FB98 7E                  MOV    A,M             ; GET FORMAT INFO
1489   FB99 322FFC              STA    BFORMAT         ; STORE IN CURRENT FORMAT
1490                   DPSEL4:
1491   FB9C 3A31FC              LDA    BTRACK
1492   FB9F 3C                  INR    A               ; CURR TRACK IS FF ON FIRST ACCESS AFTER DISK
1493   FBA0 C0                  RNZ                    ;  SYSTEM RESET. IF NOT, CONSIDER DRIVE READY.
1494                   DPSEL5:
1495   FBA1 DB61                IN     QPTRKRG         ; FORCE TYPE 1 STATUS
1496   FBA3 CD27FB              CALL   DPSEEKN
1497                                                   ; ON FIRST ACCESS TO A DRIVE AFTER A DISK
1498                                                   ;  SYSTEM RESET, MAKE SURE DISK IS IN
1499                                                   ;  DRIVE RIGHT SIDE UP BY LOOKING FOR A
1500                                                   ;  CHANGE IN THE INDEX PULSE STATUS. ON A
1501                                                   ;   TIMEOUT, DRIVE IS CONSIDERED NOT READY.
1502   FBA6 21409C              LXI    H,QKDRCNT
1503                   DPSEL6:
1504   FBA9 DB60                IN     QPCOMD          ; READ STATUS ONCE TO GET RID OF SPURIOUS
1505   FBAB DB60                IN     QPCOMD          ;  INDEX STATUS
1506   FBAD E602                ANI    QMINDEX         ; SAVE STARTING INDEX BIT STATE
1507   FBAF 47                  MOV    B,A
1508   FBB0 CDB8FB              CALL   DPSELS          ; WAIT FOR 1 CHANGE IN INDEX
1509   FBB3 CDB8FB              CALL   DPSELS          ; WAIT FOR 2ND CHANGE
1510   FBB6 A7                  ANA    A               ; CLEAR CARRY
1511   FBB7 C9                  RET                    ; DISK IS OK
1512                   DPSELS:                        ; SUBROUTINE: CHECKS FOR CHANGE IN INDEX BIT
CP/M MACRO ASSEM 2.0    #033    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

1513   FBB8 2B                  DCX    H
1514   FBB9 7C                  MOV    A,H
1515   FBBA B5                  ORA    L
1516                            JRZ    DPSELS1         ; TIMEOUT: DISK NOT READY
1517   FBBB+2809                DB     28H,DPSELS1-$-1
1518   FBBD DB60                IN     QPCOMD
1519   FBBF E602                ANI    QMINDEX
1520   FBC1 B8                  CMP    B
1521   FBC2 47                  MOV    B,A             ; SAVE NEW INDEX STATUS
1522   FBC3 C0                  RNZ                    ; AHA-- CHANGE HAS OCCURRED.
1523                            JR     DPSELS
1524   FBC4+18F2                DB     18H,DPSELS-$-1
1525                   DPSELS1:
1526   FBC6 C1                  POP    B
1527                   DPSELS2:
1528   FBC7 DB61                IN     QPTRKRG         ; GET CURRENT TRACK #
1529   FBC9 CD23FB              CALL   DPSEKH0         ; UNLOAD HEADS
```

