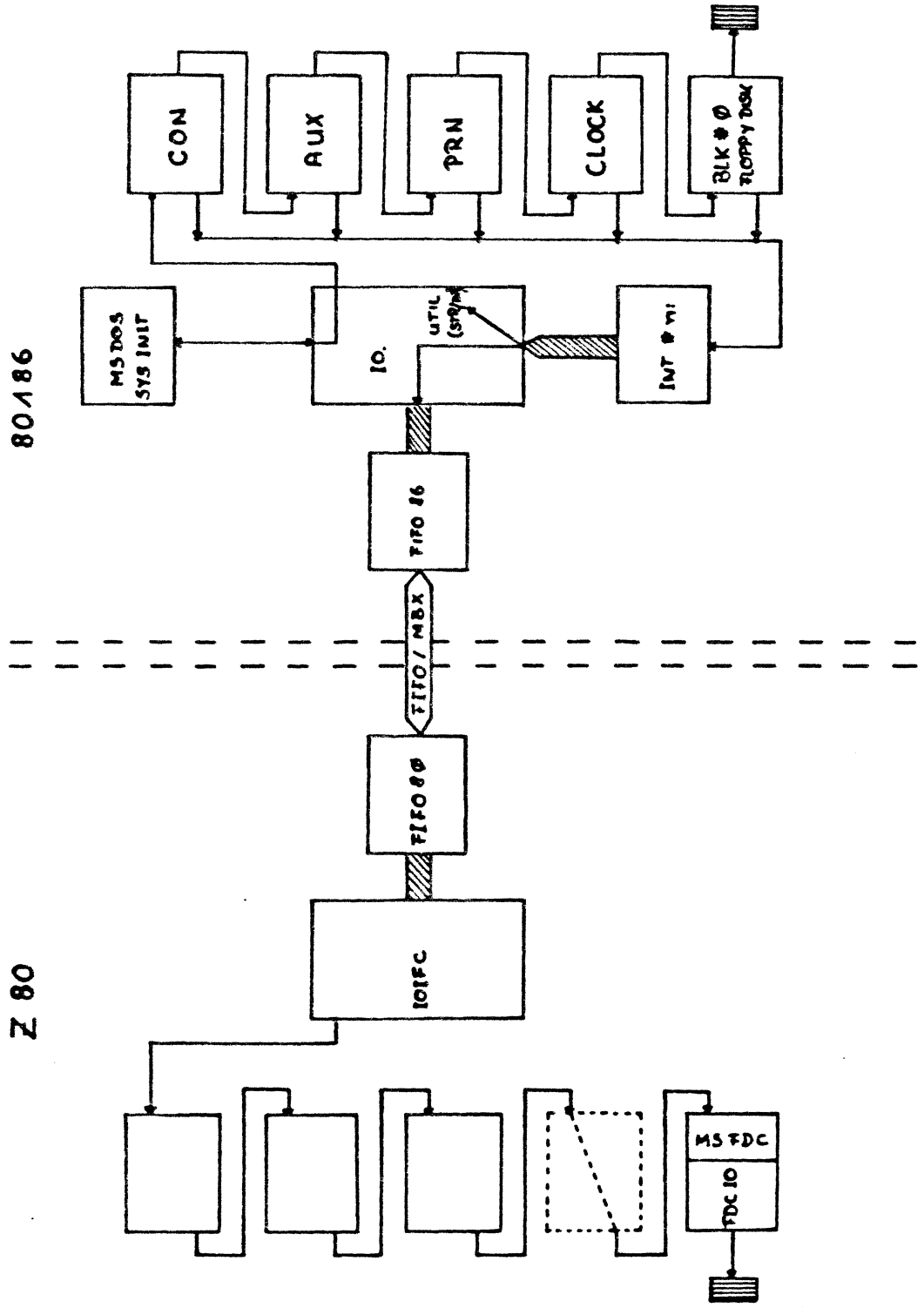
