

IBM PC/XT CREF86 V1.0

INPUT FILES: INITIAL.OBJ BOOT.OBJ RS232.OBJ KEYBOARD.OBJ KEYINT.OBJ PRINTER.OBJ FLOPPY.OBJ GRAPHICS.OBJ
 VIDEO.OBJ EQUIPMNT.OBJ USEFUL.OBJ INTS.OBJ TIME.OBJ PRINTSCN.OBJ BIOSDATA.OBJ STACK.OBJ

OUTPUT FILE: BIOS.CRF

CONTROLS SPECIFIED: PAGELNGTH(80) PAGEWIDTH(132) PRINT(BIOS.CRF) TITLE((C) Display Telecommunications Corp., 1983)

MODULES INCLUDED:

FILE NAME	MODULE NAME(S)
INITIAL.OBJ:	INITIAL
BOOT.OBJ:	BOOT
RS232.OBJ:	RS232
KEYBOARD.OBJ:	KEYBOARD
KEYINT.OBJ:	KEYINT
PRINTER.OBJ:	PRINTER
FLOPPY.OBJ:	FLOPPY
GRAPHICS.OBJ:	GRAPHICS
VIDEO.OBJ:	VIDEO
EQUIPMNT.OBJ:	EQUIPMNT
USEFUL.OBJ:	USEFUL
INTS.OBJ:	INTS
TIME.OBJ:	TIME
PRINTSCN.OBJ:	PRINTSCREEN
BIOSDATA.OBJ:	PUBLICS
STACK.OBJ:	STACK

SYMBOL NAME	SYMBOL TYPE	DEFINING MODULE; REFERRING MODULE(S)
ACTIVECARD.	UNKNOWN	PUBLICS; INITIAL VIDEO
ACTIVEPAGE.	UNKNOWN	PUBLICS; VIDEO
ALTINPUT.	UNKNOWN	PUBLICS; KEYINT
BASICTRAPADDR	UNKNOWN	INTS; INITIAL BOOT
BEEP.	UNKNOWN	USEFUL; INITIAL VIDEO INTS
BIOSBREAK	UNKNOWN	PUBLICS; KEYINT
BOOTDRIVER.	UNKNOWN	BOOT; INTS
BUFFEREND	UNKNOWN	PUBLICS; INITIAL KEYBOARD KEYINT
BUFFERSTART	UNKNOWN	PUBLICS; INITIAL KEYBOARD KEYINT
BYTEOUT	UNKNOWN	USEFUL; INITIAL
CASSETTEDRIVER.	UNKNOWN	EQUIPMNT; INTS
CHAROUT	UNKNOWN	USEFUL; INITIAL INTS
CLEARSCREEN	UNKNOWN	USEFUL; INITIAL BOOT INTS
COMMSDRIVER	UNKNOWN	RS232; INTS
CRTCOLUMNS.	UNKNOWN	PUBLICS; INITIAL VIDEO
CRTCRLF	UNKNOWN	USEFUL; INITIAL
CRTLENGTH	UNKNOWN	PUBLICS; VIDEO
CRTMODE	UNKNOWN	PUBLICS; INITIAL KEYINT VIDEO
CRTMODESET.	UNKNOWN	PUBLICS; INITIAL VIDEO
CRTPALETTE.	UNKNOWN	PUBLICS; VIDEO
CRTSTART.	UNKNOWN	PUBLICS; VIDEO
CURSORMODE.	UNKNOWN	PUBLICS; VIDEO
CURSORPOSN.	UNKNOWN	PUBLICS; VIDEO
DISKETTESTAT.	UNKNOWN	PUBLICS; FLOPPY
EQUIPDRIVER	UNKNOWN	EQUIPMNT; INTS
EQUIPFLAG	UNKNOWN	PUBLICS; INITIAL VIDEO EQUIPMNT USEFUL
FDCSTATUS	UNKNOWN	PUBLICS; FLOPPY
FIXEDDISK1.	UNKNOWN	PUBLICS; PUBLICS
FIXEDDISK2.	UNKNOWN	UNKNOWN
FLOPPYDRIVER.	UNKNOWN	FLOPPY; INTS
FLOPPYHDWRINT	UNKNOWN	FLOPPY; INTS
FLOPPYPARAMSPINTER	UNKNOWN	FLOPPY; BOOT INTS
FLOPPYPARAMSTRAPADDR.	UNKNOWN	INTS; BOOT FLOPPY
ILLEGALINT.	UNKNOWN	INTS; INITIAL
INTRFLAG.	UNKNOWN	PUBLICS; INTS
IOROMINIT	UNKNOWN	PUBLICS; INITIAL
IROMSEGMENT.	UNKNOWN	PUBLICS; INITIAL
KEYBOARDDRIVER.	UNKNOWN	KEYBOARD; INTS
KEYBOARDFLAG1	UNKNOWN	PUBLICS; KEYBOARD KEYINT
KEYBOARDFLAG2	UNKNOWN	PUBLICS; KEYINT
KEYBOARDHDWRINT	UNKNOWN	KEYINT; INTS
KEYBUFEND	UNKNOWN	PUBLICS
KEYBUFFER	UNKNOWN	PUBLICS; INITIAL KEYINT
KEYBUFHEAD.	UNKNOWN	PUBLICS; INITIAL KEYBOARD KEYINT
KEYBUFTAIL.	UNKNOWN	PUBLICS; INITIAL KEYBOARD KEYINT
KEYIN	UNKNOWN	USEFUL; INITIAL BOOT INTS
MEMORYSIZE.	UNKNOWN	PUBLICS; INITIAL EQUIPMNT INTS
MEMORYSIZEDRIVER.	UNKNOWN	EQUIPMNT; INTS
MEMORYTEST.	UNKNOWN	USEFUL; INITIAL
MF6ERRFLAG.	UNKNOWN	PUBLICS; INITIAL
MF6TEST	UNKNOWN	PUBLICS
MOTORCOUNT.	UNKNOWN	PUBLICS; FLOPPY TIME
MOTORSTATUS	UNKNOWN	PUBLICS; FLOPPY TIME
NIBBLEOUT	UNKNOWN	USEFUL; INTS
NMIINT.	UNKNOWN	INTS; INITIAL USEFUL
NMITRAPADDR	UNKNOWN	INTS; INITIAL
POENTRY.	UNKNOWN	INITIAL; INTS
POSITIONCURSOR.	UNKNOWN	USEFUL; INITIAL INTS
PRINTERBASE	UNKNOWN	PUBLICS; INITIAL PRINTER
PRINTERDRIVER	UNKNOWN	PRINTER; INTS
PRINTMESSAGE.	UNKNOWN	USEFUL; INITIAL BOOT INTS
PRINTSCNSTATUS.	UNKNOWN	PUBLICS; PRINTSCREEN

PRINTSCREENINT.	UNKNOWN	PRINTSCREEN;	INITIAL	
PRINTSCREENTRAPADDR . .	UNKNOWN	INTS;	INITIAL	
PRINTTIMEOUT.	UNKNOWN	PUBLICS;	INITIAL	PRINTER
RAMVECTORS.	UNKNOWN	INTS;	INITIAL	
RESEENTRY.	UNKNOWN	INITIAL;	KEYINT	INTS
RESETFLAG	UNKNOWN	PUBLICS;	INITIAL	KEYINT
ROMCHECK8K.	UNKNOWN	USEFUL;	INITIAL	
ROMCHECKCX.	UNKNOWN	USEFUL;	INITIAL	
ROMVECTORS.	UNKNOWN	INTS;	INITIAL	
RS232BASE	UNKNOWN	PUBLICS;	INITIAL	RS232
RS232TIMEOUT.	UNKNOWN	PUBLICS;	INITIAL	RS232
SEEKSTATUS.	UNKNOWN	PUBLICS;	FLOPPY	
STACKTOP.	UNKNOWN	STACK;	INITIAL	
TIMERHDWRINT.	UNKNOWN	TIME;	INTS	
TIMERHIGH	UNKNOWN	PUBLICS;	TIME	
TIMERLOW.	UNKNOWN	PUBLICS;	TIME	
TIMEROVERFLOW	UNKNOWN	PUBLICS;	TIME	
TODDRIVER	UNKNOWN	TIME;	INTS	
VIDEODRIVER	UNKNOWN	VIDEO;	INTS	
VIDEGRAPHICSPINTER. .	UNKNOWN	GRAPHICS;	VIDEO	
VIDEGRAPHICSTRAPADDR .	UNKNOWN	INTS;	INITIAL	VIDEO
VIDEOINIT	UNKNOWN	USEFUL;	INITIAL	INTS
VIDEOTRAPADDR	UNKNOWN	INTS;	INITIAL	
VIDPARAMSPINTER. . . .	UNKNOWN	VIDEO;	INTS	
VIDPARAMSTRAPADDR . . .	UNKNOWN	INTS;	VIDEO	
WORDOUT	UNKNOWN	USEFUL;	INITIAL	INTS

IBM PC/XT 0086/0087/0088 MACRO ASSEMBLER V1.1 ASSEMBLY OF MODULE IBMINC
 OBJECT MODULE PLACED IN IBMINC.OBJ
 ASSEMBLER INVOKED BY: ASMB6 IBMINC.SRC

```

LOC OBJ          LINE    SOURCE
                1 +1  $Title ('DTC/PC BIOS IBMINC Include File Dummy Assembly V1.0')
                2 +1  $Pagelength (80) Pagewidth (132) Debug Nogen
                3      Name Ibminc
                4
                5
                6      ;   Author:      Don K. Harrison
                7
                8      ;   Start date:  December 28, 1983      Last edit:  December 28, 1983
                9
               10
               11      ;
               12      ;           *****
               13      ;           * Module Description *
               14      ;           *****
               15      ;           This module is a dummy assembly of the include file used
               16      ;           by all modules.
               17
               18
               19
               20
               21
               22      ;           (c) Display Telecommunications Corporation, 1983
               23      ;           All Rights Reserved
               24
               25      $Eject
    
```

LDC	OBJ	LINE	SOURCE
		26	
		27	
		28	;
		29	;
		30	*****
		31	;
		32	
		33	
		34	
		35	
		36	
		37	
		38	\$Eject

LOC	OBJ	LINE	SOURCE
		39	; IBMInc.tmp file same as IBMInc except \$Nolist command commented out
		40	\$Include (Ibminc.tmp)
=1		41	; *****
=1		42	; * Global Include File *
=1		43	; *****
=1		44	;\$Nolist
=1		45	
=1		46	
=1		47	; *****
=1		48	; * Macros *
=1		49	; *****
=1		50	
=1		51	;%Define (Bus(Value))
=1			(Mov AL,%Value
=1			Out 0C0H,AL)
=1		52	
=1		53	
=1		54	\$Eject

```

LOC OBJ          LINE    SOURCE
=1 55
=1 56      ;          *****
=1 57      ;          * Equates *
=1 58      ;          *****
=1 59
=1 60      ;          *****
=1 61      ;          * System Board Peripherals *
=1 62      ;          *****
=1 63
=1 64      ;          *****
=1 65      ;          * DMA Controller *
=1 66      ;          *****
=1 67
0008      =1 68  PortDMACCommand Equ  8      ;Command register
0008      =1 69  PortDMAMode     Equ  0BH    ;Mode register
0009      =1 70  PortDMARequest Equ  09H    ;Software request register
000A      =1 71  PortDMAMaskSngl Equ  0AH    ;Single channel mask
000F      =1 72  PortDMAMaskGlb1 Equ  0FH    ;Global channel mask
0008      =1 73  PortDMAStatus Equ  08H    ;Status register
0000      =1 74  PortDMACH0Base Equ  0      ;Channel 0 base register
0001      =1 75  PortDMACH0Count Equ  1      ;Channel 0 word count register
0002      =1 76  PortDMACH1Base Equ  2      ;Channel 1 base register
0003      =1 77  PortDMACH1Count Equ  3      ;Channel 1 word count register
0004      =1 78  PortDMACH2Base Equ  4      ;Channel 2 base register
0005      =1 79  PortDMACH2Count Equ  5      ;Channel 2 word count register
0006      =1 80  PortDMACH3Base Equ  6      ;Channel 3 base register
0007      =1 81  PortDMACH3Count Equ  7      ;Channel 3 word count register
000C      =1 82  PortDMAToggle Equ  0CH    ;Toggle byte pointer flip flop
000D      =1 83  PortDMAReset  Equ  0DH    ;Master clear
000E      =1 84  PortDMAClrMask Equ  0EH    ;Global mask clear
0083      =1 85  PortPageChan1 Equ  83H    ;Channel 1 upper page reg. address
0081      =1 86  PortPageChan2 Equ  81H    ;Channel 2 upper page reg. address
0082      =1 87  PortPageChan3 Equ  82H    ;Channel 3 upper page reg. address
=1 88
=1 89
=1 90      ;          *****
=1 91      ;          * Interrupt Controller *
=1 92      ;          *****
=1 93
0020      =1 94  PortPICICW1   Equ  20H    ;Initialization command word 1
0021      =1 95  PortPICICW2   Equ  21H    ;Initialization command word 2
0021      =1 96  PortPICICW3   Equ  21H    ;Initialization command word 3
0021      =1 97  PortPICICW4   Equ  21H    ;Initialization command word 4
0021      =1 98  PortPICOCW1   Equ  21H    ;Operation command word 1
0020      =1 99  PortPICOCW2   Equ  20H    ;Operation command word 2
0020      =1 100 PortPICOCW3   Equ  20H    ;Operation command word 3
00A0      =1 101 PortNMIMask  Equ  0A0H   ;Non maskable interrupt mask
0000      =1 102 NMIMask      Equ  0      ;Mask command
0000      =1 103 NMIUnmask  Equ  80H    ;Unmask command
0020      =1 104 PicEoi       Equ  20H    ;End of interrupt
=1 105
=1 106      ;          *****
=1 107      ;          * Counter Timer *
=1 108      ;          *****
=1 109
0040      =1 110 PortCTCLoadCh0 Equ  40H    ;Load channel 0
0041      =1 111 PortCTCLoadCh1 Equ  41H    ;Load channel 1
0042      =1 112 PortCTCLoadCh2 Equ  42H    ;Load channel 2
0043      =1 113 PortCTCMode   Equ  43H    ;Write mode word
0040      =1 114 PortCTCReadCh0 Equ  40H    ;Read channel 0
0041      =1 115 PortCTCReadCh1 Equ  41H    ;Read channel 1
0042      =1 116 PortCTCReadCh2 Equ  42H    ;Read channel 2
0012      =1 117 CTCRefreshDiv Equ  18      ;Real time clock divisor constant
=1 118      ;          *Eject

```

```

LOC OBJ          LINE    SOURCE
=====
=1 119           ;          *****
=1 120           ;          * Programmable Peripheral Interface *
=1 121           ;          *****
=1 122           ;
=1 123           ;          *****
=1 124           ;          * Port A *
=1 125           ;          *****
0060 126 PortPPIPortA Equ 60H           ;Parallel Port A
0060 127 PortKBYDScan Equ 60H           ;Port A is keyboard scan code
=1 128           ;
=1 129           ;          *****
=1 130           ;          * Port B *
=1 131           ;          *****
0061 132 PortPPIPortB Equ 61H           ;Parallel Port B
=1 133           ;
=1 134           ;          *****
=1 135           ;          * Control and Flag Bits (B Port) *
=1 136           ;          *****
0001 137 PPIBSpkrGate Equ 0000001B      ;Speaker gate bit
0002 138 PPIBSpkrData Equ 0000010B      ;Speaker data bit
0004 139 PPIBSpare Equ 0000100B      ;Unused bit position
0008 140 PPIBConfigS1ct Equ 00001000B     ;Select lower (0) or upper (1) switches
0010 141 PPIBDisSysParit Equ 00010000B    ;System board parity disable bit
0020 142 PPIBDisIOParity Equ 00100000B    ;I/O channel parity disable bit
0040 143 PPIBRelKbydClk Equ 01000000B    ;Keyboard clock line release bit
0080 144 PPIBClearKbyd Equ 10000000B    ;Keyboard clear
00A5 145 PPIBStartUpMode Equ 10100101B    ;PPIBSpkrGate +
=1 146           ;...PPIBSpare +
=1 147           ;...PPIBDisIOParity +
=1 148           ;...PPIBClearKbyd
=1 149           ;          *****
=1 150           ;          * Port C *
=1 151           ;          *****
0062 152 PortPPIPortC Equ 62H           ;Parallel Port C
0062 153 PortConfigSw Equ 62H           ;Port C (lower) are config. switches
=1 154           ;
=1 155           ;          *****
=1 156           ;          * Control and Flag Bits (C Port) *
=1 157           ;          *****
0001 158 PPICLoopPost Equ 0000001B      ;Loop on post bit
0002 159 PPICNPIInstalled Equ 0000010B     ;Numeric processor installed
000C 160 PPICRAM Equ 00001100B      ;Ram size bits
0003 161 PPICDisplay Equ 0000011B      ;Display type bits
000C 162 PPICDiskettes Equ 00001100B    ;Floppy disk type bits
0010 163 PPICSpk Equ 00010000B      ;Speaker feed back bit
0020 164 PPICCTCh2Out Equ 00100000B     ;Output of CTC channel 2
0040 165 PPICIOChk Equ 01000000B     ;Status of I/O check flip flop
0080 166 PPICRAMChk Equ 10000000B     ;Status of system board parity f/f
=1 167           ;
=1 168           ;          *****
=1 169           ;          * Control Port *
=1 170           ;          *****
0063 171 PortPPIMode Equ 63H           ;Write mode word
0099 172 PPIBStdMode Equ 99H           ;Standard mode word = 99H
=1 173           ;
=1 174 $Eject

```



```

LOC OBJ          LINE    SOURCE
                =1 175      ;
                =1 176      ;
                =1 177      ;
                =1 178      ;
                =1 179      ;
                =1 180      ;
                =1 181      ;
                =1 182      ;
0201             =1 183      PortGCAButtons Equ 201H      ;Game paddle button status
0201             =1 184      PortGCADoneShots Equ 201H      ;Game paddle one shot outputs
0201             =1 185      PortGCAFire Equ 201H      ;Fire all one shots
                =1 186      ;
                =1 187      ;
                =1 188      ;
                =1 189      ;
                =1 190      ;
0210             =1 191      PortEXPBusTest Equ 210H      ;Write/read bus test port
0211             =1 192      PortEXPHiAdrTst Equ 211H      ;Read high order address latch
0211             =1 193      PortEXPWaitTest Equ 211H      ;Clear wait test latch
0212             =1 194      PortEXPLOAdrTst Equ 212H      ;Read high order address latch
0213             =1 195      PortEXPEnable Equ 213H      ;1=enable expansion unit, 0=disable
0213             =1 196      PortEXPStatus Equ 213H      ;Read status of expansion unit
                =1 197      ;
                =1 198      ;
                =1 199      ;
                =1 200      ;
                =1 201      ;
0001             =1 202      EXPEnStatus Equ 00000001B      ;1=enabled, 0=disabled
0002             =1 203      EXPWaitRqFlag Equ 00000010B      ;State of wait state request flag
00F0             =1 204      EXPSwitches Equ 11110000B      ;Dip switch state
                =1 205      ;
                =1 206      $Eject
    
```

```

LOC OBJ          LINE    SOURCE
                =1 207      ; *****
                =1 208      ; * Asynchronous Communications Controller *
                =1 209      ; *****
                =1 210      ;
                =1 211      ; *****
                =1 212      ; * Primary Adapter *
                =1 213      ; *****
                =1 214      ;
03F8            =1 215      PortSio1Rxdata Equ 3F8H      ;Receive data DLAB=0
03F8            =1 216      PortSio1Txdata Equ 3F8H      ;Transmit data DLAB=0
03F8            =1 217      PortSio1BaudLo Equ 3F8H      ;Receive data DLAB=1
03F9            =1 218      PortSio1BaudHi Equ 3F9H      ;Receive data DLAB=1
03F9            =1 219      PortSio1IntReg Equ 3F9H      ;Interrupt enable register
03FA            =1 220      PortSio1IntID  Equ 3FAH      ;Interrupt ID register
03FB            =1 221      PortSio1LCR   Equ 3FBH      ;Line control register
03FC            =1 222      PortSio1MCR   Equ 3FCH      ;Modem control register
03FD            =1 223      PortSio1LSR   Equ 3FDH      ;Line status register
03FE            =1 224      PortSio1MSE   Equ 3FEH      ;Modem status register
                =1 225      ;
                =1 226      ; *****
                =1 227      ; * Secondary Adapter *
                =1 228      ; *****
                =1 229      ;
02F8            =1 230      PortSio2Rxdata Equ 2F8H      ;Receive data DLAB=0
02F8            =1 231      PortSio2Txdata Equ 2F8H      ;Transmit data DLAB=0
02F8            =1 232      PortSio2BaudLo Equ 2F8H      ;Receive data DLAB=1
02F9            =1 233      PortSio2BaudHi Equ 2F9H      ;Receive data DLAB=1
02F9            =1 234      PortSio2IntReg Equ 2F9H      ;Interrupt enable register
02FA            =1 235      PortSio2IntID  Equ 2FAH      ;Interrupt ID register
02FB            =1 236      PortSio2LCR   Equ 2FBH      ;Line control register
02FC            =1 237      PortSio2MCR   Equ 2FCH      ;Modem control register
02FD            =1 238      PortSio2LSR   Equ 2FDH      ;Line status register
02FE            =1 239      PortSio2MSE   Equ 2FEH      ;Modem status register
                =1 240      $Eject
    
```

```

LOC OBJ          LINE    SOURCE
=====
=1 241          ;          *****
=1 242          ;          * Control and Flag Bits *
=1 243          ;          *****
=1 244          ;
0000            =1 245      Sio5Bits      Equ      0000000B      ;5 data bits
0001            =1 246      Sio6Bits      Equ      0000001B      ;6 data bits
0002            =1 247      Sio7Bits      Equ      0000010B      ;7 data bits
0003            =1 248      Sio8Bits      Equ      0000011B      ;8 data bits
0004            =1 249      Sio2StopBits  Equ      0000100B      ;Set 2 stop bits
0008            =1 250      SioEnableParity Equ      00001000B     ;Enable parity
0010            =1 251      SioEvenParity Equ      00010000B     ;Set even parity if parity enabled
0020            =1 252      SioStickParity Equ      00100000B     ;Parity bit follows bit 4 of this byte
0040            =1 253      SioBreakBit   Equ      0100000B     ;1 sets Tx output marking
0080            =1 254      SioAccessBrgDiv Equ      10000000B     ;Set hi to access BRG divisor latch
0001            =1 255      SioRxReady    Equ      00000001B     ;Receiver ready
0002            =1 256      SioOverrun    Equ      00000010B     ;Receive overrun
0004            =1 257      SioParityError Equ      00000100B     ;Receive parity error
0008            =1 258      SioFramingErr Equ      00001000B     ;Receive parity error
0010            =1 259      SioBreakDetect Equ      00010000B     ;Receive line is in break condition
0020            =1 260      SioTxReady    Equ      00100000B     ;Transmitter holding register empty
0020            =1 261      SioTxEmpty    Equ      00100000B     ;All data bits have cleared transmitter
0001            =1 262      SioNoIntPending Equ      00000001B     ;No interrupt condition exists
0006            =1 263      SioRxLineInt  Equ      00000110B     ;Receiver caused interrupt
0004            =1 264      SioRxDataInt  Equ      00000100B     ;Reception of data char. caused int.
0002            =1 265      SioTxReadyInt Equ      00000010B     ;Transmitter register went ready
0000            =1 266      SioModemInt   Equ      00000000B     ;Modem status change caused interrupt
0001            =1 267      SioEnabRxInt  Equ      00000001B     ;Enable data available interrupt
0002            =1 268      SioEnabTxInt  Equ      00000010B     ;Enable Tx register empty interrupt
0004            =1 269      SioEnabStatInt Equ      00000100B     ;Enable status change interrupt
0008            =1 270      SioEnabModemInt Equ      00001000B     ;Enable modem change interrupt
0001            =1 271      SioEnabDTR    Equ      00000001B     ;Turn on DTR
0002            =1 272      SioEnabRTS    Equ      00000010B     ;Turn on RTS
0004            =1 273      SioEnabOUT1   Equ      00000100B     ;Turn on OUT1
0008            =1 274      SioEnabOUT2   Equ      00001000B     ;Turn on OUT2
0010            =1 275      SioEnabLoop   Equ      00010000B     ;Turn on Loop
0001            =1 276      SioDeltaCTS   Equ      00000001B     ;Set if CTS input changed state
0002            =1 277      SioDeltaDSR   Equ      00000010B     ;Set if DSR input changed state
0004            =1 278      SioDeltaRI    Equ      00000100B     ;Set if RI input changed state
0008            =1 279      SioDeltaRX    Equ      00001000B     ;Set if RX input changed state
0010            =1 280      SioCTS        Equ      00010000B     ;State of CTS input
0020            =1 281      SioDSR        Equ      00100000B     ;State of DSR input
0040            =1 282      SioRI         Equ      01000000B     ;State of RI input
0080            =1 283      SioRX         Equ      10000000B     ;State of RX input
=1 284          ;
=1 285          ;          *****
=1 286          ;          * Baud Rate Divisor Values *
=1 287          ;          *****
=1 288          ;
0900            =1 289      Sio50Baud     Equ      2304       ;Divisor value for 50 baud
0600            =1 290      Sio75Baud     Equ      1536       ;Divisor value for 75 baud
0417            =1 291      Sio110Baud    Equ      1047       ;Divisor value for 110 baud
0359            =1 292      Sio134Baud    Equ      857        ;Divisor value for 134 baud
0300            =1 293      Sio150Baud    Equ      768        ;Divisor value for 150 baud
0180            =1 294      Sio300Baud    Equ      384        ;Divisor value for 300 baud
00C0            =1 295      Sio600Baud    Equ      192        ;Divisor value for 600 baud
0060            =1 296      Sio1200Baud   Equ      96         ;Divisor value for 1200 baud
0040            =1 297      Sio1800Baud   Equ      64         ;Divisor value for 1800 baud
003A            =1 298      Sio2000Baud   Equ      58         ;Divisor value for 2000 baud
0030            =1 299      Sio2400Baud   Equ      48         ;Divisor value for 2400 baud
0020            =1 300      Sio3600Baud   Equ      32         ;Divisor value for 3600 baud
0018            =1 301      Sio4800Baud   Equ      24         ;Divisor value for 4800 baud
0010            =1 302      Sio7200Baud   Equ      16         ;Divisor value for 7200 baud
000C            =1 303      Sio9600Baud   Equ      12         ;Divisor value for 9600 baud
=1 304          $Eject
    
```

```

LOC OBJ          LINE    SOURCE
                =1 305      ;
                =1 306      ;
                =1 307      ;
                =1 308      ;
0300             =1 309      PortProtoBase Equ 300H      ;Base address of prototype card
                =1 310      ;
                =1 311      ;
                =1 312      ;
                =1 313      ;
                =1 314      ;
0320             =1 315      PortHDCData   Equ 320H      ;Read and write data port
0321             =1 316      PortHDCStatus Equ 321H      ;Read controller status
0321             =1 317      PortHDCReset  Equ 321H      ;Reset the controller (write)
0322             =1 318      PortHDCSelect Equ 322H      ;Generate controller-select pulse
0323             =1 319      PortHDCDMAInt Equ 323H      ;Write pattern to DMA and Int register
                =1 320      ;
                =1 321      ;
                =1 322      ;
                =1 323      ;
                =1 324      ;
0378             =1 325      PortPrint2Data Equ 378H      ;Printer data
0379             =1 326      PortPrint2Stat Equ 379H      ;Printer status
037A             =1 327      PortPrint2Cnt1 Equ 37AH      ;Printer control
                =1 328      ;
                =1 329      ;
                =1 330      ;
                =1 331      ;
                =1 332      ;
0001             =1 333      PrintStrobe   Equ 0000001B  ;Printer strobe (low active)
0002             =1 334      PrintAutoFeed Equ 0000010B  ;Auto line feed (low active)
0004             =1 335      PrintInit      Equ 0000100B  ;Initialize printer (low active)
0008             =1 336      PrintSelectIn Equ 0001000B  ;Select the printer (low active)
0010             =1 337      PrintIntEnable Equ 0010000B  ;Enable printer interrupts
0008             =1 338      PrintError     Equ 0001000B  ;Printer error status bit
0010             =1 339      PrintSelectOut Equ 0010000B  ;Printer selected bit
0020             =1 340      PrintPaperOut Equ 0010000B  ;Printer paper out status bit
0040             =1 341      PrintAck      Equ 0100000B  ;Printer ACK signal state (low active)
0080             =1 342      PrintBusy     Equ 1000000B  ;Printer busy signal state (low active)
                =1 343      %Eject

```

```

LOC OBJ          LINE    SOURCE
                =1 344      ; *****
                =1 345      ; * Monochrome Video Adapter *
                =1 346      ; *****
                =1 347
03B4            =1 348      PortMonoIndex Equ 3B4H      ;6845 Index register
03B5            =1 349      PortMonoData  Equ 3B5H      ;6845 Control registers
03B8            =1 350      PortMonoCntl1 Equ 3B8H      ;Mode control port
03BA            =1 351      PortMonoStatus Equ 3BAH      ;Sync and video states
03BC            =1 352      PortPrint1Data Equ 3BCH      ;Mono board printer data
03BD            =1 353      PortPrint1Stat Equ 3BDH      ;Mono board printer status
03BE            =1 354      PortPrint1Cntl Equ 3BEH      ;Mono board printer control
0000000000000000 =1 355      MonoScreen   Equ 0B0000H    ;Start of monochrom adapter memory
                =1 356
                =1 357      ; *****
                =1 358      ; * Control and Flag Bits *
                =1 359      ; *****
                =1 360
0001            =1 361      MonoResMode  Equ 0000001B    ;Set high resolution mode
0008            =1 362      MonoVideoEnable Equ 00001000B    ;Enable monochrome video
0020            =1 363      MonoBlinkEnable Equ 00100000B    ;Enable blinking
0001            =1 364      MonoHorizState Equ 0000001B    ;Horizontal pulse state
0008            =1 365      MonoVideoState Equ 00001000B    ;Video output state
                =1 366
                =1 367      ; *****
                =1 368      ; * Register Pointer Values *
                =1 369      ; *****
                =1 370
0000            =1 371      MonoHorizTotal Equ 0          ;Pointer to horizontal total register
0001            =1 372      MonoHorizDispl Equ 1          ;Pointer to horizontal displayed
0002            =1 373      MonoHorizSyncP Equ 2          ;Pointer to horizontal sync position
0003            =1 374      MonoHorizSyncW Equ 3          ;Pointer to horizontal sync width
0004            =1 375      MonoVertTotal Equ 4          ;Pointer to vertical total
0005            =1 376      MonoVertTotAdj Equ 5          ;Pointer to vertical total adjust
0006            =1 377      MonoVertDispl Equ 6          ;Pointer to vertical displayed
0007            =1 378      MonoVertSync Equ 7          ;Pointer to vertical sync position
0008            =1 379      MonoInterlace Equ 8          ;Pointer to interlace mode bit
0009            =1 380      MonoMaxScanLine Equ 9          ;Pointer to maximum scan line number
000A            =1 381      MonoCursorStart Equ 10         ;Pointer to cursor start scan line
000B            =1 382      MonoCursorEnd Equ 11         ;Pointer to cursor end scan line
000C            =1 383      MonoStartHigh Equ 12         ;Pointer to screen start address high
000D            =1 384      MonoStartLow Equ 13         ;Pointer to screen start address low
000E            =1 385      CursorHigh Equ 14         ;Pointer to cursor position address hi
000F            =1 386      CursorLow Equ 15         ;Pointer to cursor position address low
                =1 387      $Eject
    
```

```

LOC OBJ          LINE    SOURCE
=1 388           ;          *****
=1 389           ;          * Color Graphics Adapter *
=1 390           ;          *****
=1 391
03D8             =1 392   PortColorMode Equ 3D8H      ;Color board mode select register
03D9             =1 393   PortColorColor Equ 3D9H     ;Color select register
03DA             =1 394   PortColorStatus Equ 3DAH    ;Color status register
03DB             =1 395   PortColorLPClr Equ 3DBH     ;Clear light pen detector
03DC             =1 396   PortColorLPSet Equ 3DCH    ;Preset light pen detector
03D4             =1 397   PortColorIndex Equ 3D4H    ;6845 Index register
03D5             =1 398   PortColorData Equ 3D5H    ;6845 Control registers
=1 399
=1 400           ;          *****
=1 401           ;          * Control and Flag Bits *
=1 402           ;          *****
=1 403
002C             =1 404   Color40x25ABW Equ 2CH      ;40x25 alphanumeric black and white
0028             =1 405   Color40x25ACo Equ 28H      ;40x25 alphanumeric color
002C             =1 406   Color80x25ABW Equ 2CH      ;80x25 alphanumeric black and white
0029             =1 407   Color80x25ACo Equ 29H      ;80x25 alphanumeric color
000E             =1 408   Color320x200GBW Equ 0EH     ;320x200 graphics black and white
000A             =1 409   Color320x200GCo Equ 0AH     ;320x200 graphics color
001E             =1 410   Color640x200GBW Equ 1EH     ;640x200 graphics black and white
000F             =1 411   ColorBkgnd Equ 00001111B ;Background or boarder selection
0010             =1 412   ColorIntense Equ 00010000B ;Select intensified colors
0020             =1 413   ColorPalette Equ 00100000B ;Palette toggle bit
=1 414
=1 415   $Eject
    
```

```

LOC OBJ          LINE    SOURCE
                =1 416      ; *****
                =1 417      ; * Flexible Diskette Controller *
                =1 418      ; *****
03F5             =1 419      PortFDCData   Equ    3F5H      ;Floppy controller data register
03F4             =1 421      PortFDCStatus Equ    3F4H      ;Floppy controller status register
03F2             =1 422      PortFDCAdptMode Equ    3F2H      ;Adapter mode register
                =1 423
                =1 424      ; *****
                =1 425      ; * Control and Flag Bits *
                =1 426      ; *****
                =1 427
0004             =1 428      FDCModeDrive0 Equ    0000100B  ;Select drive 0
0005             =1 429      FDCModeDrive1 Equ    00000101B  ;Select drive 1
0006             =1 430      FDCModeDrive2 Equ    00000110B  ;Select drive 2
0007             =1 431      FDCModeDrive3 Equ    00000111B  ;Select drive 3
0000             =1 432      FDCModeReset Equ    00000000B  ;Reset FDCcontroller
000C             =1 433      FDCModeIntDwaEn Equ    00001100B  ;Enable FDC interrupts and dwa action
0014             =1 434      FDCModeMotor0 Equ    00010100B  ;Turn on drive 0
0024             =1 435      FDCModeMotor1 Equ    00100100B  ;Turn on drive 1
0044             =1 436      FDCModeMotor2 Equ    01000100B  ;Turn on drive 2
0084             =1 437      FDCModeMotor3 Equ    10000100B  ;Turn on drive 3
0020             =1 438      FDCContSK    Equ    00100000B  ;Skip deleted mark bit
0040             =1 439      FDCContMF    Equ    01000000B  ;MFM Select bit
0080             =1 440      FDCContMT    Equ    10000000B  ;Multi-track bit
0001             =1 441      FDCContUS0   Equ    00000001B  ;Unit select bit 0
0002             =1 442      FDCContUS1   Equ    00000010B  ;Unit select bit 1
0004             =1 443      FDCContHD    Equ    00000100B  ;Head select bit
00C0             =1 444      FDCStatIC    Equ    11000000B  ;Interrupt code status bits
0020             =1 445      FDCStatSE    Equ    00100000B  ;Seek end status bit
0010             =1 446      FDCStatEC    Equ    00010000B  ;Equipment check status bit
0008             =1 447      FDCStatNR    Equ    00001000B  ;Not ready status bit
0004             =1 448      FDCStatHD    Equ    00000100B  ;Head state bit
0003             =1 449      FDCStatDS    Equ    00000011B  ;Drive state bits
0080             =1 450      FDCStatEN    Equ    10000000B  ;End of cylinder status bit
0020             =1 451      FDCStatDE    Equ    00100000B  ;Data error status bit
0010             =1 452      FDCStatOR    Equ    00010000B  ;Overrun status bit
0004             =1 453      FDCStatND    Equ    00000100B  ;No data status bit
0002             =1 454      FDCStatNW    Equ    00000010B  ;Not writable status bit
0001             =1 455      FDCStatMA    Equ    00000001B  ;Missing address mark status bit
00C0             =1 456      FDCStatCM    Equ    11000000B  ;Control mark status bit
0020             =1 457      FDCStatDD    Equ    00100000B  ;Data error in data field status bit
0010             =1 458      FDCStatWC    Equ    00010000B  ;Wrong cylinder status bit
0008             =1 459      FDCStatSH    Equ    00001000B  ;Scan equal hit status bit
0004             =1 460      FDCStatSN    Equ    00000100B  ;Scan not satisfied status bit
0003             =1 461      FDCStatBC    Equ    00000011B  ;Bad cylinder status bit
0080             =1 462      FDCStatMD    Equ    10000000B  ;Missing address mark in status bit
0020             =1 463      FDCStatFT    Equ    00100000B  ;Fault status bit
0010             =1 464      FDCStatWP    Equ    00010000B  ;Write protect status bit
0004             =1 465      FDCStatRY    Equ    00000100B  ;Ready status bit
0002             =1 466      FDCStatT0    Equ    00000010B  ;Track 0 status bit
0001             =1 467      FDCStatTS    Equ    00000001B  ;Two side status bit
0002             =1 468      FDCN         Equ    2          ;Number of data bytes per sector
0008             =1 469      FDCSC        Equ    8          ;Sectors per cylinder
000F             =1 470      FDC HUT      Equ    15         ;Head unload time
000C             =1 471      FDCSRT       Equ    12         ;Step rate time
0005             =1 472      FDCBPLFormat Equ    5          ;Gap length (during format)
002A             =1 473      FDCBPLReadWrite Equ    42        ;Gap length (during read or write)
0001             =1 474      FDCHLT       Equ    1          ;Head load time
                =1 475
                =1 476
                =1 477      $Eject

```

```

LOC OBJ          LINE      SOURCE
=1 478
=1 479          ;
=1 480          ; *****
=1 481          ; * Vector Location Definitions *
=1 482          ; *****
0000 =1 483 TrapZeroDiv Equ 0 ;Divide by zero vector
0001 =1 484 TrapSngStp Equ 1 ;Single step vector
0002 =1 485 TrapNMI Equ 2 ;Non maskable interrupt vector
0003 =1 486 TrapBkpt Equ 3 ;Breakpoint interrupt vector
0004 =1 487 TrapOvf Equ 4 ;Overflow interrupt vector
0005 =1 488 TrapPrintScreen Equ 5 ;Print screen vector
0006 =1 489 TrapUndef1 Equ 6 ;Undefined vector #1
0007 =1 490 TrapUndef2 Equ 7 ;Undefined vector #2
0008 =1 491 TrapTimer Equ 8 ;Timer interrupt
0009 =1 492 TrapKeyboard Equ 9 ;Keyboard interrupt
000A =1 493 TrapIRQ2 Equ 10 ;Interrupt 2 interrupt
000B =1 494 TrapSio2 Equ 11 ;Serial channel 2 interrupt
000C =1 495 TrapSio1 Equ 12 ;Serial channel 1 interrupt
000D =1 496 TrapHDisk Equ 13 ;Hard disk interrupt
000E =1 497 TrapFDisk Equ 14 ;Floppy disk interrupt
000F =1 498 TrapPrinter Equ 15 ;Printer interrupt
0010 =1 499 TrapVideo Equ 16 ;Video driver trap vector
0011 =1 500 TrapEquip Equ 17 ;Equipment query vector
0012 =1 501 TrapMemorySize Equ 18 ;Memory size query vector
0013 =1 502 TrapFDDriver Equ 19 ;Diskette driver vector
0014 =1 503 TrapSIODriver Equ 20 ;Serial interface vector
0015 =1 504 TrapCassette Equ 21 ;Cassette (dummy routine)
0016 =1 505 TrapKeyDrive Equ 22 ;Keyboard driver vector
0017 =1 506 TrapPrintDrive Equ 23 ;Printer driver vector
0018 =1 507 TrapBasic Equ 24 ;Cassette basic vector
0019 =1 508 TrapBoot Equ 25 ;Bootstrap loader trap vector
001A =1 509 TrapTOD Equ 26 ;Time of day vector
001B =1 510 TrapKeyBreak Equ 27 ;Keyboard break address
001C =1 511 TrapTimerBreak Equ 28 ;Timer break address
001D =1 512 TrapVidParams Equ 29 ;Video parameters
001E =1 513 TrapDiskParams Equ 30 ;Disk parameters
001F =1 514 TrapVideoExt Equ 31 ;Video external pointer
=1 515
=1 516
=1 517 $Eject
    
```


LOC	OBJ	LINE	SOURCE
		=1 518	
		=1 519	;
		=1 520	;
		=1 521	;
		=1 522	*****
0000		=1 523	DiskCmdReset Equ 0
0001		=1 524	DiskCmdStatus Equ 1
0002		=1 525	DiskCmdRead Equ 2
0003		=1 526	DiskCmdWrite Equ 3
0004		=1 527	DiskCmdVerify Equ 4
0005		=1 528	DiskCmdFormat Equ 5
		=1 529	
		=1 530	*Eject

LOC	OBJ	LINE	SOURCE
		=1 531	
		=1 532	; *****
		=1 533	; * Video Driver Command Equates *
		=1 534	; *****
		=1 535	
0000		=1 536	VidCmdInit Equ 0
0001		=1 537	VidCmdCurType Equ 1
0002		=1 538	VidCmdCurPos Equ 2
0003		=1 539	VidCmdRdCurPos Equ 3
0004		=1 540	VidCmdRdLPenPos Equ 4
0005		=1 541	VidCmdSelPage Equ 5
0006		=1 542	VidCmdScrollUp Equ 6
0007		=1 543	VidCmdScrollDn Equ 7
0008		=1 544	VidCmdRdCurChAt Equ 8
0009		=1 545	VidCmdWrCurChAt Equ 9
000A		=1 546	VidCmdWrCurCh Equ 10
000B		=1 547	VidCmdSetPalet Equ 11
000C		=1 548	VidCmdWrDot Equ 12
000D		=1 549	VidCmdRdDot Equ 13
000E		=1 550	VidCmdWrTTY Equ 14
000F		=1 551	VidCmdInfo Equ 15
		=1 552	
		=1 553	\$Eject

LOC	OBJ	LINE	SOURCE
		=1 554	
		=1 555	;
		=1 556	*****
		=1 557	* Serial I/O Driver Command Equates *
		=1 558	*****
0000		=1 559	SioCmdInit Equ 0
0001		=1 560	SioCmdSend Equ 1
0002		=1 561	SioCmdReceive Equ 2
0003		=1 562	SioCmdStatus Equ 3
		=1 563	\$List
		564	
		565	End