```
                                MVIX    QMNRDY,STATUS   ; SET NOT READY STATUS
FBCC+DD360180                   DB      0DDH,36H,STATUS,QMNRDY
FBD0 37                         STC                     ; CARRY MEANS NOT READY
FBD1 C9                         RET
                        ;
                        ;
                        ; GETRAK -      GET CURRENT TRACK NUMBER FROM TRACK TABLE
                        ;
                        ; ENTRY:        A = DRIVE NUMBER
                        ;
                        ; EXIT:         A = TRACK NUMBER
                        ;               HL= TRACK TABLE POINTER FOR DRIVE
                        ;
                        GETRAK:
FBD2 E602                       ANI     2               ; ** FOR RX-50 COUPLED DRIVES
FBD4 4F                         MOV     C,A
FBD5 0600                       MVI     B,0
FBD7 21F0FF                     LXI     H,TTRACK        ; BASE OF TRACK TABLE
FBDA 09                         DAD     B               ; OFFSET INTO TRACK TABLE
FBDB 7E                         MOV     A,M             ; GET TRACK FOR THIS DRIVE
FBDC C9                         RET
                        ;
                        ;
                        ;
                        ;
                        ; DPCOMD  -  EXECUTE A TYPE 1 COMMAND
                        ;
                        ; WAITS UNTIL COMMAND IS COMPLETED BEFORE RETURNING
                        ;
                        ; ENTRY: COMMAND CODE IS IN 'A' REG.
                        ;
                        DPCOMD:
FBDD D360                       OUT     QPCOMD          ; SEND COMMAND TO FDC
                        DPCOMD1:
FBDF DB40                       IN      QPSTAT          ; GET GENERAL STATUS
FBE1 87                         ADD     A               ; SHIFT INT. BIT TO SIGN
FBE2 F8                         RM                      ; EXIT ON INTERRUPT
                                JR      DPCOMD1         ; LOOP UNTIL DONE
FBE3+18FA                       DB      18H,DPCOMD1-$-1
CP/M MACRO ASSEM 2.0    #034    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

                        ;
                        ;
                        ; DPRWSET  - SET UP FDC AND SEND TYPE 2 OR 3 (READ/WRITE) COMMAND
                        ; FUNCTION:  SENDS SECTOR TO FDC.  ADDS HEAD LOAD BIT TO COMMAND
                        ; IF HEAD IS NOT ALREADY LOADED.  SETS UP REGISTERS FOR INI AND
                        ; OUTI INSTRUCTIONS.  SENDS COMMAND TO FDC.
                        ;
                        ;
                        ; ENTRY: FDC COMMAND IS IN "E" REG.  SECTOR IS IN PACKET.
                        ;
                        ; EXIT: N/A
                        ;
                        DPRWSET:
FBE5 DB60                       IN      QPCOMD          ; GET CONTROLLER STATUS
```

```
1583   FBE7 E601              ANI     QMBUSY          ; BUSY?
1584                          JRNZ    DPRWSET         ;  YES - WAIT UNTIL NOT BUSY
1585   FBE9+20FA              DB      20H,DPRWSET-$-1
1586   FBEB E3                XTHL                    ; NEED SOME DELAY. WE'LL UNDO IT SOON
1587   FBEC 3ED0              MVI     A,QCTERM        ; SEND TERMINATE COMMAND
1588   FBEE D360              OUT     QPCOMD          ;  TO FORCE TYPE 1 STATUS
1589   FBF0 E3                XTHL                    ; DELAY - MATCHES THE ONE ABOVE
1590   FBF1 0E63              MVI     C,QPDATA        ; SET UP DATA PORT NUMBER
1591                          LDX     L,DMALOW        ; SET UP DMA ADDRESS
1592   FBF3+DD6E04            DB      0DDH,L*8+46H,DMALOW
1593                          LDX     H,DMAHI
1594   FBF6+DD6605            DB      0DDH,H*8+46H,DMAHI
1595                          LDX     A,SECTN         ; GET SECTOR NUMBER
1596   FBF9+DD7E02            DB      0DDH,A*8+46H,SECTN
1597   FBFC E61F              ANI     SECNUM          ; ISOLATE SECTOR NUMBER
1598   FBFE D362              OUT     QPSECRG         ; SEND IT TO FDC
1599   FC00 E3                XTHL                    ; DELAY BEFORE STATUS READ
1600   FC01 E3                XTHL
1601   FC02 E3                XTHL
1602   FC03 E3                XTHL
1603   FC04 E3                XTHL
1604   FC05 E3                XTHL
1605   FC06 F3                DI                      ; DISABLE FOR DISK I/O
1606   FC07 DB60              IN      QPCOMD          ; GET FDC STATUS
1607   FC09 E620              ANI     QMHLT           ; HEAD ALREADY LOADED?
1608   FC0B 7B                MOV     A,E             ; GET COMMAND
1609                          JRNZ    DPRWSET4        ;;  SKIP IF HEAD ALREADY LOADED
1610   FC0C+2002              DB      20H,DPRWSET4-$-1
1611   FC0E F604              ORI     QMEFLG          ; ADD HEAD LOAD FLAG
1612                  DPRWSET4:
1613   FC10 D360              OUT     QPCOMD          ; SEND TO CONTROLLER
1614   FC12 C9                RET
1615                  ;
1616                  ;
1617                  ; GETDRV  -  GET DRIVE NUMBER FROM PACKET
1618                  ;
1619                  ; ENTRY: IX = PACKET ADDRESS
1620                  ;
1621                  ; EXIT: A = DRIVE NUMBER
1622                  ;
1623                  GETDRV:
1624                          LDX     A,DRIVEN        ; GET DRIVE NUMBER
CP/M MACRO ASSEM 2.0    #035    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