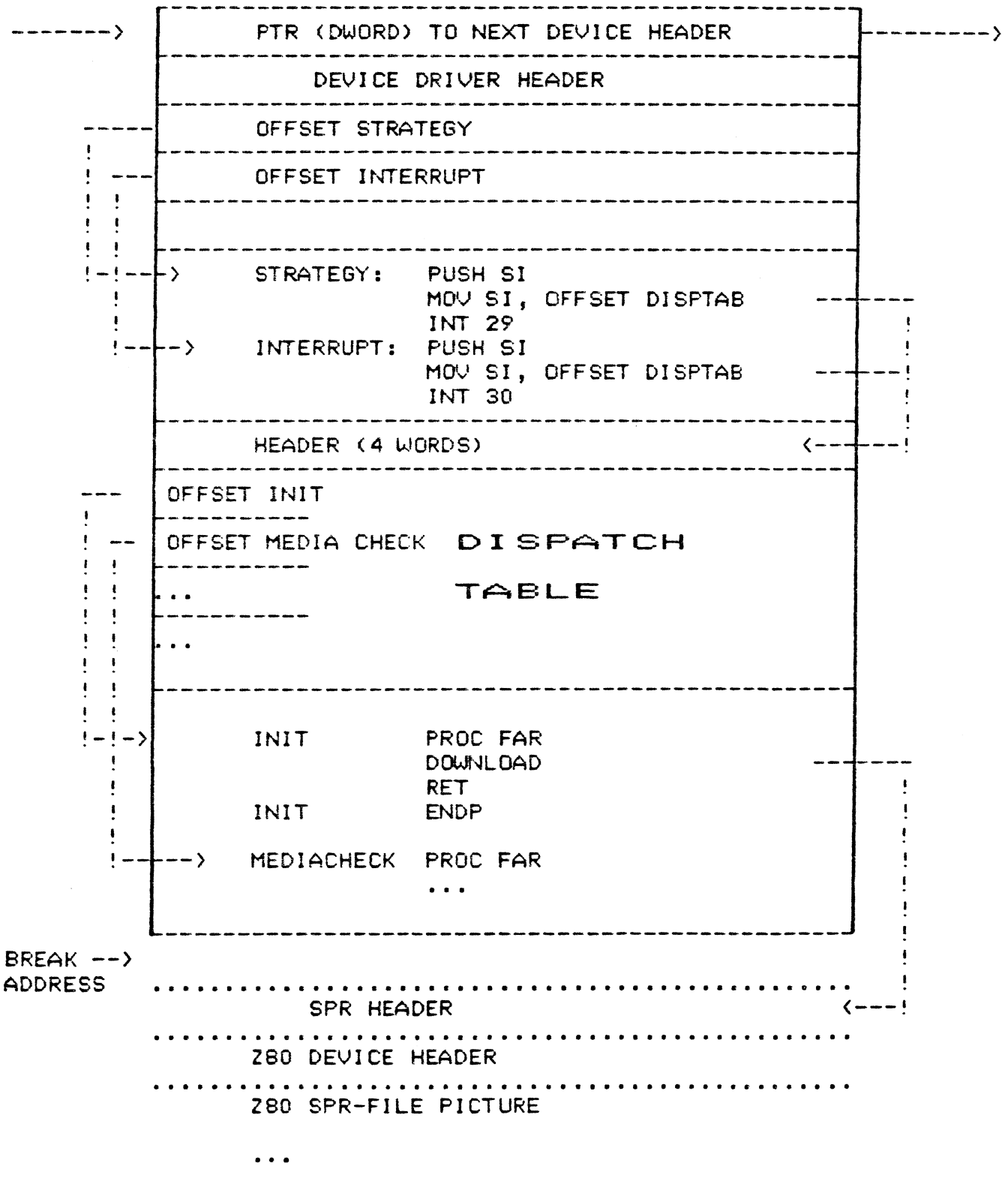
BIOS PARAMETER BLOCK (BPB)