ASSEMBLY COMPLETE, NO ERRORS FOUND

IBM PC/XT 8086/8087/8088 MACRO ASSEMBLER V1.1 ASSEMBLY OF MODULE PUBLICS
OBJECT MODULE PLACED IN BIOSDATA.OBJ
ASSEMBLER INVOKED BY: ASM86 BIOSDATA.SRC

```
LOC OBJ          LINE    SOURCE
                1 +1  $Title ('DTC/PC BIOS Public Data Definitions V1.0')
                2 +1  $Pagelength (80) Pagewidth (132) Debug Nogen
                3      Name Publics
                4
                5
                6      ;   Author:      Don K. Harrison
                7
                8      ;   Start date:  October 31, 1983      Last edit:  December 7, 1983
                9
               10
               11
               12      ;           *****
               13      ;           * Module Description *
               14      ;           *****
               15
               16      ;           This module contains variable (data) definitions for the
               17      ;           bios software
               18
               19
               20
               21
               22      ;           (c) Display Telecommunications Corporation, 1983
               23      ;           All Rights Reserved
               24
               25  $Eject
```

LOC	OBJ	LINE	SOURCE
		26	
		27	
		28	*****
		29	: * Revision History *
		30	: *****
		31	:
		32	
		33	
		34	
		35	
		36	
		37	
		38	*Eject

LOC	OBJ	LINE	SOURCE
		39	
		40	;
		41	;
		42	;
		43	*****
		44	Public RS232Base, PrinterBase, EquipFlag, MfgTest
		45	Public MemorySize, MfgErrFlag, KeyboardFlag1, KeyboardFlag2
		46	Public AltInput, KeyBufHead, KeyBufTail, KeyBuffer
		47	Public KeyBufEnd, SeekStatus, MotorStatus, MotorCount
		48	Public DisketteStat, FDCStatus, CrtMode, CrtColumns
		49	Public CrtLength, CrtStart, CursorPosn, CursorMode
		50	Public ActivePage, ActiveCard, CrtModeSet, CrtPalette
		51	Public IOROMInit, IOROMSegment, IntrFlag, TimerLow
		52	Public TimerHigh, TimerOverflow, BiosBreak, ResetFlag
		53	Public FixedDisk1, FixedDisk2, PrintTimeOut, RS232TimeOut
		54	Public BufferStart, BufferEnd, PrintScnStatus
		55	\$Eject

```

LOC OBJ          LINE    SOURCE
                    56      ;
                    57      ; *****
                    58      ; * Bios Data Segment *
                    59      ; *****
0000            60      BiosDataArea Segment Public
0000            61      RS232Base Label Word
0000 (1          62      RS2321Addr Dw 1 Dup(?) ;Active RS232 Card 1
    ????)
    )
0002 (1          63      RS2322Addr Dw 1 Dup(?) ;Active RS232 Card 2
    ????)
    )
0004 (1          64      RS2323Addr Dw 1 Dup(?) ;Active RS232 Card 3
    ????)
    )
0006 (1          65      RS2324Addr Dw 1 Dup(?) ;Active RS232 Card 4
    ????)
    )
                    66
0008            67      PrinterBase Label Word
0008 (1          68      Printer1Addr Dw 1 Dup(?) ;Active Printer Card 1
    ????)
    )
000A (1          69      Printer2Addr Dw 1 Dup(?) ;Active Printer Card 2
    ????)
    )
000C (1          70      Printer3Addr Dw 1 Dup(?) ;Active Printer Card 3
    ????)
    )
000E (1          71      Printer4Addr Dw 1 Dup(?) ;Active Printer Card 4
    ????)
    )
                    72
0010 (1          73      EquipFlag Dw 1 Dup(?) ;Configuration status
    ????)
    )
0012 (1          74      MfgTest Db 1 Dup(?) ;Not used
    ??)
    )
0013 (1          75      MemorySize Dw 1 Dup(?) ;Memory size in K bytes
    ????)
    )
0015 (1          76      MfgErrFlag Db 1 Dup(?) ;Not used
    ??)
    )
0016 (1          77      Db 1 Dup(?) ;Not used
    ??)
    )
0017 (1          78      KeyboardFlag1 Db 1 Dup(?) ;Keyboard
    ??)
    )
0018 (1          79      KeyboardFlag2 Db 1 Dup(?) ;...status
    ??)
    )
0019 (1          80      AltInput Db 1 Dup(?) ;Keypad data
    ??)
    )
001A (1          81      KeyBufHead Dw 1 Dup(?) ;Pointer to head of keyboard buffer
    ????)
    )
001C (1          82      KeyBufTail Dw 1 Dup(?) ;Pointer to tail of keyboard buffer
    ????)
    )
001E (31         83      KeyBuffer Db 31 Dup (?) ;Keyboard buffer area
    ??)
    )
003D (1          84      KeyBufEnd Db 1 Dup(?) ;...and last byte of it
    ??)
    )
003E (1          85      SeekStatus Db 1 Dup(?) ;Drives 0-3 recalibrate status
    ??)
    )
003F (1          86      MotorStatus Db 1 Dup(?) ;Drives 0-3 motor state
    ??)
    )

```

LOC	OBJ	LINE	SOURCE			
0040	(1 ??)	87	MotorCount	Db	1 Dup(?)	;Drive turn off counter
0041	(1 ??)	88	DisketteStat	Db	1 Dup(?)	;Returned status byte
0042	(7 ??)	89	FDCStatus	Db	7 Dup(?)	;765 status bytes storage
0049	(1 ??)	90	CrtMode	Db	1 Dup(?)	;Current mode
004A	(1 ????)	91	CrtColumns	Dw	1 Dup(?)	;Number of columns
004C	(1 ????)	92	CrtLength	Dw	1 Dup(?)	;Linear length of screen
004E	(1 ????)	93	CrtStart	Dw	1 Dup(?)	;Screen starting address
0050	(8 ????)	94	CursorPosn	Dw	8 Dup(?)	;Cursor position on each page
0060	(1 ????)	95	CursorMode	Dw	1 Dup(?)	;Cursor mode
0062	(1 ??)	96	ActivePage	Db	1 Dup(?)	;Which page active
0063	(1 ????)	97	ActiveCard	Dw	1 Dup(?)	;I/O address of active video
0065	(1 ??)	98	CrtModeSet	Db	1 Dup(?)	;Mode register setting
0066	(1 ??)	99	CrtPalette	Db	1 Dup(?)	;Palette setting
0067	(1 ????)	100	IOROMInit	Dw	1 Dup(?)	;Not used
0069	(1 ????)	101	IOROMSegment	Dw	1 Dup(?)	;Not used
006B	(1 ??)	102	IntrFlag	Db	1 Dup(?)	;Not used
006C	(1 ????)	103	TimerLow	Dw	1 Dup(?)	;Timer
006E	(1 ????)	104	TimerHigh	Dw	1 Dup(?)	;...storage
0070	(1 ??)	105	TimerOverflow	Db	1 Dup(?)	;Timer overflow flag
0071	(1 ??)	106	BiosBreak	Db	1 Dup(?)	;If D7=1, break key hit
0072	(1 ????)	107	ResetFlag	Dw	1 Dup(?)	;Set to 1234 after initialization
0074	(1 ????)	108	FixedDisk1	Dw	1 Dup(?)	;Used by hard disk
0076	(1 ????)	109	FixedDisk2	Dw	1 Dup(?)	;Used by hard disk
0078		110	PrintTimeOut	Label	Word	
0078	(1 ??)	112	PrintT01	Db	1 Dup(?)	;Printer 1 time out value
0079	(1	113	PrintT02	Db	1 Dup(?)	;Printer 2 time out value

LOC	OBJ	LINE	SOURCE			
	??					
007A	(1	114	PrintT03	Db	1 Dup(?)	;Printer 3 time out value
	??					
007B	(1	115	PrintT04	Db	1 Dup(?)	;Printer 4 time out value
	??					
)					
007C	(1	116	RS232TimeOut	Label	Word	
007C	(1	117	RS232T01	Db	1 Dup(?)	;Sio 1 time out value
	??					
)					
007D	(1	119	RS232T02	Db	1 Dup(?)	;Sio 2 time out value
	??					
)					
007E	(1	120	RS232T03	Db	1 Dup(?)	;Sio 3 time out value
	??					
)					
007F	(1	121	RS232T04	Db	1 Dup(?)	;Sio 4 time out value
	??					
)					
		122				
0080	(1	123	BufferStart	Dw	1 Dup(?)	;Temporary keyboard driver data area
	????					
)					
0082	(1	124	BufferEnd	Dw	1 Dup(?)	;Temporary keyboard driver data area
	????					
)					
		125				
0100		126		Org	100H	;Physical address = 500H
		127				
0100	(1	128	PrintScnStatus	Db	1 Dup(?)	;Print screen driver status byte
	??					
)					
		129				
---		130	BiosDataArea	Ends		
		131	End			

ASSEMBLY COMPLETE, NO ERRORS FOUND

IBM PC/XT 8086/8087/8088 MACRO ASSEMBLER V1.1 ASSEMBLY OF MODULE STACK
OBJECT MODULE PLACED IN STACK.OBJ
ASSEMBLER INVOKED BY: ASMB6 STACK.SRC

```
LOC OBJ          LINE    SOURCE
                1 +1  $Title ('DTC/PC BIOS Stack V1.0')
                2 +1  $Pagelength (80) Pagewidth (132) Debug Nogen
                3      Name Stack
                4
                5
                6      ;   Author:      Don K. Harrison
                7
                8      ;   Start date:  October 25, 1983      Last edit:  October 30, 1983
                9
               10
               11
               12
               13      ;           *****
               14      ;           * Module Description *
               15      ;           *****
               16
               17      ;   This module contains the stack segment.
               18
               19
               20
               21
               22      ;           (c) Display Telecommunications Corporation, 1983
               23      ;           All Rights Reserved
               24      $Eject
```

LOC	OBJ	LINE	SOURCE
		25	;
		26	;
		27	;
		28	*****
		29	* Revision History *
		30	*****
		31	
		32	
		33	
		34	
		35	\$Eject

LOC	OBJ	LINE	SOURCE
		36	
		37	
		38	;
		39	;
		40	*****
		41	;
		42	Public StackTop
		43	
		44	*****
		45	;
		46	*****
		47	;
		48	BiosStack Segment Public
0000	{128	49	Dw 128 Dup (?) ;128 Level Temporary Stack
	???)		
0100)	50	StackTop Label Word
		51	BiosStack Ends
		52	
		53	End

ASSEMBLY COMPLETE, NO ERRORS FOUND

IBM PC/XT 8086/8087/8088 MACRO ASSEMBLER V1.1 ASSEMBLY OF MODULE INITIAL
OBJECT MODULE PLACED IN INITIAL.OBJ
ASSEMBLER INVOKED BY: ASM86 INITIAL.SRC

```
LOC OBJ          LINE    SOURCE
                1 +1  $Title ('DTC/PC BIOS Initialization Module V1.0')
                2 +1  $Pagelength (80) Pagewidth (132) Debug Nogen
                3      Name Inifial
                4
                5
                6      ;   Author:      Don K. Harrison
                7
                8      ;   Start date:  October 18, 1983      Last edit:  December 26, 1983
                9
               10
               11      ;           *****
               12      ;           * Module Description *
               13      ;           *****
               14
               15      ;           The Initialization module serves to initialize the system
               16      ;           on power up, or during a keyboard or manual reset.  Initialization
               17      ;           occurs for the system board peripherals and all attached option
               18      ;           adapters.  Global memory location values are initialized by this module.
               19
               20
               21
               22      ;           (c) Display Telecommunications Corporation, 1983
               23      ;           All Rights Reserved
               24
               25      $Eject
```

LOC	OBJ	LINE	SOURCE
		26	
		27	
		28	;
		29	;
		30	*****
		31	* Revision History *
		32	*****
		33	
		34	
		35	
		36	
		37	
		38	\$Eject

```

LOC OBJ          LINE      SOURCE
                39      ;          *****
                40      ;          * Public Symbols *
                41      ;          *****
                42
                43      Public ResetEntry,POREntry
                44
                45
                46      ;          *****
                47      ;          * Equates *
                48      ;          *****
                49
0001             50      BiosCksErr   Equ    0000001B      ;Bios rom checksum error
0002             51      MemErr       Equ    0000010B      ;System memory error
0004             52      VidMemErr    Equ    0000100B      ;Video memory error
0008             53      RefreshErr   Equ    00001000B     ;Memory refresh error
0010             54      First2kErr   Equ    00010000B     ;Error in first 2k of RAM
0020             55      IORDMErr    Equ    00100000B     ;Error in expansion ch. ROM
000D             56      AsciiCarriage Equ    0DH          ;Carriage return
000A             57      AsciiLineFeed Equ    0AH          ;Line feed
                58
                59
                60      $Include (IbmInc)
=1             61      ;          *****
=1             62      ;          * Global Include File *
=1             63      ;          *****
=1             64      $Nolist
584            584      $Eject

```

```

LOC OBJ          LINE    SOURCE
                585
                586 ;          *****
                587 ;          * Data Segments *
                588 ;          *****
                589
                590
-----         591 BiosDataArea Segment Public
                592 Extn MemorySize:Word, ResetFlag:Word, EquipFlag:Word
                593 Extn KeyBuffer:Byte, KeyBufHead:Word, KeyBufTail:Word
                594 Extn BufferStart:Word, BufferEnd:Word
                595 Extn PrintTimeOut:Word, RS232TimeOut:Word
                596 Extn PrinterBase:Word, RS232Base:Word
                597 Extn IOROMInit:Word, IOROMSegment:Word
                598 Extn MfgErrFlag:Byte, PositionCursor:Near, ClearScreen:Near
                599 Extn CrtMode:Byte, CrtColumns:Word
                600 Extn CrtModeSet:Byte, ActiveCard:Word
                601
                602
-----         603 BiosDataArea Ends
                604
-----         605 BiosStack Segment Public
                606 Extn StackTop:Word
                607 BiosStack Ends
                608
-----         609 HiStack Segment
                610 HiStack Ends
                611
-----         612 IntSegment Segment Public
                613 Extn RAMVectors:DWord, ROMVectors:DWord, VideoTrapAddr:DWord
                614 Extn BasicTrapAddr:DWord, NMITrapAddr:DWord
                615 Extn VideoGraphicsTrapAddr:DWord, PrintScreenTrapAddr:DWord
-----         616 IntSegment Ends
                617
-----         618 BasicSeg Segment Common
                619 Assume Cs:BasicSeg
                620 Org 0
0000           621 Basic Label Far
0000           622 BasicSeg Ends
                623
-----         624 IOROMSeg Segment Common
                625 IOROMSeg Ends
                626
-----         627 FlasherSegment Segment
                628 DsZero Label Byte
                629 FlasherSegment Ends
                630
-----         631 MonoSeg Segment Public
                632 MonoSeg Ends
                633
-----         634 ColorSeg Segment Public
                635 ColorSeg Ends
                636
                637
                638 $Eject

```


LOC	OBJ	LINE	SOURCE
		639	;
		640	;
		641	*****
		642	*****
----		643	ReleaseDate Segment byte
		644	
		645	*****
		646	*****
		647	*****
		648	
0000	31322F32332F38	649	
	33444B48	650	Db '12/23/83DKH'
		651	
----		652	ReleaseDate Ends
		653	%Eject

```

LOC  OBJ          LINE  SOURCE
                                654      ;
                                655      ;
                                656      ;
                                657      ;
                                658      ;
                                659      ;
                                660      ;
-----
                                661      Bios      Segment Common
                                662
                                663      Extrn PrintScreenInt:Near, NMIInt:Near, IllegalInt:Near
                                664      Extrn WordOut:Near, ByteOut:Near, KeyIn:Near
                                665      Extrn PrintMessage:Near, Beep:Near, CharOut:Near
                                666      Extrn MemoryTest:Near, RomCheckCx:Near
                                667      Extrn ROMCheck8K:Near, VideoInit:Near
                                668      Extrn CrtCrLf:Near
                                669
E000      670      Drg      0E000H      ;Begin BIOS
E000      671      StartOfBios Label  Byte
E000      672      LogonMsg  Db      '      MEGA-BIOS      V1.0'
                                20202020202020
                                20202020204045
                                47412042494F53
                                2020202056312E
                                30
E01D      673      Db      AsciiCarriage,AsciiLineFeed,AsciiLineFeed
E01E      0A
E01F      0A
E020      674      Db      '(C) Display Telecommunications ',0
                                28432920446973
                                706C6179205465
                                6C65636F6D6D75
                                6E69636174696F
                                6E7320
E03F      00
E040      675      Logon2ndHalf Db      'Corporation, 1983'
                                436F72706F7261
                                74696F6E2C2031
                                393833
E051      676      Db      AsciiCarriage,AsciiLineFeed,AsciiLineFeed,0
E052      0A
E053      0A
E054      00
                                677
                                678
                                679      ;
                                680      ;
                                681      ;
                                682      ;
                                683      Assume Cs:BIOS      ;Tell ASMB6 what we think
E055      684      PossiblePtrs:
E055      BC03      685      Dw      3BCH      ;Printer on mono card
E057      7803      686      Dw      37BH      ;Parallel printer card (pri.)
E059      7802      687      Dw      27BH      ;Parallel printer card (sec.)
                                688
E05B      689      Drg      0E05BH      ;Align with PC and XT
                                690
                                691      ;
                                692      ;
                                693      ;
                                694      ;
E05B      695      POREntry  Label far
                                696
                                697      ;      Prepare for Cold Boot
                                698
                                699      Assume Ds:BIOSDataArea
E05B      B8----      R      700      Mov      Ax,BIOSDataArea      ;Point at our memory
E05E      8ED8      701      Mov      Ds,Ax      ;...segment
E060      C70600000000      E      702      Mov      ResetFlag,0      ;...and be sure cold boot
                                703      Assume Ds:Nothing
                                704
                                705      ;
                                706      ;
                                707      ;
                                708      ;
E065      709      ResetEntry Label  Far
                                710
                                711      ;      Turn off interrupts
                                712

```

```

LOC OBJ          LINE    SOURCE
E066 FA          713          Cli          ;Reset interrupts till ready
                714
                715          ;          Test 8088 Flags
                716
E067 33C0        717          Xor          Ax,Ax          ;Cy,Of,Sf=0, Z,Pf=1
E069 7248        718          Jc           Err8088      ;...jump if carry set
E06B 7046        719          Jo           Err8088      ;...jump if overflow
E06D 7844        720          Js           Err8088      ;...jump if sign
E06F 7542        721          Jnz          Err8088      ;...jump if not zero
E071 7B40        722          Jpo          Err8088      ;...jump if parity odd
E073 050100      723          Add          Ax,1         ;Z=0,Pf=0
E076 743B        724          Jz           Err8088      ;...jump if zero stuck
E078 7A39        725          Jpe          Err8088      ;...jump if parity stuck
E07A 2D0280      726          Sub          Ax,8002H     ;Z=0,Pf=1,Sf=0,Of=0
E07D 7834        727          Js           Err8088      ;...jump if sign set
E07F 40          728          Inc          Ax           ;...cause overflow
E080 7131        729          Jno          Err8088      ;...jump if not overflow
E082 D1E0        730          Shl          Ax,1         ;Test shifting
E084 732D        731          Jnc          Err8088      ;...should shift into cy
E086 752B        732          Jnz          Err8088      ;...and leave Ax clear
E088 D1E0        733          Shl          Ax,1         ;Shift zero in
E08A 7227        734          Jc           Err8088      ;...should clear carry
                735
                736          ;          Test Registers
                737
E08C B85555      738          Mov          Bx,5555H     ;Alternating pattern
E08F           739          RegTest:
E08F 8BEB        740          Mov          Bp,Bx        ;.....
E091 8BCD        741          Mov          Cx,Bp        ;.
E093 8BE1        742          Mov          Sp,Cx        ;.  Shift  .
E095 8BD4        743          Mov          Dx,Sp        ;.  thru   .
E097 8ED2        744          Mov          Ss,Dx        ;.  Registers .
E099 8CDE        745          Mov          Si,Ss        ;.
E09B 8EC6        746          Mov          Es,Si        ;.
E09D 8CC7        747          Mov          Di,Es        ;.
E09F 8EDF        748          Mov          Ds,Di        ;.
E0A1 8CD8        749          Mov          Ax,Ds        ;.....
E0A3 3D5555      750          Cmp          Ax,5555H     ;Like we started?
E0A6 7506        751          Jne          Not5555      ;...jump if no
E0A8 F7D0        752          Not          Ax           ;Invert it
E0AA 8BD8        753          Mov          Bx,Ax        ;...and move to starting reg
E0AC EBE1        754          Jump         RegTest      ;...and do again
E0AE           755          Not5555:
E0AE 3DAAAA      756          Cmp          Ax,0AAAAH    ;Second pass result?
E0B1 7401        757          Je           Ok8088      ;...jump if ok
E0B3           758          Err8088:
E0B3 F4          759          Hlt          ;Common error halt
E0B4           760          Ok8088:
                761
                762          ;          Begin the Initialization
                763
E0B4 FC          764          Cld          ;Clear direction flag
E0B5 B000        765          Mov          Al,NMIMask   ;Mask NMI
E0B7 E6A0        766          Out         PortNMIMask,Al ;...interrupts till ready
                767
                768          ;          Stabilize Color and Monochrome Adapters
                769
E0B9 BAD803      770          Mov          Dx,PortColorMode ;Preliminarily
E0BC B000        771          Mov          Al,0         ;...disable
E0BE EE          772          Out         Dx,Al        ;...color adapter
E0BF BAB803      773          Mov          Dx,PortMonoCntl1 ;...and disable
E0C2 FEC0        774          Inc          Al           ;...monochrome adapter
E0C4 EE          775          Out         Dx,Al        ;...but enable Hi Res
                776
                777          ;          Setup Miscellaneous on Board I/O
                778
E0C5 B099        779          Mov          Al,PPIBStdMode ;Mode PPI
E0C7 E663        780          Out         PortPPIMode,Al ;...for A,C=in, B=out
                781
                782          ;Speaker gate = open
                783          ;...Speaker data = off
                784          ;...Spare = hi
E0C9 B0A5        785          Mov          Al,PPIBStartUpMode ;...Config. sw = sw0 - sw3
E0CB E661        786          Out         PortPPIPortB,Al ;...Sys parity = enabled
                787          ;...I/O Check = disabled

```

```

LOC OBJ                LINE    SOURCE
                                788                                ;...Keyboard clk = released
                                789                                ;...Keyboard clr = released
                                790
                                791                                ; Turn on Memory Refresh and Init DMAC
                                792
E0CD B054              793      Mov     Al,01010100B      ;Setup timer 1
E0CF E643              794      Out    PortCTCMode,Al    ;...for LSB only, Mode 2, Bin
E0D1 B012              795      Mov     Al,CTCRefreshDiv ;Load timer 1
E0D3 E641              796      Out    PortCTCLoadCh1,Al ;...with refresh divisor
E0D5 B000              797      Mov     Al,0             ;Set DMA channel registers
E0D7 E683              798      Out    PortPageChan1,Al ;...to base page
E0D9 E681              799      Out    PortPageChan2,Al ;...do all three
E0DB E682              800      Out    PortPageChan3,Al ;...channels just in case
E0DD E60D              801      Out    PortDMAReset,Al   ;Reset DMA controller
E0DF B058              802      Mov     Al,01011000B     ;Mode DMA
E0E1 E60B              803      Out    PortDMAMode,Al    ;..Channel 0 = Read, auto-init
E0E3 B041              804      Mov     Al,01000001B     ;Mode DMA
E0E5 E60B              805      Out    PortDMAMode,Al    ;...Channel 1 = Verify
E0E7 B042              806      Mov     Al,01000010B     ;Mode DMA
E0E9 E60B              807      Out    PortDMAMode,Al    ;...Channel 2 = Verify
E0EB B043              808      Mov     Al,01000011B     ;Mode DMA
E0ED E60B              809      Out    PortDMAMode,Al    ;...Channel 3 = Verify
E0EF B0FF              810      Mov     Al,0FFH         ;Load max count in
E0F1 E601              811      Out    PortDMACH0Count,Al ;...channel 0 for
E0F3 E601              812      Out    PortDMACH0Count,Al ;...refresh of memories
E0F5 B000              813      Mov     Al,00000000B     ;Enable DMAC,
E0F7 E60B              814      Out    PortDMACommand,Al ;...and unmask channel 0
E0F9 E60A              815      Out    PortDMAMaskSng1,Al ;...turning on refresh
                                816
                                817                                ;
                                818                                ; Test if refresh working
E0FB B9000A            819      Mov     Cx,0A00H         ;Un-noticeable wait time
E0FE                820      RefTstLoop:
E0FE E2FE              821      Loop   RefTstLoop      ;Loop for a short while
E100 E401              822      In     Al,PortDMACH0Count ;Get count low byte
E102 BAE0              823      Mov     Ah,Al           ;Save it
E104 E401              824      In     Al,PortDMACH0Count ;Get count high byte
E106 B6C4              825      Xchg   Al,Ah           ;...put them right
E108 F7D8              826      Neg    Ax              ;How far have we counted?
E10A BD0000            827      Mov     Bp,0            ;Clear error flag
E10D 3D7002            828      Cmp    Ax,270H         ;Lowest allowed = 270H
E110 7205              829      Jb     DmaErr          ;...jump if below
E112 3D9002            830      Cmp    Ax,290H         ;Highest allowed = 290H
E115 7603              831      Jbe    RefOk           ;Jump if ok and continue
E117                832      DmaErr:
E117 BD0800            833      Mov     Bp,RefreshErr   ;Error = Refresh Error
                                834
                                835                                ;
                                836                                ; Initialize Real Time Clock (Channel 0 of 8253)
E11A                836      RefOk:
E11A B036              837      Mov     Al,00110110B     ;Mode = 2 byte, Sq. Wave
E11C E643              838      Out    PortCTCMode,Al    ;...binary counting
E11E B000              839      Mov     Al,0             ;Load counter with 0 for
E120 E640              840      Out    PortCTCLoadCh0,Al ;...maximum count and
E122 E640              841      Out    PortCTCLoadCh0,Al ;...slowest clock available
                                842
                                843                                ;
                                844                                ; Enable expansion box
E124 BA1302            845      Mov     Dx,PortEXPEEnable ;Enable any
E127 B001              846      Mov     Al,1            ;...expansion
E129 EE                847      Out    Dx,Al           ;...chassis
                                848
                                849                                ;
                                850                                ; Retrieve warm start flag
                                851
E12A B8-----        R    852      Assume Ds:BiosDataArea, Es:Nothing
E12D B8D8              853      Mov     Ax,BiosDataArea ;Point at our data area
E12F B8360000          E    854      Mov     Si,ResetFlag     ;...and get
                                855                                ;
                                856                                ;
                                857                                ; Clear Memory (initializing parity bits)
E133 33C0              858      Xor    Ax,Ax            ;Writing a zero
E135 B8D8              859      Mov     Bx,Ax           ;Bx is segment pointer
E137 B855AA            860      Mov     Dx,0A855H       ;Load Dx with pattern
E13A FC                861      Cld                               ;Set forward direction
E13B                862      ClearLoop:

```

LOC	OBJ	LINE	SOURCE			
E13B	33FF	863		Xor	Di,Di	;Clear byte pointer
E13D	8EC3	864		Mov	Es,Bx	;Set segment register
E13F	268915	865		Mov	Es:[Di],Dx	;Store pattern
E142	263B15	866		Cmp	Dx,Es:[Di]	;...and test, is memory
E145	750D	867		Jne	TopFound	;...there? If not, jump
E147	890020	868		Mov	Cx,02000H	;Count 8K words, or
E14A	F3	869	Rep	Stosw		;...16K bytes to write zeros
E14B	AB					
E14C	80C704	870		Add	Bh,4	;Bump to next 16K segment
E14F	80FFA0	871		Cmp	Bh,0A0H	;...Is it last one?
E152	75E7	872		Jne	ClearLoop	;...If not, loop, else done
E154		873	TopFound:			
E154	89360000	E 874		Mov	ResetFlag,SI	;Put warm boot flag back after
		875				;...clearing memory
		876				
		877				; Perform memory test on first 2K, regardless of warmth
		878				
E158	33C0	879		Xor	Ax,Ax	;Segment starts at 0
E15A	8EC0	880		Mov	Es,Ax	;...load Es
E15C	88----	R 881		Mov	Ax,HiStack	;Setup Hi mem stack
E15F	8ED0	882		Mov	Ss,Ax	;...segment and
E161	BC0000	E 883		Mov	Sp,Offset StackTop	;...stack pointer
E164	55	884		Push	Bp	;Save error from refresh test
E165	53	885		Push	Bx	;Save Memory Size
E166	BD0200	886		Mov	Bp,2	;Count = 2K
E169	E80000	E 887		Call	MemoryTest	;Go test memory
E16C	58	888		Pop	Ax	;Restore Memory Size
E16D	B106	889		Mov	C1,6	;Convert from # of 64K's
E16F	D3E8	890		Shr	Ax,C1	;...to # of 1K's
E171	A30000	E 891		Mov	MemorySize,Ax	;Store memory size
E174	58	892		Pop	Ax	;Restore error from refresh
E175	7302	893		Jnc	First2kOk	;...jump if Ok
E177	0C10	894		Or	Al,First2kErr	;...else set error bit
E179		895	First2kOk:			
E179	A20000	E 896		Mov	MfgErrFlag,Al	;Save error status in memory
E17C	33C0	897		Xor	Ax,Ax	;Clear stack to zero
E17E	50	898		Push	Ax	;By pushing some zeros
E17F	50	899		Push	Ax	;By pushing some zeros
E180	50	900		Push	Ax	;By pushing some zeros
E181	50	901		Push	Ax	;By pushing some zeros
E182	50	902		Push	Ax	;By pushing some zeros
		903				
		904		Assume	Ss:BiosStack, Ds:BiosDataArea, Es:Nothing	
E183	88----	R 905		Mov	Ax,BiosStack	;Setup bios stack
E186	8ED0	906		Mov	Ss,Ax	;...segment and
E188	BC0000	E 907		Mov	Sp,Offset StackTop	;...stack pointer
		908				
		909				; Perform Checksum on Bios Rom
		910				
E18B	1E	911		Push	Ds	;Save our data segment
E18C	BB00E0	912		Mov	Bx,Offset StartOfBios	;Point at our ROM
E18F	0E	913		Push	Cs	;Set Ds = Cs
E190	1F	914		Pop	Ds	;...required byte subroutine
E191	B401	915		Mov	Ah,1	;...do 1 8k segment
E193	E80000	E 916		Call	ROMCheck8K	;Check it
E196	1F	917		Pop	Ds	;Restore Ds
E197	7405	918		Jz	BiosRomOk	;...and jump if all ok
		919				
		920				; Set error flag
		921				
E199	800E00001	E 922	BiosRomOk:	Or	MfgErrFlag,BiosCksErr	;Error type = Bios checksum
E19E		923				
		924				
		925				; Initialize Interrupt Controller
		926				
E19E	FA	927		Cli		;Insure interrupts off
E19F	B013	928		Mov	Al,00010011B	;Edge trig, 4 byte vector,
E1A1	E620	929		Out	PortPICICW1,Al	;...single, ICW4 needed
E1A3	B008	930		Mov	Al,8	;Address 8
E1A5	E621	931		Out	PortPICICW2,Al	;...to start vectors
E1A7	B009	932		Mov	Al,00010011B	;Buffered mode,
E1A9	E621	933		Out	PortPICICW4,Al	;...8086/8088 mode
E1AB	B0FF	934		Mov	Al,11111111B	;Mask off all
E1AD	E621	935		Out	PortPICOCW1,Al	;...interrupts for now
		936				

```

LOC OBJ          LINE    SOURCE
                937          ;      Initialize Interrupt Vectors
                938
                939          Assume Es:IntSegment
E1AF 1E          940          Push Ds          ;Save Ds value
E1B0 B8----- R      941          Mov Ax,IntSegment ;Load segment register
E1B3 8ED0        942          Mov Es,Ax        ;...with interrupt segment
E1B5 0E          943          Push Cs        ;Get fixed seg value = Cs
E1B6 1F          944          Pop Ds         ;...keep in Ax, load Ds
E1B7 B90000     945          Mov Cx,8        ;Setup first 8 vectors to
E1BA BF0000     E      946          Mov Di,Offset RAMVectors ;Start at first interrupt
E1BD            947          NullIntLoop:
E1BD 800000     E      948          Mov Ax,Offset IllegalInt ;Get offset of illegal int
E1C0 AB          949          Stosw        ;Store offset
E1C1 8CC8        950          Mov Ax,Cs        ;Get segment of illegal int
E1C3 AB          951          Stosw        ;Store segment
E1C4 E2F7        952          Loop NullIntLoop ;Loop till done
E1C6 BE0000     E      953          Mov Si,Offset ROMVectors ;From rom based vectors
E1C9 B91800     954          Mov Cx,24        ;Initializing 24 more vectors
E1CC            955          VectorLoop:
E1CC A5          956          Movsw         ;Move data from ROM to RAM
E1CD 8CC8        957          Mov Ax,Cs        ;Get segment of illegal int
E1CF AB          958          Stosw        ;Store SEG value = CS
E1D0 E2FA        959          Loop VectorLoop ;Loop till 32 vectors written
                960
                961          ;      Test if Basic exists and plug vector if it does
                962
E1D2 B8----- R      963          Mov Ax,BasicSeg   ;Point
E1D5 8ED8        964          Mov Ds,Ax        ;...at Basic
E1D7 B80000     965          Mov Bx,Offset Basic ;Offset of basic in our segment
E1DA B404        966          Mov Ah,4         ;...there are 4 8k segments
E1DC            967          BasicCkLoop:
E1DC E00000     E      968          Call ROMCheck8K   ;Check one
E1DF 7511        969          Jne NoBasic      ;...jump if checksum fails
E1E1 FECC        970          Dec Ah          ;...count - 1
E1E3 75F7        971          Jnz BasicCkLoop ;...loop till done
E1E5 1F          972          Pop Ds         ;Restore data segment register
E1E6 BF0000     E      973          Assume Ds:BiosDataArea ;...and inform ASM86
E1E9 B80000     974          Mov Di,Offset BasicTrapAddr ;Point at basic trap address
E1EC AB          975          Mov Ax,Offset Basic ;Offset = 0
E1ED B8----- R      976          Stosw        ;...store it
E1F0 AB          977          Mov Ax,BasicSeg ;Segment = F600
E1F1 1E          978          Stosw        ;...store it
E1F2            979          Push Ds        ;Keeping the stack in order
E1F2 1F          980          NoBasic:
E1F2 1F          981          Pop Ds         ;Restore data segment
                982          Assume Ds:BiosDataArea ;...and inform ASM86
                983
                984          ;      Plug NMI Vector
                985
E1F3 26C70600000000 E 986          Mov Es:Word Ptr NMITrapAddr,Offset NMIInt
                987
                988          ;      Plug Print Screen Vector
                989
E1FA 26C70600000000 E 990          Mov Es:Word Ptr PrintScreenTrapAddr,Offset PrintScreenInt
                991
                992          ;      Plug User Graphics Pointer with 0000
                993
E201 26C70600000000 E 994          Mov Es:Word Ptr VideoGraphicsTrapAddr, 0
E208 26C70602000000 E 995          Mov Es:Word Ptr VideoGraphicsTrapAddr + 2, 0
                996
                997          ;      Enable NMI Interrupts
                998
E20F BA6100     999          Mov Dx,PortPPIPortB ;Point at PPI port B
E212 EC         1000         In Al,Dx         ;Reset parity check edge
E213 0C30       1001         Or Al,PPIBDisSysParit or PPIBDisIOParity ;...trigger
E215 EE         1002         Out Dx,Al       ;...flip flops
E216 24CF       1003         And Al,Not (PPIBDisSysParit or PPIBDisIOParity) ;Release
E218 EE         1004         Out Dx,Al       ;...flip flop clear inputs
E219 B080       1005         Mov Al,NMIUnmask ;Ok, carefully
E21B E6A0       1006         Out PortNMIMask,Al ;...unmask NMI interrupts
                1007
                1008          ;      Initialize Both Mono and Color Adapters, Regardless
                1009          ;      of Their Presence
                1010
E21D B83000     1011         Mov Ax,30H      ;Switch value if mono

```

```

LOC OBJ          LINE      SOURCE
E220 A30000      E       1012      Mov      EquipFlag,Ax          ;...adapter
E223 B400        1013      Mov      Ah,VidCmdInit      ;Command to initialize
E225 CD10        1014      Int      TrapVideo          ;Initialize monochrome adapter
E227 B82000      1015      Mov      Ax,20H             ;Switch value if color
E22A A30000      E       1016      Mov      EquipFlag,Ax      ;...adapter
E22D B400        1017      Mov      Ah,VidCmdInit      ;Command to initialize
E22F CD10        1018      Int      TrapVideo          ;Initialize color adapter
1019
1020      ;      Setup Configuration Data
1021
E231 E462        1022      In       Al,PortPPIPortC    ;Get first 4 switches
E233 240F        1023      And     Al,00001111B        ;...into upper
E235 0AE0        1024      Mov     Ah,Al               ;...Ax
E237 B0AD        1025      Mov     Al,PPIBConfigSlct + PPIBStartUpMode ;Select upper
E239 E661        1026      Out     PortPPIPortB,Al     ;...configuration switches
E23B E462        1027      In     Al,PortPPIPortC    ;Get next 4 switches
E23D B104        1028      Mov     Cl,4                ;Shift it
E23F D2E0        1029      Shl    Al,Cl               ;...into upper nibble
E241 0AC4        1030      Or     Al,Ah               ;...and OR with lower switches
E243 B400        1031      Mov     Ah,0                ;Store configuration
E245 A30000      E       1032      Mov     EquipFlag,Ax      ;...word (upper byte = 0)
E248 2430        1033      And     Al,00110000B        ;Isolate display bits
E24A 7509        1034      Jnz    DisplayExistsJmp    ;...and jump if display exists
E24C B823E4      1035      Mov     Ax,Offset NullReturn ;Get offset of null int return
E24F 26A30000    E       1036      Mov     Es:Word ptr VideoTrapAddr,Ax ;...and force into video vector
E253 EB03        1037      Jmp     Short DontInitVideos ;Don't init the videos
E255           1038      DisplayExistsJmp:
1039
1040      ;      Set Video Card Per Switches
1041
E255 E80000      E       1042      Call    VideoInit          ;Proc init's per switches
1043
1044      ;      Initialize Keyboard and Keyboard Buffer
1045      DontInitVideos:
E258 B008        1046      Mov     Al,00001000B        ;Clock line = 0
E25A E661        1047      Out     PortPPIPortB,Al     ;...to signal remote 8048
E25C B95629      1048      Mov     Cx,10582            ;Delay for approx.
E25F           1049      KeyResetLoop:
E25F E2FE        1050      Loop   KeyResetLoop        ;...20 milliseconds
E261 B0CB        1051      Mov     Al,11001000B        ;Release clock and clear
E263 E661        1052      Out     PortPPIPortB,Al     ;...shift register and int
E265 3480        1053      Xor    Al,10000000B        ;...request f/f, then enable
E267 E661        1054      Out     PortPPIPortB,Al     ;...shift register output
E269 B80000      E       1055      Mov     Ax,Offset KeyBuffer ;Setup keyboard buffer
E26C A30000      E       1056      Mov     KeyBufHead,Ax       ;Head pointer
E26F A30000      E       1057      Mov     KeyBufTail,Ax      ;Tail pointer
E272 A30000      E       1058      Mov     BufferStart,Ax      ;Starting address of buffer
E275 052000      1059      Add     Ax,32                ;Length of buffer = 32
E278 A30000      E       1060      Mov     BufferEnd,Ax        ;...store it
E27B EB4990      1061      Jmp     OverNMIEntry        ;Jump over PC NMI entry point
1062
E2C3           1063      Org     0E2C3H              ;Align with PC
1064
E2C3 E90000      E       1065      Jmp     NMIInt              ;Jump to parity err. service
E2C6           1066      OverNMIEntry:
1067
1068      ;      Initialize Default RS-232 and Printer Timeout Values
1069
E2C6 B81414      1070      Mov     Ax,1414H            ;Translated to 4 bytes of
E2C9 A30000      E       1071      Mov     PrintTimeOut[0],Ax  ;...decimal 20s corresponding
E2CC A30200      E       1072      Mov     PrintTimeOut[2],Ax  ;...to 4 possible printers
E2CF B80101      1073      Mov     Ax,0101H            ;Translated to 4 bytes of
E2D2 A30000      E       1074      Mov     RS232TimeOut[0],Ax  ;...decimal 1s corresponding
E2D5 A30200      E       1075      Mov     RS232TimeOut[2],Ax  ;...to 4 possible RS-232 ports
1076
1077      ;      Determine Printer Hardware and Setup Base I/O Addr.
1078
E2D8 BE55E0      1079      Mov     Si,Offset PossiblePtrs ;Point at list of printers
E2DB BF0000      1080      Mov     Di,0                ;Reset index value
E2DE B90300      1081      Mov     Cx,3                ;Only three printers
E2E1           1082      PrintBaseLoop:
E2E1 2EBB14      1083      Mov     Dx,Cs:[Si]          ;Load Ax with first (next) base
E2E4 B0AA        1084      Mov     Al,0AAH             ;Try to write bit pattern
E2E6 EE         1085      Out     Dx,Al               ;...to printer port
E2E7 B0FF        1086      %Bus   (0FFH)              ;Force floating bus hi