1625   FC13+DD7E02            DB      0DDH,A*8+46H,DRIVEN
1626   FC16 E660              ANI     DRVNUM          ; ISOLATE IT
1627   FC18 07                RLC                     ; MOVE INTO BITS 0-2
1628   FC19 07                RLC
1629   FC1A 07                RLC
1630   FC1B C9                RET
1631                  ;
1632                  ;
1633                  ; SETRAK  -  STORE TRACK NUMBER IN TRACK TABLE
1634                  ;
1635                  ; ENTRY: A = TRACK NUMBER
```

```
                        ;        IX = PACKET ADDRESS
                        ;
                        ; EXIT: TTRACK CONTAINS NEW TRACK VALUE
                        ;
                        SETRAK:
FC1C 3231FC                     STA     BTRACK          ; CURRENT TRACK FOR CURRENT DRIVE
FC1F 47                         MOV     B,A             ; SAVE TRACK
FC20 CD13FC                     CALL    GETDRV          ; GET DRIVE NUMBER
FC23 E602                       ANI     2               ; ** FOR RX-50 COUPLING
FC25 5F                         MOV     E,A             ; MAKE 16-BIT DRIVE NR
FC26 1600                       MVI     D,0
FC28 21F0FF                     LXI     H,TTRACK        ; BASE OF TRACK TABLE
FC2B 19                         DAD     D               ; OFFSET INTO TABLE
FC2C 70                         MOV     M,B             ; STORE TRACK
FC2D C9                         RET
                        ;
                        ;
                                PAGE
CP/M MACRO ASSEM 2.0    #036    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION


                        ;
                        ;
                        ; DATA AREA
                        ;
FC2E FF                 BCURDRV DB      0FFH    ; CURRENT DRIVE NUMBER
FC2F 00                 BFORMAT DB      0       ; FORMAT FOR CURRENT DRIVE (NONZERO = VT-180)
FC30 FF                 TCURDRV DB      0FFH    ;CURRENT DRIVE # WITH PRE COMP SET IF NECESSARY
FC31 FF                 BTRACK  DB      0FFH    ; CURRENT TRACK OF SELECTED DRIVE ("TRUE" TRACK)
                        ;
                        ;TTRACK DB      0FFH,0FFH,0FFH,0FFH     ; TRACK TABLE
                        ;TFORMAT        DB      0,0,0,0         ; FORMAT TABLE (NONZERO = VT180)
                        ;** NOTE: TTRACK AND TFORMAT ARE IN THE DATA BLOCK. THEY ARE DEFINED
                        ;         AT THE END OF THIS MODULE (Z80CODE.ASM), NEXT TO THE PACKET
                        ;         POINTER DEFINITIONS (Z80PKT AND I88PKT)
                        ;
                        ;
                                PAGE
CP/M MACRO ASSEM 2.0    #037    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION


                        ; *********************************************************
                        ;
                        ; NAME: ZSTART
                        ;
                        ; FUNCTION: THIS ROUTINE WILL SET UP THE STACK AND ALLOW THE Z80
                        ;           TO EXECUTE AN APPLICATION.
                        ;
                        ; ENTRY: IX = PACKET ADDRESS
                        ;
                        ; EXIT:
                        ;
                        ZSTART:
FC32 F3                         DI                              ; DISABLE INTERRUPTS WHILE
FC33 319CFC                     LXI     SP,STACK                ;   SWAPPING STACKS
```