BPB:	512	BYTES / SECTOR
	2	SECTOR / BLOCK
	1	RESERVED SECTOR (BOOT)
	2	FAT - STRUCTURES
	128	DIRECTORY - ENTRIES
	1600	SECTORS / DISK
	0F9H	MEDIA ID / FAT ID
	3	SECTORS / FAT
	20	SECTORS / CYLINDERS
		NO LOGICAL SKEW


CON
AUX
PRN
CLOCK
BLK # 0
FLOPPY DISK
MS DOS
SYS INIT
IO.
UTIL
INT # n
80186
FIFO 16
FIFO/MBX
FIFO 00
IOIFC
Z 80
MS FDC
FDC IO

DRIVER CONSTRUCTION

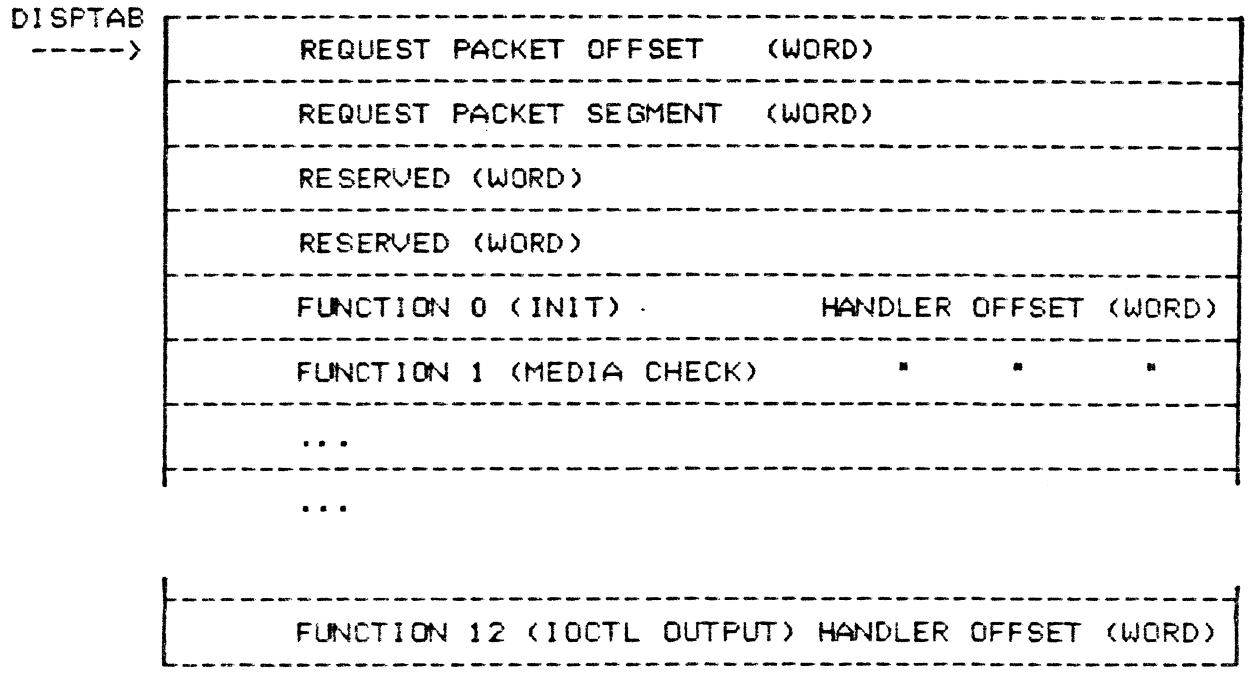


DEVICE DRIVER HEADER FORMAT

DWORD Pointer to next Device (Set to -1,-1 on installable devices)
WORD Attributes Bit 15 = 1 if char device or 0 if block device if bit 15 is 1: Bit 0 = 1 if Current sti device Bit 1 = 1 if Current sto device Bit 2 = 1 if Current NUL device Bit 3 = 1 if Current CLOCK dev. Bit 4 = 1 if Special device bit 14 is the IOCTL bit bit 13 is NON IBM Format Bit
WORD Pointer to Device STRATEGY Entry Point
WORD Pointer to Device INTERRUPT Entry Point
8 BYTE Character Device Name Field Left Justified Space Filled Upper Case Characters For Block Devices, first byte is Number of Units Defined