```

LOC	OBJ	LINE	SOURCE
E2EB	EC	1088	In Al,Dx ;...then read data back
E2EC	3CAA	1089	Cmp Al,0AAH ;...and see if it took
E2EE	7506	1090	Jne PrinterNotThere ;...jump if not
E2F0	89950000	E 1091	Mov PrinterBase[Dil],Dx ;...else, store base address
E2F4	47	1092	Inc Di ;...and inc dest. pointer
E2F5	47	1093	Inc Di ;...to next slot
E2F6		1094	PrinterNotThere:
E2F6	46	1095	Inc Si ;Increment source to
E2F7	46	1096	Inc Si ;...next try
E2F8	E2E7	1097	Loop PrintBaseLoop ;...and loop (max of 3 times)
E2FA	8BC7	1098	Mov Ax,Di ;Calculate and put
E2FC	B103	1099	Mov Cl,3 ;...into position
E2FE	D2C8	1100	Ror Al,Cl ;...by rotating end around
E300	A20100	E 1101	Mov Byte ptr EquipFlag+1,Al ;Save the # of printers
		1102	
		1103	; Determine RS-232 Hardware and Setup Base I/O Addresses
		1104	
E303	BF0000	1105	Mov Di,0 ;Reset index value
E306	BAFB03	1106	Mov Dx,PortSio1LCR ;Use LCR for test
E309	B01A	1107	Mov Al,00011010B ;Setup LCR for standard mode
E30B	EE	1108	Out Dx,Al ;...as a test of it's existence
E30C	B0FF	1109	%Bus (0FFH) ;Force floating bus hi
E310	EC	1111	In Al,Dx ;...then read back data
E311	3C1A	1112	Cmp Al,00011010B ;...and see if it took
E313	7508	1113	Jne Sio1NotThere ;...jump if not else store
E315	C7850000F803	E 1114	Mov RS232Base[Dil],PortSio1RxData ;...store base address
E31B	47	1115	Inc Di ;...and inc dest. pointer
E31C	47	1116	Inc Di ;...to next slot
E31D		1117	Sio1NotThere:
E31D	BAFB02	1118	Mov Dx,PortSio2LCR ;Use LCR for test
E320	B01A	1119	Mov Al,00011010B ;Setup LCR for standard mode
E322	EE	1120	Out Dx,Al ;...as a test of it's existence
E323	B0FF	1121	%Bus (0FFH) ;Force floating bus hi
E327	EC	1123	In Al,Dx ;...then read back data
E328	3C1A	1124	Cmp Al,00011010B ;...and see if it took
E32A	7508	1125	Jne Sio2NotThere ;...jump if not else, store
E32C	C7850000F802	E 1126	Mov RS232Base[Dil],PortSio2RxData ;...store base address
E332	47	1127	Inc Di ;...and inc dest. pointer
E333	47	1128	Inc Di ;...to next slot
E334		1129	Sio2NotThere:
E334	8BC7	1130	Mov Ax,Di ;Or in the number of RS232
E336	00050100	E 1131	Or Byte ptr EquipFlag+1,Al ;...ports into equipflag
		1132	
		1133	; Determine Game Card Equipment and Signify it
		1134	
E33A	BA0102	1135	Mov Dx,PortGCAButtons ;Read buttons
E33D	EC	1136	In Al,Dx ;...and check
E33E	A80F	1137	Test Al,00001111B ;...that all are = 0
E340	7505	1138	Jnz NoGameCard ;Jump if no game card, else
E342	800E010010	E 1139	Or Byte ptr EquipFlag+1,00010000B ;...signify it
E347		1140	NoGameCard:
		1141	; Test memory if required
		1142	
E347	F606000001	E 1143	Test Byte Ptr EquipFlag,1 ;Test Post switch
E34C	740B	1144	Jz NoMemTest ;...jump if post off
E34E	813E00003412	E 1145	Cmp ResetFlag,1234H ;Test warm boot
E354	7403	1146	Je NoMemTest ;...and jump if warm
E356	E8E800	1147	Call SysMemTest ;...else do a memory test
E359		1148	NoMemTest:
		1149	; Initialize Expansion Modules, if Present
		1150	
E359	BA----	R 1151	Mov Dx,IORDMSeg ;2K ROMs from C0000 to F6000
		1152	Assume Ds:Nothing
E35C	1E	1153	Push Ds ;Save data segment
E35D		1154	ExpansionChk:
E35D	8EDA	1155	Mov Ds,Dx ;Make ROM data segment
E35F	BB0000	1156	Mov Bx,0 ;Bx = pointer to first word
E362	BB07	1157	Mov Ax,[Bx] ;...Get it
E364	3D55AA	1158	Cmp Ax,0AA55H ;Is signature correct?
E367	7537	1159	Jnz SignatureNG ;...Jump if it isn't
		1160	Assume Es:BiosDataArea
E369	B8----	R 1161	Mov Ax,BiosDataArea ;Point Es at our data segment
E36C	8EC0	1162	Mov Es,Ax
E36E	32E4	1163	Xor Ah,Ah ;Clear accum upper and put
E370	8A4702	1164	Mov Al,[Bx+2] ;...length in accum lower

LOC	OBJ	LINE	SOURCE
E373	B105	1165	Mov C1,5 ;Multiply Ax by 32 to get
E375	D3E0	1166	Shl Ax,C1 ;...actual length in paragraphs
E377	03D0	1167	Add Dx,Ax ;Add it to pointer
E379	B104	1168	Mov C1,4 ;Convert length from paragraphs
E37B	D3E0	1169	Shl Ax,C1 ;...to bytes
E37D	8BC8	1170	Mov Cx,Ax ;Move count to cx for proc.
E37F	E80000	E 1171	Call ROMCheckCx ;Check rom checksum
E382	7516	1172	Jnz ROMErrJmp ;...if bad, skip initialization
E384	52	1173	Push Dx ;Save pointer during init
E385	26C70600000300	E 1174	Mov IOROMInit,3 ;Offset into jump vector
E38C	268C1E0000	E 1175	Mov IOROMSegment,DS ;...Segment into jump vector
E391	26FF1E0000	E 1176	Call Dword ptr IOROMInit ;...and call init code
E396	5A	1177	Pop Dx ;Restore pointer
E397	E80B90	1178	Jmp ROMDone ;...and jump, rom init'ed
E39A		1179	ROMErrJmp:
E39A	26800E000020	E 1180	Or MfgErrFlag, IOROMErr ;Indicate Error
E3A0		1181	
E3A0	81C28000	1182	SignatureNG: Add Dx,80H ;Skip to next ROM space
E3A4		1184	ROMDone:
E3A4	81FA----	R 1185	Cmp Dx,BasicSeg ;Was that our last ROM?
E3A8	7CB3	1186	Jl ExpansionChk ;...if Dx < F600, do some more
E3AA	1F	1187	Pop Ds ;Restore data segment
		1188	Assume Ds:BiosDataArea ;...and tell ASM86
		1189	
		1190	; Enable Keyboard, Floppy and RTC Interrupts
		1191	
E3AB	E421	1192	In Al,PortPICOCW1 ;Unmask timer, floppy
E3AD	24BC	1193	And Al,10111100B ;...and keyboard
E3AF	E621	1194	Out PortPICOCW1,Al ;...interrupts
		1195	
		1196	; Signal that we are warm
		1197	
E3B1	C70600003412	E 1198	Mov ResetFlag,1234H ;Warmth code
		1199	
		1200	; Beep once
		1201	
E3B7	B302	1202	Mov Bl,2 ;Medium length Beep
E3B9	E80000	E 1203	Call Beep ;Prod operator
E3BC	33C9	1204	Xor Cx,Cx ;Pause
		1205	
		1206	; Report any errors
		1207	
E3BE	F6060000FF	E 1208	Test MfgErrFlag,1111111B ;Any Bits set?
E3C3	743F	1209	Jz BootItUp ;...jump if not
		1210	
		1211	; Beep again for error
E3C5		1212	BeepPause:
E3C5	90	1213	Nop ;...a
E3C6	E2FD	1214	Loop BeepPause ;...while
E3C8	B302	1215	Mov Bl,2 ;Medium length Beep
E3CA	E80000	E 1216	Call Beep ;Prod operator
E3CD	1E	1217	Push Ds ;Save Ds for later
E3CE	0E	1218	Push Cs ;Load code pointer
E3CF	1F	1219	Pop Ds ;...into data pointer
E3D0	BE24E490	1220	Mov Si,Offset ErrorMessage ;Point at error message
E3D4	E80000	E 1221	Call PrintMessage ;...and print it
E3D7	1F	1222	Pop Ds ;Restore Ds
		1223	
		1224	; Get response
		1225	
E3D8	B80E00	1226	Mov Ax,000EH ;Position cursor to end
E3DB	E80000	E 1227	Call PositionCursor ;...for printing error number
E3DE	A00000	E 1228	Mov Al,MfgErrFlag ;Get the error
E3E1	C606000000	E 1229	Mov MfgErrFlag,0 ;...clear the error
E3E6	E80000	E 1230	Call ByteOut ;Print byte
E3E9	B81D00	1231	Mov Ax,001DH ;Position cursor to end
E3EC	E80000	E 1232	Call PositionCursor ;...for retrieving answer
E3EF	E80000	E 1233	Call KeyIn ;Get keyboard data
E3F2	50	1234	Push Ax ;Save Ax
E3F3	E80000	E 1235	Call CharOut ;Echo keyboard response
E3F6	58	1236	Pop Ax ;Restore Ax
E3F7	3C59	1237	Cmp Al,'Y' ;...Y?
E3F9	7409	1238	Je BootItUp ;...jump and continue if yes
E3FB	3C79	1239	Cmp Al,'y' ;...y?

LOC	OBJ	LINE	SOURCE
E3FD	7405	1240	Je BootItUp ;...jump and continue if yes
E3FF	EA66E0----	1241	Jmp ResetEntry ;Re-boot
		1242	
		1243	
E404		1244	BootItUp:
		1245	;
		1246	Logon with copyright notice
E404	E80000	1247	Call ClearScreen ;Clear the screen
E407	A00000	1248	Mov Al,Byte Ptr CrtColumns ;Save column count
E40A	50	1249	Push Ax ;...in stack
E40B	0E	1250	Push Cs ;Make Ds
E40C	1F	1251	Pop Ds ;...Cs
E40D	BE00E0	1252	Mov Si,Offset LogOnMsg ;Point at message
E410	E80000	1253	Call PrintMessage ;...and send it
E413	58	1254	Pop Ax ;Restore column count
E414	3C50	1255	Cmp Al,80 ;...is it >= 80?
E416	7303	1256	Jae SkipBootCr ;Jump if 80 or above
E418	E80000	1257	Call CrtCrLf ;...else do a carriage return
E41B		1258	SkipBootCr:
E41B	BE40E0	1259	Mov Si,Offset LogOn2ndHalf ;Point at message
E41E	E80000	1260	Call PrintMessage ;...and send it
		1261	
		1262	;
		1263	Boot it Up
E421	CD19	1264	Int TrapBoot <i>E6F2</i>
E423		1265	NullReturn:
E423	CF	1266	Iret ;Null return from interrupt
E424	53797374656D20 4572726F722023 20202C20436F6E 74696E75653F20	1267	ErrorMsg Db 'System Error # , Continue? ',0
E440	00	1268	\$Eject

```

LOC OBJ                LINE    SOURCE
                                1269    ;
                                1270    ;
                                1271    ;
                                1272    ;
                                1273    ;
                                1274    ;
                                1275    ;
                                1276    ;
                                1277    ;
                                1278    ;
                                1279    ;
                                1280    ;
                                1281    ;
                                1282    ;
                                1283    ;
                                1284    ;
                                1285    ;
                                1286    ;
                                1287    ;
                                1288    ;
                                1289    ;
                                1290    ;
                                1291    ;
                                1292    ;
                                1293    ;
                                1294    ;
                                1295    ;
                                1296    ;
                                1297    ;
                                1298    ;
                                1299    ;
                                1300    ;
                                1301    ;
                                1302    ;
                                1303    ;
                                1304    ;
                                1305    ;
                                1306    ;
                                1307    ;
                                1308    ;
                                1309    ;
                                1310    ;
                                1311    ;
                                1312    ;
                                1313    ;
                                1314    ;
                                1315    ;
                                1316    ;
                                1317    ;
                                1318    ;
                                1319    ;
                                1320    ;
                                1321    ;
                                1322    ;
                                1323    ;
                                1324    ;
                                1325    ;
                                1326    ;
                                1327    ;
                                1328    ;
                                1329    ;
                                1330    ;
                                1331    ;
                                1332    ;
                                1333    ;
                                1334    ;
                                1335    ;
                                1336    ;
                                1337    ;
                                1338    ;
                                1339    ;
                                1340    ;
                                1341    ;
                                1342    ;
                                1343    ;

E441                    1273    SysMemTest Proc Near
E441 B401                1274    Mov Ah,VidCmdCurType ;Turn off cursor
E443 B90720              1275    Mov Cx,2007H ;...by putting it on an
E446 CD10                1276    Int TrapVideo ;...imaginary scan line
E448 BEB9E490            1277    Mov Si,Offset MemTstMsg ;Point at message
E44C 1E                  1278    Push Ds ;Save current Ds
E44D 0E                  1279    Push Cs ;Send message to screen
E44E 1F                  1280    Pop Ds ;...telling about
E44F E80000              E 1281    Call PrintMessage ;...memory test

                                1282    ;
                                1283    ; Determine which video memory
                                1284    ;
E452 1F                  1285    Pop Ds ;Restore data segment
E453 A00000              E 1286    Mov Al,CrtMode ;Which monitor?
E456 3C07                1287    Cmp Al,7 ;...Monochrome?
E458 BB-----           R 1288    Mov Bx,ColorSeg ;Point at color
E45B B81000              1289    Mov Ax,16 ;...and say 16K
E45E 7505                1290    Jne ColorMon ;...jump if color
E460 BB-----           R 1291    Mov Bx,MonoSeg ;...else point at monochrome
E463 B004                1292    Mov Al,4 ;...and only 4K
E465                    1293    ColorMon:
E465 53                  1294    Push Bx ;...and save for memory test
E466 50                  1295    Push Ax ;...save length of video mem

                                1296    ;
                                1297    ; Test system memory
                                1298    ;
E467 8B2E0000            E 1299    Mov Bp,MemorySize ;Get # of 1k Blocks in mem
E46B 4D                  1300    Dec Bp ;...leave the first one alone
E46C 4D                  1301    Dec Bp ;...and the second one
E46D 1E                  1302    Push Ds ;Save data segment
E46E 43                  1303    Inc Bx ;...make seg 2
E46F 43                  1304    Inc Bx ;...paragraphs into display
E470 8EDB                1305    Mov Ds,Bx ;Load segment for message
                                1306    Assume Ds:FlasherSegment ;Dummy segment for DsZero def.
E472 BB8000              1307    Mov Bx,2048 / 16 ;Point at 3rd
E475 8EC3                1308    Mov Es,Bx ;...1K block with Es
E477 C60600005C          1309    Mov DsZero,05CH ;Flasher character "\
E47C                    1310    OuterMemLoop:
E47C F7C50700            1311    Test Bp,111B ;Flip only if lower three bits
E480 7505                1312    Jnz SkipFlip ;...of counter are = 0
E482 8036000073          1313    Xor DsZero,073H ;...alternate between \ and /
E487                    1314    SkipFlip:
E487 E80000              E 1315    Call MemoryTest ;...test memory
E48A 7207                1316    Jc LoadMemErr ;...and jump if bad
E48C 4D                  1317    Dec Bp ;Bp has count
E48D 75ED                1318    Jnz OuterMemLoop ;...jump and loop
E48F 1F                  1319    Pop Ds ;Restore data segment
E490 EB0790              1320    Jmp TestVidMem ;...and go test video
E493                    1321    LoadMemErr:
E493 1F                  1322    Pop Ds ;Restore data segment
                                1323    Assume Ds:BiosDataArea ;...and tell ASM86
E494 800E000002          E 1324    Or MfgErrFlag,MemErr ;Load error number

                                1325    ;
                                1326    ; Test video memory
                                1327    ;
E499                    1328    TestVidMem:
E499 A00000              E 1328    Mov Al,CrtModeSet ;Current setting of video reg
E49C 24F7                1329    And Al,11110111B ;...turn off video
E49E 8B160000            E 1330    Mov Dx,ActiveCard ;...on
E4A2 83C204              1331    Add Dx,4 ;...active
E4A5 EE                  1332    Out Dx,Al ;...video card
E4A6 5D                  1333    Pop Bp ;Length in 1K blocks
E4A7 07                  1334    Pop Es ;...segment of video memory
E4A8                    1335    OuterVidLoop:
E4A8 E80000              E 1336    Call MemoryTest ;...do the test
E4AB 4D                  1337    Dec Bp ;Bp has count
E4AC 75FA                1338    Jnz OuterVidLoop ;...jump and loop
E4AE 7305                1339    Jnc MemTstRtn ;Return if ok
E4B0 800E000004          E 1340    Or MfgErrFlag,VidMemErr ;...else, set error code

                                1341    ;
                                1342    ; Turn Video Back On
E4B5                    1343    MemTstRtn:

```

LOC	OBJ		LINE	SOURCE		
E4B5	E00000	E	1344		Call	VideoInit ;Initialize the video
E4B8	C3		1345		Ret	;...else return
E4B9	54657374696E67 204D656D6F7279 20202F		1346	MemTstMsg	Db	'Testing Memory /',0 ;Memory test message
E4CA	00		1347	SysMemTest	Endp	
			1348			
----			1349	Bios	Ends	
			1350	End	POREntry	

ASSEMBLY COMPLETE, NO ERRORS FOUND

IBM PC/XT 8086/8087/8088 MACRO ASSEMBLER V1.1 ASSEMBLY OF MODULE BOOT
OBJECT MODULE PLACED IN BOOT.OBJ
ASSEMBLER INVOKED BY: ASM86 BOOT.SRC

```
LOC OBJ          LINE    SOURCE
                1 +1  $Title ('DTC/PC BIOS Bootstrap Loader V1.0')
                2 +1  $Pagelength (80) Pagewidth (132) Debug Nogen
                3      Name Boot
                4
                5
                6      ;   Author:      Don K. Harrison
                7
                8      ;   Start date:  October 25, 1983      Last edit:  December 27, 1983
                9
               10
               11
               12      ;           *****
               13      ;           * Module Description *
               14      ;           *****
               15
               16      ;           This module is called by interrupt 19 (TrapBoot).
               17      ;           It's function is to read in the boot image from the diskette
               18      ;           and transfer control to it.
               19
               20
               21
               22      ;           (c) Display Telecommunications Corporation, 1983
               23      ;           All Rights Reserved
               24
               25      $Eject
```

LOC	OBJ	LINE	SOURCE
		26	
		27	
		28	;
		29	;
		30	*****
		31	;
		32	
		33	
		34	
		35	
		36	
		37	
		38	\$Eject

```
LOC OBJ          LINE    SOURCE
                 39
                 40          ; *****
                 41          ; * Public Symbols *
                 42          ; *****
                 43
                 44          Public BootDriver
                 45
                 46          ; *****
                 47          ; * Equates *
                 48          ; *****
                 49
7C00             50      BootOffset   Equ    7C00H          ;Location of loaded boot code
000D             51      AsciiCarriage Equ    0DH           ;Carriage return code
000A             52      AsciiLineFeed Equ    0AH           ;Line feed code
                 53
                 54      $Include (IbmInc)
=1              55          ; *****
=1              56          ; * Global Include File *
=1              57          ; *****
=1              58      $Nolist
                 578     $Eject
```

```
LOC OBJ          LINE    SOURCE
                    579
                    580      ;          *****
                    581      ;          * Data Segments *
                    582      ;          *****
                    583
-----          584      IntSegment      Segment Public
                    585                      Extern FloppyParamsTrapAddr:DWord, BasicTrapAddr:DWord
                    586
-----          587      IntSegment      Ends
                    588
-----          589      BootSegment     Segment at 0
-----          590      BootSegment     Ends
                    591
-----          592      BasicSeg        Segment Common
                    593                      Assume Cs:BasicSeg
0000             594                      Org      0
0000             595      Basic          Label Far
-----          596      BasicSeg        Ends
                    597      $Eject
```


LOC	OBJ	LINE	SOURCE
		598	
		599	;
		600	*****
		601	* Code Segment *
		602	*****
		603	;
----		604	Bios Segment Common
		605	
		606	Extrn FloppyParamsPointer:DWORD, ClearScreen:Near
		607	Extrn PrintMessage:Near, KeyIn:Near
		608	
		609	Assume Cs:Bios, Ds:IntSegment
		610	
E6F2		611	Org 0E6F2H ;Align with Xt and Pc
		612	
E6F2		613	BootDriver:
E6F2	E90BFF	614	Jmp BootContinue ;Jump to code
		615	
E600		616	Org 0E600H ;Place code above init
E600		617	BootContinue:
E600	FB	618	Sti ;Enable Interrupts
E601	B0----	619	Mov Ax,IntSegment ;Point at vector segment
E604	8ED8	620	Mov Ds,Ax ;...make it data segment
		621	
		622	;
		623	Initialize pointer to disk tables
E606	C70600000000	624	Mov Word ptr FloppyParamsTrapAddr, Offset FloppyParamsPointer
E60C	8C0E0200	625	Mov Word ptr FloppyParamsTrapAddr + 2, Cs
		626	
		627	;
		628	Load from diskette
E610	B00400	629	Mov Ax,4 ;Reset try = 4 times
E613		630	RetryLoop:
E613	50	631	Push Ax ;Save count
E614	B400	632	Mov Ah,0 ;Reset the diskette
E616	CD13	633	Int TrapFDDriver ;...system
E618	7219	634	Jc BootFailed ;Carry set on error, try again
E61A	B001	635	Mov Al,1 ;Read in 1 sector
E61C	B402	636	Mov Ah,DiskCmdRead ;Command in Ah
E61E	BA0000	637	Mov Dx,BootSegment ;Point Es to bootstrap segment
E621	8EC2	638	Mov Es,Dx ;... (which is at zero)
E623	BB007C	639	Mov Bx,BootOffset ;Load Bx with address of
E626	B101	640	Mov Cl,1 ;...first byte in boot area
E628	B500	641	Mov Ch,0 ;Cl = sector 1, Ch = track 0
E62A	CD13	642	Int TrapFDDriver ;Execute disk I/O software
E62C	7205	643	Jc BootFailed ;Jump and retry if error
		644	Assume Cs:BootSegment ;Assume causes Jmp instr. to
E62E	EA007C----	645	Jmp Far ptr BootOffset ;...reference BootSegment
		646	Assume Cs:Bios ;Back to Bios
E633		647	BootFailed:
E633	50	648	Pop Ax ;Restore count
E634	FEC8	649	Dec Al ;...and reduce it by 1
E636	75DB	650	Jnz RetryLoop ;...keep trying till 0
		651	
		652	;
		653	WhatGives:
E638	0AE4	654	Or Ah,Ah ;Second time around?
E63A	7517	655	Jnz TryBasic ;...try to load basic
E63C	E00000	656	Call ClearScreen ;Clear the screen
E63F	0E	657	Push Cs ;Print
E640	1F	658	Pop Ds ;...message
E641	BE70E690	659	Mov Si,Offset LoadDiskMsg ;...telling him
E645	E00000	660	Call PrintMessage ;...to insert diskette
E648	E00000	661	Call Keyin ;Get his response
E64B	E00000	662	Call ClearScreen ;Clear the screen
E64E	B004FF	663	Mov Ax,0FF04H ;Try 4 more times, set flag
E651	EBC0	664	Jmp RetryLoop ;...try again
		665	
		666	;
		667	Try to load basic
E653		668	TryBasic:
E653	33C0	669	Xor Ax,Ax ;Point at interrupt segment
E655	8ED8	670	Mov Ds,Ax ;...to check
E657	C4060000	671	Les Ax,BasicTrapAddr ;...if basic available
E65B	BCC3	672	Mov Bx,Es ;...for booting

LOC	OBJ	LINE	SOURCE
E65D	3D0000	673	Cmp Ax,Offset Basic ;Check offset
E660	880000	674	Mov Ax,0 ;Clear flag incase failure
E663	75D3	675	Jne WhatSives ;...jump if not basic
E665	81FB----	676	Cmp Bx,BasicSeg ;Check segment
E669	75CD	677	Jne WhatSives ;...jump if not basic
E66B	EA0000----	678	Jmp Basic ;...else startup basic
		679	
E670	496E7365727420 64697368657474 6520696E204452 49564520412E	680	LoadDiskMsg Db 'Insert diskette in DRIVE A.',AsciiCarriage
E68B	0D		
E68C	0A	681	Db AsciiLineFeed,' Press any key.',0
E68D	20202020507265 737320616E7920 6865792E		
E69F	00	682	
----		683	Bios Ends
		684	
		685	End

ASSEMBLY COMPLETE, NO ERRORS FOUND

IBM PC/XT 8086/8087/8088 MACRO ASSEMBLER V1.1 ASSEMBLY OF MODULE RS232
OBJECT MODULE PLACED IN RS232.OBJ
ASSEMBLER INVOKED BY: ASM86 RS232.SRC

LOC	OBJ	LINE	SOURCE
		1 +1	\$Title ('DTC/PC BIOS RS232 Driver V1.0')
		2 +1	\$Pagelength (80) Pagewidth (132) Debug Nogen
		3	Name RS232
		4	
		5	
		6	; Author: Don K. Harrison
		7	
		8	; Start date: October 30, 1983 Last edit: December 28, 1983
		9	
		10	
		11	; *****
		12	; * Module Description *
		13	; *****
		14	
		15	; This module contains the serial interface driver. The driver
		16	; is entered via interrupt 20 (TrapSIODriver).
		17	
		18	
		19	
		20	
		21	
		22	; (c) Display Telecommunications Corporation, 1983
		23	; All Rights Reserved
		24	
		25	\$Eject

LOC	OBJ	LINE	SOURCE
		26	
		27	
		28	;
		29	;
		30	*****
		31	* Revision History *
		32	*****
		33	
		34	
		35	
		36	
		37	
		38	\$Eject

LOC	OBJ	LINE	SOURCE
		39	
		40	;
		41	;
		42	*****
		43	*****
		44	Public CommsDriver
		45	
		46	;
		47	*****
		48	*****
		49	*****
		50	;
		51	All Equates in include file: IbmInc
		52	\$Include (IbmInc)
=1		53	;
=1		54	*****
=1		55	*****
=1		56	*****
		576	\$Nolist
			\$Eject

LOC	OBJ	LINE	SOURCE
		577	
		578	;
		579	;
		580	*****
		581	*****
---		582	IntSegment Segment Public
---		583	
---		584	IntSegment Ends
---		585	
---		586	BiosDataArea Segment Public
---		587	Extrn RS232Base:Word, RS232TimeOut:Byte
---		588	BiosDataArea Ends
		589	\$Eject

```

LOC OBJ          LINE    SOURCE
                    590
                    591      ; *****
                    592      ; * Code Segment *
                    593      ; *****
                    594
----             595      Bios      Segment Common
                    596
                    597      Assume  Cs:Bios,Ds:BiosDataArea
                    598
E729             599      Org      0E729H      ;Align with Pc and Xt
                    600
E729 1704       601      BaudTable Dw      SI0110Baud      ;110 Baud
E72B 0003       602      Dw      SI0150Baud      ;150 Baud
E72D 8001       603      Dw      SI0300Baud      ;300 Baud
E72F C000       604      Dw      SI0600Baud      ;600 Baud
E731 6000       605      Dw      SI01200Baud     ;1200 Baud
E733 3000       606      Dw      SI02400Baud     ;2400 Baud
E735 1800       607      Dw      SI04800Baud     ;4800 Baud
E737 0C00       608      Dw      SI09600Baud     ;9600 Baud
                    609
E739             610      CommsDriver Proc  Far
E739 FB         611      Sti
E73A 1E         612      Push  Ds      ;Restore interrupts
E73B 52         613      Push  Dx      ;Save all but Ax
E73C 56         614      Push  Si      ;...Dx has card #
E73D 57         615      Push  Di
E73E 51         616      Push  Cx
E73F 53         617      Push  Bx
E740 8B-----  618      Mov   Bx,BiosDataArea ;Setup our
E743 8EDB       619      Mov   Ds,Bx      ;...data segment
E745 8BFA       620      Mov   Di,Dx      ;Save card # for send/receive
E747 8BDA       621      Mov   Bx,Dx      ;Calculate index
E749 D1E3       622      Shl  Bx,1        ;...into base addresses
E74B 8B970000   623      Mov   Dx,RS232Base [Bx] ;Fetch base address to Dx
E74F 0BD2       624      Or   Dx,Dx      ;Test for existence
E751 7410       625      Jz   SI0Return   ;...and jump if not there
E753 0AE4       626      Or   Ah,Ah      ;Command = 0?
E755 7413       627      Jz   SI0Init    ;...if it is, jump to init
E757 FECC       628      Dec  Ah         ;Command = 1?
E759 743B       629      Jz   SI0Send    ;...if it is, jump to send
E75B FECC       630      Dec  Ah         ;Command = 2?
E75D 7454       631      Jz   SioReceive ;...if it is, jump to receive
E75F FECC       632      Dec  Ah         ;Command = 3?
E761 7464       633      Jz   SioStatus  ;...if it is, return status
E763             634      SI0Return:
E763 5B         635      Pop   Bx      ;Restore registers
E764 59         636      Pop   Cx
E765 5F         637      Pop   Di
E766 5E         638      Pop   Si
E767 5A         639      Pop   Dx
E768 1F         640      Pop   Ds
E769 CF         641      Iret      ;...and return
                    642
                    643      ; *****
                    644      ; * SI0 Initialization *
                    645      ; *****
                    646
E76A             647      SI0Init:
                    648      ;      Set the baud rate
                    649
E76A 50         650      Push  Ax      ;Save mode
E76B 8AD8       651      Mov   Bl,Al     ;Save parameters
E76D 83C203     652      Add   Dx,3      ;Point at SI0 control reg
E770 B080       653      Mov   Al,SioAccessBrgDiv ;...and point at baud port
E772 EE         654      Out  Dx,Al     ;... (Dlab = 1)
E773 B104       655      Mov   Cl,4      ;Rotate parameter left 4 places
E775 D2C3       656      Rol  Bl,Cl     ;...placing baud index into
E777 81E30E00   657      And  Bx,0000000000001110B ;...2nd, 3rd and 4th bits of Bx
E77B 2E8B8729E7 658      Mov   Ax,Cs:BaudTable[Bx] ;...and retrieve divisor
E780 83EA03     659      Sub  Dx,3      ;Restore I/O pointer to base
E783 EE         660      Out  Dx,Al     ;Send lower divisor to SI0
E784 42         661      Inc  Dx      ;Set I/O pointer to base+1
E785 8AC4       662      Mov  Al,Ah     ;Send upper divisor
E787 EE         663      Out  Dx,Al     ;...to SI0
                    664

```

```

LOC OBJ          LINE    SOURCE
                665      ;      Send line control
                666
E788 58          667      Pop     Ax           ;Restore mode
E789 42          668      Inc     Dx           ;Point at line
E78A 42          669      Inc     Dx           ;...control register
E78B 241F        670      And     Al,00000011111B ;Strip baud bits from
E78D EE          671      Out    Dx,Al        ;...mode and send to SIO
                672
                673      ;      Disable all interrupts
                674
E78E 8000        675      Mov     Al,0         ;Turn off all
E790 4A          676      Dec     Dx           ;Point back at interrupt enable
E791 4A          677      Dec     Dx           ;...register
E792 EE          678      Out    Dx,Al        ;...and set to no ints
                679
                680      ;      Point back at base
                681
E793 4A          682      Dec     Dx           ;Point back at base
E794 EB31        683      Jmp    Short SIOStatus ;Jump and return status
                684
                685      ;
                686      ;      * SIO Send Character *
                687      ;
                688
E796            689      SioSend:
E796 50          690      Push   Ax           ;Save the character
E797 B003        691      Mov     Al,SioEnabRTS Or SioEnabDTR ;Al has modem setup
E799 B730        692      Mov     Bh,SioCTS Or SioDSR ;Bh has value for modem reg
E79B B320        693      Mov     Bl,SioTxReady ;Bl has value for line reg
E79D EB4900      694      Call   SetupandWait ;Setup port and wait
E7A0 7509        695      Jnz    SendTimeout ;If non zero, timeout
E7A2 83EA05      696      Sub     Dx,5         ;Point back to data port
E7A5 59          697      Pop     Cx           ;Recover char, saving Ah
E7A6 8AC1        698      Mov     Al,C1        ;...then move it into
E7A8 EE          699      Out    Dx,Al        ;...Al and output it
E7A9 EBB8        700      Jmp    SioReturn     ;...and return Ah = status
E7AB            701      SendTimeout:
E7AB 59          702      Pop     Cx           ;Recover data byte
E7AC 8AC1        703      Mov     Al,C1        ;...into Al, preserving Ah
E7AE            704      ReceiveTimeout:
E7AE B0CC00      705      Or     Ah,10000000B ;Return error bit and
E7B1 EBB0        706      Jmp    SioReturn     ;...return
                707
                708      ;
                709      ;      * SIO Receive Character *
                710      ;
                711
E7B3            712      SIOReceive:
E7B3 B001        713      Mov     Al,SioEnabDTR ;Al has modem setup
E7B5 B720        714      Mov     Bh,SioDSR ;Bh has value for modem reg
E7B7 B301        715      Mov     Bl,SioRxReady ;Bl has value for line reg
E7B9 EB2D00      716      Call   SetupandWait ;Setup port and wait
E7BC 75F0        717      Jnz    ReceiveTimeout ;If non zero, timeout
E7BE 80E41E      718      And     Ah,00011110B ;Mask off error flags
E7C1 83EA05      719      Sub     Dx,5         ;Point back at data register
E7C4 EC          720      In     Al,Dx        ;Get the character
E7C5 EB9C        721      Jmp    SioReturn     ;...and return with it Ah=stat.
                722
                723      ;
                724      ;      * SIO Status Return Procedure *
                725      ;      * Ah = Line control register *
                726      ;
                727
E7C7            728      SIOStatus:
E7C7 83C205      729      Add     Dx,5         ;Point at line status
E7CA EC          730      In     Al,Dx        ;...and get it
E7CB BAE0        731      Mov     Ah,Al        ;...and put in Ah for return
E7CD 42          732      Inc     Dx           ;Point at modem status
E7CE EC          733      In     Al,Dx        ;...get it and return it
E7CF EB92        734      Jmp    SioReturn     ;...in Al
                735      CommsDriver Endp
                736
                737
                738      ;
                739      ;      * Proc used by Setup and Wait *

```



```

LOC OBJ          LINE      SOURCE
                740      ;          * Ah = Line control register *
                741      ;          *****
                742
E7D1             743      TimeoutProc Proc Near
E7D1 8A9D0000    E         744      Mov      Bl,RS232Timeout[Di]    ;Di=Card # (dx at entry)
E7D5             745      SioTimeouter:
E7D5 2BC9        746      Sub      Cx,Cx          ;Maximum loop count
E7D7             747      SioTimeInner:
E7D7 EC         748      In      Al,Dx          ;Get status
E7D8 8AE0        749      Mov      AH,Al          ;...into Ah
E7DA 22C7        750      And     Al,Bh          ;And with Bh mask
E7DC 3AC7        751      Cmp     Al,Bh          ;...and test if it matches
E7DE 7408        752      Je      MatchReturn    ;...jump if it does
E7E0 E2F5        753      Loop   SioTimeInner    ;Loop till something happens
E7E2 FECB        754      Dec     Bl             ;...decrementing outer pointer
E7E4 75EF        755      Jnz    SioTimeouter    ;...and looping till timed out
E7E6 0AFF        756      Or     Bh,Bh           ;Return timed out, z flag clear
E7E8             757      MatchReturn:
E7E8 C3         758      Ret
                759      TimeoutProc Endp
                760
                761      ;          *****
                762      ;          * Modem Setup and Status Wait Routine *
                763      ;          * Ah = Line control register *
                764      ;          *****
                765
E7E9             766      SetupAndWait Proc Near
E7E9 83C204      767      Add     Dx,4           ;Move from base to modem control
E7EC EE         768      Out     Dx,Al          ;...register and output command
E7ED 42         769      Inc     Dx             ;Move I/O pointer to modem status
E7EE 42         770      Inc     Dx             ;...register
E7EF 53         771      Push   Bx             ;Save Bl
E7F0 E8DEFF      772      Call   TimeoutProc    ;Wait for timeout or correct status
E7F3 5B         773      Pop     Bx            ;Restore Bx regardless
E7F4 7506        774      Jnz    TimedOut       ;Jump if not timed out
E7F6 4A         775      Dec     Dx             ;Point back at line status
E7F7 8AFB        776      Mov     Bh,Bl          ;Match for line register
E7F9 E8D5FF      777      Call   TimeoutProc    ;Wait for timeout or correct status
E7FC             778      TimedOut:
E7FC C3         779      Ret
                780      SetupAndWait Endp
                781
                782      Bios      Ends
                783
                784      End

```

ASSEMBLY COMPLETE, NO ERRORS FOUND

IBM PC/XT 8086/8087/8088 MACRO ASSEMBLER V1.1 ASSEMBLY OF MODULE KEYBOARD
OBJECT MODULE PLACED IN KEYBOARD.OBJ
ASSEMBLER INVOKED BY: ASM86 KEYBOARD.SRC

```
LOC OBJ          LINE    SOURCE
                1 +1  $Title ('DTC/PC BIOS Keyboard Interface V1.0')
                2 +1  $Pagelength (80) Pagewidth (132) Debug Nogen
                3      Name Keyboard
                4
                5
                6      ;   Author:      Don K. Harrison
                7
                8      ;   Start date:  November 3, 1983      Last edit:  December 17, 1983
                9
               10
               11      ;           *****
               12      ;           * Module Description *
               13      ;           *****
               14
               15      ;           This module handles the keyboard interface through interrupt
               16      ;           16 (TrapKeyDrive).
               17
               18
               19
               20
               21      ;           (c) Display Telecommunications Corporation, 1983
               22      ;           All Rights Reserved
               23
               24      $Eject
```

LOC	OBJ	LINE	SOURCE
		25	
		26	
		27	; *****
		28	; * Revision History *
		29	; *****
		30	
		31	
		32	
		33	
		34	
		35	
		36	
		37	\$Eject

LOC	OBJ	LINE	SOURCE
		38	
		39	;
		40	;
		41	;
		42	*****
		43	Public KeyboardDriver
		44	
		45	
		46	
		47	\$Eject

LOC	OBJ	LINE	SOURCE
		48	
		49	;
		50	;
		51	;
		52	
---		53	BiosDataArea Segment Public
		54	Extrn KeyboardFlag1:Byte
		55	Extrn KeyBufHead:Word, KeyBufTail:Word
		56	Extrn BufferStart:Word, BufferEnd:Word
---		57	BiosDataArea Ends
		58	\$Eject

```

LOC  OBJ          LINE    SOURCE
                                     59
                                     60 ;
                                     61 ;
                                     62 ;
                                     63 ;
----- 64 Bios Segment Common
                                     65
                                     66 Assume Cs:Bios, Ds:BiosDataArea
                                     67
E82E    68 Org 0E82EH ;Align with PC and XT
                                     69
E82E    70 KeyboardDriver Proc Far
E82E FB 71 Sti ;Restore interrupts
E82F 1E 72 Push Ds ;Save Registers
E830 53 73 Push Bx
E831 BB---- R 74 Mov Bx,BiosDataArea ;Our data segment
E834 8EDB 75 Mov Ds,Bx ;...point to it
                                     76
                                     77 ; Do Case Ah;
                                     78
E836 0AE4 79 Or Ah,Ah ;Case = 0
E838 740B 80 Jz KeyRead ;...read ascii translation
E83A FECC 81 Dec Ah ;Case = 1
E83C 742D 82 Jz KeyStatus ;...read key status
E83E FECC 83 Dec Ah ;Case = 2
E840 743A 84 Jz KeyShift ;...read shift status
E842 85 KeyDrvReturn:
E842 5B 86 Pop Bx ;Case = ??? and general return
E843 1F 87 Pop Ds ;Restore registers
E844 CF 88 Iret ;...and return
                                     89
                                     90 ;
                                     91 ;
                                     92 ;
                                     93 ;
E845    94 KeyRead:
E845 FA 95 Cli ;No ints while reading pointer
E846 8B1E0000 E 96 Mov Bx,KeyBufHead ;If heads = tails, buffer
E84A 3B1E0000 E 97 Cmp Bx,KeyBufTail ;...is empty
E84E 7503 98 Jne KeyBufRdy ;Jump if buffer has data
E850 FB 99 Sti ;...else, restore ints and loop
E851 EBF2 100 Jmp Keyread ;...till something comes in
E853 101 KeyBufRdy:
E853 8B07 102 Mov Ax,[Bx] ;Get Ah=scan code Al=ascii
E855 43 103 Inc Bx ;Remove char from buffer
E856 43 104 Inc Bx ;...by adjusting pointers
E857 891E0000 E 105 Mov KeyBufHead,Bx ;...and
E85B 3B1E0000 E 106 Cmp Bx,BufferEnd ;...circling
E85F 75E1 107 Jne KeyDrvReturn ;Jump and return if not at end
E861 8B1E0000 E 108 Mov Bx,BufferStart ;...else, move the pointer
E865 891E0000 E 109 Mov KeyBufHead,Bx ;...to the start
E869 EBD7 110 Jmp KeyDrvReturn ;...and return
                                     111
                                     112 ;
                                     113 ;
                                     114 ;
                                     115 ;
E86B    116 KeyStatus:
E86B FA 117 Cli ;No ints while reading pointer
E86C 8B1E0000 E 118 Mov Bx,KeyBufHead ;If heads = tails, buffer
E870 3B1E0000 E 119 Cmp Bx,KeyBufTail ;...is empty
E874 8B07 120 Mov Ax,[Bx] ;Get the data (if any)
E876 FB 121 Sti ;...restore interrupts
E877 5B 122 Pop Bx ;Restore registers, but
E878 1F 123 Pop Ds ;...keep the flags intact
E879 CA0200 124 Ret 2 ;(Like Iret, but flags not affected)
                                     125 ;..Note: Interrupts leave enabled!!
                                     126
                                     127 ;
                                     128 ;
                                     129 ;
                                     130 ;
E87C    131 KeyShift:
E87C A00000 E 132 Mov Al,KeyboardFlag1 ;Get shift state flag
E87F EBC1 133 Jmp KeyDrvReturn ;...and return with it

```

LOC	OBJ	LINE	SOURCE
		134	
		135	KeyboardDriver Endp
		136	
----		137	Bios Ends
		138	
		139	End

ASSEMBLY COMPLETE, NO ERRORS FOUND

IBM PC/XT 0086/8087/8088 MACRO ASSEMBLER V1.1 ASSEMBLY OF MODULE KEYINT
OBJECT MODULE PLACED IN KEYINT.OBJ
ASSEMBLER INVOKED BY: ASM86 KEYINT.SRC

```
LOC OBJ          LINE    SOURCE
                1 +1  $Title ('DTC/PC BIOS Keyboard Interrupt Service Routine V1.0')
                2 +1  $Pagelength (80) Pagewidth (132) Debug Nogen
                3      Name KeyInt
                4
                5
                6      ;   Author:      Don K. Harrison
                7
                8      ;   Start date:  November 3, 1983      Last edit:  December 28, 1983
                9
               10
               11      ;           *****
               12      ;           * Module Description *
               13      ;           *****
               14
               15      ;   This module processes the keyboard interrupt, interrupt 9.
               16
               17
               18
               19
               20
               21
               22      ;           (c) Display Telecommunications Corporation, 1983
               23      ;           All Rights Reserved
               24
               25      $Eject
```


LOC	OBJ	LINE	SOURCE
		26	
		27	
		28	;
		29	;
		30	;
		31	*****
		32	
		33	
		34	
		35	
		36	
		37	
		38	\$Eject

```

LOC OBJ          LINE    SOURCE
                39
                40      ; *****
                41      ; * Public Symbols *
                42      ; *****
                43
                44      Public KeyboardHdwrInt
                45
                46
                47      ; *****
                48      ; * Equates *
                49      ; *****
                50
                51      ; Equates in include file: IbmInc
                52
                53      $Include (IbmInc)
=1 54      ; *****
=1 55      ; * Global Include File *
=1 56      ; *****
=1 57      $Nolist
0080 577      InsBit      Equ      10000000B      ;Flag bits
0040 578      CapsBit     Equ      01000000B      ;...indicating
0020 579      NumBit      Equ      00100000B      ;...shift
0010 580      ScrollBit    Equ      00010000B      ;...s
0008 581      AltBit       Equ      00001000B      ;...t
0004 582      ControlBit   Equ      00000100B      ;...a
0002 583      LeftShiftBit Equ      00000010B      ;...t
0001 584      RightShiftBit Equ      00000001B      ;...e
0008 585      HoldBit      Equ      00001000B      ;Hold state
0080 586      BreakBit     Equ      10000000B      ;Upper bit of scan
0080 587      BiosBit      Equ      10000000B      ;Break Key Flag
                588
                589      ; Scan Code Definitions Used By Routine
                590
0003 591      TwoKey       Equ      03H          ;Scan code for 2 key
0037 592      PrtScnKey    Equ      37H          ;Scan code for print screen key
0045 593      NumKey       Equ      45H          ;Scan code for num lock key
0046 594      ScrollKey    Equ      46H          ;Scan code for scroll lock key
0047 595      HomeKey      Equ      47H          ;Scan code for home key
0052 596      InsKey       Equ      52H          ;Scan code for insert key
0053 597      DelKey       Equ      53H          ;Scan code for delete key
001A 598      ControlZ     Equ      1AH          ;Ascii code for control Z
                599
                600
                601
                602      $Eject