```
  1687
  1688   FC36 210000            LXI     H,0                     ; PUSH 00 ONTO STACK FOR CP/M-80
  1689                                                          ;   TRANSIENTS THAT END WITH 'RET'
  1690                                                          ;   SO IT WILL BE SAME AS ENDING
  1691                                                          ;   WITH 'JUMP 0'
  1692
  1693   FC39 E5                PUSH    H
  1694
  1695   FC3A 3EFF              MVI     A,TRUE                  ; SET DONE FLAG
  1696   FC3C 329DFC            STA     DONEFL
  1697
  1698                         LDX     L,STADRL                ; GET START ADDRESS FROM PACKET
  1699   FC3F+DD6E02            DB      0DDH,L*8+46H,STADRL
  1700                         LDX     H,STADRH
  1701   FC42+DD6603            DB      0DDH,H*8+46H,STADRH
  1702
  1703   FC45 FB                EI                              ; ENABLE INTERRUPTS
  1704
  1705   FC46 E9                PCHL                            ; JUMP TO START ADDRESS
  1706                         PAGE
CP/M MACRO ASSEM 2.0    #038    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

  1707
  1708                  ; *****************************************************************
  1709                  ;
  1710                  ; NAME: ZMOVE
  1711                  ;
  1712                  ; FUNCTION: THIS ROUTINE WILL MOVE A BLOCK OF DATA FROM ANYWHERE IN
  1713                  ;           Z80 MEMORY TO ANYWHERE ELSE IN Z80 MEMORY.
  1714                  ;
  1715                  ; ENTRY: IX = PACKET ADDRESS
  1716                  ;
  1717                  ; EXIT: IX = PACKET ADDRESS
  1718                  ;
  1719                  ZMOVE:
  1720                         LDX     L,SCADRL                ; GET SOURCE ADDRESS INTO HL
  1721   FC47+DD6E02            DB      0DDH,L*8+46H,SCADRL
  1722                         LDX     H,SCADRH
  1723   FC4A+DD6603            DB      0DDH,H*8+46H,SCADRH
  1724
  1725                         LDX     E,DSADRL                ; GET DESTINATION ADDRESS INTO DE
  1726   FC4D+DD5E04            DB      0DDH,E*8+46H,DSADRL
  1727                         LDX     D,DSADRH
  1728   FC50+DD5605            DB      0DDH,D*8+46H,DSADRH
  1729
  1730                         LDX     C,BYCNTL                ; GET BYTE COUNT INTO BC
  1731   FC53+DD4E06            DB      0DDH,C*8+46H,BYCNTL
  1732                         LDX     B,BYCNTH
  1733   FC56+DD4607            DB      0DDH,B*8+46H,BYCNTH
  1734
  1735                         LDIR                            ; MOVE THE BLOCK OF DATA
  1736   FC59+EDB0              DB      0EDH,0B0H
  1737   FC5B C9                RET                             ;   AND RETURN
  1738                         PAGE
CP/M MACRO ASSEM 2.0    #039    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION
```

```
  1739
  1740                  ; *****************************************************************
  1741                  ; *****************************************************************
  1742
  1743   FC9C =         STACK   EQU     $+40H                   ; SERVICE ROUTINE STACK POINTER
  1744
  1745   FC9D =         DONEFL  EQU     STACK+1                 ; DONE FLAG
  1746
  1747   FFFC =         Z80PKT  EQU     INTFPTR                 ; PACKET POINTER FOR PACKET FROM Z80
  1748
  1749   FFFE =         I88PKT  EQU     INTFPTR+2               ; PACKET POINTER FOR PACKET FROM 8088
  1750
  1751                  ; FOLLOWING TWO DEFINITIONS ARE FOR THE DISK I/O ROUTINES.
  1752                  ; THE TABLES ARE LOCATED IN THE DATA BLOCK, INITIALIZED BY THE BIOS
  1753                  ; COLD BOOT ROUTINE AND MOVED BY Z80CCP WHEN A CONFIGURATION CHANGE OCCURS.
  1754   FFF0 =         TTRACK  EQU     INTFPTR-12              ; TRACK TABLE
  1755   FFF4 =         TFORMAT EQU     INTFPTR-8               ; FORMAT TABLE
  1756
  1757   FC9E                   ORG     DONEFL+1
  1758
  1759   FC9E                   END
CP/M MACRO ASSEM 2.0    #040    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