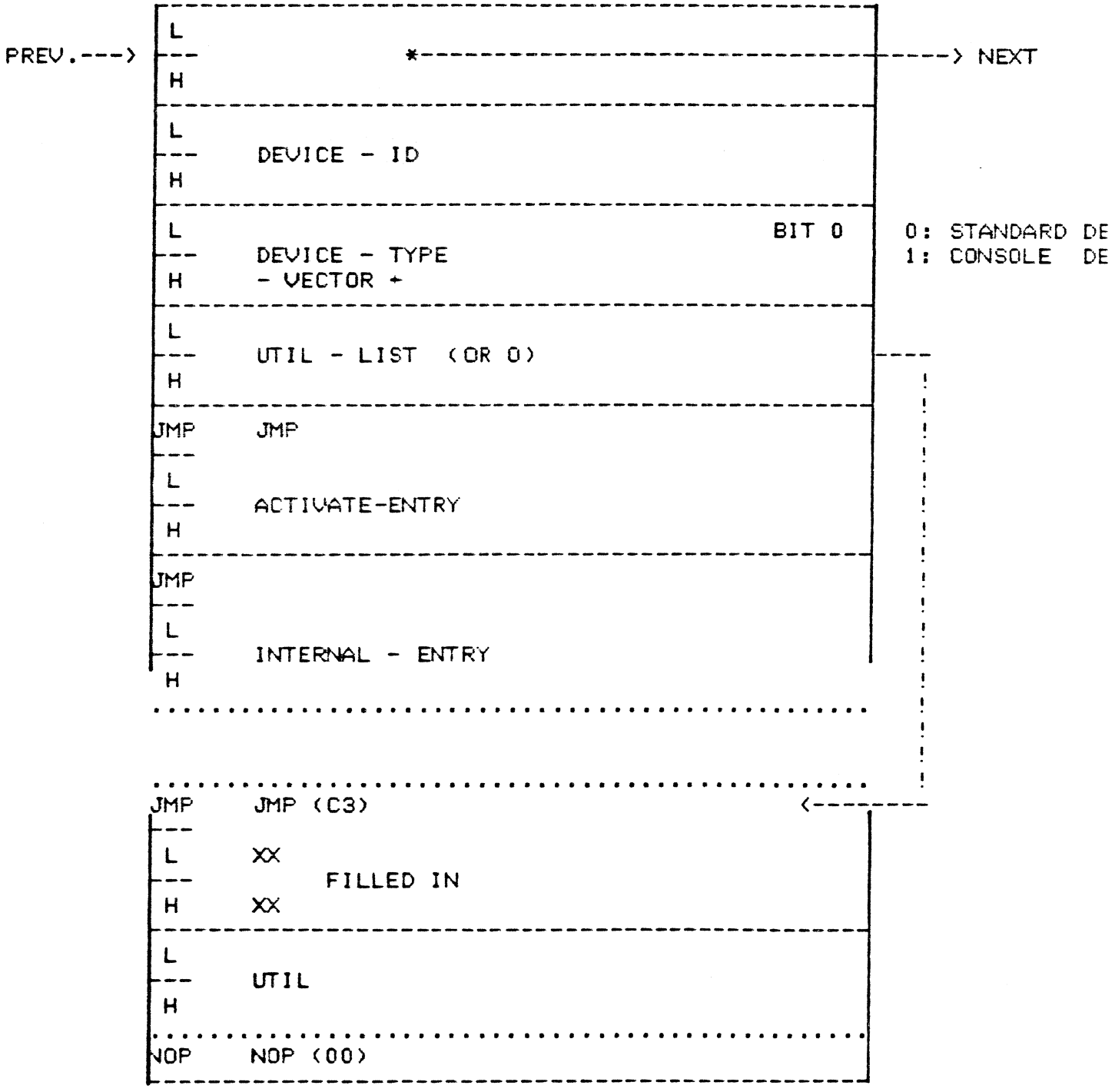
Note that the device entry points are words. They must be offsets from the same segment number used to point to this table. I.e. if XXX:YYY points to the start of this table, then XXX:STRATEGY and XXX:INTERRUPT are the entry points.