```

LOC OBJ

LINE

SOURCE

```

603 ; *****
604 ; * Keyboard Map *
605 ; *****
606 ;
607 ; Scan Lower Upper Control Alt Table
608 ; Code Case Case Case Case Entry
609 ; ---
610 ; 01 1BXX ESC 1BXX 1B 1BXX ESC NO VALUE 37
611 ; 02 31XX 1 21XX ! NO VALUE 0078 ALT 1 2E
612 ; 03 32XX 2 40XX @ 0003 NUL 0079 ALT 2 20
613 ; 04 33XX 3 23XX # NO VALUE 007A ALT 3 2F
614 ; 05 34XX 4 24XX $ NO VALUE 007B ALT 4 30
615 ; 06 35XX 5 25XX Percent NO VALUE 007C ALT 5 31
616 ; 07 36XX 6 5EXX ^ 1EXX RS 007D ALT 6 21
617 ; 08 37XX 7 26XX & NO VALUE 007E ALT 7 32
618 ; 09 38XX 8 2AXX * NO VALUE 007F ALT 8 33
619 ; 0A 39XX 9 28XX ( NO VALUE 0080 ALT 9 34
620 ; 0B 30XX 0 29XX ) NO VALUE 0081 ALT 0 35
621 ; 0C 2DXX - 5FXX _ 1FXX US 0082 ALT - 22
622 ; 0D 3DXX = 2BXX + NO VALUE 0083 ALT = 36
623 ; 0E 08XX BS 08XX BS 7FXX DEL NO VALUE 38
624 ; 0F 09XX TAB 000F BK TB NO VALUE NO VALUE 3E
625 ; 10 71XX q 51XX Q 11XX DC1 0010 ALT q 11
626 ; 11 77XX w 57XX W 17XX ETB 0011 ALT w 17
627 ; 12 75XX e 55XX E 05XX ENG 0012 ALT e 05
628 ; 13 72XX r 52XX R 12XX DC2 0013 ALT r 12
629 ; 14 74XX t 54XX T 14XX DC4 0014 ALT t 14
630 ; 15 79XX y 59XX Y 19XX EM 0015 ALT y 19
631 ; 16 75XX u 55XX U 15XX NAK 0016 ALT u 15
632 ; 17 6AXX i 4AXX I 09XX HT 0017 ALT i 09
633 ; 18 6FXX o 4FXX O 0FXX SI 0018 ALT o 0F
634 ; 19 70XX p 50XX P 10XX DLE 0019 ALT p 10
635 ; 1A 5BXX [ 7BXX { 1BXX ESC NO VALUE 39
636 ; 1B 5DXX ] 7DXX } 1DXX GS NO VALUE 3A
637 ; 1C 0DXX CR 0DXX CR 0AXX LF NO VALUE 3B
638 ; 1D -- CTRL -- CTRL -- CTRL 84 1000100
639 ; 1E 61XX a 41XX A 01XX SOH 001E ALT a 01
640 ; 1F 73XX s 53XX S 13XX DC3 001F ALT s 13
641 ; 20 64XX d 44XX D 04XX EOT 0020 ALT d 04
642 ; 21 66XX f 46XX F 06XX ACK 0021 ALT f 06
643 ; 22 67XX g 47XX G 07XX BEL 0022 ALT g 07
644 ; 23 68XX h 48XX H 08XX BS 0023 ALT h 08
645 ; 24 6AXX j 4AXX J 0AXX LF 0024 ALT j 0A
646 ; 25 6BXX k 4BXX K 0BXX VT 0025 ALT k 0B
647 ; 26 6CXX l 4CXX L 0CXX FF 0026 ALT l 0C
648 ; 27 3BXX ; 3AXX : NO VALUE NO VALUE 3F
649 ; 28 27XX ' 22XX " NO VALUE NO VALUE 40
650 ; 29 60XX ` 7EXX ~ NO VALUE NO VALUE 41
651 ; 2A -- SHIFT -- SHIFT -- SHIFT 82 1000010
652 ; 2B 5CXX \ 7CXX | 1CXX FS NO VALUE 3C
653 ; 2C 7AXX z 58XX Z 1AXX SUB 002C ALT z 1A
654 ; 2D 78XX x 58XX X 18XX CAN 002D ALT x 18
655 ; 2E 63XX c 43XX C 03XX ETX 002E ALT c 03
656 ; 2F 76XX v 56XX V 16XX SYN 002F ALT v 16
657 ; 30 62XX b 42XX B 02XX STX 0030 ALT b 02
658 ; 31 6EXX n 4EXX N 0EXX SO 0031 ALT n 0E
659 ; 32 6DXX m 4DXX M 0DXX CR 0032 ALT m 0D
660 ; 33 2CXX , 3CXX ( NO VALUE NO VALUE 42
661 ; 34 2EXX . 3EXX ) NO VALUE NO VALUE 43
662 ; 35 2FXX / 3FXX ? NO VALUE NO VALUE 44
663 ; 36 -- SHIFT -- SHIFT -- SHIFT 81 1000001
664 ; 37 2AXX * SPCL PRT SC 0072 PR ECH NO VALUE 3D
665 ; 38 -- ALT -- ALT -- ALT 88 10001000
666 ; 39 20XX SPACE 20XX SPACE 20XX SPACE 20XX SPACE 2D
667 ; 3A -- CAP LK -- CAP LK -- CAP LK C0 11000000
668 ; 3B 003B F1 0054 SHF F1 005E CTL F1 0068 ALT F1 23
669 ; 3C 003C F2 0055 SHF F2 005F CTL F2 0069 ALT F2 24
670 ; 3D 003D F3 0056 SHF F3 0060 CTL F3 006A ALT F3 25
671 ; 3E 003E F4 0057 SHF F4 0061 CTL F4 006B ALT F4 26
672 ; 3F 003F F5 0058 SHF F5 0062 CTL F5 006C ALT F5 27
673 ; 40 0040 F6 0059 SHF F6 0063 CTL F6 006D ALT F6 28
674 ; 41 0041 F7 005A SHF F7 0064 CTL F7 006E ALT F7 29
675 ; 42 0042 F8 005B SHF F8 0065 CTL F8 006F ALT F8 2A
676 ; 43 0043 F9 005C SHF F9 0066 CTL F9 0070 ALT F9 2B
677 ; 44 0044 F10 005D SHF F10 0067 CTL F10 0071 ALT F10 2C
    
```

LOC OBJ

LINE

SOURCE

```
678 ;45 -- NUM LK -- NUM LK -- NUM LK -- NUM LK 00 10100000
679 ;46 -- SCL LK -- SCL LK -- SCL LK -- SCL LK 90 10010000
680 $Eject
```

```

LOC OBJ          LINE    SOURCE
681              ;          *****
682              ;          * Keyboard Map By Value *
683              ;          *****
684              ;
685              ;          Directly Encodable Values
686
687      ;Scan Lower Upper      Control      Alt      Table
688      ;Code Case Case      Case      Case      Entry
689      ;-----
690      ;1E 61XX a      41XX A      01XX SOH      001E ALT a      01
691      ;30 62XX b      42XX B      02XX STX      0030 ALT b      02
692      ;2E 63XX c      43XX C      03XX ETX      002E ALT c      03
693      ;20 64XX d      44XX D      04XX EOT      0020 ALT d      04
694      ;12 75XX e      55XX E      05XX ENQ      0012 ALT e      05
695      ;21 66XX f      46XX F      06XX ACK      0021 ALT f      06
696      ;22 67XX g      47XX G      07XX BEL      0022 ALT g      07
697      ;23 68XX h      48XX H      08XX BS      0023 ALT h      08
698      ;17 6AXX i      4AXX I      09XX HT      0017 ALT i      09
699      ;24 6AXX j      4AXX J      0AXX LF      0024 ALT j      0A
700      ;25 6BXX k      4BXX K      0BXX VT      0022 ALT k      0B
701      ;26 6CXX l      4CXX L      0CXX FF      0026 ALT l      0C
702      ;32 6DXX m      4DXX M      0DXX CR      0032 ALT m      0D
703      ;31 6EXX n      4EXX N      0EXX SO      0031 ALT n      0E
704      ;18 6FXX o      4FXX O      0FXX SI      0018 ALT o      0F
705      ;19 70XX p      50XX P      10XX DLE      0019 ALT p      10
706      ;10 71XX q      51XX Q      11XX DC1      0010 ALT q      11
707      ;13 72XX r      52XX R      12XX DC2      0013 ALT r      12
708      ;1F 73XX s      53XX S      13XX DC3      001F ALT s      13
709      ;14 74XX t      54XX T      14XX DC4      0014 ALT t      14
710      ;16 75XX u      55XX U      15XX NAK      0016 ALT u      15
711      ;2F 76XX v      56XX V      16XX SYN      002F ALT v      16
712      ;11 77XX w      57XX W      17XX ETB      0011 ALT w      17
713      ;2D 78XX x      58XX X      18XX CAN      002D ALT x      18
714      ;15 79XX y      59XX Y      19XX EM      0015 ALT y      19
715      ;2C 7AXX z      5AXX Z      1AXX SUB      002C ALT z      1A
716      $Eject

```

```

LOC OBJ          LINE    SOURCE
717
718              ;          Level Two Values
719
720              ;Scan Lower Upper      Control      Alt          Table
721              ;Code Case Case      Case          Case          Entry
722              -----
723              ;03 32XX 2      40XX @      0003 NUL      0079 ALT 2      20
724              ;07 36XX 6      5EXX ^      1EXX RS      007D ALT 6      21
725              ;0C 20XX -      5FXX        1FXX US      0082 ALT -      22
726              ;3B 003B F1     0054 SHF F1  005E CTL F1   0068 ALT F1     23
727              ;3C 003C F2     0055 SHF F2  005F CTL F2   0069 ALT F2     24
728              ;3D 003D F3     0056 SHF F3  0060 CTL F3   006A ALT F3     25
729              ;3E 003E F4     0057 SHF F4  0061 CTL F4   006B ALT F4     26
730              ;3F 003F F5     0058 SHF F5  0062 CTL F5   006C ALT F5     27
731              ;40 0040 F6     0059 SHF F6  0063 CTL F6   006D ALT F6     28
732              ;41 0041 F7     005A SHF F7  0064 CTL F7   006E ALT F7     29
733              ;42 0042 F8     005B SHF F8  0065 CTL F8   006F ALT F8     2A
734              ;43 0042 F9     005C SHF F9  0066 CTL F9   0070 ALT F9     2B
735              ;44 0044 F10    005D SHF F10 0067 CTL F10 0071 ALT F10    2C
736              ;39 20XX SPACE 20XX SPACE  20XX SPACE    20XX SPACE     2D
737              ;02 31XX 1      21XX !      NO VALUE      0078 ALT 1      2E
738              ;04 33XX 3      23XX #      NO VALUE      007A ALT 3      2F
739              ;05 34XX 4      24XX $      NO VALUE      007B ALT 4      30
740              ;06 35XX 5      25XX Percent NO VALUE      007C ALT 5      31
741              ;08 37XX 7      26XX &      NO VALUE      007E ALT 7      32
742              ;09 38XX 8      2AXX *      NO VALUE      007F ALT 8      33
743              ;0A 39XX 9      2BXX (      NO VALUE      0080 ALT 9      34
744              ;0B 30XX 0      29XX )      NO VALUE      0081 ALT 0      35
745              ;0D 3DXX =      2BXX +      NO VALUE      0083 ALT =      36
746              ;01 1BXX ESC    1BXX 1B     1BXX ESC     NO VALUE      37
747              ;0E 08XX BS     08XX BS     7FXX DEL     NO VALUE      38
748              ;1A 5BXX [      7BXX {      1BXX ESC     NO VALUE      39
749              ;1B 5DXX ]      7DXX }      1DXX GS      NO VALUE      3A
750              ;1C 0DXX CR     0DXX CR     0AXX LF      NO VALUE      3B
751              ;2B 5CXX \      7CXX |      1CXX FS      NO VALUE      3C
752              ;37 2AXX *      SPCL PRT SC 0072 PR ECH  NO VALUE      3D
753              ;0F 09XX TAB    000F BK TB  NO VALUE      NO VALUE      3E
754              ;27 3BXX ;      3AXX :      NO VALUE      NO VALUE      3F
755              ;28 27XX '      22XX "      NO VALUE      NO VALUE      40
756              ;29 60XX `      7EXX ~      NO VALUE      NO VALUE      41
757              ;33 2CXX ,      3CXX (      NO VALUE      NO VALUE      42
758              ;34 2EXX .      3EXX )      NO VALUE      NO VALUE      43
759              ;35 2FXX /      3FXX ?      NO VALUE      NO VALUE      44
760
761              ;          Shift Keys
762
763              ;Scan Lower Upper      Control      Alt          Table
764              ;Code Case Case      Case          Case          Entry
765              -----
766              ;36 -- SHIFT -- SHIFT -- SHIFT -- SHIFT 81 10000001
767              ;2A -- SHIFT -- SHIFT -- SHIFT -- SHIFT 82 10000010
768              ;1D -- CTRL  -- CTRL  -- CTRL  -- CTRL  84 10000100
769              ;38 -- ALT   -- ALT   -- ALT   -- ALT   88 10001000
770              ;46 -- SCL LK -- SCL LK -- SCL LK -- SCL LK 90 10010000
771              ;45 -- NUM LK -- NUM LK -- NUM LK -- NUM LK A0 10100000
772              ;3A -- CAP LK -- CAP LK -- CAP LK -- CAP LK C0 11000000
773              $Eject

```

```
LOC OBJ          LINE    SOURCE
                774
                775          ;          *****
                776          ;          * Data Segments *
                777          ;          *****
                778
-----          779      BiosDataArea  Segment Public
                780          Extrn KeyboardFlag1:Byte, KeyboardFlag2:Byte
                781          Extrn AltInput:Byte, KeyBufHead:Word, KeyBufTail:Word
                782          Extrn KeyBuffer:Byte, ResetFlag:Word, BufferStart:Word
                783          Extrn BufferEnd:Word, BiosBreak:Byte
                784          Extrn CrtMode:Byte, CrtModeSet:Byte
-----          785      BiosDataArea  Ends
                786          $Eject
```


LOC	OBJ	LINE	SOURCE
0052		862	Db 0E3H,0E4H,0E5H,0E6H,0E7H,020H,005H,005H
005A		863	Db 005H,005H,005H,005H,005H,005H,005H,01BH
0062		864	Db 07FH,01BH,01DH,00AH,01CH,0F2H,005H,005H
006A		865	Db 005H,005H,005H,005H,005H
		866	
		867	; ALT Case Tables
		868	
006F		869	ALTCASE Db 0F9H,0FDH,002H,0E8H,0E9H,0EAH,0EBH,0ECH
0077		870	Db 0EDH,0EEH,0EFH,0F0H,0F1H,020H,0F8H,0FAH
007F		871	Db 0FBH,0FCH,0FEH,0FFH,000H,001H,003H,005H
0087		872	Db 005H,005H,005H,005H,005H,005H,005H,005H
008F		873	Db 005H,005H,005H,005H,005H
		874	
		875	LevelTwoStruc Ends
		876	
		877	\$Eject

```
LOC OBJ          LINE      SOURCE
                878      ;
                879      ;
                880      ;
                881      ;
                882      PadStruc   Struc
0000            883      Shift      Db      '789-456+1230.'
000D            884      Control    Db      0F7H,005H,004H,005H,0F3H,005H,0F4H,005H
0015            885      Control    Db      0F5H,005H,0F6H,005H,005H
001A            886      Base       Db      0C7H,0C8H,0C9H,02DH,0CBH,005H,0CDH,02BH
0022            887      Base       Db      0CFH,0D0H,0D1H,0D2H,0D3H
                888      PadStruc   Ends
                889
                890      ;
                891      ;
                892      ;
                893      ;
E885            894      Org        0E987H - (Size(LevelOneStruc) + Size(LevelTwoStruc) + Size(PadStruc))
                895
                896      $Nolist
                901
E986            902      LastTableByte Equ    $-1          ;This byte should be 0E986H
                903
                904      $Eject
```

LOC	OBJ	LINE	SOURCE
		905	*****
		906	;
		907	* Keyboard Driver Code *
		908	*****
		909	Assume Cs:Bios, Ds:BiosDataArea
E987		910	
		911	Org 0E987H ;Align with Pc and Xt
E987		912	
E987 FB		913	KeyboardHdwrInt Proc Far
E988 50		914	Sti ;Restore interrupts
E989 53		915	Push Ax ;Save all registers
E98A 51		916	Push Bx ;...R
E98B 52		917	Push Cx ;...e
E98C 56		918	Push Dx ;...g
E98D 57		919	Push Si ;...l
E98E 1E		920	Push Di ;...s
E98F 06		921	Push Ds ;...t
		922	Push Es ;...e
		923	;
		924	;
E990 FC		925	Cld ;Set string direction forward
E991 B8	R	926	Mov Ax,BiosDataArea ;Load data segment
E994 8ED8		927	Mov Ds,Ax ;...register with bios data
E996 E460		928	In Al,PortPPIPortA ;Get keyboard scan code
E998 50		929	Push Ax ;...and save it for later
E999 E461		930	In Al,PortPPIPortB ;Get B port data
E99B 50		931	Push Ax ;...and save it
E99C 0C80		932	Or Al,10000000B ;Set reset bit
E99E E661		933	Out PortPPIPortB,Al ;...clearing int rq f/f
E9A0 58		934	Pop Ax ;Restore B port value
E9A1 E661		935	Out PortPPIPortB,Al ;...and reset keyboard i/f
E9A3 58		936	Pop Ax ;Restore scan code
E9A4 8AE0		937	Mov Ah,Al ;...and make a copy in AH
		938	;
		939	; FF in scan code indicates an overrun
		940	;
E9A6 3CFF		941	Cmp Al,0FFH ;Is it overrun?
E9A8 7510		942	Jne KeyNotOverrun ;...jump if not
E9AA E94102		943	Jmp KeyBeepReturn ;...else beep and return
E9AD		944	KeyIntReturn:
E9AD B020		945	Mov Al,PICED0I ;Restore interrupts
E9AF E620		946	Out PortPICDCW2,Al ;...for further keyboard entry
E9B1		947	KeyReturnNoEOI:
E9B1 07		948	Pop Es ;Restore the
E9B2 1F		949	Pop Ds ;...R
E9B3 5F		950	Pop Di ;...e
E9B4 5E		951	Pop Si ;...g
E9B5 5A		952	Pop Dx ;...l
E9B6 59		953	Pop Cx ;...s
E9B7 5B		954	Pop Bx ;...t
E9B8 58		955	Pop Ax ;...e
E9B9 CF		956	Iret ;...r
		957	;
		958	;
E9BA		959	KeyNotOverrun:
E9BA 247F		960	And Al,07FH ;Remove shift bit
E9BC 3C46		961	Cmp Al,ScrollKey ;Keypad, or regular key?
E9BE 7603		962	Jbe InTable ;...jump if regular
E9C0 E92B01		963	Jmp KeyPadState ;...else process keypad
E9C3		964	InTable:
E9C3 BB85E8		965	Mov Bx,Offset LevelOneTable ;Point [Bx] to table
E9C6 2ED7		966	Xlat Cs:ByteType ;...and convert scan from table
E9C8 0AC0		967	Or Al,Al ;Is value shift code?
E9CA 7807		968	Js ModifierTest ;...jump if it is
E9CC 0AE4		969	Or Ah,Ah ;Is break bit on? (key release)
E9CE 78DD		970	Js KeyIntReturn ;...if it is, ignore and return
E9D0 EB4E90		971	Jmp KeyContinuel ;...else, process non-modifier
		972	;
		973	*****
		974	;
		975	* Process Modifier Keys *
		976	*****
		977	;
		978	Note: The insert key is both a modifier and a data key.
		979	It is processed in the data key section as a special case.

```

LOC OBJ                LINE    SOURCE
E9D3                    980    ModifierTest:
E9D3 247F               981      And    Al,7FH                ;Remove Modifier flag bit
E9D5 0AE4               982      Or     Ah,Ah                ;Test scan code if make/break
E9D7 7821               983      Js     BreakingModifier    ;...and jump if key released
                               984
                               985      ;      Modifier Key Being Pressed
                               986
E9D9 3C10               987      Cmp   Al,ScrollBit        ;Test if toggle type or normal
E9DB 7306               988      Jae   ToggleMake          ;...and jump if toggle
                               989
                               990      ;      Normal (not a toggle) Modifier
                               991
E9DD 08060000           E 992      Or     KeyboardFlag1,Al    ;...else set bit indicating
E9E1 EBCA               993      Jmp   KeyIntReturn        ;...a modifier is pressed
                               994
                               995      ;      Toggle Modifier
                               996
E9E3                    997    ToggleMake:
E9E3 F606000004         E 998      Test  KeyboardFlag1,ControlBit ;Do not toggle if control
E9E8 7536               999      Jnz   KeyContinue1        ;...key (ctl num lock, etc...)
E9EA 84060000           E 1000     Test  Al,KeyboardFlag2    ;Is the key depressed now?
E9EE 75BD               1001     Jnz   KeyIntReturn        ;...jump if it is and ignore
E9F0 08060000           E 1002     Or     KeyboardFlag2,Al    ;...else show dep. in flag 2
E9F4 30060000           E 1003     Xor   KeyboardFlag1,Al    ;Toggle state bit if flag 1
E9F8 EBB3               1004     Jmp   KeyIntReturn        ;...and return
                               1005
                               1006     ;      Modifier Key Being Released
                               1007
E9FA                    1008    BreakingModifier:
E9FA 3C10               1009     Cmp   Al,ScrollBit        ;Test if toggle type or normal
E9FC 731A               1010     Jae   ToggleBreak         ;...and jump if toggle
                               1011
                               1012     ;      Normal (not a toggle) Modifier - Breaking
                               1013
E9FE F6D0               1014     Not   Al                  ;Turn off key depressed
EA00 20060000           E 1015     And   KeyboardFlag1,Al    ;...bit
EA04 3CF7               1016     Cmp   Al,Not AltBit       ;(Value of alt key mask here)
EA06 75A5               1017     Jne   KeyIntReturn        ;Jump if not alt key release
                               1018
                               1019     ;      Return collected data on release of ALT key
                               1020
EA08 A00000             E 1021     Mov   Al,AltInput         ;Get the data
EA0B 32E4               1022     Xor   Ah,Ah               ;Get a zero
EA0D 88260000           E 1023     Mov   AltInput,Ah         ;...and zero data for next try
EA11 3AC4               1024     Cmp   Al,Ah               ;If input = 0, don't
EA13 7498               1025     Je    KeyIntReturn        ;...return any data
EA15 E97701             1026     Jmp   KeyBufLoad          ;...else return data w/scan=0
                               1027
                               1028     ;      Toggle Modifier - Breaking
                               1029
EA18                    1030    ToggleBreak:
EA18 F6D0               1031     Not   Al                  ;Turn off toggle key
EA1A 20060000           E 1032     And   KeyboardFlag2,Al    ;...depressed bit
EA1E EB8D               1033     Jmp   KeyIntReturn        ;...and return
                               1034
                               1035     ;      *****
                               1036     ;      * Test Special Conditions *
                               1037     ;      *****
                               1038
EA20                    1039    KeyContinue1:
                               1040     ;      Control Num Lock Hold State Test
                               1041
EA20 F606000008         E 1042     Test  KeyboardFlag2,HoldBit ;Is hold state bit set?
EA25 740D               1043     Jz    KeyContinue4        ;...jump if not
                               1044
                               1045     ;      Restore from hold state
                               1046
EA27 80FC45             1047     Cmp   Ah,NumKey           ;Is this control num?
EA2A 7405               1048     Je    HoldIntReturn       ;...if so, don't restore
EA2C 80260000F7         E 1049     And   KeyboardFlag2,Not HoldBit ;...else clear hold state
EA31                    1050    HoldIntReturn:
EA31 E979FF             1051     Jmp   KeyIntReturn        ;...and return
                               1052
                               1053     ;      *****
                               1054     ;      * What Shift State are We In? *

```

```

LOC OBJ          LINE    SOURCE
                1055    ;
                1056    ; *****
                1057    ;     Ah = Scan Code, Al = Value From Level 1 Table
                1058
EA34            1059    KeyContinue4:
EA34 F606000008  E    1060    Test   KeyboardFlag1,AltBit      ;Alt?
EA39 7514            1061    Jnz   AltState                 ;...Yes, jump
EA3B F606000004  E    1062    Test   KeyboardFlag1,ControlBit ;Control?
EA40 7520            1063    Jnz   ControlState            ;...Yes, jump, else test shift
EA42 F606000003  E    1064    Test   KeyboardFlag1,LeftShiftBit Or RightShiftBit
EA47 7503            1065    Jnz   ShiftStateShort         ;...Yes, jump
EA49 E90F00        1066    Jmp   BaseCaseState           ;No shift state active
EA4C            1067    ShiftStateShort:
EA4C EB6C90        1068    Jmp   ShiftState              ;Convert to near jump
                1069
                1070
                1071    ; *****
                1072    ; * Alt State Processor *
                1073    ; *****
                1074
EA4F            1075    AltState:
EA4F 3C1A            1076    Cmp   Al,ControlZ             ;Is Level 1 (= Control Z?
EA51 7705            1077    Ja   AltNotAZ                ;...jump if not and continue
EA53 B000            1078    Mov   Al,0                    ;...else return extended
EA55 E97201        1079    Jmp   KeyBufLoad2            ;...Al = 0, Ah = Scan
EA58            1080    AltNotAZ:
EA58 B83BE9        1081    Mov   Bx,Offset LevelTwoTable. ;Load Level Two Pointer
EA5B 2C20            1082    Sub   Al,20H                  ;Remove level two char offset
EA5D 2ED7            1083    Xlat  Cs:ByteType             ;Translate level 1 to level 2
EA5F E92D01        1084    Jmp   KeyBufLoad             ;Jump and load buffer
                1085
                1086    ; *****
                1087    ; * Control State Processor *
                1088    ; *****
                1089
                1090    ;     Process specials
                1091
                1092    ;     Control Break?
EA62            1093    ControlState:
EA62 80FC46        1094    Cmp   Ah,ScrollKey           ;Are we processing Scroll Key?
EA65 7515            1095    Jnz   KeyContinue5           ;...jump if not
EA67 C606000008  E    1096    Mov   BiosBreak,BiosBit      ;Turn on Bios Break Bit
EA6C A10000        E    1097    Mov   Ax,BufferStart         ;Clear keyboard buffer
EA6F A30000        E    1098    Mov   KeyBufTail,Ax          ;...
EA72 A30000        E    1099    Mov   KeyBufHead,Ax          ;...
EA75 CD1B            1100    Int   TrapKeyBreak           ;Trap to keyboard break driver
EA77 2BC0            1101    Sub   Ax,Ax                   ;...Incase of return
EA79 E91301        1102    Jmp   KeyBufLoad             ;...send some data back
                1103
                1104    ;     Set Hold State?
                1105
EA7C            1106    KeyContinue5:
EA7C 80FC45        1107    Cmp   Ah,NumKey              ;Are we processing Num key?
EA7F 7521            1108    Jnz   KeyContinue3           ;...jump if not
EA81 800E000008  E    1109    Or   KeyboardFlag2,HoldBit   ;...else set hold bit
EA86 B020            1110    Mov   Al,PIC0I               ;Restore interrupts
EA88 E620            1111    Out  PortPICOCW2,Al          ;...for futher keyboard entry
EA8A 803E000007  E    1112    Cmp   CrtMode,7              ;Test if monochrome adapter
EA8F 7407            1113    Je   HoldLoop                ;...and jump if it is
EA91 BAD003        1114    Mov   Dx,PortColorMode       ;...else, turn color card
EA94 A00000        E    1115    Mov   Al,CrtModeSet          ;...on during keyboard pause
EA97 EE            1116    Out  Dx,Al                   ;...so user can see screen
EA98            1117    HoldLoop:
EA98 F606000008  E    1118    Test  KeyboardFlag2,HoldBit   ;Wait for recursive interrupt
EA9D 75F9            1119    Jnz   HoldLoop               ;...to clear flag
EA9F E90FFF        1120    Jmp   KeyReturnNoEOI         ;...and return
                1121
                1122    ;     Null?
EA92            1123    KeyContinue3:
EA92 80FC03        1124    Cmp   Ah,TwoKey              ;Control 2?
EA95 7505            1125    Jne   KeyContinue7           ;...jump if not
EA97 B000            1126    Mov   Al,0                    ;...else return a null
EA99            1127    CtlBufLoad:
EA99 E91E01        1128    Jmp   KeyBufLoad2           ;Jump and load buffer
                1129

```

```

LOC OBJ          LINE    SOURCE
                                     ;      Else process normal control key
EAC          1130
EAC          1131      KeyContinue7:      ;
EAC 3C1A      1132      Cmp      Al,ControlZ      ;Is Level 1 (= Control Z?)
EAE 76F9      1133      Jbe      CtlBufLoad      ;...return level 1 val. if so
EAB0 B816E9   1134      Mov      Bx,Offset LevelTwoTable.ControlCase ;Load level 2
EAB3 2C20      1135      Sub      Al,20H          ;Remove level two char offset
EAB5 2ED7      1136      Xlat     Cs:ByteType     ;Translate level 2 data into Al
EAB7 E9D500   1137      Jmp      KeyBufLoad      ;...and load into buffer
                                     ;
                                     ; *****
                                     ; * Shift State Processor *
                                     ; *****
EABA          1142      ShiftState:
EABA          1143
EABA          1144      ;      Handle Special Cases
EABA          1145
EAB8 80FC37   1146      Cmp      Ah,PrtScnKey    ;Are we processing print scn
EABD 7509      1147      Jnz      KeyContinue6    ;...jump if not
EABF B020      1148      Mov      Al,PIC0EI       ;Allow futher interrupts
EAC1 E620      1149      Out      PortPIC0CW2,Al  ;...and keystrokes
EAC3 CD05      1150      Int      TrapPrintScreen ;Perform print screen
EAC5 E9E9FE   1151      Jmp      KeyReturnNoEoi  ;...and return
EAC8          1152
EAC8          1153      KeyContinue6:
EAC8 3C1A      1154      Cmp      Al,ControlZ    ;Is Level 1 (= Control Z?)
EACA 7705      1155      Ja       ShiftNotAZ     ;...jump if not and continue
EACC 0440      1156      Add      Al,40H          ;...else return extended
EACE E9BE00   1157      Jmp      KeyBufLoad     ;...Al = L1+40H Ah = Scan
EAD1          1158      ShiftNotAZ:
EAD1 BBF1E8   1159      Mov      Bx,Offset LevelTwoTable.UpperCase ;Load level 2
EAD4 2C20      1160      Sub      Al,20H          ;Remove level two char offset
EAD6 2ED7      1161      Xlat     Cs:ByteType     ;Translate level 2 data into Al
EAD8 E9B400   1162      Jmp      KeyBufLoad     ;Jump and load buffer
EAD8          1163
EAD8          1164      ; *****
EAD8          1165      ; * Base Case Processor *
EAD8          1166      ; *****
EAD8          1167
EAD8          1168      BaseCaseState:
EAD8 3C1A      1169      Cmp      Al,ControlZ    ;Is Level 1 (= Control Z?)
EADD 7705      1170      Ja       BaseNotAZ      ;...jump if not and continue
EADF 0460      1171      Add      Al,60H          ;...else return extended
EAE1 E9AB00   1172      Jmp      KeyBufLoad     ;...Al = L1+60H Ah = Scan
EAE4          1173      BaseNotAZ:
EAE4 BBCCE8   1174      Mov      Bx,Offset LevelTwoTable.LowerCase ;Load level 2
EAE7 2C20      1175      Sub      Al,20H          ;Remove level two char offset
EAE9 2ED7      1176      Xlat     Cs:ByteType     ;Translate level 2 data into Al
EAEB E9A100   1177      Jmp      KeyBufLoad     ;Jump and load buffer
EAEB          1178
EAEB          1179      ; *****
EAEB          1180      ; * Keypad Processor *
EAEB          1181      ; *****
EAEB          1182
EAE          1183      KeypadState:
EAE 2C47      1184      Sub      Al,HomeKey      ;Calculate table offset
EAF0 8A1E0000  1185      Mov      Bl,KeyboardFlag1 ;Use Bl for faster action
EAF4 F6C308   1186      Test     Bl,AltBit       ;Alt?
EAF7 7519      1187      Jnz      PadALTState     ;...Yes, jump process numeric
EAF9 F6C304   1188      Test     Bl,ControlBit   ;Control?
EAFC 754B      1189      Jnz      PadControlState ;...Yes, jump, else test shift
EAFE F6C320   1190      Test     Bl,NumBit       ;Num Lock?
EB01 7407      1191      Jz       KeyPadContinue  ;...Jump if not
EB03 F6C303   1192      Test     Bl,LeftShiftBit Or RightShiftBit ;Shifted?
EB06 754D      1193      Jnz      PadBaseState    ;...Shifted numloc=BaseState
EB08 EB79      1194      Jmp      Short PadShiftState ;...Unshifted numloc=ShiftState
EB0A          1195      KeyPadContinue:
EB0A F6C303   1196      Test     Bl,LeftShiftBit Or RightShiftBit ;Shifted?
EB0D 7574      1197      Jnz      PadShiftState    ;...Yes, jump
EB0F EB4490   1198      Jmp      PadBaseState    ;No shift state active
EB12          1199      PadALTState:
EB12 0AE4      1200      Or       Ah,Ah           ;Affect flags
EB14 7830      1201      Js       PadReturn        ;...ignore if breaking
                                     ;
                                     ;      Test for Reset sequence
                                     ;
EAE          1202
EAE          1203
EAE          1204

```

LOC	OBJ	LINE	SOURCE
		1205	; Are we in the control and alt shift state?
		1206	
EB16	F606000004	E 1207	Test KeyboardFlag1,ControlBit ;Is control pressed?
EB1B	7410	1208	Jz KeyPadCont2 ;.....jump if not
EB1D	80FC53	1209	Cmp Ah,DelKey ;Are we processing DEL?
EB20	750B	1210	Jne KeyPadCont2 ;...jump if not
EB22	C70600003412	E 1211	Mov ResetFlag,1234H ;...else, set reset flag
EB28	EA0000----	E 1212	Jmp ResetEntry ;...and reset the system
EB2D		1213	KeyPadCont2:
EB2D	BB60E9	1214	Mov Bx,Offset PadTable.Shift ;Point at numeric translation
EB30	2ED7	1215	Xlat Cs:ByteType ;Translate from table
EB32	3C30	1216	Cmp Al,'0' ;Test if (0 (.-)
EB34	7210	1217	Jb PadReturn ;...and return null if it is
EB36	2C30	1218	Sub Al,'0' ;...else remove ascii offset
EB38	8AD8	1219	Mov Bl,Al ;Free up accumulator
EB3A	A00000	E 1220	Mov Al,AltInput ;Get keypad accumulator
EB3D	B40A	1221	Mov Ah,10 ;...and shift it left 1 decimal
EB3F	F6E4	1222	Mul Ah ;...then add
EB41	02C3	1223	Add Al,B1 ;...this number
EB43	A20000	E 1224	Mov AltInput,Al ;...in
EB46		1225	PadReturn:
EB46	E964FE	1226	Jmp KeyIntReturn ;...and return
		1227	
EB49		1228	PadControlState:
EB49	0AE4	1229	Or Ah,Ah ;Affect flags
EB4B	78F9	1230	Js PadReturn ;...ignore if breaking
EB4D	BB6DE9	1231	Mov Bx,Offset PadTable.Control ;Use control table
EB50	2ED7	1232	Xlat Cs:ByteType ;Translate from scan to data
EB52	EB3B90	1233	Jmp KeyBufLoad
EB55		1234	PadBaseState:
EB55	80FCD2	1235	Cmp Ah,InsKey Or 80H ;Breaking INS key?
EB58	7507	1236	Jne NotBrkIns ;...jump if not
EB5A	802600007F	E 1237	And KeyboardFlag2,Not InsBit ;Show INS key released
EB5F	EBE5	1238	Jmp PadReturn ;...and return
EB61		1239	NotBrkIns:
EB61	0AE4	1240	Or Ah,Ah ;Affect flags
EB63	78E1	1241	Js PadReturn ;...ignore if breaking
EB65	80FC52	1242	Cmp Ah,InsKey ;Special case of insert key
EB68	7511	1243	Jne NotInsKey ;...jump if not that key
EB6A	F606000080	E 1244	Test KeyboardFlag2,InsBit ;Is the insert key pressed now?
EB6F	75D5	1245	Jnz PadReturn ;...if it is, ignore repeat
EB71	8036000080	E 1246	Xor KeyboardFlag1,InsBit ;...else toggle Ins shift bit
EB76	800E000080	E 1247	Or KeyboardFlag2,InsBit ;...and show key depressed
		1248	...then go ahead and send code
EB7B		1249	NotInsKey:
EB7B	BB7AE9	1250	Mov Bx,Offset PadTable.Base ;Use base table
EB7E	2ED7	1251	Xlat Cs:ByteType ;Translate from scan to data
EB80	EB0D90	1252	Jmp KeyBufLoad
EB83		1253	PadShiftState:
EB83	0AE4	1254	Or Ah,Ah ;Affect flags
EB85	78BF	1255	Js PadReturn ;...ignore if breaking
EB87	BB60E9	1256	Mov Bx,Offset PadTable.Shift ;Use Shift table
EB8A	2ED7	1257	Xlat Cs:ByteType ;Translate from scan to data
EB8C	EB0190	1258	Jmp KeyBufLoad
		1259	\$Eject

```

LOC OBJ          LINE    SOURCE
                1260    ;
                1261    ;
                1262    ;
                1263    ;
                1264    ;
                1265    ;
                1266    ;
                1267    ;
                1268    ;
                1269    ;
                1270    ;
                1271    ;
                1272    ;
                1273    ;
                1274    ;
EB8F            1275    KeyBufLoad:
EB8F 3C05        1276    Cmp    Al,5                ;No value?
EB91 7458        1277    Jc    BufLoadReturn      ;...return and ignore
EB93 3C04        1278    Cmp    Al,4                ;Is value (= 4?
EB95 7704        1279    Ja    BufNotSpcl        ;...jump if not
EB97 0C80        1280    Or    Al,10000000B      ;Turn on upper bit
EB99 EB06        1281    Jmp    Short BufContinue ;...and continue below
EB9B            1282    BufNotSpcl:
EB9B A880        1283    Test   Al,10000000B     ;Is upper bit set?
EB9D 7406        1284    Jz    BufContinue2      ;...jump if not
EB9F 247F        1285    And   Al,01111111B     ;Turn off upper bit
EBA1            1286    BufContinue:
EBA1 8AE0        1287    Mov    Ah,Al             ;...and move it in as scan
EBA3 B000        1288    Mov    Al,0              ;Indicate extended code
                1289    ;
                1290    ;
                1291    ;
                1292    ;
EBA5            1292    BufContinue2:
EBA5 F606000040 E 1293    Test   KeyboardFlag1,CapsBit ;Are we in capslock?
EBA7 741E        1294    Jz    BufLoad            ;...jump if not
EBAC F606000003 E 1295    Test   KeyboardFlag1,LeftShiftBit Or RightShiftBit ;Shift?
EBB1 740D        1296    Jz    BufLowerToUpper    ;...jump if not, conv. lo to up
                1297    ;
                1298    ;
                1299    ;
EBB3 3C41        1300    Cmp    Al,'A'             ;)= A
EBB5 7213        1301    Jb    BufLoad            ;...jump if not and load buffer
EBB7 3C5A        1302    Cmp    Al,'Z'             ;)= Z
EBB9 770F        1303    Ja    BufLoad            ;...jump if not and load buffer
EBBB 0420        1304    Add   Al,'a'-'A'         ;Add offset to lower
EBBD EB0B90      1305    Jmp    BufLoad            ;...and go load buffer
                1306    ;
                1307    ;
                1308    ;
                1309    ;
EBC0            1309    BufLowerToUpper:
EBC0 3C61        1310    Cmp    Al,'a'             ;)= a
EBC2 7206        1311    Jb    BufLoad            ;...jump if not and load buffer
EBC4 3C7A        1312    Cmp    Al,'z'             ;)= z
EBC6 7702        1313    Ja    BufLoad            ;...jump if not and load buffer
EBC8 2C20        1314    Sub   Al,'a'-'A'         ;Add offset to lower
                1315    ;
                1316    ;
                1317    ;
                1318    ;
EBCA            1318    KeyBufLoad2:
EBCA            1319    BufLoad:
EBCA 8B1E0000    E 1320    Mov    Bx,KeyBufTail     ;Get the tail (target address)
EBCE 8BF8        1321    Mov    Di,Bx             ;Di points where we'll put char
EBD0 43          1322    Inc    Bx                ;Bump the tail
EBD1 43          1323    Inc    Bx                ;...
EBD2 3B1E0000    E 1324    Cmp    Bx,BufferEnd      ;Are we at the end?
EBD6 7504        1325    Jne   BufCont2          ;...jump if not
EBD8 8B1E0000    E 1326    Mov    Bx,BufferStart    ;...else, move the pointer
EBDC            1327    BufCont2:
EBDC 3B1E0000    E 1328    Cmp    Bx,KeyBufHead     ;Tail = Head?
EBE0 7503        1329    Jne   BufCont3          ;...jump if not
EBE2 EB0A90      1330    Jmp    KeyBeepReturn     ;...else beep and return
EBE5            1331    BufCont3:
EBE5 8905        1332    Mov    [Di],Ax           ;Store scan and ascii in buf.
EBE7 891E0000    E 1333    Mov    KeyBufTail,Bx     ;...move the pointer
EBEB            1334    BufLoadReturn:

```



```

LOC OBJ          LINE      SOURCE
EBEB E9BFFD      1335          Jmp      KeyIntReturn      ;...and return
                  1336
                  1337          ; *****
                  1338          ; * Keyboard Beep *
                  1339          ; *****
                  1340
EBEE             1341      KeyBeepReturn:
EBEE B020        1342          Mov      Al,PICEDI          ;Restore interrupts
EBF0 E620        1343          Out     PortPICOCW2,Al     ;...and futher keyboard
EBF2 B88000      1344          Mov     Bx,128             ;Short tone, same as PC
EBF5 E461        1345          In     Al,PortPPIPortB    ;Get audio control state
EBF7 50          1346          Push   Ax                 ;...and save it
EBF8             1347
EBF8 24FC        1348          And     Al,01111100B      ;Turn off gate
EBFA E661        1349          Out     PortPPIPortB,Al   ;...and data
EBFC             1350
EBFC B96400      1351          Mov     Cx,100            ;Half cycle time
EBFF             1352
EBFF E2FE        1353          Loop   BeepLoop           ;...loop for half a cycle
EC01 3402        1354          Xor     Al,0000010B      ;Turn on speaker
EC03 E661        1355          Out     PortPPIPortB,Al   ;...speaker data
EC05 A802        1356          Test   Al,0000010B      ;Is this second pass?
EC07 74F3        1357          Jz     BeepHalfCycle     ;...loop if not
EC09 4B          1358          Dec    Bx                 ;Decrement cycle count
EC0A 75EC        1359          Jnz   BeepCycle          ;...and cycle till done
EC0C 58          1360          Pop    Ax                 ;Restore B port value
EC0D E661        1361          Out     PortPPIPortB,Al   ;...from stack and output it
EC0F B93200      1362          Mov     Cx,50            ;Pause at end of beep
EC12             1363
EC12 E2FE        1364          Loop   PauseBeep         ;...for tonal quality
EC14 E99AFD      1365          Jmp     KeyReturnNoEDI    ;...and return
                  1366
                  1367      KeyboardHdwrInt Endp
                  1368
                  1369      Bios      Ends
                  1370      End

```

ASSEMBLY COMPLETE, NO ERRORS FOUND

IBM PC/XT 8086/8087/8088 MACRO ASSEMBLER V1.1 ASSEMBLY OF MODULE FLOPPY
OBJECT MODULE PLACED IN FLOPPY.OBJ
ASSEMBLER INVOKED BY: ASM86 FLOPPY.SRC

```
LOC OBJ          LINE    SOURCE
                1 +1    $Title ('DTC/PC BIOS Floppy Disk Driver V1.0')
                2 +1    $Pagelength (80) Pagewidth (132) Debug Nogen
                3        Name Floppy
                4
                5
                6        ;    Author:      Don K. Harrison
                7
                8        ;    Start date:  November 17, 1983      Last edit:  December 22, 1983
                9
               10
               11        ;
               12        ;          *****
               13        ;          * Module Description *
               14        ;          *****
               15        ;
               16        ;          This module contains the floppy disk driver routines and
               17        ;          the floppy disk interrupt service routines.  The driver is
               18        ;          accessed via interrupt 14 (TrapFDisk).  The interrupt service
               19        ;          is via interrupt 19 (TrapFDDriver).
               20
               21
               22        ;          (c) Display Telecommunications Corporation, 1983
               23        ;          All Rights Reserved
               24
               25    $Eject
```

LOC	OBJ	LINE	SOURCE
		26	
		27	
		28	;
		29	;
		30	*****
		31	* Revision History *
		32	*****
		33	
		34	
		35	
		36	
		37	
		38	\$Eject

```

LOC OBJ          LINE      SOURCE
                39
                40          ;
                41          ; *****
                42          ; * Public Symbols *
                43          ; *****
                44          Public FloppyDriver, FloppyHdwrInt, FloppyParamsPointer
                45
                46
                47          ; *****
                48          ; * Equates *
                49          ; *****
                50
                51          ; *****
                52          ; * Stack Frame Definitions *
                53          ; *****
                54
000C[]          55          PointerSegment Equ [BP+12]
000A[]          56          PointerOffset Equ [BP+10]
0009[]          57          DTL Equ Byte Ptr [BP+9]
0008[]          58          GapSize Equ Byte Ptr [BP+8]
0007[]          59          EDT Equ Byte Ptr [BP+7]
0006[]          60          SectorSize Equ Byte Ptr [BP+6]
0005[]          61          SectorNumber Equ Byte Ptr [BP+5]
0004[]          62          HeadNumber Equ Byte Ptr [BP+4]
0003[]          63          TrackNumber Equ Byte Ptr [BP+3]
0002[]          64          DriveNumber Equ Byte Ptr [BP+2]
0001[]          65          FDCCOMMAND Equ Byte Ptr [BP+1]
0000[]          66          NumberOfSectors Equ Byte Ptr [BP]
                67
                68          ; *****
                69          ; * Disk Parameter Definitions *
                70          ; *****
                71
0000:[]        72          Specify1 Equ Es:Byte Ptr [Si]
0001:[]        73          Specify2 Equ Es:Byte Ptr [Si+1]
0002:[]        74          MotorWait Equ Es:Byte Ptr [Si+2]
0007:[]        75          FormatGap Equ Es:Byte Ptr [Si+7]
0008:[]        76          FormatFiller Equ Es:Byte Ptr [Si+8]
0009:[]        77          HeadSettle Equ Es:Byte Ptr [Si+9]
000A:[]        78          MotorStart Equ Es:Byte Ptr [Si+10]
0000:[]        79          SoftParams Equ Es:Byte Ptr [Si]
                80
                81          ; *****
                82          ; * Status Bits *
                83          ; *****
                84
0000           85          DkStat Equ 00000000B ;Dk status
0000           86          TimeOutStat Equ 10000000B ;Timeout
0040           87          BadSeekStat Equ 01000000B ;Seek error
0020           88          BadFDCStat Equ 00100000B ;Floppy controller bad
0010           89          BadCrcStat Equ 00010000B ;CRC error
0009           90          DMAPageStat Equ 00001001B ;Dma requested to cross 64k
0008           91          BadDMAStat Equ 00001000B ;Dma probably bad
0004           92          SectNotFndStat Equ 00000100B ;Sector not found
0003           93          WriteProtStat Equ 00000011B ;Disk write protected
0002           94          BadAddMarkStat Equ 00000010B ;Address mark not found
0001           95          BadCmdStat Equ 00000001B ;Unrecognizable command
                96
                97          ; *****
                98          ; * Modes *
                99          ; *****
                100
0042           101         DMAVerifyMode Equ 01000010B ;Channel 2 verify normal
0046           102         DMAReadMode Equ 01000110B ;Channel 2 read normal
004A           103         DMAWriteMode Equ 01001010B ;Channel 2 write normal
                104
                105          ; *****
                106          ; * Commands *
                107          ; *****
                108
00E6           109         FDCReadCMD Equ 11100110B ;Read command
00C5           110         FDCWriteCMD Equ 11000101B ;Write command
004D           111         FDCFormatCMD Equ 01001101B ;Format command
0007           112         FDCRecalCMD Equ 00000111B ;Recalibrate command
0003           113         FDCSpecifyCMD Equ 00000011B ;Specify command (timers)
    
```

LOC	OBJ	LINE	SOURCE
0008		114	FDCSenseIntCMD Equ 00001000B ;Sense interrupt command
000F		115	FDCSeekCMD Equ 00001111B ;Seek command
		116	
		117	\$Include (IbmInc)
=1		118	;
=1		119	;
=1		120	;
=1		121	;
		641	\$Nolist
			\$Eject

LOC	OBJ	LINE	SOURCE
		642	
		643	;
		644	; * Data Segments *
		645	;
		646	;
----		647	IntSegment Segment Public
		648	Extrn FloppyParamsTrapAddr:Dword
----		649	IntSegment Ends
		650	
----		651	BiosStack Segment Public
----		652	BiosStack Ends
		653	
----		654	BiosDataArea Segment Public
		655	Extrn FDCStatus:Byte, DisketteStat:Byte, SeekStatus:Byte
		656	Extrn MotorStatus:Byte, MotorCount:Byte
----		657	BiosDataArea Ends
		658	\$Eject

```

LOC OBJ          LINE    SOURCE
659              ;          *****
660              ;          * Stack Frame After Initialization *
661              ;          *****
662
663
664              ;          -----
665              ;          | Return Address | BP+22
666              ;          |-----|
667              ;          | BP | BP+20
668              ;          |-----|
669              ;          | SI | BP+18
670              ;          |-----|
671              ;          | DI | BP+16
672              ;          |-----|
673              ;          | DS | BP+14
674              ;          |-----|
675              ;          | DMA Memory Pointer (Segment) | BP+12
676              ;          |-----|
677              ;          | DMA Memory Pointer (Offset ) | BP+10
678              ;          |-----|
679              ;          | DTL | GAP | BP+8
680              ;          |-----|
681              ;          | EOT | Sector Size | BP+6
682              ;          |-----|
683              ;          | Sector # | Head # | BP+4
684              ;          |-----|
685              ;          | Track # | Drive # | BP+2
686              ;          |-----|
687              ;          | Command | # of Sectors | (- BP
688              ;          |-----|
689              ;          $Eject
    
```

```

LOC OBJ                LINE    SOURCE
-----
                                690      ;
                                691      * Code Segment *
                                692      ;
                                693      ;
                                694      Bios      Segment Common
                                695      Assume  Cs:Bios, Ds:BiosDataArea, Ss:BiosStack
EC59                    696      Org      0EC59H      ;Align with PC / XT
                                697
                                698      ;      Initialization of stack frame
EC59                    699      FloppyDriver Proc Far
EC59 FB                 700      StI                    ;Restore higher interrupts
EC5A 55                 701      Push Bp                ;Save Block pointers Bp
EC5B 56                 702      Push Si                ;...Si
EC5C 57                 703      Push Di                ;...Di
EC5D 1E                 704      Push Ds                ;...Data segment
EC5E 06                 705      Push Es                ;...Memory pointer SEG
EC5F 53                 706      Push Bx                ;.....OFFSET
EC60 8BF8              707      Mov Di,Ax              ;Save accumulator
                                708
                                709      ;      Point ES:SI at disk parameters
                                710
                                711      Assume Ds:IntSegment ;DS refers to segment at zero
EC62 33C0              712      Xor Ax,Ax              ;Fill Command block
EC64 8ED8              713      Mov Ds,Ax              ;...from data
EC66 C4360000         E 714      Les Si,FloppyParamsTrapAddr ;...pointed to by int 30
                                715      Assume Ds:BiosDataArea ;Tell ASM86 to use bios data
EC6A 8A-----        R 716      Mov Ax,BiosDataArea    ;...set segment register
EC6D 8ED8              717      Mov Ds,Ax              ;...to bios data
                                718
                                719      ;      Move values from disk parameters
                                720
EC6F 8B0500           721      Mov Bx,5                ;...starting at byte 5
EC72 268B00           722      Mov Ax,Es:[Bx][Si]     ;Load ax
EC75 50               723      Push Ax                ;...and push (DTL & GAP)
EC76 4B               724      Dec Bx                  ;Decrement index
EC77 4B               725      Dec Bx                  ;...by 2 and
EC78 268B00           726      Mov Ax,Es:[Bx][Si]     ;Load ax
EC7B 50               727      Push Ax                ;...and push (EOT & Sec Sz)
EC7C 86CE             728      Xchg C1,Dh              ;Re-arrange for
EC7E 86D1             729      Xchg D1,C1              ;...command block
EC80 52               730      Push Dx                 ;Save C1, Dh (Sec# & HD#)
EC81 51               731      Push Cx                 ;Save Ch, D1 (TK# & DV#)
EC82 57               732      Push Di                 ;Save Ah, A1 (Cmd & #Sectors)
EC83 8BEC             733      Mov Bp,Sp               ;Point at base of stack frame
                                734
                                735      ;      Parse and dispatch command
                                736
EC85 E82200           737      Call ParseAndGo         ;Parse command and execute
                                738
                                739      ;      Set the motor timer and affect carry with status
                                740
EC88 268A6402         E 741      Mov Ah,MotorWait        ;Plug timer to keep
EC8C 8A260000         E 742      Mov MotorCount,Ah       ;...drives running
EC90 8A260000         E 743      Mov Ah,DisketteStat     ;Affect carry flag
EC94 80FC01           744      Cmp Ah,1                ;if >1, carry will be clear
EC97 F5               745      Cmc                     ;...not, if >1, carry set
                                746
                                747      ;      Dissolve stack frame
                                748
EC98 5B               749      Pop Bx                  ;Pop Ah,A1 (Cmd & #Sectors)
EC99 59               750      Pop Cx                  ;Pop TK# & DV#
EC9A 5A               751      Pop Dx                  ;Pop SEC# & HD#
EC9B 86D1             752      Xchg D1,C1              ;Restore back to
EC9D 86CE             753      Xchg C1,Dh              ;...calling convention
EC9F 5B               754      Pop Bx                  ;Pop EOT & SEC SIZE
ECA0 5B               755      Pop Bx                  ;Pop DTL & GAP
ECA1 5B               756      Pop Bx                  ;Restore DMA offset pointer
ECA2 07               757      Pop Es                  ;Restore DMA segment pointer
ECA3 1F               758      Pop Ds                  ;Restore data segment pointer
ECA4 5F               759      Pop Di                  ;Restore pointers Di
ECA5 5E               760      Pop Si                  ;...Si
ECA6 5D               761      Pop Bp                  ;...Bp
ECA7 CA0200           762      Ret 2                   ;Return without restoring flags
                                763      FloppyDriver Endp
                                764      $Eject

```


LOC	OBJ	LINE	SOURCE
		765	;
		766	;
		767	*****
		768	*****
ECAA		769	ParseAndGo Proc Near
ECAA 8A4601		770	Mov A1,FDCCommand ;Parse the command
ECA0 0AC0		771	Or A1,A1 ;...Command = reset?
ECAF 741B		772	Jz ResetCommand ;...jump if Ah=0
ECB1 FEC0		773	Dec A1 ;...Command = status?
ECB3 7413		774	Jz StatusCommand ;...jump if Ah=1
ECB5 807E0203		775	Cmp DriveNumber,3 ;Drive # must be 0 1 2 or 3
ECB9 7704		776	Ja BadDriveRtn ;...jump if out of range
ECBB 3C05		777	Cmp A1,DiskCmdFormat ;Command = (Format (highest)
ECBD 7606		778	Jbe DiskIOShort ;...jump if 2 thru 5
ECBF		779	BadDriveRtn:
ECBF C606000001	E	780	Mov DisketteStat,BadCmdStat ;...else, unrecognizable
ECC4 C3		781	Ret ;...return with bad status
ECC5		782	DiskIOShort:
ECC5 E98200		783	Jmp DiskIO ;Jump and execute r/w/v/f
		784	ParseAndGo Endp
		785	
		786	
		787	;
		788	*****
		789	*****
		790	*****
ECC8		791	StatusCommand Proc Near
ECC8 A00000	E	792	Mov A1,DisketteStat ;Return the status of
ECCB C3		793	Ret ;...the last operation
		794	StatusCommand Endp
		795	
		796	
		797	\$Eject

LOC	OBJ	LINE	SOURCE
		798	;
		799	;
		800	*****
		801	* Reset Command Handler *
		802	* Returns are via ResetReturn or error conditions in *
		803	* FDCOut, ProbeInterrupt, and GetStatus causing 2 level * * return from those procs. *
		804	*****
		805	ResetCommand Proc
ECCC		806	Mov Dx,PortFDCAdptMode ;Point at adapter mode port
ECCC	8AF203	807	Cli ;Prevent int level motor change
ECCF	FA	808	And MotorStatus,00001111B ;Strip all but motor bits
ECD0	80260000F	E 809	Mov Al,MotorStatus ;...and get it in Al
ECD5	A00000	E 810	Mov Cl,4 ;Shift motor to upper for i/o
ECD8	B104	811	Shl Al,Cl ;...to mode port
ECD8	D2E0	812	Test Al,00100000B ;Motor 1?
ECD8	A820	813	Jnz MotorOneBit ;...jump if yes
ECD8	750C	814	Test Al,01000000B ;Motor 2?
ECE0	A840	815	Jnz MotorTwoBit ;...jump if yes
ECE2	7506	816	Test Al,10000000B ;Motor 3?
ECE4	A880	817	Jz MotorZeroBit ;...jump if no (0 or no motor)
ECE6	7406	818	Inc Al ;Inc 3 times to sel drive 3
ECE8	FEC0	819	Inc Al ;Inc 2 times to sel drive 2
ECEA	FEC0	820	Inc Al ;Inc 1 time to sel drive 1
ECEC	FEC0	821	Mov SeekStatus,0 ;Require resets on all drives
ECEE	FEC0	822	Mov DisketteStat,OkStat ;...set OK status
ECEE	C606000000	E 823	Or Al,00001000B ;Interrupt/dma bit
ECEE	C606000000	E 824	Out Dx,Al ;...turn on motor, select and reset
ECF3	C606000000	E 825	Out Dx,Al ;...turn off reset bit
ECF8	0C08	826	Out Dx,Al ;...FDC now reset
ECFA	EE	827	Sti ;Restore interrupts and
ECFB	0C04	828	Call WaitForInt ;...wait for one
ECFD	EE	829	Call ProbeInterrupt ;Get status
ECFE	FB	830	Mov Al,FDCStatus ;Ready transition?
ECFF	E8B701	831	Cmp Al,11000000B ;...indicated by C0 status
ED02	E86602	832	Jz ResetContinue ;Jump if ok
ED05	A00000	E 833	; FDC Error Return
ED08	3CC0	834	Mov DisketteStat,BadFDCStat ;...if not, FDC may be bad
ED0A	7408	835	Jmp ResetReturn ;...jump and error out
		836	; ResetContinue:
		837	Mov Al,FDCSpecifyCmd ;Specify timer settings
		838	Call FDCOut ;...to FDC
ED0C	C606000020	E 839	Mov Al,Specify1 ;Send SPT and HUT
ED11	EB1390	840	Call FDCOut ;...send it to FDC
ED14	B003	841	Mov Al,Specify2 ;Send HLT and ND
ED16	E87C02	842	Call FDCOut ;...send it to FDC
ED19	268A04	843	Mov Al,Specify1 ;Send HLT and ND
ED1C	E87602	844	Call FDCOut ;...send it to FDC
ED1F	268A4401	845	Mov Al,Specify2 ;Send HLT and ND
ED23	E86F02	846	Call FDCOut ;...send it to FDC
ED26		847	; ResetReturn:
ED26	C3	848	Ret ;Return, FDC will not respond
		849	; ResetCommand
		850	Endp
		851	\$Eject

```

LOC OBJ          LINE    SOURCE
-----
                                852
                                853
                                854
                                855
                                856
                                857
                                858
                                859
                                860
                                861
                                ; Command and Mode Table
                                862
                                863
                                TableStruc  Struc
0000          864          Command      Db      FDCSpecifyCMD      ;0 Specify command
0001          865          Db      0                    ;1 No command for status
0002          866          Db      FDCReadCMD       ;2 Read command
0003          867          Db      FDCWriteCMD      ;3 Write command
0004          868          Db      FDCReadCMD       ;4 Verify command
0005          869          Db      FDCFormatCMD     ;5 Format command
0006          870          DMAMode      Db      0                    ;0 Dma Not used
0007          871          Db      0                    ;1 Dma Not used
0008          872          Db      DMAReadMode     ;2 Read command
0009          873          Db      DMAWriteMode    ;3 Write command
000A          874          Db      DMAVerifyMode   ;4 Verify command
000B          875          Db      DMAWriteMode    ;5 Format command
000C          876          Write          Db      0                    ;0 Reset
000D          877          Db      0                    ;1 Status
000E          878          Db      0                    ;2 Read
000F          879          Db      80H                ;3 Write
0010          880          Db      0                    ;4 Verify
0011          881          Db      80H                ;5 Format
0012          882          Bit          Db      1,2,4,8          ;Quick bits
0016          883          TheirError Db      10000000B         ;End of track
0017          884          Db      00100000B         ;Data error
0018          885          Db      00010000B         ;Overrun error
0019          886          Db      00000100B         ;Sector not found
001A          887          Db      00000010B         ;Write protect error
001B          888          Db      00000001B         ;Missing address mark
001C          889          OurError     Db      SectNotFndStat     ;Sector not found
001D          890          Db      BadCrcStat        ;Crc error
001E          891          Db      BadDmaStat        ;Dma error
001F          892          Db      SectNotFndStat     ;Sector not found
0020          893          Db      WriteProtStat     ;Write protect
0021          894          Db      BadAddMarkStat    ;Address mark not found
0022          895          Db      BadFDCStat        ;None of the above (??)
                                896
                                TableStruc  Ends
                                897
                                $Nolist
                                898
                                901
                                DiskIO      Proc      Near
ED4A          902          ED4A FA          Cli                    ;Clear ints during I/O and
ED4B          903          ED4B C606000000    E      Mov      DisketteStat,OkStat ;...set OK status
ED50          904          ED50 8A4601        Mov      Al,FDCCommand    ;Get command
ED53          905          ED53 32E4          Xor     Ah,Ah            ;...into Ah
ED55          906          ED55 8BF8          Mov     Di,Ax            ;...and then into Di
ED57          907          ED57 E60C          Out     PortDmaToggle,Al   ;Set first/last f/f = first
ED59          908          ED59 2E8A852DED    Mov     Al,Table.DMAMode[Di] ;Get DMA mode byte
ED5E          909          ED5E E60B          Out     PortDmaMode,Al    ;...and output it to DMA
                                910
                                911
                                ; Translate SEG:OFF to physical address
                                912
                                913
ED60          914          ED60 8B460C        Mov     Ax,PointerSegment ;Get segment part of pointer
ED63          915          ED63 B104          Mov     Cl,4              ;Multiply by 16 (bytes)
ED65          916          ED65 D3C0          Rol     Ax,Cl             ;...in a paragraph)
ED67          917          ED67 8AE8          Mov     Ch,Al             ;Save upper nibble
ED69          918          ED69 80E50F        And     Ch,00001111B     ;...in Ch
ED6C          919          ED6C 24F0          And     Al,11110000B     ;Isolate upper nibble in Al
ED6E          920          ED6E 03460A        Add     Ax,PointerOffset  ;Add offset in
ED71          921          ED71 80D500        Adc     Ch,0              ;...and carry into Ch
ED74          922          ED74 8BD0          Mov     Dx,Ax            ;Save lower 16 bits of address
                                923
                                924
                                ; Send address to DMAC
                                925
ED76          926          ED76 E604          Out     PortDMACH2Base,Al ;...and output it
ED78          927          ED78 8AC4          Mov     Al,Ah            ;...to DMA
ED7A          928          ED7A E604          Out     PortDmaCH2Base,Al ;...base registers

```

```

LOC OBJ          LINE      SOURCE
ED7C 8AC5        929      Mov    Al,Ch          ;Now output top
ED7E E681        930      Out    PortPageChan2,Al ;...address to page registers
          931
          932      ;    Calculate Word Count
          933
ED80 8A6600      934      Mov    Ah,NumberOfSectors ;Form # of sectors
ED83 32C0        935      Xor    Al,Al          ;...times 128
ED85 D1E8        936      Shr    Ax,1           ;...in Ax
ED87 8A4E06      937      Mov    C1,SectorSize   ;Multiply by the
ED8A D3E0        938      Shl    Ax,C1           ;...number of bytes in sector
ED8C 48          939      Dec    Ax              ;Correct for dma requirement
          940
          941      ;    Output it to DMAC
          942
ED8D E605        943      Out    PortDMACH2Count,Al ;...and output
ED8F 86C4        944      Xchg   Al,Ah           ;...to the
ED91 E605        945      Out    PortDMACH2Count,Al ;...DMA controller
          946
          947      ;    Test for word count error
          948
ED93 86C4        949      Xchg   Al,Ah           ;Restore count in Ax
ED95 03C2        950      Add    Ax,Dx           ;Set carry if crossing 64k
ED97 7309        951      Jnc    NoCrossing     ;...and jump if so
          952
          953      ;    Crossed 64k, Error return
          954
ED99 FB          955      Sti    ;Interrupts Ok now
ED9A C606000009  956      Mov    DisketteStat,DMAPageStat ;...else loadup error code
ED9F E91401      957      Jmp    ErrorReturn    ;...and return
          958
EDA2            959      NoCrossing:
EDA2 B002        960      Mov    Al,0000010B    ;Get mask bit for DMAC
EDA4 E60A        961      Out    PortDMAMaskSngl,Al ;...and enable the transfer
          962
          963      ;    Turn on drive motor and select drive
          964
EDA6 C6060000FF  965      Mov    MotorCount,255 ;Set large value during I/D
          966
          967      ;    Calculate Drive Bit from Drive Number
          968
EDAB 8A5E02      969      Mov    B1,DriveNumber ;Get drive number in
EDAE 32FF        970      Xor    Bh,Bh          ;...Bx
EDB0 2EBA8739ED  971      Mov    Al,Table.Bit[Bx] ;...and calculate bit number
EDB5 8AE8        972      Mov    Ch,Al          ;...save in ch for seek
          973
          974      ;    Indicate on for compatibility
          975
EDB7 08060000   976      Or     MotorStatus,Al ;Indicate in status byte
          977
          978      ;    Actually turn it on and select drive
          979
EDBB B104        980      Mov    C1,4           ;Move motor bit to
EDBD D2E0        981      Shl    Al,C1          ;...upper nibble
EDBF 0AC3        982      Or     Al,B1          ;...then or in drive number
EDC1 0C0C        983      Or     Al,00001100B  ;Add enable and not reset
EDC3 BAF203      984      Mov    Dx,PortFDCAdptMode ;...point at adapter I/D port
EDC6 EE         985      Out    Dx,Al         ;...and output the data
          986
          987      ;    Restore interrupts from DMA section above
          988
EDC7 FB         989      Sti    ;Restore interrupts from above
          990
          991      ;    If write, wait for motor up to speed
          992
EDC8 2EBA8533ED  993      Mov    Al,Table.Write[Dil] ;Test if command involves write
EDCD 08060000   994      Or     MotorStatus,Al ;... (for compatibility)
EDD1 0AC0        995      Or     Al,Al         ;...and affect flags
EDD3 7912        996      Jns    WaitDone      ;...jump if read (upper bit=0)
          997
          998      ;    Write command, wait for motor up to speed
          999
EDD5 268A640A   1000     Mov    Ah,MotorStart  ;Ah=# of 125ms to wait
EDD9 0AE4        1001     Or     Ah,Ah          ;Affect flags
EDDB 740A        1002     Jz     WaitDone      ;...if no wait (??), comply
EDDD 51          1003     Push   Cx            ;Save bit position in Ch
    
```

NB:
he teste
pas si le
motteur est
deja
monte
Attend p
consequen
125ms p
écrit

```

LOC OBJ          LINE      SOURCE
EDDE            1004      OuterLoop:
EDDE 33C9       1005      Xor      Cx,Cx          ;Max count in Cx
EDE0            1006      EighthSecLoop:
EDE0 E2FE       1007      Loop    EighthSecLoop  ;Loop for 125 ms
EDE2 FECC       1008      Dec     Ah              ;Decrement outer loop
EDE4 75F8       1009      Jnz    OuterLoop      ;...and jump to the top of it
EDE6 59         1010      Pop     Cx              ;Restore bit position in Ch
EDE7            1011      WaitDone:
                1012
                1013      ;      Seek to track Ch = Motor Bit, Bl = Drive #
                1014
ED E7 842E0000   E 1015      Test    SeekStatus,Ch  ;Need recal?
EDEB 7514       1016      Jnz    NoRecalReqd    ;...jump if no
EDED 082E0000   E 1017      Or     SeekStatus,Ch  ;Show as recal
EDF1 B007       1018      Mov     Al,FDCRecalCMD ;...and send recal
EDF3 E89F01     1019      Call   FDCOut         ;...command to FDC
EDF6 BAC3       1020      Mov     Al,Bl          ;Then send drive number
EDF8 E89A01     1021      Call   FDCOut         ;...which is in Bl from above
EDFB E8BB00     1022      Call   WaitForInt     ;Wait for results
EDFE E86A01     1023      Call   ProbeInterrupt ;Get status
EE01            1024      NoRecalReqd:
EE01 B00F       1025      Mov     Al,FDCSeekCMD  ;Send seek command
EE03 E88F01     1026      Call   FDCOut         ;...to FDC
EE06 BAC3       1027      Mov     Al,Bl          ;Send drive number
EE08 E88A01     1028      Call   FDCOut         ;...to FDC
EE0B 8A4603     1029      Mov     Al,TrackNumber ;Send track number
EE0E E88401     1030      Call   FDCOut         ;...to FDC
EE11 E8A500     1031      Call   WaitForInt     ;Wait for results
EE14 E85401     1032      Call   ProbeInterrupt ;Get status
EE17 268A4409   1033      Mov     Al,HeadSettle ;Get head settling time
EE1B 0AC0       1034      Or     Al,Al          ;...none (??), comply
EE1D 7409       1035      Jz     MStimerDone    ;...and end timer
EE1F            1036      MStimerDone:
EE1F B92602     1037      Mov     Cx,550        ;1 ms timer
EE22            1038      OneMsLoop:
EE22 E2FE       1039      Loop    OneMsLoop     ;Loop for a ms
EE24 FEC8       1040      Dec     Al              ;Decrement outer loop
EE26 75F7       1041      Jnz    MStimerDone    ;Loop till done
EE28            1042      MStimerDone:
                1043
                1044      ;      Perform the operation
                1045
EE28 2E8A8527ED 1046      Mov     Al,Table.Command[Di] ;Get operation command
EE2D E86501     1047      Call   FDCOut         ;...and send it
EE30 8A4604     1048      Mov     Al,HeadNumber  ;Get head number
EE33 2401       1049      And    Al,00000001B   ;...make sure 0 or 1
EE35 D0E0       1050      Shl    Al,1           ;...move it into
EE37 D0E0       1051      Shl    Al,1           ;...position
EE39 0AC3       1052      Or     Al,Bl          ;...Or in drive number
EE3B E85701     1053      Call   FDCOut         ;...and output second byte
EE3E 807E0105   1054      Cmp    FDCCommand,DiskCmdFormat ;Are we formatting?
EE42 751D       1055      Jne    NotFormatting  ;...jump if not
                1056
                1057      ;      Load parameters for format
                1058
EE44 8A4606     1059      Mov     Al,SectorSize  ;FDC "N" parameter
EE47 E84B01     1060      Call   FDCOut         ;...send it out
EE4A 8A4607     1061      Mov     Al,EDT         ;FDC "SC" parameter
EE4D E84501     1062      Call   FDCOut         ;...send it out
EE50 268A4407   1063      Mov     Al,FormatGap   ;FDC "BPL" parameter
EE54 E83E01     1064      Call   FDCOut         ;...send it out
EE57 268A4408   1065      Mov     Al,FormatFiller ;FDC "D" parameter
EE5B E83701     1066      Call   FDCOut         ;...send it out
EE5E E80F90     1067      Jmp    CommandLoaded  ;Continue below
                1068
                1069      ;      Load parameters for read / write / verify
                1070
EE61            1071      NotFormatting:
EE61 B90700     1072      Mov     Cx,7           ;Transfer 7 bytes from
EE64 BF0300     1073      Mov     Di,3           ;...stack frame starting at 9
EE67            1074      FrameLoop:
EE67 8A03       1075      Mov     Al,[Bp][Di]    ;Get byte from frame
EE69 E82901     1076      Call   FDCOut         ;...and send it
EE6C 47         1077      Inc     Di              ;...inc index to next byte
EE6D E2F8       1078      Loop   FrameLoop      ;Loop till 7 bytes moved

```

```

LDC OBJ          LINE      SOURCE
                1079
                1080          ;      Wait for interrupt
                1081
EE6F            1082      CommandLoaded:
EE6F E84700     1083          Call   WaitForInt      ;Wait for end of op. interrupt
EE72 E8FB00     1084          Call   GetStatus      ;...Read in the status
                1085
                1086          ;      Test for error
                1087
EE75 A00000     E      1088          Mov    Al,FDCStatus      ;Get first status byte
EE78 24C0       1089          And    Al,11000000B     ;...normal termination?
EE7A 7426       1090          Jz     OperationOk     ;...jump if ok
EE7C 3C40       1091          Cmp   Al,01000000B     ;...abnormal termination?
EE7E 7408       1092          Je     Translate        ;...jump if yes
EE80 C606000020 E      1093          Mov    DisketteStat,BadFDCStat ;...else error return
EE85 EB1B90     1094          Jmp   ErrorRtnNum      ;...actual transferred
                1095
                1096          ;      Translate FDC error to our error
EE88            1097      Translate:
EE88 A00100     E      1098          Mov    Al,FDCStatus+1  ;Look at second byte
EE8B B90600     1099          Mov    Cx,6             ;6 bits to translate
EE8E 33D8       1100          Xor   Bx,Bx            ;...point at byte 0
EE90            1101      ErrorLoop:
EE90 2E84873DED 1102          Test  Al,Table.TheirError[Bx] ;Test a bit from table
EE95 7503       1103          Jnz   ErrorFound      ;...and jump if it is set
EE97 43         1104          Inc   Bx               ;...bump pointer
EE98 E2F6       1105          Loop  ErrorLoop       ;...and loop till done
                1106
                1107          ;      Falling thru to here means no bits were set (FDC error)
                1108
EE9A            1109      ErrorFound:
EE9A 2E8A8743ED 1110          Mov    Al,Table.OurError[Bx] ;Translate from table
EE9F A20000     E      1111          Mov    DisketteStat,Al  ;...and into DisketteStat
                1112
                1113          ;      Calculate # of sectors transferred and return in Al
EEA2            1114      ErrorRtnNum:
EEA2            1115      OperationOk:
EEA2 A00300     E      1116          Mov    Al,FDCStatus[3]  ;Get track we ended up on
EEA5 3A4603     1117          Cmp   Al,TrackNumber   ;...compare it to starting tk
EEA8 A00500     E      1118          Mov    Al,FDCStatus[5] ;...get last sector
EEAB 7405       1119          Je     OnSameTrack     ;...jump if didn't roll over
EEAD 8A4607     1120          Mov    Al,EDT          ;Get last track
EEB0 FEC0       1121          Inc   Al               ;...plus 1 into Al
EEB2            1122      OnSameTrack:
EEB2 2A4605     1123          Sub   Al,SectorNumber  ;Return end minus start
EEB5 C3         1124          Ret                    ;...equals num transferred
                1125
                1126          ;      Return for pre-operation errors
EEB6            1127      ErrorReturn:
EEB6 B000       1128          Mov    Al,0             ;No sectors transferred
EEB8 C3         1129          Ret                    ;Return, operation failed
                1130      DiskIO
                1131          Endp
                1132          ;
                1133          ;      *****
                1134          ;      * Wait for Interrupt and Return Result *
                1135          ;      *****
EEB9            1136      WaitForInt Proc   Near
EEB9 FB         1137          Sti                    ;Ints on
EEBA 33C9       1138          Xor   Cx,Cx            ;Setup for 2 second wait
EEBC B002       1139          Mov   Al,2             ;...for interrupt
EEBE            1140      IntWaitLoop:
EEBE F606000080 E      1141          Test  SeekStatus,1000000B ;Interrupt will set this
EEC3 F8         1142          Clc                    ;...clear carry for return
EEC4 7510       1143          Jnz   IntOccurred     ;...jump if interrupt
EEC6 E2F6       1144          Loop IntWaitLoop     ;Loop till int occurs
EEC8 FEC8       1145          Dec   Al               ;Decrement outer loop
EECA 75F2       1146          Jnz   IntWaitLoop     ;...and loop some more
EECC C606000080 E      1147          Mov    DisketteStat,TimeOutStat ;Set error status
EED1 58         1148          Pop   Ax               ;Discard return address
EED2 32C0       1149          Xor   Al,Al           ;Indicate 0 bytes xferred
EED4 F9         1150          Stc                    ;Set carry for error and
EED5 C3         1151          Ret                    ;...return 2 levels up
EED6            1152      IntOccurred:
EED6 802600007F E      1153          And   SeekStatus,0111111B ;Turn off interrupt bit

```

```

LOC OBJ          LINE    SOURCE
EEDB C3          1154      Ret                ;...return
                1155      WaitForInt        Endp
                1156
                1157      ;
                1158      ; *****
                1159      ; * Read Data from Controller *
                1160      ; *****

EEDC             1161      FDCIn            Proc    Near
EEDC 51          1162      Push            Cx                ;Save Cx
EEDD 33C9        1163      Xor             Cx,Cx            ;Maximum timeout
EEDF BAF403      1164      Mov            Dx,PortFDCStatus ;Pointer to FDC Status port
EEE2            1165      FDCInLoop1:
EEE2 EC          1166      In             Al,Dx            ;Get status
EEE3 0AC0        1167      Or             Al,Al            ;Affect flags
EEE5 7809        1168      Js             MasterReady      ;If upper bit set, jump
EEE7 E2F9        1169      Loop           FDCInLoop1      ;Loop till master set or T.O.
EEE9 C606000080 E 1170      Mov            DisketteStat,TimeOutStat;FDC did not respond, error
EEEE EB09        1171      Jmp            Short FDCInRtn   ;...return
EEF0            1172      MasterReady:
EEF0 A840        1173      Test           Al,01000000B     ;Correct direction?
EEF2 7508        1174      Jnz            CorrectDir       ;...jump if so
EEF4 C606000020 E 1175      Mov            DisketteStat,BadFDCStat ;...else chip is bad, error out
EEF9            1176      FDCInRtn:
EEF9 59          1177      Pop            Cx
EEFA F9          1178      Stc                ;Indicate error
EEFB C3          1179      Ret                ;...and return
                1180
                1181      ; Data ready
                1182
EEFC            1183      CorrectDir:
EEFC 42          1184      Inc            Dx                ;Point at data port
EEFD EC          1185      In             Al,Dx            ;Get a byte
EEFE 50          1186      Push           Ax                ;...and save it
EEFF B90A00      1187      Mov            Cx,10            ;Pause and let FDC get another
EF02            1188      PauseLoop:
EF02 E2FE        1189      Loop           PauseLoop        ;...byte ready if it has one
EF04 4A          1190      Dec            Dx                ;Point back at status
EF05 EC          1191      In             Al,Dx            ;...and get it
EF06 A810        1192      Test           Al,00010000B     ;...test if still busy. Return
EF08 F8          1193      Clc                ;...carry clear and
EF09 58          1194      Pop            Ax                ;...restore data
EF0A 59          1195      Pop            Cx                ;...and registers, return
EF0B C3          1196      Ret                ;...Z=1 if last byte
                1197      FDCIn            Endp
                1198
                1199
                1200      ;
                1201      ; *****
                1202      ; * Disk Interrupt Service Routine *
                1203      ; *****
EF57            1204      Drg            0EF57H
EF57            1205
EF57            1206      FloppyHdwrInt Proc    Far
EF57 FB          1207      Sti                ;Restore interrupts
EF58 1E          1208      Push            Ds                ;Save Data segment
EF59 50          1209      Push            Ax                ;Save an intermediate register
EF5A B8----- R 1210      Mov            Ax,BiosDataArea ;Load data segment
EF5D 8ED8        1211      Mov            Ds,Ax            ;...with bios segment
EF5F 800E000080 E 1212      Or             SeekStatus,10000000B ;Turn on indicator bit
EF64 B020        1213      Mov            Al,PicEDI        ;Interrupt ack to Pic
EF66 E620        1214      Out            PortPICDCW2,Al ;...to restore higher ints
EF68 58          1215      Pop            Ax                ;Restore
EF69 1F          1216      Pop            Ds                ;...registers
EF6A CF          1217      Iret             ;Restore flags and return
                1218      FloppyHdwrInt Endp
                1219
                1220      ;
                1221      ; *****
                1222      ; * Read Operation Result *
                1223      ; *****
EF6B            1224      ResultProc      Proc    Near
EF6B            1225      ProbeInterrupt:
EF6B B008        1226      Mov            Al,FDCSenseIntCMD ;Send command to sense int
EF6D E82500      1227      Call           FDCOut           ;...to FDC
EF70            1228      GetStatus:

```

```

LOC OBJ          LINE      SOURCE
EF70 53          1229          Push    Bx          ;Save Bx
EF71 51          1230          Push    Cx          ;...and Cx
EF72 B90700      1231          Mov     Cx,7        ;Get 7 bytes max from FDC
EF75 B00000      1232          Mov     Bx,0        ;...set pointer to first byte
EF78            1233          ResultLoop:
EF78 E861FF      1234          Call   FDCIn        ;Get a byte
EF7B 720E        1235          Jc     StatErrRtn   ;...jump if an error occurred
EF7D 88870000    E 1236          Mov     FDCStatus[Bx],Al ;...and store it
EF81 740F        1237          Jz     StatusReturn ;Jump if the FDC is empty
EF83 43         1238          Inc     Bx          ;...else bump pointer and
EF84 E2F2        1239          Loop   ResultLoop  ;...loop
EF86 C606000020 E 1240          Mov     DisketteStat,BadFDCStat ;More than 7 bytes is an error
EF8B            1241          StatErrRtn:
EF8B F9         1242          Stc                    ;Set carry for error return
EF8C 59         1243          Pop     Cx          ;Restore Cx
EF8D 5B         1244          Pop     Bx          ;...and Bx
EF8E 58         1245          Pop     Ax          ;Discard return address
EF8F 32C0        1246          Xor     Al,Al       ;Indicate 0 bytes xferred
EF91 C3         1247          Ret                    ;...and return 2 levels up
EF92            1248          StatusReturn:
EF92 59         1249          Pop     Cx          ;Restore Cx
EF93 5B         1250          Pop     Bx          ;...and Bx
EF94 C3         1251          Ret                    ;and return error
                1252          ResultProc
                1253          Endp
                1254          ;
                1255          ; *****
                1256          ; * Write Data to Controller *
                1257          ; *****
EF95            1258          FDCOut
EF95 51          1259          Push   Cx          ;Save
EF96 52          1260          Push   Dx          ;...registers
EF97 50          1261          Push   Ax          ;...on stack
EF98 33C9        1262          Xor    Cx,Cx       ;Maximum timeout
EF9A BAF403      1263          Mov    Dx,PortFDCStatus ;Pointer to FDC Status port
EF9D            1264          FDCOutLoop1:
EF9D EC         1265          In     Al,Dx       ;Get status
EF9E 0AC0        1266          Or     Al,Al       ;Affect flags
EFA0 7809        1267          Js     OutMasterRdy ;If upper bit set, jump
EFA2 E2F9        1268          Loop  FDCOutLoop1 ;Loop till master set or T.O.
EFA4 C606000080 E 1269          Mov    DisketteStat,TimeoutStat ;FDC did not respond, error
EFA9 EB12        1270          Jmp   Short OutErrRtn ;...jump and return error
EFAB            1271          OutMasterRdy:
EFAB A840        1272          Test   Al,01000000B ;Correct direction?
EFAD 7407        1273          Jz     OutCorrectDir ;...jump if so
EFAF C606000020 E 1274          Mov    DisketteStat,BadFDCStat ;...else chip is bad, error out
EFB4 EB07        1275          Jmp   Short OutErrRtn ;...jump and return error
EFB6            1276          OutCorrectDir:
EFB6 42         1277          Inc    Dx          ;Point at data port
EFB7 58         1278          Pop    Ax          ;Restore data to output
EFB8 EE         1279          Out    Dx,Al       ;...and output it
EFB9 F8         1280          Clc                    ;Clear carry
EFBA 5A         1281          Pop    Dx          ;Restore Dx
EFBB 59         1282          Pop    Cx          ;...and Cx
EFBC C3         1283          Ret                    ;...and return
EFBD            1284          OutErrRtn:
EFBD 58         1285          Pop    Ax          ;Restore Ax
EFBE 5A         1286          Pop    Dx          ;Restore Dx
EFBF 59         1287          Pop    Cx          ;...and Cx
EFC0 58         1288          Pop    Ax          ;Discard return address
EFC1 32C0        1289          Xor    Al,Al       ;Indicate 0 bytes xferred
EFC3 F9         1290          Stc                    ;...and set carry flag
EFC4 C3         1291          Ret                    ;...and return 2 levels back
                1292          FDCOut
                1293          Endp
                1294          ;
                1295          ; *****
                1296          ; * Disk Base Parameters *
                1297          ; *****
EFC7            1298          Org     0EFC7H      ;Align with PC / XT
                1299
EFC7 CF         1300          FloppyParamsPointer Db 11001111B ;Specify byte 1
EFC8 02         1301          Db 00000010B ;Specify byte 2
EFC9 25         1302          Db 37 ;Motor timeout wait
EFCB 02         1303          Db 2 ;512 bytes per sector