BC              0000
BCURDRV         FC2E   851  1084  1441  1451  1479  1659#
BDCS            0002    35#   72
BFORMAT         FC2F   858   949   994  1288  1489  1660#
BIOCS           0001    34#   79   263
BOOT            F736   226   246#
BTRACK          FC31   985  1003  1285  1481  1491  1641  1662#
BYCNTH          0007   440# 1732  1733
BYCNTL          0006   439# 1730  1731
CONIN           F762   229   270#
CONOUT          F76A    89   230   274#
CONST           F74D    90   228   258#
CONST1          F75A   260   261   265#
CSFLAG          FFE7    33#   71    78   262
DE              0002
DELAY7          FA2E  1087  1094#
DKCHECK         F8F2   544   814#
DKCHECK1        F901   818   819   823#
DKCHECK2        F904   821   822   827#
DKCHECK3        F927   825   826   842   843   846   847   849#
DKREAD          F93B   531   869#
DKRW20          F94C   873   879#  892   893
DKRW30          F958   883   884   888#
DKRW40          F95B   886   887   890#
DKRW90          F968   875   876   902#
DKRW95          F96D   898   899   906#
DKRWS10         F9BB   987   988   993#
DKRWS20         F9C5   998   999  1001#
DKRWSET         F9A8   871   982#
DKWRIT          0014   427#
DKWRITE         93B    534   870#
```

```
DMAHI           0005   686#  934   935   937   938  1593  1594
DMALOW          0004   685# 1591  1592
DONEFL          FC9D   472   591   627   648  1696  1745# 1757
DPCOMD          FBDD  1241  1253  1312  1394  1561#
DPCOMD1         FBDF  1563# 1567  1568
DPRDADR         FA2F  1108# 1361
DPRDADRX        FA32  1111#
DPREAD          F9CB   885  1020#
DPREADX         F9CE  1023#
DPREADZ         F9D1   841  1025#
DPRECAL         FAA1   820   989  1176  1234# 1294  1344
DPRECAL1        FAA7  1239# 1242  1243
DPRECALX        FAB2  1235  1244  1251#
DPRECALX1       FAC2  1257  1258  1263#
DPRWE10         FA55  1142  1143  1150# 1171  1184  1204  1205
DPRWE15         FA57  1152# 1162  1163
DPRWE20         FA69  1145  1146  1170#
DPRWE80         FA9B  1177  1178  1180  1181  1197  1198  1200  1201  1209#
DPRWE90         FAA0  1137  1157  1217#
DPRWEX          FA47  1021  1051  1109  1134#
DPRWSET         FBE5  1031  1063  1115  1581# 1584  1585
DPRWSET4        FC10  1609  1610  1612#
DPSEEK          FAC7   874  1284#
DPSEEK1         FACC  1179  1199  1287#
DPSEEKE         FAF4  1306  1307  1319# 1358  1359
CP/M MACRO ASSEM 2.0    #041    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

DPSEEKN         FB27  1360  1388# 1432  1476  1496
DPSEEKV         FAF8  1290  1338#
DPSEEKVX        FB09  1339  1340  1346  1347  1353#
DPSEEKX         FADF  1192  1291  1296  1302#
DPSEKH0         FB23  1383# 1474  1529
DPSEKN3         FB29  1385  1386  1390#
DPSEL1          FB41  1416  1417  1419#
DPSEL2          FB47  1423# 1429  1430
DPSEL3          FB5E  1426  1427  1440#
DPSEL3A         FB7E  1457  1458  1472#
DPSEL3C         FB88  1437  1438  1448  1449  1464  1465  1478#
DPSEL4          FB9C  1443  1444  1490#
DPSEL5          FBA1  1494#
DPSEL6          FBA9  1503#
DPSELDR         FB32   817   983  1409#
DPSELS          FBB8  1508  1509  1512# 1523  1524
DPSELS1         FBC6  1516  1517  1525#
DPSELS2         FBC7  1435  1436  1527#
DPWRITE         F9E7   889  1050#
DPWRT3          FA17  1075  1076  1078#
DPWRTX          F9EA  1053#
DRIVEN          0002   682#  832   833  1624  1625
DRVNUM          0060   689#  834  1626
DSADRH          0005   438# 1727  1728
DSADRL          0004   437# 1725  1726
DSKFNC          0010   414#
ERROR           F890   549#
FALSE           0000     3#    4     6   471   626
```