DISPATCH TABLE DEFINITION



0 INIT
1 MEDIA CHECK (Block only, NOP for character)
2 BUILD BPB - " -
3 IOCTL INPUT (Only called if device has IOCTL)
4 INPUT (Read)
5 NON-DESTRUCTIVE INPUT NO WAIT (Char devices only)
6 INPUT STATUS - " -
7 INPUT FLUSH - " -
8 OUTPUT (write)
9 OUTPUT (write) with verify
10 OUTPUT STATUS - " -
11 OUTPUT FLUSH - " -
12 IOCTL OUTPUT (Only called if device has IOCTL)

Z80 DEVICE HEADER



CALL OF DIPATCH TABLE FUNCTIONS

FAR CALL with

Parameters:

CL	UNIT NUMBER
ES:DI	POINTER TO REQUEST HEADER TAIL (REQ. HEADER ADDR. + 13)
DS	DEVICE HANDLER CS

RESULT

AX	STATUS WORD

MS-DOS KERNEL

INT 29 STRATEGY CALL:

```
call:   PUSH SI
        MOV SI, OFFSET DISPTAB
        INT 29
```

INT 30 INTERRUPT CALL:

```
call:   PUSH SI
        MOV SI, OFFSET DISPTAB
        INT 30
```

INT 31

```
call:   ON CALL DL

        DL= Function No.

        0: DOWNLOAD
           ES:SI = Pointer to SPR-File
           ~~CX = New device ID~~
           Result
           AL    = ERRCODE

        1: ACTIVATE
           CX    = device ID
           AL    = ERRCODE

        2:-7:  MESSAGE I/O

        8: RESOLVE A REQUEST
           DS    = Data segment (=CS I.A.)
```

```
================================================================================
  MS-DOS IO-KERNEL (DEFINITION OF THE SYSTEM FUNCTIONS)
================================================================================

  System interrupts definitions

          STRATEGY            INT 29
          INTERRUPT           INT 30
          SYSFUNC             INT 31

  Dipatch table constants

          DTAB                BUF    8  Dispatch table buffer size

  System function numbers definitions

          DOWNLOAD            SF     0  Z80 SPR-File download function

                              ON CALL:  SI = POINTER (OFFSET) TO THE SPR-
                                            HEADER
                                        ES = POINTER (SEGMENT) TO THE SPR-
                                            HEADER
                              RETURNS:  AL = ERROR CODE, 000H = NO ERROR
          --------------------------------------------------------------------
          ACTIVATE            SF     1  Z80 Device handler activation

                              ON CALL:  CX = DEVICE IDENTIFIER
                              RETURNS:  AL = ERROR CODE, 000H = NO ERROR
          --------------------------------------------------------------------
          BYTEMSGSEND         SF     2  Byte message send function

                              ON CALL:  AL = BYTE MESSAGE
          --------------------------------------------------------------------
          BYTEMSGRECV         SF     3  Byte message recive function

                              RETURNS:  AL = BYTE MESSAGE
          --------------------------------------------------------------------
          WORDMSGSEND         SF     4  Word message send function

                              ON CALL:  CX = WORD MESSAGE
          --------------------------------------------------------------------
          WORDMSGRECV         SF     5  Word message recive function

                              RETURNS:  BX = WORD MESSAGE
          --------------------------------------------------------------------
          STRMSGSEND          SF     6  String message send function

                              ON CALL:  CX = STRING MESSAGE LENGTH
                                        SI = POINTER (OFFSET) TO STRING
                                            MESSAGE
                                        ES = POINTER (SEGMENT) TO STRING
                                            MESSAGE
```