```


LOC	OBJ	LINE	SOURCE
EFCB	08	1304	Db 8 ;Last sector on a track
EFCC	2A	1305	Db 42 ;Gap length
EFCD	FF	1306	Db 0FFH ;DTL
EFCE	50	1307	Db 80 ;Format gap length
EFCF	F6	1308	Db 0F6H ;Format fill byte
EFD0	19	1309	Db 25 ;Head settle time
EFD1	04	1310	Db 4 ;Motor start time
		1311	
---		1312	Bios Ends
		1313	
		1314	End

ASSEMBLY COMPLETE, NO ERRORS FOUND

IBM PC/XT 0086/0087/0088 MACRO ASSEMBLER V1.1 ASSEMBLY OF MODULE PRINTER
 OBJECT MODULE PLACED IN PRINTER.OBJ
 ASSEMBLER INVOKED BY: ASM86 PRINTER.SRC

```

LOC OBJ          LINE    SOURCE
                1 +1 $Title ('DTC/PC BIOS Parallel Printer Driver V1.0')
                2 +1 $Pagelength (80) Pagewidth (132) Debug Nogen
                3   Name Printer
                4
                5
                6   ;   Author:      Don K. Harrison
                7
                8   ;   Start date:  November 25, 1983      Last edit:  December 20, 1983
                9
               10
               11   ;           *****
               12   ;           * Module Description *
               13   ;           *****
               14
               15   ;           This module handles the parallel printer driver, accessed via
               16   ;           interrupt 17H.
               17
               18
               19
               20
               21
               22   ;           (c) Display Telecommunications Corporation, 1983
               23   ;           All Rights Reserved
               24
               25   $Eject
    
```

LOC	OBJ	LINE	SOURCE
		26	
		27	
		28	;
		29	;
		30	*****
		31	* Revision History *
		32	*****
		33	
		34	
		35	
		36	
		37	
		38	\$Eject

```
LOC OBJ          LINE    SOURCE
                 39
                 40          ;
                 41          ; *****
                 42          ; * Public Symbols *
                 43          ; *****
                 44          ;
                 45          Public PrinterDriver
                 46          ;
                 47          ; *****
                 48          ; * Equates *
                 49          ; *****
                 50          ;
                 51          ; All Equates in include file: IbmInc
                 52          ;
                 53          $Include (IbmInc)
=1 54          ; *****
=1 55          ; * Global Include File *
=1 56          ; *****
=1 57          $Nolist
577 $Eject
```

LOC	OBJ	LINE	SOURCE
		578	
		579	;
		580	* Data Segments *
		581	;
		582	;
----		583	BiosDataArea Segment Public
----		584	Exfrn PrintTimeOut:Byte, PrinterBase:Word
		585	BiosDataArea Ends
		586	\$Eject

```

LOC OBJ          LINE    SOURCE
                    587
                    588 ;
                    589 ; *****
                    590 ; * Code Segment *
                    591 ; *****
-----          592 Bios      Segment Common
                    593
                    594 Assume  Cs:Bios, Ds:BiosDataArea
                    595
EFD2             596 Org      0EFD2H
                    597
EFD2             598 PrinterDriver Proc  Far
EFD2 FB         599 StI      ;Restore interrupts
EFD3 1E         600 Push     Ds      ;Save registers
EFD4 53         601 Push     Bx      ;...that
EFD5 51         602 Push     Cx      ;...we
EFD6 52         603 Push     Dx      ;...will use
EFD7 8B----- R    604 Mov      Bx,BiosDataArea ;Point at our
EFD8 8EDB       605 Mov      Ds,Bx   ;...data segment
EFD9 8BDA       606 Mov      Bx,Dx   ;Use Bx as index of ports
EFDE D1E3       607 Shl     Bx,1     ;... (ports are words)
EFE0 8B970000   E    608 Mov     Dx,PrinterBase[Bx] ;...and get our port
EFE4 0BD2       609 Or      Dx,Dx    ;If none, return
EFE6 740C       610 Jz      PrinterReturn ;...no particular status
                    611
                    612 ; Do case on command
                    613
EFE8 0AE4       614 Or      Ah,Ah    ;Case = 0?
EFEA 740D       615 Jz      PrintChar ;...if so, jump and print char
EFEC FECC       616 Dec     Ah      ;Case = 1?
EFEE 743B       617 Jz      PrintInitialize ;...if so, init printer
EFF0 FECC       618 Dec     Ah      ;Case = 2?
EFF2 742B       619 Jz      PrintStatus ;...if so, jump and return status
                    620
                    621 ; If illegal command, just return
                    622
EFF4             623 PrinterReturn:
EFF4 5A         624 Pop     Dx      ;Restore registers we
EFF5 59         625 Pop     Cx      ;...used
EFF6 5B         626 Pop     Bx      ;...and
EFF7 1F         627 Pop     Ds      ;...return
EFF8 CF         628 Iret    ;...to caller
                    629
                    630 ;
                    631 ; *****
                    632 ; * Print Character in Al *
                    633 ; *****
                    634
EFF9             634 PrintChar Proc  Near
EFF9 EE         635 Out     Dx,Al   ;Send char to printer
EFAA 42         636 Inc     Dx      ;...then point at status
EFBB 8ABF0000   E    637 Mov     Bh,PrintTimeOut[Bx] ;Get # of loops in Bh
EFFF 8AE0       638 Mov     Ah,Al   ;...save print character
F001             639 PrintOuterLoop:
F001 33C9       640 Xor     Cx,Cx   ;Maximum loop count
F003             641 PrintInnerLoop:
F003 EC         642 In     Al,Dx   ;Get the status
F004 0AC0       643 Or     Al,Al   ;...and affect flags
F006 7B0D       644 Js     StrobeChar ;...jump if not busy
F008 E2F9       645 Loop   PrintInnerLoop ;...else loop till not busy
F00A FECF       646 Dec     Bh     ;Decrement outer loop
F00C 75F3       647 Jnz    PrintOuterLoop ;...and loop
                    648
                    649 ; Timeout
                    650
F00E 0C01       651 Or     Al,1     ;...set timeout error
F010 24F9       652 And   Al,11111001B ;...mask unused bits
F012 EB1190     653 Jmp   StatusReturn2 ;...and return status
F015             654 StrobeChar:
F015 42         655 Inc     Dx      ;Point at command port
F016 B00D       656 Mov     Al,00001101B ;Strobe = high
F018 EE         657 Out     Dx,Al   ;...to printer
                    658
                    659 ; This is an entry point from PrinterInitialize
                    660
F019             661 StatusReturn3:

```

LOC	OBJ	LINE	SOURCE	
F019	800C	662	Mov	Al,00001100B ;Strobe = low
F01B	EE	663	Out	Dx,Al ;...to printer
F01C	4A	664	Dec	Dx ;Point back at status
F01D	EB03	665	Jmp	Short StatusReturn1 ;...and return with it
		666	PrintChar	Endp
		667		
		668	;	*****
		669	;	* Return Printer Status *
		670	;	*****
		671		
F01F		672	PrintStatus	Proc Near
F01F	8AE0	673	Mov	Ah,Al ;Preserve Al
F021	42	674	Inc	Dx ;Increment to status port
F022		675	StatusReturn1:	
F022	EC	676	In	Al,Dx ;...and get status
F023	24F8	677	And	Al,011111000B ;...turn off unused bits
F025		678	StatusReturn2:	
F025	3448	679	Xor	Al,01001000B ;...and flip sense of some
F027	86C4	680	Xchg	Al,Ah ;...Al=char, Ah=status
F029	EBC9	681	Jmp	PrinterReturn ;...jump and return status
		682	PrintStatus	Endp
		683		
		684	;	*****
		685	;	* Initialize Printer *
		686	;	*****
		687		
F02B		688	PrintInitialize	Proc Near
F02B	8AE0	689	Mov	Ah,Al ;Preserve Al
F02D	42	690	Inc	Dx ;Point at command
F02E	42	691	Inc	Dx ;...port
F02F	B008	692	Mov	Al,00001000B ;Init bit
F031	EE	693	Out	Dx,Al ;...to printer
F032	B9DC05	694	Mov	Cx,1500 ;Delay
F035		695	InitDelay:	
F035	E2FE	696	Loop	InitDelay ;...for reset pulse
F037	EBE0	697	Jmp	StatusReturn3 ;...then jump and turn off
		698		;...init bit and return status
		699		
		700	PrintInitialize	Endp
		701		
		702	PrinterDriver	Endp
		703		
----		704	Bios	Ends
		705		
		706	End	

ASSEMBLY COMPLETE, NO ERRORS FOUND

IBM PC/XT 8086/8087/8088 MACRO ASSEMBLER V1.1 ASSEMBLY OF MODULE VIDEO
OBJECT MODULE PLACED IN VIDEO.OBJ
ASSEMBLER INVOKED BY: ASM86 VIDEO.SRC

LOC	OBJ	LINE	SOURCE
		1 +1	\$Title ('DTC/PC BIOS Video Driver V1.0')
		2 +1	\$Pagelength (80) Pagewidth (132) Debug
		3	Name Video
		4	
		5	
		6	; Author: Don K. Harrison
		7	
		8	; Start date: November 26, 1983 Last edit: December 27, 1983
		9	
		10	
		11	; *****
		12	; * Module Description *
		13	; *****
		14	
		15	; This module, accessed via interrupt 10H, handles both alpha
		16	; and graphics video modes of either type video interface card.
		17	
		18	
		19	
		20	
		21	
		22	; (c) Display Telecommunications Corporation, 1983
		23	; All Rights Reserved
		24	
		25 +1	\$Eject

LOC	OBJ	LINE	SOURCE
		26	
		27	
		28	;
		29	*****
		30	* Revision History *
		31	*****
		32	;
		33	
		34	
		35	
		36	
		37	
		38	+1 \$Eject

```
LOC OBJ          LINE    SOURCE
                39
                40      ;          *****
                41      ;          * Public Symbols *
                42      ;          *****
                43
                44      Public VideoDriver, VidParamsPointer
                45
                46
                47
                48      ;          *****
                49      ;          * Equates *
                50      ;          *****
                51
0000             52      AsciiBackspace Equ    08H          ;Ascii characters
000A             53      AsciiLineFeed  Equ    0AH          ;...for
000D             54      AsciiCarriage  Equ    0DH          ;...TTY
0007             55      AsciiBell     Equ    07H          ;...procedure
                56
=1              57 +1  $Include (IbmInc)
                58      ;          *****
=1              59      ;          * Global Include File *
                60      ;          *****
=1              61 +1  $Nolist
=1              581 +1 $Eject
```

LOC	OBJ	LINE	SOURCE
		582	
		583	;
		584	;
		585	;
		586	;
---		587	IntSegment Segment Public
---		588	Extrn VidParamsTrapAddr:DWord, VideoGraphicsTrapAddr:DWord
---		589	IntSegment Ends
---		590	
---		591	BiosDataArea Segment Public
		592	Extrn CrtMode:Byte, CrtColumns:Word, CrtLength:Word
		593	Extrn CrtStart:Word, CursorPosn:Word, CursorMode:Word
		594	Extrn ActivePage:Byte, ActiveCard:Word, CrtModeSet:Byte
		595	Extrn CrtPalette:Byte, EquipFlag:Word
---		596	BiosDataArea Ends
---		597	
---		598	MonoSeg Segment Public
---		599	MonoSeg Ends
---		600	
---		601	ColorSeg Segment Public
---		602	ColorSeg Ends
		603	
		604	+1 \$Eject

```

LOC OBJ          LINE      SOURCE
605
606              ;
607              ; *****
608              ; * Stack Frame *
609              ; *****
610              ; -----
611              ; |           Es           | Bp[16]
612              ; -----
613              ; |           Ds           | Bp[14]
614              ; -----
615              ; |           Si           | Bp[12]
616              ; -----
617              ; |           Di           | Bp[10]
618              ; -----
619              ; |           Dh           | Bp[8]
620              ; |           Ch           | Bp[6]
621              ; |           Bh           | Bp[4]
622              ; |           Command      | Bp[2]
623              ; |           Video Segment | Bp[0]
624              ; -----
625              ;
626              ; *****
627              ; * Stack Frame Equates *
628              ; *****
0003[ 633 Command      Equ      Byte Ptr [Bp+3] ;All - command
0002[ 634 Mode        Equ      Byte Ptr [Bp+2] ;Set mode/state - mode in/out
0007[ 635 CursStartLine Equ      Byte Ptr [Bp+7] ;Curs type- start line
0006[ 636 CursEndLine     Equ      Byte Ptr [Bp+6] ;Curs type- end line
0006[ 637 CursCommand   Equ      Word Ptr [Bp+6] ;Curs type- start and end
0009[ 638 SetRow         Equ      Byte Ptr [Bp+9] ;Set curs- row
0008[ 639 SetCol         Equ      Byte Ptr [Bp+8] ;Set curs- column
0008[ 640 SetRowCol      Equ      Word Ptr [Bp+8] ;Set curs -row and column
0009[ 641 ReadRow       Equ      Byte Ptr [Bp+9] ;Read curs- row
0008[ 642 ReadCol       Equ      Byte Ptr [Bp+8] ;Read curs- column
0008[ 643 ReadRowCol   Equ      Word Ptr [Bp+8] ;Read curs -row and column
0006[ 644 ReturnCursMode Equ      Word Ptr [Bp+6] ;Read curs -current cursor mode
0009[ 645 LpRow         Equ      Byte Ptr [Bp+9] ;Read Lp - row
0008[ 646 LpCol         Equ      Byte Ptr [Bp+8] ;Read Lp - col
0008[ 647 LpRowCol     Equ      Word Ptr [Bp+8] ;Read Lp - row and column
0007[ 648 LpRasterLine  Equ      Byte Ptr [Bp+7] ;Read Lp - Raster line
0004[ 649 LpPixel       Equ      Word Ptr [Bp+4] ;Read Lp - Pixel #
0003[ 650 LpStatus      Equ      Byte Ptr [Bp+3] ;Read Lp - Exit status
0002[ 651 NewPage      Equ      Byte Ptr [Bp+2] ;Sel Active - new active page
0002[ 652 ScrollNumRows Equ      Byte Ptr [Bp+2] ;Scroll - # of rows to blank
0007[ 653 ScrollUpperRow Equ      Byte Ptr [Bp+7] ;Scroll - Upper row
0009[ 654 ScrollLowerRow Equ      Byte Ptr [Bp+9] ;Scroll - Lower row
0006[ 655 ScrollUpperCol Equ      Byte Ptr [Bp+6] ;Scroll - Upper column
0008[ 656 ScrollLowerCol Equ      Byte Ptr [Bp+8] ;Scroll - Lower column
0006[ 657 ScrollUpper   Equ      Word Ptr [Bp+6] ;Scroll - Upper row and column
0008[ 658 ScrollLower   Equ      Word Ptr [Bp+8] ;Scroll - Lower row and column
0005[ 659 ScrollAttrib  Equ      Byte Ptr [Bp+5] ;Scroll - blanking attribute
0005[ 660 DisplayPage   Equ      Byte Ptr [Bp+5] ;Char handling - display page
0002[ 661 Char          Equ      Byte Ptr [Bp+2] ;Char handling - char in/out
0003[ 662_ATTRIBOut    Equ      Byte Ptr [Bp+3] ;Char handling - attrib out
0006[ 663 CharCount    Equ      Word Ptr [Bp+6] ;Char handling - repeat count
0004[ 664_ATTRIBIn     Equ      Byte Ptr [Bp+4] ;Char handling - attrib in
0005[ 665 ColorId     Equ      Byte Ptr [Bp+5] ;Set Palette - Color Id
0004[ 666 ColorValue   Equ      Byte Ptr [Bp+4] ;Set Palette - Color value
0008[ 667 DotRow      Equ      Word Ptr [Bp+8] ;Read/write dot - row number
0006[ 668 DotCol       Equ      Word Ptr [Bp+6] ;Read/write dot - col number
0002[ 669 Dot         Equ      Byte Ptr [Bp+2] ;Read/write dot - dot in/out
0004[ 670 TTYForeground Equ      Byte Ptr [Bp+4] ;TTY - foreground color
0003[ 671 Columns     Equ      Byte Ptr [Bp+3] ;State - # of columns
0000[ 672 VideoSegment Equ      Word Ptr [Bp+0] ;Video segment
673
674
675
676 +1 $Eject

```

```

LOC OBJ          LINE    SOURCE
-----
                                677      ;          *****
                                678      ;          * Code Segment *
                                679      ;          *****
                                680
                                681      Bios      Segment Common
                                682      Extrn   VideoGraphicsPointer:Byte, Beep:Near
                                683      Assume Cs: Bios, Ds: BiosDataArea, Es: Nothing
                                684      Org    0F045H
F045              685      JumpTable Label Word
F045 15F1         686      Dw      Offset SetMode          ;Change modes
F047 C3F1         687      Dw      Offset SetCursorType    ;Set cursor type
F049 D0F1         688      Dw      Offset SetCursorPos     ;Move cursor to position
F04B E8F1         689      Dw      Offset ReadCursorPos    ;Read current cursor position
F04D 04F2         690      Dw      Offset ReadLpPos        ;Read Light pen position
F04F 85F2         691      Dw      Offset ActivatePage     ;Activate new page
F051 A5F2         692      Dw      Offset Scroll           ;Scroll UP
F053 A5F2         693      Dw      Offset Scroll           ;Scroll DOWN
F055 F5F3         694      Dw      Offset ReadWrite        ;Read char and attrib at cursor
F057 F5F3         695      Dw      Offset ReadWrite        ;Write char and attrib at cursor
F059 F5F3         696      Dw      Offset ReadWrite        ;Write char at cursor
F05B CFF5         697      Dw      Offset SetColor         ;Set color and background
F05D FAF5         698      Dw      Offset WriteDot         ;Write a dot
F05F 3EF6         699      Dw      Offset ReadDot          ;Read a dot
F061 67F6         700      Dw      Offset WriteTTY         ;Write glass tty
F063 D9F6         701      Dw      Offset VideoState       ;Return current video state
                                702
F065              703      VideoDriver Proc Far
F065 FB           704      Sti                      ;Restore interrupts
F066 FC           705      Cld                      ;Clear direction
F067 55           706      Push Bp                  ;.....
F068 06           707      Push Es                  ;.
F069 1E           708      Push Ds                  ;.
F06A 56           709      Push Si                  ;. Save .
F06B 57           710      Push Di                  ;.
F06C 52           711      Push Dx                  ;. Registers .
F06D 51           712      Push Cx                  ;.
F06E 53           713      Push Bx                  ;.
F06F 50           714      Push Ax                  ;.....
F070 BB-----   R    715      Mov Bx, BiosDataArea     ;Load our segment
F073 8EDB        E    716      Mov Ds, Bx               ;...register
F075 8A1E0000    E    717      Mov Bl, Byte Ptr EquipFlag ;Get display type
F079 80E330      718      And Bl, 00110000B        ;...isolate bits
F07C 80FB30      719      Cmp Bl, 00110000B        ;...and test if monochrome
F07F BB-----   R    720      Mov Bx, ColorSeg         ;Pre-load Bx with color card seg
F082 7503        721      Jne SegNotMono           ;Jump if not monochrome
F084 BB-----   R    722      Mov Bx, MonoSeg          ;...else use mono segment
F087              723      SegNotMono:
F087 53           724      Push Bx                  ;Push video segment into stack
                                725
F088 8BEC        726      Mov Bp, Sp               ;Bp points at stack frame
F08A E87700      727      Call CommandDispatch     ;Perform the command
                                728
F08D 5E           729      Pop Si                   ;Toss video segment
F08E 58           730      Pop Ax                   ;.....
F08F 5B           731      Pop Bx                   ;.
F090 59           732      Pop Cx                   ;.
F091 5A           733      Pop Dx                   ;. Restore .
F092 5F           734      Pop Di                   ;.
F093 5E           735      Pop Si                   ;. Registers .
F094 1F           736      Pop Ds                   ;.
F095 07           737      Pop Es                   ;.
F096 5D           738      Pop Bp                   ;.....
F097 CF         739      Iret                    ;And return
                                740      VideoDriver Endp
                                741
                                742 +1 $Eject

```

```

LOC OBJ          LINE    SOURCE
                743
                744 ; *****
                745 ; * Mode Tables - Aligned with PC and Xt *
                746 ; *****
                747
-----
                748 ParameterStruc Struc
0000            749 Alpha40x25    Db    038H,028H,02DH,00AH,01FH,006H,019H,01CH
0008            750                Db    002H,007H,006H,007H,000H,000H,000H,000H
0010            751 Alpha80x25    Db    071H,050H,05AH,00AH,01FH,006H,019H,01CH
0018            752                Db    002H,007H,006H,007H,000H,000H,000H,000H
0020            753 Graphics      Db    038H,028H,02DH,00AH,07FH,006H,064H,070H
0028            754                Db    002H,001H,006H,007H,000H,000H,000H,000H
0030            755 AlphaMono    Db    061H,050H,052H,00FH,019H,006H,019H,019H
0038            756                Db    002H,00DH,00BH,00CH,000H,000H,000H,000H
0040            757 Mem40x25      Dw    2048
0042            758 Mem80x25      Dw    4096
0044            759 MemGraphics   Dw    16384
0046            760                Dw    16384
0048            761 NumCols      Db    028H,028H,050H,050H,028H,028H,050H,050H
0050            762 ModeSets     Db    02CH,028H,02DH,029H,02AH,02EH,01EH,029H
-----
                763 ParameterStruc Ends
                764 ParamOffStruc Struc
0000            765                Db    0 ;Offset in parameter structure for mode 0
0001            766                Db    0 ;Offset in parameter structure for mode 1
0002            767                Db    16 ;Offset in parameter structure for mode 2
0003            768                Db    16 ;Offset in parameter structure for mode 3
0004            769                Db    32 ;Offset in parameter structure for mode 4
0005            770                Db    32 ;Offset in parameter structure for mode 5
0006            771                Db    32 ;Offset in parameter structure for mode 6
0007            772                Db    48 ;Offset in parameter structure for mode 7
-----
                773 ParamOffStruc Ends
                774 ; Table definition - done under %NoList
                775
F004            776                Org    0F0A4H ;Align with PC and Xt
                777 +1 %NoList
                782
                783 +1 %Eject
    
```

```
LDC OBJ          LINE  SOURCE
                  784  ;
                  785  ;
                  786  ;
                  787  ;
F104              788  CommandDispatch Proc Near
F104 80FC0F       789  Cmp Ah,15 ;Is command in range?
F107 7601         790  Jbe ComInRange ;...jump if it is
F109 C3           791  Ret ;...else return
F10A              792  ComInRange:
F10A D0E4         793  Shl Ah,1 ;Multiply command by 2
F10C 8ADC         794  Mov Bl,Ah ;Put in index register
F10E 32FF         795  Xor Bh,Bh ;...and extend to word
F110 2EFA745F0   796  Jmp JumpTable[Bx] ;...and jump to routine
                  797  CommandDispatch Endp
                  798 +1 $Eject
```

```

LOC OBJ          LINE      SOURCE
                799      ;
                800      ;
                801      ;
                802      ;
F115             803      SetMode Proc Near          ;Change mode
                804
                805      ; Set ActiveCard Global Variable
                806
F115 A00000      E        807      Mov Al,Byte Ptr EquipFlag ;Get switches
F118 BAB403      808      Mov Dx,PortMonoIndex     ;...pre-load with mono card
F11B 2430        809      And Al,00110000B        ;Isolate
F11D 3C30        810      Cmp Al,00110000B        ;...and test switches
F11F B001        811      Mov Al,MonoResMode       ;...(null mode command)
F121 B307        812      Mov Bl,7                 ;Get internal mode for mono
F123 7407        813      Je SetModeJump1         ;...jump if monochrome
F125 BA5E02      814      Mov Bl,Mode              ;Get new mode into Bl
F128 B2D4        815      Mov Dl,Low(PortColorIndex) ;...else load color address
F12A FECA        816      Dec Al                   ;...modify null mode for color
F12C             817      SetModeJump1:
F12C 89160000    E        818      Mov ActiveCard,Dx        ;Save base address in global
                819
                820      ; Reset card to null mode
                821
F130 80C204      822      Add Dl,4                 ;...point at control register
F133 EE          823      Out Dx,Al               ;...and setup null mode
F134 881E0000    E        824      Mov CrtMode,Bl          ;Set mode
                825
                826      ; Point at 6845 parameters
                827
                828      Assume Ds:IntSegment ;Point ASM86 at int segment
F138 1E          829      Push Ds                 ;Save Ds
F139 33C0        830      Xor Ax,Ax               ;Go there
F13B 8ED8        831      Mov Ds,Ax               ;...ourselves with data segment
F13D C4360000    E        832      Les Si,VidParamsTrapAddr ;Point Ds:Si to parameters
F141 1F          833      Pop Ds                  ;Restore data segment
                834      Assume Ds:BiosDataArea ;...and tell ASM86
                835
                836      ; Pump parameters into 6845
                837
F142 32FF        838      Xor Bh,Bh               ;Form index into table (Bl=mode)
F144 53          839      Push Bx                 ;Save Bx=crt mode
F145 2E8A9FFCF0  840      Mov Bl,OffsetTable[Bx] ;Load the offset
F14A 03F3        841      Add Si,Bx               ;...and point at right list
F14C B91000      842      Mov Cx,16               ;Do 16 bytes
                843
                844      ; Ah is 6845 index, initialized to zero from above
                845
F14F             846      PumpModeLoop:
F14F 268A04      847      Mov Al,Es:[Si]          ;Get byte from table
F152 E8B205      848      Call Load6845           ;...send it
F155 FEC4        849      Inc Ah                  ;Increment 6845 index
F157 46          850      Inc Si                  ;Increment memory pointer
F158 E2F5        851      Loop PumpModeLoop      ;...and loop till 16 done
                852
                853      ; Clear the memory
                854
F15A 8B5E00      855      Mov Bx,VideoSegment     ;Set segment
F15D 8EC3        856      Mov Es,Bx               ;...Es points to video buffer
F15F BF0000      857      Mov Di,0                ;Byte pointer
F162 E88705      858      Call WhichMode          ;Which mode are we in?
F165 B90020      859      Mov Cx,8192             ;Memory size on color card
F168 B00000      860      Mov Ax,0                ;Clear for graphics
F16B 7208        861      Jc SetModeClr3         ;...carry set if graphics
F16D 7503        862      Jnz SetModeClr1        ;...Z set if monochrome
                863
F16F             864      SetModeClr2:
F16F B90008      865      Mov Cx,2048             ;Memory size on mono card
F172             866      SetModeClr1:
F172 B82007      867      Mov Ax,0000011100100000B ;Alpha mode space & attrib=7
F175             868      SetModeClr3:
F175 F3          869      Rep Stosw              ;Repeat and store data
F176 AB          870
                871      ; Set Mode from Table
                872

```


LOC	OBJ	LINE	SOURCE
F177	8B160000	E 873	Mov Dx,ActiveCard ;Get port address
F178	80C204	874	Add Dl,4 ;...and point at mode port
F17E	5B	875	Pop Bx ;Get CrtMode from stack
F17F	2E8A87F4F0	876	Mov Al,VidParamsPointer.ModeSets[Bx] ;Get mode from table
F184	EE	877	Out Dx,Al ;.....and output it
F185	A20000	E 878	Mov CrtModeSet,Al ;.....and seg global
		879	
		880	; Set overscan mode
		881	
F188	42	882	Inc Dx ;Point at register
F189	B030	883	Mov Al,00110000B ;Default value
F18B	80FB06	884	Cmp Bl,6 ;640x200 mode
F18E	7502	885	Jne OverscanHires ;...jump if is
F190	B03F	886	Mov Al,00111111B ;...else set special mode
F192		887	OverscanHires:
F192	A20000	E 888	Mov CrtPalette,Al ;...store it and
F195	EE	889	Out Dx,Al ;...send it
		890	
		891	; Init Some Globals
		892	
F196	33C0	893	Xor Ax,Ax ;Get a zero
F198	A30000	E 894	Mov CrtStart,Ax ;...Set start address
F19B	A20000	E 895	Mov ActivePage,Al ;...and active page
F19E	890800	896	Mov Cx,8 ;Clear cursor positions
F1A1	8F0000	E 897	Mov Di,Offset CursorPosn ;Point at them
F1A4		898	CursPosClr:
F1A4	8905	899	Mov [Di],Ax ;Clear one
F1A6	47	900	Inc Di ;...increment pointer
F1A7	E2FB	901	Loop CursPosClr ;...and loop
F1A9	C70600000706	E 902	Mov CursorMode,607H ;Cursor mode line 6-7
F1AF	2E8A87ECF0	903	Mov Al,VidParamsPointer.NumCols[Bx] ;Init CrtColumns from
F1B4	A30000	E 904	Mov CrtColumns,Ax ;.....table
F1B7	80E3FE	905	And Bl,11111110B ;Make mode even
F1BA	2E8B87E4F0	906	Mov Ax,VidParamsPointer.Mem40x25[Bx] ;Init size
F1BF	A30000	E 907	Mov CrtLength,Ax ;.....from table
F1C2	C3	908	Ret ;Return, init complete
		909	SetMode
		910 +1	\$Eject
			Endp

```

LOC OBJ          LINE    SOURCE
                911      ;
                912      ;
                913      ;
                914      ;
F1C3            915      SetCursorType Proc Near          ;Set cursor type
F1C3 8B4E06     916      Mov Cx,CursCommand      ;Get parameter
F1C6 890E0000   E 917      Mov CursorMode,Cx      ;Store new global value
F1CA B40A       918      Mov Ah,10              ;Address of 6845 Cursor spec.
F1CC E82F05     919      Call Load6845Double    ;...send it
F1CF C3         920      Ret                    ;...and return
                921      SetCursorType Endp
                922
                923      ;
                924      ;
                925      ;
                926      ;
F1D0            927      SetCursorPos Proc Near          ;Move cursor to position
F1D0 8A5E05     928      Mov Bl,DisplayPage     ;Get page parameter
F1D3 D0E3       929      Shl Bl,i              ;...multiply by 2
F1D5 32FF       930      Xor Bh,Bh             ;Clear the top
F1D7 8B4608     931      Mov Ax,SetRowCol      ;Get new row and column
F1DA 89870000   E 932      Mov CursorPosn[Bx],Ax ;...and store it
F1DE 381E0000   E 933      Cmp ActivePage,Bl     ;Page now on screen?
F1E2 7503       934      Jne SetCursorReturn   ;...jump if not
F1E4 E86305     935      Call LoadCursor       ;...else load the cursor
F1E7           936      SetCursorReturn:
F1E7 C3         937      Ret                    ;...to caller
                938      SetCursorPos Endp
                939
                940      ;
                941      ;
                942      ;
                943      ;
F1E8            944      ReadCursorPos Proc Near        ;Read current cursor position
F1E8 8A5E05     945      Mov Bl,DisplayPage     ;Get page parameter
F1EB D0E3       946      Shl Bl,i              ;...multiply by 2
F1ED 32FF       947      Xor Bh,Bh             ;Clear the top
F1EF 8B870000   E 948      Mov Ax,CursorPosn[Bx] ;Get cursor position
F1F3 894608     949      Mov ReadRowCol,Ax     ;...and store it in stack frame
F1F6 A10000     E 950      Mov Ax,CursorMode     ;Get cursor mode
F1F9 894606     951      Mov ReturnCursMode,Ax ;...and store it in stack frame
F1FC C3         952      Ret                    ;Then return
                953      ReadCursorPos Endp
                954 +1   $Eject
    
```

```

LOC OBJ          LINE    SOURCE
                955      ;
                956      ;
                957      ;
                958      ;
                959      Skew      Db      3,3,5,5,3,3,4

F1FD 03
F1FE 03
F1FF 05
F200 05
F201 03
F202 03
F203 04

                960
F204          961      ReadLpPos  Proc      Near          ;Read Light pen position
F204 8B160000  E          962      Mov      Dx,ActiveCard ;Point at 6845
F208 80C205          963      Add     D1,6           ;...and the status register
F20B C6460300          964      Mov     LpStatus,0     ;Returned status = not pressed
F20F EC            965      In      A1,Dx          ;Get status
F210 A004          966      Test   A1,00000100B   ;...and test bit
F212 7468          967      Jz      ResetTrigRtn  ;If not set, return and reset
F214 A002          968      Test   A1,00000010B   ;If not trigger, return no reset
F216 7501          969      Jnz    LpPosJmp1      ;...jump over return if trigger
F218 C3            970      Ret                    ;...return not triggered
F219          971      LpPosJmp1:
F219 B410          972      Mov     Ah,16          ;Fetch value from 6845
F21B E83D05          973      Call   Fetch6845Double ;...into Cx
F21E 8A1E0000  E          974      Mov     Bl,CrtMode     ;Load Bx with mode
F222 8ACB          975      Mov     Cl,B1          ;...and keep it,
F224 32FF          976      Xor     Bh,Bh          ;...then
F226 2E8A9FFDF1      977      Mov     Bl,Skew[Bx]    ;...load it with skew value
F22B 2BCB          978      Sub     Cx,Bx          ;Subtract skew
F22D 7902          979      Jns    LpPosJmp2      ;Jump if negative
F22F 33C0          980      Xor     Ax,Ax          ;...Don't go past origin
F231          981      LpPosJmp2:
F231 E88804          982      Call   WhichMode      ;Which mode are we in?
F234 7325          983      Jnc    LpAlpha         ;...jump if alpha mode
                984
                985      ;      Graphics Light Pen
                986
F236 B528          987      Mov     Ch,40          ;Divide into 40 columns
F238 F6F2          988      Div     D1             ;...per row
                989
                990      ;      A1=0-99 rows   Ah=0-39 columns
                991
F23A 8ADC          992      Mov     Bl,Ah          ;Return pixels in Bx
F23C 32FF          993      Xor     Bh,Bh          ;...calculated as 8
F23E B103          994      Mov     Cl,3           ;...times
F240 D3E3          995      Shl    Bx,Cl          ;...column value
F242 8AEB          996      Mov     Ch,A1          ;Scan line in Ch
F244 D0E5          997      Shl    Ch,1           ;...double for interlace
F246 8AD4          998      Mov     D1,Ah          ;Char column in D1
F248 8AF0          999      Mov     Dh,A1          ;Char row in Dh
F24A D0EE          1000     Shr     Dh,1           ;Get char row value in
F24C D0EE          1001     Shr     Dh,1           ;...range (div by 4)
                1002
                1003     ;      Adjust for Hi-Res
                1004
F24E 803E000006  E          1005     Cmp     CrtMode,6      ;Hi-Res?
F253 751A          1006     Jne    LpReturnValues ;...jump if no and return
F255 D0E2          1007     Shl    D1,1           ;Column # * 2
F257 D1E3          1008     Shl    Bx,1           ;Pixels * 2
F259 EB14          1009     Jmp    Short LpReturnValues ;Return
                1010
                1011     ;      Alpha Light Pen
                1012
F25B          1013     LpAlpha:
F25B F6360000  E          1014     Div     Byte Ptr CrtColumns ;Divide address by columns
F25F 86C4          1015     Xchg   A1,Ah          ;...Dh = rows
F261 8BD0          1016     Mov     Dx,Ax          ;...D1 = columns
F263 B103          1017     Mov     Cl,3           ;Calculate scan lines
F265 D2E4          1018     Shl    Ah,Cl          ;...from rows * 8
F267 8AEC          1019     Mov     Ch,Ah          ;...and return it in Ch
F269 8AD8          1020     Mov     Bl,A1          ;Calculate pixels
F26B 32FF          1021     Xor     Bh,Bh          ;...from rows * 8
F26D D3E3          1022     Shl    Bx,Cl          ;...and return in Bx
                1023

```

LOC	OBJ	LINE	SOURCE
		1024	; Return with Values
		1025	
		1026	LpReturnValues:
F26F	C6460301	1027	Mov LpStatus,1 ;Signal trigger
F273	895608	1028	Mov LpRowCol,Dx ;Store Dx
F276	895E04	1029	Mov LpPixel,Bx ;Store Bx
F279	886E07	1030	Mov LpRasterLine,Ch ;Store Ch
F27C		1031	ResetTrigRtn:
F27C	8B160000	1032	Mov Dx,ActiveCard ;Clear Lp Flip Flop
F280	83C207	1033	Add Dx,7 ;...before returning
F283	EE	1034	Out Dx,A1 ;...(data irrelevant)
F284	C3	1035	Ret ;...and return normally
		1036	ReadLpPos Endp
		1037 +1	\$Eject

LOC	OBJ	LINE	SOURCE
		1038	;
		1039	;
		1040	;
		1041	*****
		1042	ActivatePage Proc Near ;Activate new page
F285	8A4602	1043	Mov Al,NewPage ;Get the new page
F288	A20000	1044	Mov ActivePage,Al ;...store it and
F28B	32E4	1045	Xor Ah,Ah ;...put it into Ax
F28D	50	1046	Push Ax ;Save for cursor positioning
F28E	8B1E0000	1047	Mov Bx,CrtLength ;Multiply it by the
F292	F7E3	1048	Mul Bx ;...length of a page
F294	A30000	1049	Mov CrtStart,Ax ;Store start addr of page
F297	D1E8	1050	Shr Ax,1 ;Get character only length
F299	8BC8	1051	Mov Cx,Ax ;Pump it into
F29B	B40C	1052	Mov Ah,12 ;...6845
F29D	E85E04	1053	Call Load6845Double ;...double register 12
F2A0	5B	1054	Pop Bx ;Restore page to Bx
F2A1	E8A604	1055	Call LoadCursor ;...and go load the cursor
F2A4	C3	1056	Ret ;...then return
		1057	ActivatePage Endp
		1058 +1	\$Eject

```

LOC OBJ          LINE    SOURCE
1059             ;
1060             ;
1061             ;
1062             ;
F2A5             1063    Scroll Proc Near ;Scroll Down
F2A5 E84404      1064    Call WhichMode ;Graphics or Alpha?
F2A8 7303        1065    Jnc ScrollAlpha ;...jump if alpha
F2AA E99D00      1066    Jmp ScrollGraph ;...else jump to graphics
1067             Scroll Endp
1068             ;
1069             ;
1070             ;
1071             ;
1072             ;
F2AD             1073    ScrollAlpha Proc Near
F2AD FC          1074    Cld ;Clear direction
F2AE 803E000002  E 1075    Cmp CrtMode,2 ;40 x 25?
F2B3 7215        1076    Jb NoScrollWait ;Jump if not
F2B5 803E000003  E 1077    Cmp CrtMode,3 ;80 x 25?
F2BA 770E        1078    Ja NoScrollWait ;Jump if it isn't
F2BC             1079    VerticalWait:
F2BC BADA03      1080    Mov Dx,PortColorStatus ;Point at status port
F2BF             1081    VWaitLoop:
F2BF EC          1082    In Al,Dx ;Get status
F2C0 A808        1083    Test Al,00001000B ;...wait for vertical retrace
F2C2 74FB        1084    Jz VWaitLoop ;Loop till retrace
F2C4 BAD803      1085    Mov Dx,PortColorMode ;Turn off video
F2C7 B025        1086    Mov Al,00100101B ;...while
F2C9 EE          1087    Out Dx,Al ;...scrolling
1088             ;
1089             ; Calculate window start address
1090             ;
F2CA             1091    NoScrollWait:
F2CA 8B4608      1092    Mov Ax,ScrollLower ;Get lower right hand xy
F2CD 50           1093    Push Ax ;...and save it
F2CE 807E0307    1094    Cmp Command,7 ;Down?
F2D2 7403        1095    Je ScrollCont2 ;...jump if down
F2D4 8B4606      1096    Mov Ax,ScrollUpper ;...else start at upper left
F2D7             1097    ScrollCont2:
F2D7 E89B04      1098    Call Convert ;Convert it to a linear address
F2DA 03060000    E 1099    Add Ax,CrtStart ;...on current page
F2DE 8BF0        1100    Mov Si,Ax ;Store in source register
F2E0 8BF8        1101    Mov Di,Ax ;...and destination register
1102             ;
1103             ; Calculate # of rows and # of columns
1104             ;
F2E2 5A           1105    Pop Dx ;Restore upper left
F2E3 2B5606      1106    Sub Dx,ScrollUpper ;Calculate width and height
F2E6 81C20101    1107    Add Dx,0101H ;...adding 1 to form a count
1108             ;
1109             ; Get Linear difference between rows
1110             ;
F2EA 8B1E0000    E 1111    Mov Bx,CrtColumns ;# of columns
F2EE D1E3        1112    Shl Bx,1 ;...double for attributes
1113             ;
1114             ; Calculate To / From Offset
1115             ;
F2F0 1E          1116    Push Ds ;Save Ds
F2F1 8A4602      1117    Mov Al,ScrollNumRows ;Number of rows
F2F4 F6E3        1118    Mul Bl ;...difference between rows
1119             ;
1120             ; Load Source and Destination Segment Registers
1121             ;
F2F6 8B4E00      1122    Assume Ds:Nothing ;All ref. must be anonymous
F2F9 8EC1        1123    Mov Cx,VideoSegment ;Get the segment
F2FB 8ED9        1124    Mov Es,Cx ;...destination segment
1125             ;
1126             ;
1127             ; Set sign of offsets based on direction
1128             ;
F2FD 807E0306    1129    Cmp Command,6 ;Up?
F301 7405        1130    Je ScrollCont1 ;...jump if up
F303 F7D8        1131    Neg Ax ;...else make Ax negative
F305 F7DB        1132    Neg Bx ;...and make Bx negative
F307 FD          1133    Std ;Set direction