```
FDOSM            0090    32#   66
FNCCOD           0000   527   528   680#  881   882  1445  1446
FNCNG            00FF   425#  551   552
FRANGE           00F0   411#
GETDRV           FC13  1410  1623# 1643
GETRAK           FBD2  1480  1543#
GONOW            F80E    65    96   100   352   371#
GSODRV           FA19  1060  1071  1083#
GTABLE           F6BA    61   177#
HIFNC            F8B4   537   538   588#
HIFXIT           F8BC   518   595#
HL               0004
HOME             F78A   234   290#
HSAVE            F80A   135   144   358   366   369#
I88INT           0000   401#  508   566   629
I88PKT           FFFE   504   506  1749#
I88SVC           F8C7    99   251   621#
INII             A2ED   790# 1024
ININ             63DB   792#  840  1112
INOROUT          0046   787# 1026  1055  1113
INTBIT           0004   406#  570   633
INTCLR           F8D2   631#  634
INTFPTR          FFFC   391#  394# 1747  1749  1754  1755
INTSTA           0020   402#  569   632
INTSTR           F843   458#
IOBCHK           F7E0    75   259   338#
IOBYTE           0003    31#  152   166   338   343   347
IPKT             F673   111#  161   163   600   602
CP/M MACRO ASSEM 2.0    #042    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

IX               0004
IY               0004
JUMP             00C3   404#
LEGFUN           0007   422#
LIST             F772   231   278#
LISTST           F7C4   241   319#
NEXTSC10         F998   947   948   953#
NEXTSC15         F9A3   945   946   962#
NEXTSC20         F9A4   951   952   963#
NEXTSEC          F973   891   923#
NSECT            0006   687#  927   928
NUMSEC           000A   429#
OTHFNC           0020   415#
OUTII            A3ED   791# 1054
PACKET           F675    95    97    98   112#  146   148   149   151   153   248
                        249
PACKIT           F67F    60   133#  246   254   266   270   274   278   282   286
                        290   294   299   303   307   311   315   319   334
PCONS            F634    48    49    71#
PCONS1           F645    73    74    80#  264
PDRC             F649    53    54    84#
PDRC1            F652    86    87    90#
PGIOB            F7E8    56   343#
PKTPRO           F855   462   499#
PKTR1            F8A1   568#  571
```

```
PKTRET          F895   522   554#
PRIVATE         0000     6#   11   389
PSBDl           F61E    50    51    59#   76    77    91    92
PSBDOS          F600    38#  222
PSBDOSl         F606    39    43#
PSEUD           F658    67    95#
PSEUDl          F66F   104#
PSEUN           F7Fl   267   271   275   279   283   287   291   295   300   304
                       308   312   316   320   335   350#
PSEUX           F62D    67#  355
PSIOB           F7EC    58   346#
PSTACK          F841   141   376#
PSTAKSAV        F80C   138   140   361   363   370#
PUNCH           F77A   232   282#
QCRDADR         00C0   721# 1114
QCREADS         0080   718# 1030
QCREST          0008   712# 1252
QCSEEK          001C   713# 1311
QCSEEKN         0018   714# 1389
QCSEKH0         0010   715# 1384
QCSTEPIN        0048   716# 1240
QCSTEPOT        0068   717#
QCTERM          00D0   722# 1154  1587
QCWRTRK         00F0   720#
QCWRTS          00A0   719# 1062
QKCMCOM         0015   543   695#
QKDRCNT         9C40   764# 1502
QKDRETRY        0004   763# 1151
QKMXSECT        000A   768#  836   944
QKMXTRK         004F   767# 1190  1191  1305  1357
QKPCBIT         0040   766# 1418
CP/M MACRO ASSEM 2.0    #043    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