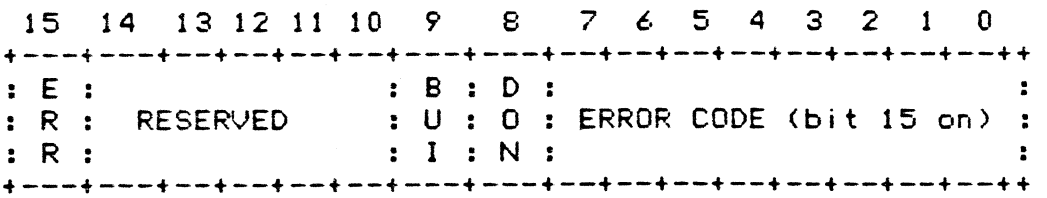
```
STRMSGRECV          SF      7   String message recive function

                    ON CALL:    CX = STRING MESSAGE LENGTH
                                DI = POINTER (OFFSET) TO STRING
                                     MESSAGE
                                ES = POINTER (SEGMENT) TO STRING
                                     MESSAGE

-------------------------------------------------------------------
RESOLVEREQ          SF      8   Resolve transfer request function

                    ON CALL:    CL = UNIT NUMBER
                                SI = DATEA ITEM LENGTH
                                DI = REQUEST PACKET PTR. OFF. (+13)
                                ES = REQUEST PACKET PTR. SEG.
                                BX = ACCES HANDLER PTR. OFF.
                                DS = ACCES HANDLER PTR. SEG.
                    RETURNS:    AX = MS-DOS ERROR STATUS

MULTIBYTE/          e.g.    :   READ ONE SECTOR FROM DISK:
MULTISECTOR
TRANSFERS                       ON CALL: BX = BLOCK NUMBER
                                         CL = DRIVE NUMBER
                                         DX = SECTOR LENGTH
                                         DI = SECTOR PTR. SEGMENT
                                         ES = SECTOR PTR. OFFSET
                                RETURNS: AX = MODIFIED ERROR CODE *

                                READONE  PROC FAR
                                           .
                                           .
                                           .
                                         RET
                                READONE  ENP
```

*) MODIFIED ERROR CODE: SAME AS STANDARD ERROR CODE BUT
WITHOUT BUSY/DONE - BITS
(ALWAYS ZERO)

REQUEST PACKET HEADER

BYTE length of record Length in bytes of this Request Structure
BYTE unit code The subunit the operation is for (drive) no meaning on character devices
BYTE command ode
WORD Status
8 bytes reserved here for two DWORD links. One will be a link for the DOS queue the other for the device queue.

STATUS WORD



The status word ist zero on entry and is always set by the driver interrupt routine on return.

Bit 8 is the done bit, it means the operation is complete. For the moment the Driver just sets it to one when it exits, in the future this will be set by an asynchronous interrupt routine to tell the DOS the operation is complete.

EXTENDED SCREEN FUNCTIONS

ANSI ESCAPE SEQUENCES

1Ä ≙ Escape

This section explains how the ANSI escape sequences are defined for MS-DOS. An ANSI escape sequence is a series of charecters (beginning with an escape character or keystroke) that you can use to define functions to DOS. Specifically, you can use ANSI escape sequences from within your program to control the movement of the cursor on the screen. You can also reassign key functions and change graphics functions. Examples on how to use ANSI escape sequences are included at the end of this section.