QKPCTRK         003D   765# 1414
QKRATE          0000   708#  712   713   714   715   716   717
QKRDCOM         0013   530   693#  880
QKROBIN         0002   769#  848
QKWTCOM         0014   533   694# 1447
QMBUSY          0001   733#  745   746   747   748   749   896   903  1212  1320
                      1425  1428  1583
QMCRC           0008   730#  746   747   748   749  1077  1144
QMDELDM         0020   741#  747
QMDRNR          0003   759#
QMDRQ           0002   740#
QMEFLG          0004   701# 1611
QMHLD           0008   699#  712   713   714   716   717
QMHLT           0020   728# 1607
QMINDEX         0002   732# 1506  1519
QMLDATA         0004   739#  747   748   749
QMNRDY          0080   726#  745   746   747   748   749  1089  1425  1434  1530
                      1531
QMON            0018   757#
QMON0           0008   756#  757
QMON1           0010   755#  757
QMPRECMP        00C0   753#
```

```
QMPSIDE          0020   754#
QMRDADR          009D   749# 1119
QMREAD           00BD   747# 1035
QMREST           0091   745# 1265
QMRNF            0010   738#  747   748   749   844  1144
QMSCOM           0002   703#
QMSEEK           0099   746# 1316
QMSKERR          0010   729#  745   746
QMSSEL           0008   702#
QMTG42           0004   758#
QMTZERO          0004   731# 1256
QMUPDT           0010   704#
QMVERF           0004   700#  713
QMWPROT          0040   727#  748
QMWRFLT          0020   737#  748
QMWRITE          00FD   748# 1067
QPCOMD           0060   778# 1034  1066  1088  1118  1155  1254  1315  1424  1433
                       1504  1505  1518  1562  1582  1588  1606  1613
QPDATA           0063   781#  792  1308  1390  1590
QPSECRG          0062   780# 1365  1598
QPSTAT           0040   777# 1086  1092  1422  1564
QPTRKRG          0061   779#  877  1029  1058  1286  1431  1473  1475  1495  1528
RDORWR           0040   786# 1032  1064  1116
READ             F7B4   239   311#
READER           F782   233   286#
REPSTAK          F67F   134#
RST3             0018   397#
RST4             0020   398#  477
RST6             0030   399#  463
SCADRH           0003   436# 1722  1723
SCADRL           0002   435# 1720  1721
SECNUM           001F   690#  943   956   964   965  1597
SECSIZ           0200   430#
SECTN            0002   683#  838   839   939   940   954   955   958   959  1595
CP/M MACRO ASSEM 2.0    #044    Z80 PSEUDO, INTERFACE AND PRIMITIVE ROUTINES - SHARED  VERSION

                       1596
SECTRAN          F7CC   242   323#
SELDSK           F792   235   294#
SETDMA           F7AC   238   307#
SETRAK           FC1C   816   831   878   908   992  1640#
SETSEC           F7A4   237   303#
SETTRK           F79A   236   298#
SHARE            FFFF    11#   16   609
STACK            FC9C   467  1686  1743# 1745
STADRH           0003   433# 1700  1701
STADRL           0002   432# 1698  1699
START            F843    40   390#  393#  444
STATUS           0001   551   552   681#  859   860   894   895   904   905  1530
                       1531
TCURDRV          FC30  1091  1421  1661#
TFORMAT          FFF4   854  1485  1755#
TRACKN           0003   684#  829   830   996   997  1027  1028  1056  1057  1186
                       1187  1190  1191  1194  1195  1303  1304  1354  1355  1366
                       1367  1412  1413
```

Rainbow™ 100
CP/M–86/80
BIOS Listings
AA–P310A–TV

READER'S COMMENTS

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

_____

_____

_____

_____

_____

Did you find errors in this manual? If so, specify the error and the page number.

_____

_____

_____

_____

_____

Please indicate the type of reader that you most nearly represent.

☐ First-time computer user
☐ Experienced computer user
☐ Application package user
☐ Programmer
☐ Other (please specify)_____

Name_____

Date_____

Organization_____

Street_____

City_____

State_____

Zip Code
or Country_____

Do Not Tear – Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE PUBLICATIONS
200 FOREST STREET MRO1–2/L12
MARLBOROUGH, MA 01752

Do Not Tear – Fold Here and Tape