Notes:

1. The default value (1) is used when no esplicit value or a value of zero is specified.

2. n represents "numeric parameter". This is a decimal number specified with ASCII digits.

3. Use hex 1B to represent ESC.

4. Use hex 5B to represent Ä ([)

CURSOR FUCTIONS

The following escape sequences affect the cursor position on the screen.

ANSI Stand. 111

CUP - Cursor Position

oben links 1,1

ESC [n;nH

↳ Decimal

HVP - Horizontal & Vertical Position

ESC [n;nf

CUP and HVP move the cursor to the screen positio indicated by the n parameters. The first parameter specifies the line number, and the second parameter specifies the column number.

EXTENDED SCREEN AND KEYBOARD FUNCTIONS

The default value is 1. When no parameters are specified, the cursor is moved to the home position.

CUU - Cursor Up

ESC [nA

This sequence moves the cursor up one line without changing columns. The value of n determines the number of lines moved. The default value for n is one. The CUU sequence is ignored if the cursor is already on the top line.

CUD - Cursor Down

ESC [nB

This sequence moves the cursor down one line without changing columns. The value of n dertemines the number of lines moved. The default value for n is 1. The CUD sequence is ignored if the cursor is already on the bottom line.

CUF - Cursor Forward

ESC [nC

The CUF sequence moves the cursor forward one column without changing lines. The value of n determines the number of columns moved. The default value for n is 1. The CUF sequence is ignored if the cursor is already in the far right column.

CUB - Cursor Bachward

ESC [nD

This escape sequence moves the cursor bach one column without changing lines. The value of n determines the number of columns moved. The default value for n is 1. The CUB sequence is ignored if the cursor is already in the far left column.

DSR - Device Status Report *

ESC [6n

The console driver will output a CPR sequence (see below) on receipt of the DSR escape sequence.

CPR - Cursor Position Report (from console driver to system) *

ESC [n;nR

The CPR sequence reports current cursor position via standard input. The first parameter specifies the current line and the second parameter specifies the current column.

SCP - Save Cursor Position

ESC [s

The current cursor position is saved. This cursor position can be restored with the RCP sequence (see below).

RCP - Restore Cursor Position

ESC [u

This sequence restores the cursor position to the value it had when the console driver receives the SCP sequence.

* not yet available

ERASING

The following escape sequences affect erase functions.

ED - Erase Display

ESC [2J

The ED sequence erases the screen, and the cursor goes to the home position.

EL - Erase Line

ESC [K

This sequence erases from the cursor to the end of the line (including the cursor position)

MODES OF OPERATION

The following escape sequences affect screen graphics.

SGR - Set Graphics Rendition

ESC [n;...;nm

The SGR escape sequence invokes the graphic functions specified by the parameter(s) describes below. The graphic functions remain until the next occurrence of an SGR escape sequence.

Patameter	Parameter Function
0	All Attributes Off
7	Reverse Video On

SM - Set Mode

```
     ESC [=nh
or   ESC [=7h
or   ESC [?7h
```

The SM escape sequence changes the screen type to one ot the following parameter:

Parameter	Function
7	wraparound at end of line

ADDITION SM5

```
     ESC [>5h   Cursor Invisible
```

RM - Reset Mode

```
     ESC [=nl
or   ESC [=7l
or   ESC [?7l
```

Parameter 7 will reset the wrap at the end of line mode (characters that extend past the end of the line are lost).

ADDITION RM5

```
     ESC [>5l   Flashing Cursor
```

CURSOR MODE CM

```
     ESC [3;nz
```

Parameter	Function
0	Flashing Rectangle
1	Standing Rectangle
2	Flashing Rectangle
3	Standing Rectangle
4	Invisible Cursor

SCROLL MODE

ESC [nw ≙ ↑ÄÄØw

Parameter	Function
1	Soft Scroll On
0	Soft Scroll Off