```

```

LOC OBJ          LINE    SOURCE
F308             1134    ScrollCont1:
1135
1136             ;
1137             ; *****
1138             ; * At this point: *
1139             ; * * * * *
1140             ; * Dh = # of rows *
1141             ; * Dl = # of cols *
1142             ; * Ax = Linear to / from offset *
1143             ; * Bx = Differences between rows *
1144             ; * Si & Di = linear start address *
1145             ; * Es & Ds = Video segment *
1146             ; *****
F308 8A4E02      1147    Mov     C1,ScrollNumRows ;Number of rows
F308 0AC9        1148    Or     C1,C1             ;...affect flags
F308 741A        1149    Jz     BlankOnly        ;Jump if blanking only
F30F 03F0        1150    Add    Si,Ax             ;Fix dif. between src and dest
F311 2A7602      1151    Sub    Dh,ScrollNumRows ;...Calculate rows to move
F314            1152    MoveLoop:
F314 32ED        1153    Xor    Ch,Ch             ;Clear counter top
F316 8ACA        1154    Mov    C1,Dl             ;Load # of columns to move
F318 57          1155    Push   Di                ;Save pointers
F319 56          1156    Push   Si                ;...that will be incremented
F31A F3         1157    Rep    Movsw              ;Move the row
F31B A5
F31C 5E          1158    Pop    Si                ;Restore pointers
F31D 5F          1159    Pop    Di                ;...to original value
F31E 03F3        1160    Add    Si,Bx             ;Point at next row
F320 03FB        1161    Add    Di,Bx             ;...both source and destination
F322 FECE        1162    Dec    Dh                ;...count down
F324 75EE        1163    Jnz    MoveLoop         ;...and loop till all moved
F326 8A7602      1164    Mov    Dh,ScrollNumRows ;Load count of rows to blank
1165
1166             ; Clear rows at bottom or top
1167
F329            1168    BlankOnly:
F329 32ED        1169    Xor    Ch,Ch             ;Clear counter top
F32B 8A6605      1170    Mov    Ah,ScrollAttrib   ;Get the attribute
F32E B020        1171    Mov    Al,' '           ;...and a space
F330            1172    BlankBotLoop:
F330 8ACA        1173    Mov    C1,Dl             ;Dl has # of columns in block
F332 57          1174    Push   Di                ;Save pointer
F333 F3         1175    Rep    Stosw              ;Clear row
F334 AB
F335 5F          1176    Pop    Di                ;Restore pointer
F336 03FB        1177    Add    Di,Bx             ;...bump to next row
F338 FECE        1178    Dec    Dh                ;Decrement count and
F33A 75F4        1179    Jnz    BlankBotLoop     ;...loop till done
F33C 1F          1180    Pop    Ds                ;Restore Bios data segment
1181    Assume Ds:BiosDataArea ;...and tell ASM86
F33D EBAC03      1182    Call   WhichMode        ;Mono card?
F340 7407        1183    Jz     ScrollReturn     ;...return if so
F342 A00000      1184    Mov    Al,CrtModeSet    ;Get current mode
F345 BAD803      1185    Mov    Dx,PortColorMode ;...and restore it
F348 EE          1186    Out   Dx,Al             ;...to re-enable video
F349            1187    ScrollReturn:
F349 C3          1188    Ret                     ;and return
1189    ScrollAlpha Endp
1190 +1 $Eject

```

```

LOC OBJ          LINE    SOURCE
1191            ;          *****
1192            ;          * Common Graphics Scroll *
1193            ;          *****
1194
F34A            1195    ScrollGraph Proc Near
F34A FC         1196            Cld                ;Clear direction for scroll up
F34B 8B4608     1197            Mov Ax,ScrollLower ;Get upper right hand corner xy
F34E 50         1198            Push Ax             ;Save upper right hand corner
F34F 807E0307   1199            Cmp Command,7     ;Scroll Down?
F353 7403       1200            Je GraphDnl        ;...jump if yes
F355 8B4606     1201            Mov Ax,ScrollUpper ;...else reload upper left
F358            1202    GraphDnl:
F358 E82B04     1203            Call ConvertGraph ;Calculate linear addr for HiRes
F35B 8BF8       1204            Mov Di,Ax          ;This is destination
1205
1206            ; Calculate Dh = # of rows, Dl = # of columns in window
1207
F35D 5A         1208            Pop Dx             ;Dx = Upper left hand corner
F35E 2B5606     1209            Sub Dx,ScrollUpper ;Calculate offset
F361 81C20101   1210            Add Dx,0101H      ;...and make it a length
F365 D0E6       1211            Shl Dh,1           ;In graphics, 4 even field rows
F367 D0E6       1212            Shl Dh,1           ;...per char. row, so shift it
1213
1214            ; Adjust for medium resolution
1215
F369 8A4603     1216            Mov Al,Command    ;Get command
F36C 803E000006 E 1217            Cmp CrtMode,6    ;Is it HiRes? (mode 6)
F371 7409       1218            Je HiresSkip      ;...jump if so
F373 D0E2       1219            Shl Dl,1          ;Twice as many bytes per column
F375 D1E7       1220            Shl Di,1          ;...offset correction for LoRes
F377 3C07       1221            Cmp Al,7          ;Down?
F379 7501       1222            Jne HiresSkip     ;...jump if not
F37B 47         1223            Inc Di            ;...bump to last byte
1224
1225            ; Adjust # of rows to scroll to even field scan lines
1226
F37C            1227    HiresSkip:
F37C 3C07       1228            Cmp Al,7          ;Down?
F37E 7504       1229            Jne DownSkip1    ;...jump if not
F380 81C7F000    1230            Add Di,240        ;Go to last row of dots if down
F384            1231    DownSkip1:
F384 8A5E02     1232            Mov Bl,ScrollNumRows ;Get character rows
F387 D0E3       1233            Shl Bl,1          ;...times 2
F389 D0E3       1234            Shl Bl,1          ;...times 2 again
1235
1236            ; Calculate byte length offset from source to dest
1237
F38B 53         1238            Push Bx           ;Save num scan lines for blank
F38C 2AF3       1239            Sub Dh,B1         ;Calculate # scans to move
F38E B050       1240            Mov Al,80         ;Calc by multiplying # scans
F390 F6E3       1241            Mul Bl            ;...times 80 bytes per scan
1242
1243            ; Set sign of offsets based on direction
1244
F392 8B001F     1245            Mov Bx,2000H-80  ;Pre-load 'UP' row offset
F395 807E0306   1246            Cmp Command,6    ;Up?
F399 7406       1247            Je ScrollGrph1   ;...jump if up
F39B F7D8       1248            Neg Ax            ;...else make Ax negative
F39D 8B5020     1249            Mov Bx,2000H+80 ;Scroll down goes up in memory
F3A0 FD         1250            Std              ;Set direction
F3A1            1251    ScrollGrph1:
1252            ; Set to / from offset
1253
F3A1 8BF7       1254            Mov Si,Di         ;Get destination and figure
F3A3 03F0       1255            Add Si,Ax         ;...Si/Di diff. by offset
1256
1257            ; Test if blanking function only
1258            ;
1259            ; Load Video Segments
1260
F3A5 58         1261            Pop Ax            ;Get # of scans to scroll
F3A6 0AC0       1262            Or Al,Al         ;...affect Z. Blank only? ---
F3A8 8B4E00     1263            Mov Cx,VideoSegment ;Load segment registers
F3AB 8ED9       1264            Mov Ds,Cx        ;...with
F3AD 8EC1       1265            Mov Es,Cx        ;...video segment

```


LOC	OBJ	LINE	SOURCE
F3AF	7428	1266	Jz GraphBlankOnly ;...jump if blanking only (←
F3B1	50	1267	Push Ax ;Save # of scans to scroll
F3B2	32ED	1268	Xor Ch,Ch ;Clear length upper byte
		1269	
		1270	; Move Loop
F3B4		1271	MoveGraph: ;
F3B4	32ED	1272	Xor Ch,Ch ;Clear counter top
F3B6	8ACA	1273	Mov Cl,Dl ;Get # of bytes (columns)
F3B8	56	1274	Push Si ;Save pointers
F3B9	57	1275	Push Di ;...for future use
F3BA	F3	1276	Rep Movsb ;Move even row
F3BB	A4		
F3BC	5F	1277	Pop Di ;Restore pointers
F3BD	5E	1278	Pop Si ;...and
F3BE	81C60020	1279	Add Si,2000H ;...point them to
F3C2	81C70020	1280	Add Di,2000H ;...odd field
F3C6	8ACA	1281	Mov Cl,Dl ;Get # of bytes (columns)
F3C8	56	1282	Push Si ;Save pointers
F3C9	57	1283	Push Di ;...for future use
F3CA	F3	1284	Rep Movsb ;Move odd row
F3CB	A4		
F3CC	5F	1285	Pop Di ;Restore pointers
F3CD	5E	1286	Pop Si ;...and 2 rows moved
F3CE	2BF3	1287	Sub Si,Bx ;Back to even field-1 row
F3D0	2BFB	1288	Sub Di,Bx ;...for both pointers
F3D2	FECE	1289	Dec Dh ;Decrement count
F3D4	75DE	1290	Jnz MoveGraph ;...and loop till moved
F3D6	58	1291	Pop Ax ;Restore # of scans to scroll
F3D7	8AF0	1292	Mov Dh,A1 ;Count using Dh
		1293	
		1294	; Blank lines at bottom or top
		1295	
F3D9		1296	GraphBlankOnly: ;
F3D9	8A4605	1297	Mov Al,ScrollAttrib ;Get the data
F3DC	32ED	1298	Xor Ch,Ch ;Clear counter top
		1299	
		1300	; Clear Loop
		1301	
F3DE		1302	ClearGraphLoop: ;
F3DE	8ACA	1303	Mov Cl,Dl ;Get # of columns
F3E0	57	1304	Push Di ;Save pointer
F3E1	F3	1305	Rep Stosb ;Store data on row in columns
F3E2	AA		
F3E3	5F	1306	Pop Di ;Restore pointer
F3E4	81C70020	1307	Add Di,2000H ;Point at odd field
F3E8	8ACA	1308	Mov Cl,Dl ;Get # of columns
F3EA	57	1309	Push Di ;Save pointer again
F3EB	F3	1310	Rep Stosb ;Store data on row in columns
F3EC	AA		
F3ED	5F	1311	Pop Di ;Restore pointer
F3EE	2BFB	1312	Sub Di,Bx ;Point at next row to clear
F3F0	FECE	1313	Dec Dh ;Decrement count of scans to do
F3F2	75EA	1314	Jnz ClearGraphLoop ;...and loop till clear
F3F4	C3	1315	Ret ;...then return
		1316	ScrollGraph Endp
		1317	+1 \$Eject

```

LOC OBJ          LINE    SOURCE
                1318    ;
                1319    ;
                1320    ;
                1321    ;
F3F5            1322    ReadWrite Proc Near
F3F5 E8F402     1323    Call WhichMode ;Which mode are we in?
F3F8 7270      1324    Jc ReadWriteGraph ;...jump if graphics
                1325
                1326    ; Convert xy to linear
                1327
F3FA 8A5E05     1328    Mov Bl,DisplayPage ;Get page number
F3FD 32FF      1329    Xor Bh,Bh ;...into Bx
F3FF 53        1330    Push Bx ;and save it
F400 E81303     1331    Call ConvertCursor ;Convert cursor to linear
F403 8BF8      1332    Mov Di,Ax ;Save result in Di
F405 58        1333    Pop Ax ;Restore page #
F406 F7260000  E 1334    Mul CrtLength ;...multiply page x length
F40A 03F8      1335    Add Di,Ax ;...and add in converted value
F40C 8BF7      1336    Mov Si,Di ;Read and write pointers
                1337
                1338    ; Get port address
                1339
F40E 8B160000  E 1340    Mov Dx,ActiveCard ;Get port address
F412 83C206     1341    Add Dx,6 ;...and point at status reg.
                1342
                1343    ; Save segments
                1344
F415 1E        1345    Push Ds ;Save data segment
F416 8B5E00     1346    Mov Bx,VideoSegment ;...get video segment
F419 8ED8      1347    Mov Ds,Bx ;...into data segment
F41B 8EC3      1348    Mov Es,Bx ;...and extra segment
                1349    Assume Ds:Nothing ;We know nothing about segment
                1350
                1351    ; Read / Write Split
                1352
F41D 8A4603     1353    Mov Al,Command ;Get command
F420 3C08      1354    Cmp Al,8 ;Command = Read?
F422 7514      1355    Jne WriteChar ;...if not then jump
                1356
                1357    ; Read Branch
                1358
F424            1359    HWaitRead:
F424 EC        1360    In Al,Dx ;Get status
F425 A001      1361    Test Al,1 ;...horiz pulse?
F427 75FB      1362    Jnz HWaitRead ;...jump and wait for it if not
F429 FA        1363    Cli ;Tight timing here
                1364
F42A            1365    HWaitRead2:
F42A EC        1365    In Al,Dx ;Get status
F42B A001      1366    Test Al,1 ;...gone back hi?
F42D 74F5      1367    Jz HWaitRead ;...wait till it does
F42F AD        1368    Lodsw ;...read it quick
F430 1F        1369    Pop Ds ;Restore Ds
                1370    Assume Ds:BiosDataArea ;...and tell ASM86
F431 804602     1371    Mov Char,Al ;...store char for return
F434 806603     1372    Mov AttribOut,Ah ;...then store attrib
F437 C3        1373    Ret ;...and return
                1374
                1375    ; Write Branch
                1376
F438            1377    WriteChar:
F438 8A5E02     1378    Mov Bl,Char ;Get Char
F43B 8A7E04     1379    Mov Bh,AttribIn ;...and attribute
F43E 8B4E06     1380    Mov Cx,CharCount ;Repeat count
                1381
                1382    ; Char only or char/attrib write split
                1383
F441 3C0A      1384    Cmp Al,10 ;Command = char only?
F443 7412      1385    Je CharOnly ;...jump if yes
                1386
                1387    ; Char and attrib branch
                1388
F445            1389    HWaitWrite1:
F445 EC        1390    In Al,Dx ;Get status
F446 A001      1391    Test Al,1 ;...horiz pulse?
F448 75FB      1392    Jnz HWaitWrite1 ;...jump and wait for it if not

```

LOC	OBJ	LINE	SOURCE	
F44A	FA	1393		Cli ;Tight timing here
F44B		1394	HWaitWrite2:	
F44B	EC	1395		In Al,Dx ;Get status
F44C	A801	1396		Test Al,1 ;...gone back hi?
F44E	74FB	1397		Jz HWaitWrite2 ;...wait till it does
F450	8BC3	1398		Mov Ax,Bx ;Get char and attribute
F452	AB	1399		Stosw ;Write char and attrib
F453	E2F0	1400		Loop HWaitWrite1 ;...do for repeat count
F455	1F	1401		Pop Ds ;Restore Ds
		1402		Assume Ds:BiosDataArea ;...and tell ASM86
F456	C3	1403		Ret ;...and return
		1404		
		1405		; Char only branch
		1406		
F457		1407	CharOnly:	
F457		1408	HWaitWrite3:	
F457	EC	1409		In Al,Dx ;Get status
F458	A801	1410		Test Al,1 ;...horiz pulse?
F45A	75FB	1411		Jnz HWaitWrite3 ;...jump and wait for it if not
F45C	FA	1412		Cli ;Tight timing here
F45D		1413	HWaitWrite4:	
F45D	EC	1414		In Al,Dx ;Get status
F45E	A801	1415		Test Al,1 ;...gone back hi?
F460	74FB	1416		Jz HWaitWrite4 ;...wait till it does
F462	8AC3	1417		Mov Al,B1 ;Get char back
F464	AA	1418		Stosb ;Write char only
F465	47	1419		Inc Di ;...skip attribute
F466	E2EF	1420		Loop HWaitWrite3 ;...and loop
F468	1F	1421		Pop Ds ;Restore Ds
		1422		Assume Ds:BiosDataArea ;...and tell ASM86
F469	C3	1423		Ret ;...and return
		1424	ReadWrite	
		1425	+1 %Eject	Endp

```

LOC OBJ          LINE    SOURCE
                1426    ;
                1427    ; *****
                1428    ; * Read Write Graphics *
                1429    ; *****
F46A             1430    ReadWriteGraph Proc Near
F46A 807E0308    1431    Cmp Command,8 ;Read?
F46E 7503        1432    Jne Writegraph ;...jump if not and write
F470 E9B800      1433    Jmp ReadGraph ;...else read
                1434    ReadWriteGraph Endp
                1435
                1436    ; *****
                1437    ; * Write Graphics *
                1438    ; *****
                1439
F473             1440    WriteGraph Proc Near
                1441    ; Convert Cursor position to linear
                1442
F473 A10000      E 1443    Mov Ax,CursorPosn[0] ;Get page 0 cursor posn
F476 E80D03      1444    Call ConvertGraph ;Convert to linear HiRes
F479 8BF8        1445    Mov Di,Ax ;...and use as destination
                1446
                1447    ; Determine image pointer
                1448
F47B 1E         1449    Push Ds ;Save data segment
F47C 8A4602      1450    Mov Al,Char ;Get char
F47F 32E4        1451    Xor Ah,Ah ;...into Ax
F481 0AC0        1452    Or Al,Al ;Affect flags
F483 7607        1453    Js UserImages ;...and jump if upper bit
                1454
                1455    ; Row images
                1456
F485 8CCA        1457    Mov Dx,Cs ;Row images in code segment
F487 BE0000      E 1458    Mov Si,Offset VideoGraphicsPointer ;Images = Dx:Si
F48A EB0C        1459    Jmp Short RWGCont1 ;Jump over
                1460
                1461    UserImages:
F48C 247F        1462    And Al,01111111B ;Strip upper bit, conv. to word
F48E 33DB        1463    Xor Bx,Bx ;...point at int segment
F490 8EDB        1464    Mov Ds,Bx ;...to find pointer
                1465    Assume Ds:IntSegment ;...to user images
F492 C5360000    E 1466    Lds Si,VideoGraphicsTrapAddr ;User images at Ds:Si
F496 8CDA        1467    Mov Dx,Ds ;.....make it Dx:Si
F498             1468    RWGCont1:
                1469    Assume Ds:BiosDataArea ;Tell ASM86 data segment
F498 1F         1470    Pop Ds ;Restore our segment
                1471
                1472    ; Figure offset into table
                1473
F499 B103        1474    Mov Cl,3 ;Multiply by 8 rows of
F49B D3E0        1475    Shl Ax,Cl ;...dots per page
F49D 03F0        1476    Add Si,Ax ;Add in image pointer
                1477
                1478    ; Load Video Segment Register
                1479
F49F 8B4600      1480    Mov Ax,VideoSegment ;Get segment pointer
F4A2 8EC0        1481    Mov Es,Ax ;...and load it
                1482
                1483    ; Load character count and toggle bit
                1484
F4A4 8B4E06      1485    Mov Cx,CharCount ;Character count
                1486
                1487    ; Mode Split and Load Segment register
                1488
F4A7 803E00006  E 1489    Cmp CrtMode,6 ;HiRes?
F4AC 1E         1490    Push Ds ;Save data segment
F4AD 8EDA        1491    Mov Ds,Dx ;Load Data Segment
                1492    Assume Ds:Nothing ;We know nothink
F4AF 7451        1493    Je HiResWrite ;...jump if yes
                1494
                1495    ; Medium Resolution Write
                1496
F4B1 D1E7        1497    Shl Di,1 ;Med res is 2 bytes per char
                1498
                1499    ; Expand color into Ax
                1500

```

LOC	OBJ	LINE	SOURCE
F4B3	8A4604	1501	Mov Al,AttribIn ;Get color
F4B6	250300	1502	And Ax,0000000000000011B ;...and isolate
F4B9	8B5555	1503	Mov Bx,0101010101010101B ;Magic multiplier
F4BC	F7E3	1504	Mul Bx ;Replicate bits across Ax
F4BE	8BD0	1505	Mov Dx,Ax ;...and keep it in Dx
F4C0	8A5E04	1506	Mov Bl,AttribIn ;Get the color attribute
		1507	
		1508	
		1509	; Ax = not used
		1510	; Bh = not used
		1511	; Bl = attribute
		1512	; Cx = character count
		1513	; Dx = expanded attribute
		1514	; Es:Di = screen pointer
		1515	; Ds:Si = image pointer
		1516	
		1517	; Write (Cx) Characters
		1518	
F4C3		1519	RepeatWrMed:
F4C3	B708	1520	Mov Bh,8 ;Write 4 odd/even pairs of rows
F4C5	57	1521	Push Di ;Save pointers
F4C6	56	1522	Push Si ;...for next write
		1523	
		1524	; Move dots
		1525	
F4C7		1526	MedResWrLoop:
F4C7	AC	1527	Lodsb ;Get even row of dots [Si]
		1528	
		1529	; Double all bits
		1530	
F4C8	51	1531	Push Cx ;Save Cx
F4C9	53	1532	Push Bx ;Save Bx (room to work)
F4CA	33DB	1533	Xor Bx,Bx ;Clear accumulator
F4CC	890800	1534	Mov Cx,8 ;8 bits to double
F4CF		1535	DoubleUp:
F4CF	D0E8	1536	Shr Al,1 ;Shift dots Rt into carry bit
F4D1	D1DB	1537	Rcr Bx,1 ;Shift accum Rt from carry
F4D3	D1FB	1538	Sar Bx,1 ;Shift accum rt and rep. bit 15
F4D5	E2F8	1539	Loop DoubleUp ;Loop till 8 bits doubled
F4D7	8BC3	1540	Mov Ax,Bx ;Get shift accum back to Ax
F4D9	5B	1541	Pop Bx ;Restore pushed registers
F4DA	59	1542	Pop Cx ;...from above
F4DB	23C2	1543	And Ax,Dx ;Convert color
F4DD	86E0	1544	Xchg Ah,Al ;Hi byte lo byte
F4DF	0ADB	1545	Or Bl,Bl ;...is attrib upper bit set?
F4E1	7903	1546	Jns MedXorSkip ;...jump if it is
F4E3	263305	1547	Xor Ax,Es:[Di] ;Xor source with destination
F4E6		1548	MedXorSkip:
F4E6	268905	1549	Mov Es:[Di],Ax ;Put the row of dots [Di]
F4E9	81F70020	1550	Xor Di,2000H ;Toggle odd/even field bit
F4ED	F7C70020	1551	Test Di,2000H ;Even field?
F4F1	7503	1552	Jnz OddFieldWrMed ;...jump if not
F4F3	83C750	1553	Add Di,80 ;Increment to next row
F4F6		1554	OddFieldWrMed:
F4F6	FECF	1555	Dec Bh ;Decrement count
F4F8	75CD	1556	Jnz MedResWrLoop ;...and loop till 1 char done
F4FA	5E	1557	Pop Si ;Restore pointers
F4FB	5F	1558	Pop Di ;...for next write
F4FC	47	1559	Inc Di ;Bump to next video position
F4FD	47	1560	Inc Di ;...for repeat loop
F4FE	E2C3	1561	Loop RepeatWrMed ;Loop till Cx written
F500	1F	1562	Pop Ds ;Restore data segment
		1563	Assume Ds:BiosDataArea ;Tell ASM86 data segment
F501	C3	1564	Ret ;...then return
		1565	
		1566	; Hi Resolution Write
		1567	
F502		1568	HiResWrite:
F502	8A5E04	1569	Mov Bl,AttribIn ;Get the color attribute
F505	BA0020	1570	Mov Dx,2000H ;Odd even bit
		1571	
		1572	; Write (Cx) Characters
F508		1573	RepeatWrLoop:
F508	B708	1574	Mov Bh,8 ;Write 4 odd/even pairs of rows
F50A	57	1575	Push Di ;Save pointers

LOC	OBJ	LINE	SOURCE		
F50B	56	1576		Push	Si ;...for next write
		1577			
		1578		;	Move dots
		1579			
F50C		1580	HiResWrLoop:		
F50C	AC	1581		Lodsb	;Get even row of dots [Si]
F50D	0ADB	1582		Or	B1,B1 ;...is attrib upper bit set?
F50F	7903	1583		Jns	XorSkip ;...jump if it is
F511	263205	1584		Xor	Al,Es:[Di] ;Xor source with destination
F514		1585	XorSkip:		
F514	268805	1586		Mov	Es:[Di],Al ;Put the row of dots [Di]
F517	33FA	1587		Xor	Di,Dx ;Toggle odd/even field bit
F519	85FA	1588		Test	Di,Dx ;Even field?
F51B	7503	1589		Jnz	OddFieldWr ;...jump if not
F51D	83C750	1590		Add	Di,80 ;Increment to next row
F520		1591	OddFieldWr:		
F520	FECF	1592		Dec	Bh ;Decrement count
F522	75E8	1593		Jnz	HiResWrLoop ;...and loop till 1 char done
F524	5E	1594		Pop	Si ;Restore pointers
F525	5F	1595		Pop	Di ;...for next write
F526	47	1596		Inc	Di ;Bump to next video position
F527	E2DF	1597		Loop	RepeatWrLoop ;Loop till Cx written
F529	1F	1598		Pop	Ds ;Restore data segment
		1599		Assume	Ds:BiosDataArea ;Tell ASM86 data segment
F52A	C3	1600		Ret	;...then return
		1601	WriteGraph	Endp	
		1602	+1	\$Eject	

```

LOC OBJ                LINE    SOURCE
1603                ;
1604                ; * Read Graphics *
1605                ;
1606                ;
F52B                1607    ReadGraph    Proc    Near
F52B FC                1608                Cld
F52C A10000            E    1609                Mov    Ax,CursorPosn[0] ;Positive direction
F52F E85402            1610                Call   ConvertGraph    ;Get page 0 cursor posn
F532 8BF0              1611                Mov    Si,Ax            ;Convert to linear HiRes
F534 83EC08            1612                Sub    Sp,8            ;...and use as source
F537 8BFC              1613                Mov    Di,Sp           ;Make room in stack for area
1614                ;
1615                ; Split for Med / Hi resolution
1616                ;
F539 803E000006        E    1617                Cmp    CrtMode,6      ;HiRes?
1618                ;
1619                ; Load segment pointer before branching
1620                ;
F53E 8B4600            1621                Mov    Ax,VideoSegment ;Video segment pointer
F541 1E                1622                Push   Ds              ;Save data segment register
F542 57                1623                Push   Di              ;Save pointer to frame
F543 8ED8              1624                Mov    Ds,Ax           ;Load it
1625                Assume Ds:Nothing      ;...we know nothing
1626                ;
1627                ; Branch
1628                ;
F545 7431              1629                Je     HiResRead      ;...jump if it is
1630                ;
1631                ; Medium Resolution Read
1632                ;
1633                ; Move data to stack frame
1634                ;
F547 B608              1635                Mov    Dh,8            ;8 lines of dots
F549 D1E6              1636                Shl   Si,1            ;2 bytes per char
F54B B80020            1637                Mov    Bx,2000H       ;Setup register with toggle bit
F54E                1638    MedReadLoop:
F54E 8B04              1639                Mov    Ax,[Si]        ;Get a row of dots from video
F550 86E0              1640                Xchg  Ah,Al           ;...Hi byte Lo byte correct
F552 B900C0            1641                Mov    Cx,1100000000000000B ;Start testing for bkgnd
F555 B200              1642                Mov    D1,0           ;...clear accumulator
F557                1643    MedReadLp2:
F557 85C1              1644                Test   Ax,Cx          ;Any bits set in pair?
F559 F8                1645                Clc
F55A 7401              1646                Jz     ShiftBitsIn   ;Prepare for no answer
F55C F9                1647                Stc
F55D                1648    ShiftBitsIn:
F55D D0D2              1649                Rcl   D1,1           ;Shift bit into accumulator
F55F D1E9              1650                Shr   Cx,1           ;Move mask
F561 D1E9              1651                Shr   Cx,1           ;...to the next bit
F563 73F2              1652                Jnc   MedReadLp2     ;Loop till mask clears reg.
1653                ;
1654                ; Store character under stack frame
1655                ;
F565 368015            1656                Mov    Ss:[Di],D1     ;Store the char
F568 47                1657                Inc    Di              ;...bump pointer
1658                ;
1659                ; Increment to next row
1660                ;
F569 33F3              1661                Xor   Si,Bx          ;Toggle even/odd field bit
F56B 85F3              1662                Test  Si,Bx          ;...and test if we need to
F56D 7503              1663                Jnz   OddFieldRd     ;...add 80
F56F 83C650            1664                Add   Si,80          ;Increment to next row
F572                1665    OddFieldRd:
F572 FECE              1666                Dec   Dh              ;Decrement count
F574 75D8              1667                Jnz   MedReadLoop    ;...and loop till done
F576 EB17              1668                Jmp   Short ImageMatch ;Go match it up
1669                ;
1670                ; Hi Resolution Read
1671                ;
F578                1672    HiResRead:
1673                ;
F578 B604              1674                Mov    Dh,4           ;4 lines of dots
F57A                1675    HiReadLoop:
F57A 8A24              1676                Mov    Ah,[Si]        ;Get odd row of dots from video
1677                ;

```

LOC	OBJ	LINE	SOURCE
		1678	; Store character under stack frame
		1679	
F57C	368825	1680	Mov Ss:[Di],Ah ;Store the char
F57F	47	1681	Inc Di ;...bump pointer
		1682	
		1683	; Increment to next row
F580	8AA40020	1684	
		1685	Mov Ah,[Si][2000H] ;Get odd row of dots from video
		1686	
		1687	; Store character under stack frame
F584	368825	1688	
F587	47	1689	Mov Ss:[Di],Ah ;Store the char
		1690	Inc Di ;...bump pointer
		1691	
		1692	; Increment to next row
		1693	
F588	83C650	1694	Add Si,80 ;Increment to next row
F58B	FECE	1695	Dec Dh ;Decrement count
F58D	75EB	1696	Jnz HiReadLoop ;...and loop till done
		1697	
		1698	; Character in stack frame, match with images
		1699	
F58F		1700	ImageMatch:
F58F	8CCA	1701	Mov Dx,Cs ;Rom images in code segment
F591	BF0000	1702	Mov Di,Offset VideoGraphicsPointer ;Images = Dx:Di
F594	8EC2	1703	Mov Es,Dx ;.....Images = Es:Di
F596	BCD2	1704	Mov Dx,Ss ;Point at stack segment
F598	8EDA	1705	Mov Ds,Dx ;...and let data seg. ref. it
F59A	5E	1706	Pop Si ;Restore pointer to frame
		1707	
		1708	; Pointers set, compare with images
		1709	
F59B	B000	1710	Mov Al,0 ;ASCII Value
F59D		1711	ImageTry2:
F59D	BA0000	1712	Mov Dx,128 ;There are 128 of them
F5A0		1713	ImageLoop:
F5A0	56	1714	Push Si ;Save pointers
F5A1	57	1715	Push Di ;...while comparing
F5A2	B90000	1716	Mov Cx,8 ;Compare 8 bytes
F5A5	F3	1717	Repe Cmps ;...for a match
F5A6	A6		
F5A7	5F	1718	Pop Di ;Restore pointers
F5A8	5E	1719	Pop Si ;...
F5A9	741C	1720	Jz MatchFound ;Jump if match found
F5AB	FECD	1721	Inc Al ;Increment ASCII
F5AD	83C708	1722	Add Di,8 ;Try next image
F5B0	4A	1723	Dec Dx ;Exhausted?
F5B1	75ED	1724	Jnz ImageLoop ;...jump if not
		1725	
		1726	; Image not in ROM images, try user
		1727	
F5B3	0AC0	1728	Or Al,Al ;Register wrap?
F5B5	7410	1729	Jz NoMatch ;...jump if so and end
		1730	
		1731	; Point at user images
		1732	
F5B7	33DB	1733	Xor Bx,Bx ;...point at int segment
F5B9	8EDB	1734	Mov Ds,Bx ;...to find pointer
		1735	Assume Ds:IntSegment ;...to user images
F5BB	C43E0000	1736	Les Di,VideoGraphicsTrapAddr ;User images at Es:Di
F5BF	8CC3	1737	Mov Bx,Es ;Test if Es:Di = 0
F5C1	0BDF	1738	Or Bx,Di ;...if so, no images there
F5C3	7402	1739	Jz NoMatch ;...jump if no match
F5C5	EBD6	1740	Jmp ImageTry2 ;Jump and re-try images
		1741	
F5C7		1742	NoMatch:
F5C7		1743	MatchFound:
F5C7	8B4602	1744	Mov Char,Al ;Return in Char
		1745	
		1746	Assume Ds:BiosDataArea ;Tell ASM86 data segment
F5CA	1F	1747	Pop Ds ;Restore our segment
F5CB	83C408	1748	Add Sp,8 ;Toss the temp area
F5CE	C3	1749	Ret ;...and return
		1750	ReadGraph
		1751	Endp
		1751	+1 \$Eject

LOC	OBJ	LINE	SOURCE
		1752	;
		1753	;
		1754	*****
		1755	* Command Procedure - Set Color *
		1756	*****
F5CF		1756	SetColor Proc Near ;Set color and background
F5CF 8B160000	E	1757	Mov Dx,ActiveCard ;Point at card
F5D3 83C205		1758	Add Dx,5 ;...and then to palette
F5D6 A00000	E	1759	Mov Al,CrtPalette ;Get the palette as it stands
F5D9 8A6605		1760	Mov Ah,ColorId
F5DC 0AE4		1761	Or Ah,Ah ;...Set background?
F5DE 8A6604		1762	Mov Ah,ColorValue ;Pre-load color value
F5E1 7509		1763	Jnz SetFore ;...jump if not
		1764	
		1765	;
		1766	Set Background
F5E3 24E0		1767	And Al,11100000B ;Strip and merge values
F5E5 80E41F		1768	And Ah,00011111B ;...old with
F5E8 0AC4		1769	Or Al,Ah ;...new
F5EA EB09		1770	Jump Short SetPalAndReturn ;...and return
		1771	
		1772	;
		1773	SetFore:
F5EC 24DF		1774	And Al,11011111B ;Palette = Grn Red Yellow
F5EE F6C401		1775	Test Ah,1 ;...Right?
F5F1 7402		1776	Jz SetPalAndReturn ;...jump if right
F5F3 0C20		1777	Or Al,00100000B ;Palette = Blue Cyan Magenta
		1778	
		1779	;
		1780	Set palette and return
F5F5		1781	SetPalAndReturn:
F5F5 A20000	E	1782	Mov CrtPalette,Al ;Set in ram and
F5F8 EE		1783	Out Dx,Al ;...send to hardware
F5F9 C3		1784	Ret ;...and return
		1785	SetColor
		1786 +1	\$Eject Endp

```

LOC OBJ          LINE      SOURCE
                1787      ;
                1788      ;
                1789      ;
                1790      ;
                1791      ;
                1792      ;
                1793      ;
                1794      ;
                1795      ;
                1796      ;
                1797      ;
                1798      ;
                1799      ;
                1800      ;
                1801      ;
                1802      ;
                1803      ;
                1804      ;
                1805      ;
                1806      ;
                1807      ;
                1808      ;
                1809      ;
                1810      ;
                1811      ;
                1812      ;
                1813      ;
                1814      ;
                1815      ;
                1816      ;
                1817      ;
                1818      ;
                1819      ;
                1820      ;
                1821      ;
                1822      ;
                1823      ;
                1824      ;
                1825      ;
                1826      ;
                1827      ;
                1828      ;
                1829      ;
                1830      ;
                1831      ;
                1832      ;
                1833      ;
                1834      ;
                1835 +1 ;

F5FA            1791      WriteDot      Proc      Near      ;Write a dot
F5FA 8B4600     1792      Mov      Ax,VideoSegment ;Get the segment value
F5FD 8EC0       1793      Mov      Es,Ax          ;...into segment register
F5FF 8B5600     1794      Mov      Dx,DotRow      ;Get Row
F602 8B4E06     1795      Mov      Cx,DotCol      ;...and column
F605 E81901     1796      Call    ConvertDot      ;Get info about dot position
F608 750E       1797      Jnz     WrDotLoRes      ;...and jump if medium res

                1798      ;
                1799      ;
                1800      ;
                1801      ;
                1802      ;
                1803      ;
                1804      ;
                1805      ;
                1806      ;
                1807      ;
                1808      ;
                1809      ;
                1810      ;
                1811      ;
                1812      ;
                1813      ;
                1814      ;
                1815      ;
                1816      ;
                1817      ;
                1818      ;
                1819      ;
                1820      ;
                1821      ;
                1822      ;
                1823      ;
                1824      ;
                1825      ;
                1826      ;
                1827      ;
                1828      ;
                1829      ;
                1830      ;
                1831      ;
                1832      ;
                1833      ;
                1834      ;
                1835 +1 ;

F60A 8A4602     1801      Mov      Al,Dot          ;Get the dot
F60D 8AD8       1802      Mov      Bl,Al          ;...and keep a copy
F60F 2401       1803      And     Al,0000001B      ;Only one bit in HiRes
F611 D0C8       1804      Ror     Al,1            ;...move it to upper bit
F613 B47F       1805      Mov     Ah,01111111B     ;Also shift mask bit same
F615 EB1090     1806      Jmp     WrDotCont2      ;...Continue below

                1807      ;
                1808      ;
                1809      ;
                1810      ;
                1811      ;
                1812      ;
                1813      ;
                1814      ;
                1815      ;
                1816      ;
                1817      ;
                1818      ;
                1819      ;
                1820      ;
                1821      ;
                1822      ;
                1823      ;
                1824      ;
                1825      ;
                1826      ;
                1827      ;
                1828      ;
                1829      ;
                1830      ;
                1831      ;
                1832      ;
                1833      ;
                1834      ;
                1835 +1 ;

F618            1809      WrDotLoRes:  ;
F618 D0E1       1810      Shl     Cl,1            ;Shifting 2 bits for lo res
F61A 8A4602     1811      Mov     Al,Dot          ;Get the dot
F61D 8AD8       1812      Mov     Bl,Al          ;...and keep a copy
F61F 2403       1813      And     Al,00000011B     ;Only one bit in HiRes
F621 D0C8       1814      Ror     Al,1            ;...move it to upper bit
F623 D0C8       1815      Ror     Al,1            ;...move it to upper bit
F625 B43F       1816      Mov     Ah,00111111B     ;Also shift mask bit same

                1817      ;
                1818      ;
                1819      ;
                1820      ;
                1821      ;
                1822      ;
                1823      ;
                1824      ;
                1825      ;
                1826      ;
                1827      ;
                1828      ;
                1829      ;
                1830      ;
                1831      ;
                1832      ;
                1833      ;
                1834      ;
                1835 +1 ;

F627            1820      WrDotCont2: ;
F627 D2CC       1821      Ror     Ah,Cl           ;...amount for anding
F629 D2E8       1822      Shr     Al,Cl           ;...out the old data
F62B 268A0C     1823      Mov     Cl,Es:[Si]      ;Get existing byte
F62E 0ADB       1824      Or     Bl,Bl           ;Affect flags with upper bit
F630 7904       1825      Jns    WrDotNoXor      ;...jump if not Xor
F632 32C8       1826      Xor    Cl,Al           ;...else XOR new with old
F634 EB04       1827      Jmp     Short XorDot    ;...and store it

                1828      ;
                1829      ;
                1830      ;
                1831      ;
                1832      ;
                1833      ;
                1834      ;
                1835 +1 ;

F636            1828      WrDotNoXor: ;
F636 22CC       1829      And     Cl,Ah           ;Strip
F638 0AC8       1830      Or     Cl,Al           ;...and or in new bits

                1831      ;
                1832      ;
                1833      ;
                1834      ;
                1835 +1 ;

F63A            1831      XorDot:      ;
F63A 26880C     1832      Mov     Es:[Si],Cl      ;Store new dot
F63D C3         1833      Ret     ;...and return

                1834      ;
                1835 +1 ;

                1834      WriteDot      Endp
                1835 +1      $Eject

```

```

LOC OBJ          LINE    SOURCE
                1836    ;
                1837    ;
                1838    ;
                1839    ;
F63E            1840    ReadDot      Proc   Near           ;Read a dot
F63E 8B4600     1841    Mov     Ax,VideoSegment ;Get the segment value
F641 8EC0       1842    Mov     Es,Ax           ;...into segment register
F643 8B5608     1843    Mov     Dx,DotRow       ;Get Row
F646 8B4E06     1844    Mov     Cx,DotCol       ;...and column
F649 E8D500     1845    Call   ConvertDot       ;Get info about dot position
F64C 268A04     1846    Mov     Al,Es:[Si]      ;Get byte containing dot
F64F 7508       1847    Jnz    RdDotLoRes       ;...and jump if medium res
                1848
                1849    ;       Read dot Hi Res
                1850
F651 D2E0       1851    Shl    Al,C1             ;Shift dot to upper bit
F653 D0C0       1852    Rol    Al,1              ;...then to lower bit
F655 2401       1853    And   Al,0000001B       ;Strip other dots
F657 EB0A       1854    Jmp   Short RdDotReturn ;...and return
                1855
                1856    ;       Read dot Med Res
                1857
F659            1858    RdDotLoRes:
F659 D0E1       1859    Shl    C1,1              ;Shifting 2 dots at once
F65B D2E0       1860    Shl    Al,C1             ;Shift dots to upper bit
F65D D0C0       1861    Rol    Al,1              ;...then to lower bit
F65F D0C0       1862    Rol    Al,1              ;...then to lower bit
F661 2403       1863    And   Al,0000011B       ;Strip other dots
F663            1864    RdDotReturn:
F663 8B4602     1865    Mov     Dot,Al           ;Return in stack frame
F666 C3         1866    Ret                     ;...and return
                1867    ReadDot
                1868 +1 $Eject

```

```

LOC OBJ          LINE    SOURCE
                1869    ;
                1870    ;
                1871    ;
                1872    ;
F667            1873    WriteTTY Proc Near ;Write glass tty
F667 B403       1874    Mov Ah,VidCmdRdCurPos ;...Recurse and read
F669 8A3E0000   E 1875    Mov Bh,ActivePage ;...cursor position
F66D CD10       1876    Int TrapVideo ;...of current screen
F66F 8A4602     1877    Mov Al,Char ;Get the char to write
                1878
                1879    ; Test special char
                1880
F672 3C00       1881    Cmp Al,AsciiBackspace ;Backspace?
F674 7422       1882    Je BSHandler ;...jump if yes
F676 3C0A       1883    Cmp Al,AsciiLinefeed ;Line feed?
F678 7434       1884    Je LFHandler ;...jump if yes
F67A 3C07       1885    Cmp Al,AsciiBell ;Bell?
F67C 7423       1886    Je BellHandler ;...jump if yes
F67E 3C0D       1887    Cmp Al,AsciiCarriage ;Carriage Return
F680 7425       1888    Je CRHandler ;...jump if yes
                1889
                1890    ; Must be printable char
                1891
F682 8A5E04     1892    Mov Bl,TTYForeground ;Get fore. col. incase graphics
F685 B40A       1893    Mov Ah,VidCmdWrCurCh ;Write at cursor
F687 B90100     1894    Mov Cx,1 ;...length of 1
F68A CD10       1895    Int TrapVideo ;Recurse to save stack frame
                1896
                1897    ; Bump Cursor in Dh Dl
                1898
F68C FEC2       1899    Inc Dl ;Column + 1
F68E 3A160000   E 1900    Cmp Dl,Byte Ptr CrtColumns ;...overflow?
F692 7515       1901    Jnz SetCursAndReturn ;...jump if not
F694 32D2       1902    Xor Dl,Dl ;Clear row count
F696 EB16       1903    Jmp Short LFHandler ;...and go do line feed
                1904
                1905    ; Special char handlers
                1906
                1907    ; Backspace
                1908
F698            1909    BSHandler:
F698 80FA00     1910    Cmp Dl,0 ;Left margin?
F69B 740C       1911    Je SetCursAndReturn ;...if so, do nothing
F69D FECA       1912    Dec Dl ;...else backup
F69F EB08       1913    Jmp Short SetCursAndReturn;...and return
                1914
                1915    ; Bell
                1916
F6A1            1917    BellHandler:
F6A1 B302       1918    Mov Bl,2 ;Call external beep routine
F6A3 E80000     E 1919    Call Beep ;...and
F6A6 C3         1920    Ret ;...return
                1921
                1922    ; Carriage Return
                1923
F6A7            1924    CRHandler:
F6A7 B200       1925    Mov Dl,0 ;Column 0
                1926
                1927    ; Set cursor and return
                1928
F6A9            1929    SetCursAndReturn:
F6A9 B402       1930    Mov Ah,VidCmdCurPos ;Set cursor
F6AB CD10       1931    Int TrapVideo ;...recurring
F6AD C3         1932    Ret ;...and return
                1933
                1934    ; Line Feed
                1935
F6AE            1936    LFHandler:
F6AE 80FE18     1937    Cmp Dh,24 ;Last line?
F6B1 7404       1938    Je TTYScroll ;...jump and scroll if so
F6B3 FED6       1939    Inc Dh ;...else go to next line
F6B5 75F2       1940    Jnz SetCursAndReturn ;...jump if not and return
                1941
                1942    ; Position cursor and read attribute
F6B7            1943    TTYScroll:

```

LOC	OBJ	LINE	SOURCE		
F6B7	B402	1944		Mov	Ah,VidCmdCurPos ;Set cursor
F6B9	CD10	1945		Int	TrapVideo ;...recurring
		1946			
		1947			
		1948		;	Mode Split
F6BB	E82E00	1949		Call	WhichMode ;Which mode are we in?
F6BE	B700	1950		Mov	Bh,0 ;Pre-load for graphics
F6C0	7206	1951		Jc	TTYGraphics ;...jump if graphics
F6C2	B408	1952		Mov	Ah,VidCmdRdCurChAt ;...else read attrib
F6C4	CD10	1953		Int	TrapVideo ;...to use during scroll
F6C6	8AFC	1954		Mov	Bh,Ah ;Scroll requires in Bh
F6C8		1955	TTYGraphics:		
F6C8	B406	1956		Mov	Ah,VidCmdScrollUp ;Scroll up command
F6CA	B001	1957		Mov	Al,1 ;...one line
F6CC	33C9	1958		Xor	Cx,Cx ;Upper left = 0
F6CE	B618	1959		Mov	Dh,24 ;Lower limit = line 24
F6D0	8A160000	1960	E	Mov	Dl,Byte Ptr CrtColumns ;Right Margin = Column
F6D4	FECA	1961		Dec	Dl ;...80 or 40
F6D6	CD10	1962		Int	TrapVideo ;Do scroll
F6D8	C3	1963		Ret	;...and return
		1964	WriteTTY	Endp	
		1965	+1 \$Eject		

```

LOC OBJ          LINE    SOURCE
                1966    ;
                1967    ;
                1968    ;
                1969    ;
                1970    ;
                1971    ;
                1972    ;
                1973    ;
                1974    ;
F6D9            1975    VideoState Proc Near ;Return current video state
F6D9 A00000     E      1976    Mov     Al,Byte Ptr CrtColumns ;Plug columns
F6DC 884603     E      1977    Mov     Columns,Al ;...into return frame
F6DF A00000     E      1978    Mov     Al,CrtMode ;Plug mode
F6E2 884602     E      1979    Mov     Mode,Al ;...into return frame
F6E5 A00000     E      1980    Mov     Al,ActivePage ;Plug page
F6E8 884605     E      1981    Mov     DisplayPage,Al ;...into return frame
F6EB C3        1982    Ret                                ;...and return
                1983    VideoState Endp
                1984    ;
                1985    ;
                1986    ;
                1987    ;
                1988    ;
                1989    ;
                1990    ;
                1991    ;
                1992    ;
F6EC            1993    WhichMode Proc Near
F6EC 50        1994    Push   Ax ;Save Ax
F6ED A00000     E      1995    Mov     Al,CrtMode ;Get the mode
F6F0 3C07     1996    Cmp     Al,7 ;Monochrome?
F6F2 7408     1997    Je     WhichReturn ;Carry = 0, Z = 1
F6F4 3C04     1998    Cmp     Al,4 ;Graphics?
F6F6 F5        1999    Cmc                                ;...Carry = 0, Z = 0
F6F7 7303     2000    Jnc    WhichReturn ;...jump if alpha on color board
F6F9 1AC0     2001    Sbb    Al,Al ;...Carry = 1, Z = 0
F6FB F9        2002    Stc                                ;...for graphics on color card
F6FC            2003    WhichReturn:
F6FC 58        2004    Pop     Ax ;Restore Ax
F6FD C3        2005    Ret                                ;Return
                2006    WhichMode Endp
                2007 +1 $Eject

```

```

LOC OBJ          LINE    SOURCE
                2008      ;
                2009      ;
                2010      ;
                2011      ;
                2012      ;
F6FE            2013      Load6845Double Proc Near
F6FE 8AC5       2014      Mov     Al,Ch           ;Get High byte
F700 E80400    2015      Call   Load6845       ;...load it
F703 FEC4       2016      Inc    Ah              ;...Inc to next byte
F705 8AC1       2017      Mov    Al,C1          ;Get Low Byte and fall through
                2018      ;
                2019      ;
                2020      ;
                2021      ;
                2022      ;
                2023      ;
F707            2024      Load6845      Proc Near
F707 52         2025      Push   Dx              ;Save Dx
F708 8B160000   E 2026      Mov    Dx,ActiveCard  ;Get port address
F70C 86C4       2027      Xchg  Al,Ah           ;...and send index from Ah
F70E EE         2028      Out   Dx,Al          ;...to point 6845 at register
F70F 86C4       2029      Xchg  Al,Ah           ;Get back data to send
F711 FEC2       2030      Inc    Dl              ;...inc to data port
F713 EE         2031      Out   Dx,Al          ;...and send it
F714 5A         2032      Pop    Dx              ;Restore Dx
F715 C3         2033      Ret                    ;...and return
                2034      Load6845      Endp
                2035      Load6845Double Endp
                2036      ;
                2037      ;
                2038      ;
                2039      ;
                2040      ;
F716            2041      ConvertCursor Proc Near
F716 32FF       2042      Xor   Bh,Bh           ;Get page number in Bx
F718 D1E3       2043      Shl   Bx,1            ;...double for word offset
F71A 8B870000   E 2044      Mov    Ax,CursorPosn[Bx] ;Get and set the cursor
F71E EB5590     2045      Jmp   Convert         ;Convert xy to linear address
                2046      ConvertCursor Endp
                2047 +1 $Eject

```

```

LOC OBJ          LINE    SOURCE
                2048    ;
                2049    ;
                2050    ;
                2051    ;
                2052    ;
                2053    ;
                2054    ;
                2055    ;
                2056    ;
                2057    ;
                2058    ;
F721            2059    ConvertDot  Proc   Near
F721 8050        2060    Mov    Al,80                ;# of bytes across
F723 33F6        2061    Xor    Si,Si                ;Pre-load start of even field
F725 D0EA        2062    Shr    Dl,1                 ;Divide rows by odd/even
F727 7303        2063    Jnc    CDEven               ;...jump if even field
F729 BE0020      2064    Mov    Si,2000H             ;Re-load start of odd field
F72C            2065    CDEven:
F72C F6E2        2066    Mul    Dl                    ;Multiply rows by 80
F72E 03F0        2067    Add    Si,Ax                ;Keep in source pointer
F730 8BD1        2068    Mov    Dx,Cx                ;Need Cx for shift
F732 B90203      2069    Mov    Cx,0302H             ;Mask & # of bits in byte - Lo
F735 803E000006 E 2070    Cmp    CrtMode,6            ;HiRes?
F73A 9C          2071    Pushf                       ;...save result for return
F73B 7503        2072    Jne    CDLoRes              ;...jump if not
F73D B90307      2073    Mov    Cx,0703H             ;Mask &# of bits in byte - Hi
F740            2074    CdLoRes:
F740 22EA        2075    And    Ch,Dl                ;Convert Ch to bit # in byte
F742 D3EA        2076    Shr    Dx,C1                ;...then shift it out
F744 03F2        2077    Add    Si,Dx                ;Add offset in line
F746 86CD        2078    Xchg  Cl,Ch                 ;...swap shift to cl
F748 9D          2079    Popf                          ;Z = 1 for HiRes
F749 C3          2080    Ret                            ;...and return
                2081    ConvertDot  Endp
                2082
                2083    ;
                2084    ;
                2085    ;
                2086    ;
                2087    ;
F74A            2087    LoadCursor Proc   Near
F74A E8C9FF      2088    Call  ConvertCursor         ;Convert cursor to linear
F74D 03060000    2089    Add    Ax,CrtStart          ;...add to start of page
F751 D1E8        2090    Shr    Ax,1                 ;...get character only address
F753 8BC8        2091    Mov    Cx,Ax                ;Send it to 6845
F755 B40E        2092    Mov    Ah,14                ;...register 14
F757 E8A4FF      2093    Call  Load6845Double        ;...and return
F75A C3          2094    Ret
                2095    LoadCursor
                2096 +1  $Eject

```



```

LOC OBJ          LINE      SOURCE
                2097      ;
                2098      * Fetch Crt Controller with Word Parameter *
                2099      *   Ah = Index   Cx = Word   *
                2100      ;
                2101      ;
F75B             2102      Fetch6845Double Proc   Near
F75B E80000      2103      Call   Fetch6845      ;Get a byte from the 6845
F75E 8AE8        2104      Mov    Ch,Al          ;...it is half of word
F760 FEC4        2105      Inc    Ah             ;...Inc to next byte
F762 E80300      2106      Call   Fetch6845      ;Get next byte from the 6845
F765 8AC8        2107      Mov    C1,Al         ;...it is next half of word
F767 C3          2108      Ret                  ;...return with word value
                2109      Fetch6845Double Endp
                2110
                2111      ;
                2112      * Fetch Crt Controller with Byte Parameter *
                2113      *   Ah = Index   Cx = Byte   *
                2114      ;
                2115      ;
F768             2116      Fetch6845   Proc   Near
F768 52          2117      Push   Dx             ;Save Dx
F769 8B160000    E  2118      Mov    Dx,ActiveCard ;Get port address
F76D 86C4        2119      Xchg  Al,Ah          ;...and send index from Ah
F76F EE          2120      Out   Dx,Al         ;...to point 6845 at register
F770 FEC2        2121      Inc    DI            ;...inc to data port
F772 EC          2122      In    Al,Dx         ;...and get byte
F773 5A          2123      Pop   Dx             ;Restore Dx
F774 C3          2124      Ret                  ;...and return
                2125      Fetch6845   Endp
                2126
                2127      ;
                2128      * Convert from Row / Column to Linear Address *
                2129      ;
F775             2130      Convert    Proc   Near
F775 53          2131      Push   Bx            ;Save a register
F776 8AD8        2132      Mov    Bl,Al         ;Get columns
F778 8AC4        2133      Mov    Al,Ah         ;Get rows in Al
F77A F6260000    E  2134      Mul   Byte Ptr CrtColumns ;...and multiply by
F77E 32FF        2135      Xor   Bh,Bh         ;Add in column
F780 03C3        2136      Add   Ax,Bx         ;...to get char address
F782 D1E0        2137      Shl   Ax,1          ;...convert to real address
F784 5B          2138      Pop   Bx            ;...restore and
F785 C3          2139      Ret                  ;...return
                2140      Convert    Endp
                2141
                2142      ;
                2143      * Graphics Convert *
                2144      ;
                2145      ;
F786             2146      ConvertGraph Proc   Near
F786 53          2147      Push   Bx            ;Using Bx, save it
F787 8AD8        2148      Mov    Bl,Al         ;Keep column value
F789 8AC4        2149      Mov    Al,Ah         ;Multiply rows by
F78B F6260000    E  2150      Mul   Byte Ptr CrtColumns ;...# columns on screen then by
F78F D1E0        2151      Shl   Ax,1          ;...4, cause in graphics mode,
F791 D1E0        2152      Shl   Ax,1          ;...there are 4 even field rows
                2153      ;...per character row
                2154      Xor   Bh,Bh         ;Add in columns to figure as if
                2155      Add   Ax,Bx         ;...each char was only 1 byte
                2156      Pop   Bx            ;...which is correct for HiRes
                2157      Ret                  ;Multiply x 2 for LoRes address
                2158      ConvertGraph Endp
                2159
                2160      Bios      Ends
                2161
                2162      End

```

ASSEMBLY COMPLETE, NO ERRORS FOUND

IBM PC/XT 0086/0087/0088 MACRO ASSEMBLER V1.1 ASSEMBLY OF MODULE EQUIPMNT
 OBJECT MODULE PLACED IN EQUIPMNT.OBJ
 ASSEMBLER INVOKED BY: ASMB6 EQUIPMNT.SRC

```

LOC OBJ          LINE    SOURCE
                1 +1 $Title ('DTC/PC BIOS Equipment Determination and Memory Size V1.0')
                2 +1 $Pagelength (80) Pagewidth (132) Debug Nogen
                3     Name Equipmnt
                4
                5
                6     ; Author:      Don K. Harrison
                7
                8     ; Start date: December 7, 1983      Last edit: December 17, 1983
                9
               10
               11     ;
               12     ;          *****
               13     ;          * Module Description *
               14     ;          *****
               15     ;
               16     ;          This module contains the memory size determination service
               17     ;          routine (Int 18) and the equipment inventory service routine Int (17).
               18     ;          A dummy routine to serve the non-existent cassette interrupt (Int 21)
               19     ;          is also included.
               20
               21
               22
               23
               24     ;          (c) Display Telecommunications Corporation, 1983
               25     ;          All Rights Reserved
               26
               27     $Eject
    
```

LOC	OBJ	LINE	SOURCE
		28	
		29	
		30	;
		31	;
		32	*****
		33	*****
		34	*****
		35	*****
		36	*****
		37	*****
		38	*****
		39	*****
		40	\$Eject

```
LOC OBJ          LINE    SOURCE
                41
                42      ;          *****
                43      ;          * Public Symbols *
                44      ;          *****
                45
                46      Public EquipDriver, MemorySizeDriver, CassetteDriver
                47
                48
                49      ;          *****
                50      ;          * Equates *
                51      ;          *****
                52
                53      ;          All Equates in include file: IbmInc
                54
                55      $Include (IbmInc)
=1 56      ;          *****
=1 57      ;          * Global Include File *
=1 58      ;          *****
=1 59      $Nolist
579      $Eject
```

LOC	OBJ	LINE	SOURCE
		580	
		581	;
		582	;
		583	;
		584	;
----		585	BiosDataArea Segment Public
		586	Extrn MemorySize:Word, EquipFlag:Word
----		587	BiosDataArea Ends
		588	\$Eject

```

LOC OBJ          LINE      SOURCE
                    589
                    590          ;
                    591          ;          *****
                    592          ;          * Code Segment *
                    593          ;          *****
-----          594      Bios      Segment Common
                    595
                    596          Assume Cs:Bios, Ds:BiosDataArea
                    597
                    598          ;          *****
                    599          ;          * Memory Size *
                    600          ;          *****
                    601
F841             602          Org      0F841H
                    603
F841             604      MemorySizeDriver Proc Far
F841 FB          605          Sti          ;Restore interrupts
F842 1E          606          Push         Ds          ;Save data segment
F843 B8-----   R      607          Mov          Ax,BiosDataArea ;...load ours
F846 8ED8        608          Mov          Ds,Ax          ;...in
F848 A10000      E      609          Mov          Ax,MemorySize ;Get the info
F84B 1F          610          Pop          Ds          ;Restore
F84C CF          611          Iret         ;...and return
                    612      MemorySizeDriver Endp
                    613
                    614
                    615          ;          *****
                    616          ;          * Equipment Inventory *
                    617          ;          *****
                    618
F84D             619          Org      0F84DH
                    620
F84D             621      EquipDriver Proc Far
F84D FB          622          Sti          ;Restore interrupts
F84E 1E          623          Push         Ds          ;Save data segment
F84F B8-----   R      624          Mov          Ax,BiosDataArea ;...load ours
F852 8ED8        625          Mov          Ds,Ax          ;...in
F854 A10000      E      626          Mov          Ax,EquipFlag ;Get the info
F857 1F          627          Pop          Ds          ;Restore
F858 CF          628          Iret         ;...and return
                    629      EquipDriver Endp
                    630
                    631          ;          *****
                    632          ;          * Dummy Cassette Driver *
                    633          ;          *****
                    634
F859             635          Org      0F859H
                    636
F859             637      CassetteDriver Proc Far
F859 F9          638          Stc          ;Error return
F85A B486        639          Mov          Ah,86H      ;...value
F85C CA0200      640          Ret          2          ;Toss flags and return
                    641      CassetteDriver Endp
                    642
-----          643      Bios      Ends
                    644
                    645      End

```

ASSEMBLY COMPLETE, NO ERRORS FOUND

IBM PC/XT 8086/8087/8088 MACRO ASSEMBLER V1.1 ASSEMBLY OF MODULE INTS
OBJECT MODULE PLACED IN INTS.OBJ
ASSEMBLER INVOKED BY: ASM86 INTS.SRC

LOC	OBJ	LINE	SOURCE
		1	+1 \$Title ('DTC/PC BIOS Interrupts V1.0')
		2	+1 \$Pagelength (80) Pagewidth (132) Debug Nogen
		3	Name Ints
		4	
		5	
		6	; Author: Don K. Harrison
		7	
		8	; Start date: October 25, 1983 Last edit: December 26, 1983
		9	
		10	
		11	
		12	; *****
		13	; * Module Description *
		14	; *****
		15	
		16	; This module contains the interrupt vector addresses.
		17	
		18	
		19	
		20	
		21	; (c) Display Telecommunications Corporation, 1983
		22	; All Rights Reserved
		23	
		24	\$Eject

LOC	OBJ	LINE	SOURCE
		25	
		26	
		27	;
		28	;
		29	;
		30	*****
		31	
		32	
		33	
		34	
		35	
		36	
		37	\$Eject


```
LOC OBJ          LINE    SOURCE
                 38
                 39      ;
                 40      * Public Symbols *
                 41      * Public Symbols *
                 42      ;
                 43      Public RamVectors, ROMVectors, VideoTrapAddr, BasicTrapAddr
                 44      Public VidParamsTrapAddr, FloppyParamsTrapAddr
                 45      Public VideoGraphicsTrapAddr, IllegalInt, NMIIInt
                 46      Public PrintScreenTrapAddr, NMITrapAddr
                 47
                 48      ;
                 49      * Equates *
                 50      * Equates *
                 51      ;
                 52      ;      All Equates in include file: IbmInc
                 53
                 54      $Include (IbmInc)
=1 55      ;
=1 56      ;      * Global Include File *
=1 57      ;
=1 58      $Nolist
578 $Eject
```

```

LOC  OBJ                LINE    SOURCE
                                579      ;
                                580      ;
                                581      ;
                                582      ;
----- 583      IntSegment      Segment Public
                                584
0000    585      RamVectors      Label  DWord
0000 (1 586      ZeroDivisionTrapAddr  Dd      1 Dup (?)      ;Divide by zero vector
      )
0004 (1 587      SingleStepTrapAddr  Dd      1 Dup (?)      ;Single step vector
      )
0008 (1 588      NMITrapAddr      Dd      1 Dup (?)      ;Non maskable interrupt vector
      )
000C (1 589      BreakpointTrapAddr  Dd      1 Dup (?)      ;Breakpoint interrupt vector
      )
0010 (1 590      OverflowTrapAddr  Dd      1 Dup (?)      ;Overflow interrupt vector
      )
0014 (1 591      PrintScreenTrapAddr  Dd      1 Dup (?)      ;Print screen vector
      )
0018 (1 592      Undef1TrapAddr  Dd      1 Dup (?)      ;Undefined vector #1
      )
001C (1 593      Undef2TrapAddr  Dd      1 Dup (?)      ;Undefined vector #2
      )
                                594
                                595      ;      Hardware interrupts
                                596
0020 (1 597      TimerHdwrTrapAddr  Dd      1 Dup (?)      ;Timer interrupt
      )
0024 (1 598      KeyboardHdwrTrapAddr  Dd      1 Dup (?)      ;Keyboard interrupt
      )
0028 (1 599      IRQ2HdwrTrapAddr  Dd      1 Dup (?)      ;Interrupt 2 interrupt
      )
002C (1 600      Sio2HdwrTrapAddr  Dd      1 Dup (?)      ;Serial channel 2 interrupt
      )
0030 (1 601      Sio1HdwrTrapAddr  Dd      1 Dup (?)      ;Serial channel 1 interrupt
      )
0034 (1 602      HarDiskHdwrTrapAddr  Dd      1 Dup (?)      ;Hard disk interrupt
      )
0038 (1 603      FloppyHdwrTrapAddr  Dd      1 Dup (?)      ;Floppy disk interrupt
      )
003C (1 604      PrinterHdwrTrapAddr  Dd      1 Dup (?)      ;Printer interrupt
      )
                                605
                                606      ;      Driver entry points
                                607
0040 (1 608      VideoTrapAddr      Dd      1 Dup (?)      ;Video trap vector
      )
0044 (1 609      EquipTrapAddr      Dd      1 Dup (?)      ;Equipment query vector
      )
0048 (1 610      MemorySizeTrapAddr  Dd      1 Dup (?)      ;Memory size query vector
      )
004C (1 611      FloppyTrapAddr      Dd      1 Dup (?)      ;Diskette vector
      )
0050 (1 612      CommsTrapAddr      Dd      1 Dup (?)      ;Serial interface vector
      )

```

LOC	OBJ	LINE	SOURCE				
0054	(1 ????????)	613	CassetteTrapAddr	Dd	1 Dup (?)		;Cassette (dummy routine)
0058	(1 ????????)	614	KeyboarTrapAddr	Dd	1 Dup (?)		;Keyboard driver vector
005C	(1 ????????)	615	PrinterTrapAddr	Dd	1 Dup (?)		;Printer driver vector
0060	(1 ????????)	616	BasicTrapAddr	Dd	1 Dup (?)		;Cassette basic vector
0064	(1 ????????)	617	BootTrapAddr	Dd	1 Dup (?)		;Bootstrap loader trap vector
0068	(1 ????????)	618	TODTrapAddr	Dd	1 Dup (?)		;Time of day vector
006C	(1 ????????)	619	KeyBreakTrapAddr	Dd	1 Dup (?)		;Keyboard break aDd ?ress
0070	(1 ????????)	620	TickTrapAddr	Dd	1 Dup (?)		;Timer break aDd ?ress
		621					
		622					
		623					
0074	(1 ????????)	624	VidParamsTrapAddr	Dd	1 Dup (?)		;Video parameters
0078	(1 ????????)	625	FloppyParamsTrapAddr	Dd	1 Dup (?)		;Disk parameters
007C	(1 ????????)	626	VideoGraphicsTrapAddr	Dd	1 Dup (?)		;Video graphics characters
----		627					
		628	IntSegment			Ends	
		629					
		630	\$Eject				

```

LOC OBJ          LINE    SOURCE
-----
                631      ;
                632      ;
                633      ;
                634      ;
                635      BiosDataArea Segment Public
                636      Extrn MemorySize:Word, IntrFlag:Byte
                637      BiosDataArea Ends
                638      ;
                639      ;
                640      ;
                641      ;
                642      ;
                643      Bios      Segment Common
                644      ;
                645      Extrn WordOut:Near, CharOut:Near, NibbleOut:Near, KeyIn:Near
                646      Extrn PrintMessage:Near, Beep:Near, TimerHdwrInt:Far
                647      Extrn KeyboardHdwrInt:Far, FloppyHdwrInt:Far
                648      Extrn VideoDriver:Far, EquipDriver:Far
                649      Extrn MemorySizeDriver:Far, FloppyDriver:Far, CommsDriver:Far
                650      Extrn CassetteDriver:Far, KeyboardDriver:Far, PrinterDriver:Far
                651      Extrn BootDriver:Far, TODDriver:Far, VidParamsPointer:Byte
                652      Extrn FloppyParamsPointer:Byte, ResetEntry:Far, PorEntry:Far
                653      Extrn VideoInit:Near, ClearScreen:Near, PositionCursor:Near
                654      ;
                655      ;
                656      ;
                657      ;
                658      ;
                659      F85F      Org      0F85FH          ;Align with PC / Xt
                660      ;
                661      Assume Cs:Bios, Ds:BiosDataArea
                662      ;
                663      F85F      NMIInt Proc      Far
                664      F85F 50      Push     Ax
                665      F860 53      Push     Bx
                666      F861 51      Push     Cx
                667      F862 52      Push     Dx
                668      F863 56      Push     Si
                669      F864 57      Push     Di
                670      F865 55      Push     Bp
                671      F866 1E      Push     Ds
                672      F867 06      Push     Es
                673      F868 E462      In       Al,PortPPIPortC ;Get NMI status port
                674      F86A A8C0      Test    Al,11000000B ;...any interrupts pending?
                675      F86C 7503      Jnz    NMIOccurred ;...jump if yes
                676      F86E E98C00      Jmp    NMIReturn ;...else, ignore and return
                677      F871      NMIOccurred:
                678      F871 B8----      R      Mov     Ax,BiosDataArea ;Point at our data area
                679      F874 8ED8      Mov     Ds,Ax ;...with Ds
                680      F876 E80000      E      Call   VideoInit ;Initialize the video
                681      F879 1E      Push     Ds ;Save our seg
                682      F87A 0E      Push     Cs ;Load code pointer
                683      F87B 1F      Pop      Ds ;...into data pointer
                684      F87C BE1BF990      Mov     Si,Offset ParityMsg ;Print error message:
                685      F880 E80000      E      Call   PrintMessage ;Parity error at: ???? Cont?
                686      F883 1F      Pop      Ds ;Restore our seg
                687      F884 B81100      Mov     Ax,0011H ;Re-position cursor
                688      F887 E80000      E      Call   PositionCursor ;...for writing address
                689      ;
                690      ;
                691      ;
                692      F88A BA0000      Mov     Dx,PortNMIMask ;Point at parity controls
                693      F88D B002      Mov     Al,NMIMask ;Disable NMI's
                694      F88F E6A0      Out    PortNMIMask,Al ;...from causing further err's
                695      F891 BA6100      Mov     Dx,PortPPIPortB ;Point at port B
                696      F894 EC      In     Al,Dx ;Get parity controls
                697      F895 0C30      Or     Al,00110000B ;Set them Hi
                698      F897 EE      Out    Dx,Al ;...clearing F/Fs
                699      F898 24CF      And    Al,11001111B ;Set them Lo
                700      F89A EE      Out    Dx,Al ;...enabling them
                701      F89B B10E      Mov     Cl,6 ;Convert memory size from
                702      F89D 8B1E0000      E      Mov     Bx,MemorySize ;...1k blocks to
                703      F8A1 D3E3      Shl    Bx,Cl ;...16 byte paragraphs
                704      F8A3 42      Inc    Dx ;Bump to port C
                705

```

```

LOC  OBJ          LINE      SOURCE
                                ;      Setup Pointers
                                706
                                707
F8A4 33C0          708          Xor      Ax,Ax          ;Start paragraph pointer at 0
F8A6 8ED8          709          Mov      Ds,Ax          ;...to test
F8A8                710      ParagraphLoop:
F8A8 B91000         711          Mov      Cx,16          ;Number of bytes in a paragraph
F8AB 33F6          712          Xor      Si,Si          ;First byte in paragraph
F8AD                713      ByteLoop:
F8AD 8A24          714          Mov      Ah,[Si]        ;Get a byte
F8AF EC           715          In      Al,Dx          ;Get interrupt status bits
F8B0 ABC0          716          Test    Al,11000000B   ;...either set?
F8B2 750F          717          Jnz     FoundError     ;Jump if error found
F8B4 46           718          Inc     Si              ;Increment byte pointer
F8B5 E2F6          719          Loop   ByteLoop        ;...else loop through paragraph
F8B7 8CD8          720          Mov     Ax,Ds          ;Get paragraph value
F8B9 40           721          Inc     Ax              ;...increment it
F8BA 8ED8          722          Mov     Ds,Ax          ;...and
F8BC 3BC3          723          Cmp    Ax,Bx          ;...compare it to end
F8BE 75EB          724          Jne    ParagraphLoop  ;...and loop till all done
F8C0 EB0D90        725          Jmp    NoErrorFound   ;...no error detected, return
F8C3                726      FoundError:
F8C3 8024          727          Mov     [Si],Ah         ;Set parity in memory
F8C5 8CD8          728          Mov     Ax,Ds          ;Get paragraph value
F8C7 E80000        729          Call   WordOut         ;...and output it to screen
F8CA 8BCE          730          Mov     Ax,Si          ;Get byte value
F8CC E80000        731          Call   NibbleOut      ;...and output it
F8CF                732      NoErrorFound:
F8CF 801D00        733          Mov     Ax,001DH       ;Position cursor to end
F8D2 E80000        734          Call   PositionCursor ;...for retrieving answer
F8D5 1E           735          Push   Ds              ;Save our seg
F8D6 0E           736          Push   Cs              ;Load code pointer
F8D7 1F           737          Pop    Ds              ;...into data pointer
F8D8 BE32F990       738          Mov     Si,Offset ContMsg ;Print error message:
F8DC E80000        739          Call   PrintMessage   ; Cont?
F8DF 1F           740          Pop    Ds              ;Restore our seg
F8E0 E421          741          In     Al,PortPicDCW1 ;Get interrupt mask
F8E2 50           742          Push   Ax              ;...and save it
F8E3 80FC          743          Mov     Al,11111100B  ;...unmask only clock and kbrd
F8E5 E621          744          Out    PortPicDCW1,AI ;...output it
F8E7 FB           745          Sti    ;Turn ints back on
F8E8 E80000        746          Call   KeyIn          ;Get keyboard data
F8EB 50           747          Push   Ax              ;Save Ax
F8EC E80000        748          Call   CharOut        ;Echo keyboard response
F8EF 58           749          Pop    Ax              ;Restore Ax
F8F0 3C59          750          Cmp    Al,'Y'         ;...Y?
F8F2 7409          751          Je     NMIReturn      ;...jump and return if yes
F8F4 3C79          752          Cmp    Al,'y'         ;...y?
F8F6 7405          753          Je     NMIReturn      ;...jump and return if yes
F8F8 EA0000----- 754          Jmp    PorEntry       ;Re-boot from cold start
F8FD                755      NMIReturn:
F8FD E80000        756          Call   ClearScreen    ;Clear the screen
F900 58           757          Pop    Ax              ;Restore mask
F901 E621          758          Out    PortPicDCW1,AI ;...like before
F903 BA6100        759          Mov     Dx,PortPPIPortB ;Point at port B
F906 EC           760          In     Al,Dx          ;Get parity controls
F907 0C30          761          Or     Al,00110000B   ;Set them Hi
F909 EE           762          Out    Dx,Al          ;...clearing F/Fs
F90A 24CF          763          And    Al,11001111B  ;Set them Lo
F90C EE           764          Out    Dx,Al          ;...parity error ints again
F90D B080          765          Mov     Al,NMIUnmask  ;Clear NMI mask
F90F E6A0          766          Out    PortNMIMask,AI ;...enabling
F911 07           767          Pop    Es              ;.....
F912 1F           768          Pop    Ds              ;.
F913 5D           769          Pop    Bp              ;.
F914 5F           770          Pop    Di              ;.      Restore
F915 5E           771          Pop    Si              ;.
F916 5A           772          Pop    Dx              ;.      Registers
F917 59           773          Pop    Cx              ;.
F918 5B           774          Pop    Bx              ;.
F919 58           775          Pop    Ax              ;.
F91A CF           776          Iret   ;...and return, restore ints
F91B 50617269747920 777      ParityMsg      Db    'Parity error at: ?????',0
        6572726F722061
        743A203F3F3F3F
        3F
    
```

LOC	OBJ	LINE	SOURCE
F931	00		
F932	20436F6E743F	778	ContMsg Db ' Cont?',0
F938	00	779	NMIInt Endp
		780	\$Eject

```

LOC OBJ          LINE    SOURCE
                781      ;
                782      ;
                783      ;
                784      ;
FEF3             785      Org      0FEF3H          ;Align with PC/Xt
                786
FEF3             787      RomVectors  Label  DWord
                788
                789      ;      Hardware interrupts
                790
FEF3 0000        E       791      Dw      TimerHdwrInt    ;Timer interrupt
FEF5 0000        E       792      Dw      KeyboardHdwrInt ;Keyboard interrupt
FEF7 23FF        E       793      Dw      IllegalInt     ;Interrupt 2 interrupt
FEF9 23FF        E       794      Dw      IllegalInt     ;Serial channel 2 interrupt
FEFB 23FF        E       795      Dw      IllegalInt     ;Serial channel 1 interrupt
FEFD 23FF        E       796      Dw      IllegalInt     ;Hard disk interrupt
FEFF 0000        E       797      Dw      FloppyHdwrInt   ;Floppy disk interrupt
FF01 23FF        E       798      Dw      IllegalInt     ;Printer interrupt
                799
                800      ;      Driver entry points
                801
FF03 0000        E       802      Dw      VideoDriver     ;Video driver trap vector
FF05 0000        E       803      Dw      EquipDriver     ;Equipment query vector
FF07 0000        E       804      Dw      MemorySizeDriver ;Memory size query vector
FF09 0000        E       805      Dw      FloppyDriver    ;Diskette driver vector
FF0B 0000        E       806      Dw      CommsDriver     ;Serial interface vector
FF0D 0000        E       807      Dw      CassetteDriver  ;Cassette (dummy routine)
FF0F 0000        E       808      Dw      KeyboardDriver  ;Keyboard driver vector
FF11 0000        E       809      Dw      PrinterDriver   ;Printer driver vector
FF13 23FF        E       810      Dw      IllegalInt     ;Cassette basic vector
FF15 0000        E       811      Dw      BootDriver      ;Bootstrap loader trap vector
FF17 0000        E       812      Dw      TODDriver       ;Time of day vector
FF19 53FF        E       813      Dw      DummyReturn     ;Keyboard break address
FF1B 53FF        E       814      Dw      DummyReturn     ;Timer break address
                815
                816      ;      Data Pointers
                817
FF1D 0000        E       818      Dw      VidParamsPointer ;Video parameters
FF1F 0000        E       819      Dw      FloppyParamsPointer ;Disk parameters
FF21 0000        E       820      Dw      0              ;Video graphics characters
                821
                822      $Eject

```

```

LOC OBJ          LINE    SOURCE
                823      ;
                824      ;
                825      ;
                826      ;
FF23             827      IllegalInt Proc Near
FF23 1E          828      Push Ds ;.....
FF24 52          829      Push Dx ;. Save Registers .
FF25 50          830      Push Ax ;.....
FF26 B8-----  R      831      Mov Ax,BiosDataArea ;Load our data
FF29 8ED8       832      Mov Ds,Ax ;...segment
FF2B B00B       833      Mov Al,00001011B ;Command to point at
FF2D E620       834      Out PortPICDCW2,Al ;...in service register
FF2F 90         835      Nop
FF30 E420       836      In Al,PortPICDCW2 ;Get in-service register
FF32 8AE0       837      Mov Ah,Al ;Save it for masking
FF34 0AC0       838      Or Al,Al ;Zero?
FF36 7504       839      Jnz NonZeroLevel ;...jump in not
FF38 B0FF       840      Mov Al,0FFH ;Flag for non-hardware
FF3A EB0A       841      Jmp Short IllegalReturn ;...jump and return
FF3C           842      NonZeroLevel:
FF3C E421       843      In Al,PortPICDCW1 ;Get Mask Register
FF3E 0AC4       844      Or Al,Ah ;...mask off service level
FF40 E621       845      Out PortPICDCW1,Al ;...in progress
FF42 B020       846      Mov Al,PICEDI ;Interrupt ack
FF44 E620       847      Out PortPICDCW2,Al ;...to 8259A
FF46           848      IllegalReturn:
FF46 88260000  E      849      Mov IntrFlag,Ah ;Set global variable
FF4A 58         850      Pop Ax ;Restore
FF4B 5A         851      Pop Dx ; Registers
FF4C 1F         852      Pop Ds ; and
FF4D CF         853      Iret ; Return
                854      IllegalInt Endp
                855
                856      ;
                857      ;
                858      ;
                859      ;
FF53             860      Org 0FF53H ;Align with Xt and Pc
FF53             861      DummyReturn:
FF53 CF         862      Iret ;Return found in Pc and Xt
                863
-----         864      Bios Ends
                865
                866      End

```

ASSEMBLY COMPLETE, NO ERRORS FOUND

IBM PC/XT 8086/8087/8088 MACRO ASSEMBLER V1.1 ASSEMBLY OF MODULE USEFUL
OBJECT MODULE PLACED IN USEFUL.OBJ
ASSEMBLER INVOKED BY: ASMB6 USEFUL.SRC

```
LOC OBJ          LINE    SOURCE
                1 +1  $Title ('DTC/PC BIOS Useful Procedures V1.0')
                2 +1  $Pagelength (80) Pagewidth (132) Debug Nogen
                3      Name Useful
                4
                5
                6      ;   Author:      Don K. Harrison
                7
                8      ;   Start date:  December 14, 1983      Last edit:  December 27, 1983
                9      ;
               10      ;
               11      ;   *****
               12      ;   * Module Description *
               13      ;   *****
               14      ;
               15      ;   This module contains some useful subroutines, including:
               16      ;
               17      ;       Beep                Beeps the bell
               18      ;       PrintMessage       Prints a message ending in 0
               19      ;       WordOut            Outputs a word value from Ax
               20      ;       ByteOut           Outputs a byte value from Al
               21      ;       CharOut            Outputs a character in Al
               22      ;       NibbleOut           Outputs a single nibble from Al
               23      ;       CrtCrLf           Outputs a carriage return line feed
               24      ;       KeyIn             Gets character from keyboard
               25      ;       VideoInit          Initialize video from switches
               26      ;       ROMCheck8K        Rom checksum procedure
               27      ;       MemoryTest          Ram memory test
               28      ;       ClearScreen          Clear screen
               29      ;       PositionCursor       Position cursor from Ax
               30      ;
               31      ;       (c) Display Telecommunications Corporation, 1983
               32      ;       All Rights Reserved
               33      ;
               34      $Eject
```

LDC	OBJ	LINE	SOURCE
		35	
		36	
		37	;
		38	;
		39	;
		40	
		41	
		42	
		43	
		44	
		45	
		46	
		47	\$Eject

```
*****  
* Revision History *  
*****
```

```
LOC OBJ          LINE    SOURCE
                48
                49          ; *****
                50          ; * Public Symbols *
                51          ; *****
                52
                53          Public ByteOut, WordOut, CharOut, NibbleOut, KeyIn
                54          Public PrintMessage, Beep, VideoInit, CrtCrLf
                55          Public ROMCheck8K, MemoryTest, ROMCheckCx
                56          Public ClearScreen, PositionCursor
                57
                58          ; *****
                59          ; * Equates *
                60          ; *****
                61
000D             62          AsciiCarriage Equ    0DH          ;Carriage return
000A             63          AsciiLineFeed Equ   0AH          ;Line feed
                64
                65          $Include (IbmInc)
=1              66          ; *****
=1              67          ; * Global Include File *
=1              68          ; *****
=1              69          $Nolist
589             $Eject
```

LOC	OBJ	LINE	SOURCE
		590	;
		591	;
		592	;
		593	*****
		594	
----		595	BiosDataArea Segment Public
		596	Extrn EquipFlag:Word
----		597	BiosDataArea Ends
		598	
		599	\$Eject

```

LOC  OBJ          LINE    SOURCE
-----
                                *****
                                * Code Segment *
                                *****
600      ;
601      ;
602      ;
603      ;
604      Bios      Segment Common
605
606      Extrn  NMIInt:Near
607      Assume Cs:Bios, Ds:BiosDataArea
608
F950     609      Org      0F950H          ;Available spot in ROM
610
611      ;
612      ;
613      ;
614      ;
615      ByteOut   Proc      Near
F950 50    616      Push   Ax          ;Save low nibble
F951 B104  617      Mov    C1,4        ;...Shift it to lower nibble
F953 D2E8  618      Shr   A1,C1        ;...and
F955 E81B00 619      Call  NibbleOut   ;...output it
F958 58    620      Pop   Ax          ;Restore lower nibble
F959 E81700 621      Call  NibbleOut   ;...and output it
F95C C3    622      Ret                    ;...then return
623      ByteOut   Endp
624
625      ;
626      ;
627      ;
628      ;
629      WordOut   Proc      Near
F95D 50    630      Push   Ax          ;Save Lo byte
F95E 8AC4  631      Mov    A1,Ah       ;Get Hi byte first
F960 E8EDFF 632      Call  ByteOut     ;...and output it
F963 58    633      Pop   Ax          ;Get lo byte
F964 E8E9FF 634      Call  ByteOut     ;...and output it
F967 C3    635      Ret                    ;...then return
636      WordOut   Endp
637
638      ;
639      ;
640      ;
641      ;
642      CharOut   Proc      Near
F968 53    643      Push   Bx          ;Save Bx
F969 50    644      Push   Ax          ;...and Ax
F96A B40E  645      Mov    Ah,VidCmdWrTTY ;Command = Write TTY
F96C B307  646      Mov    B1,7        ;Foreground color if in graph.
F96E CD10  647      Int   TrapVideo   ;Send char TTY style
F970 58    648      Pop   Ax          ;Restore
F971 58    649      Pop   Bx          ;...registers
F972 C3    650      Ret                    ;...and return
651      CharOut   Endp
652
653      ;
654      ;
655      ;
656      ;
657      NibbleOut Proc      Near
F973 50    658      Push   Ax          ;Save Ax
F974 240F  659      And   A1,0FH       ;Strip upper, just in case
F976 3C09  660      Cmp   A1,9         ;Is it greater than 9?
F978 7602  661      Jbe   LessEqu9     ;...jump if it isn't
F97A 0407  662      Add   A1,7         ;...else add diff from 9 to A
F97C      663      LessEqu9:
F97C 0430  664      Add   A1,'0'       ;Add ascii offset
F97E E8E7FF 665      Call  CharOut     ;...output it
F981 58    666      Pop   Ax          ;Restore Ax
F982 C3    667      Ret                    ;...and return
668      NibbleOut Endp
669
670      ;
671      ;
672      ;
673      ;
F983     674      CrtCrLf Proc      Near

```

```

LOC OBJ          LINE    SOURCE
F983 B00D        675      Mov     Al,AsciiCarriage
F985 E8E0FF      676      Call    CharOut
F988 B00A        677      Mov     Al,AsciiLineFeed
F98A E8DBFF      678      Call    CharOut
F98D C3          679      Ret
                680      CrtCrLf Endp
                681
                682      ;
                683      ; *****
                684      ; * Get Keyboard Character *
                685      ; *****
F98E            686      KeyIn   Proc     Near
F98E B400        687      Mov     Ah,0                ;Character command
F990 CD16        688      Int     TrapKeyDrive       ;...do procedure
F992 C3          689      Ret                          ;...and return
                690      KeyIn   Endp
                691
                692      ;
                693      ; *****
                694      ; * Print a Message *
                695      ; *****
F993            696      PrintMessage Proc     Near
F993 AC          697      LodsB                ;Get a character
F994 0AC0        698      Or      Al,Al          ;...is it 0?
F996 7501        699      Jnz     PrintMsgCont    ;...jump if not
F998 C3          700      Ret                          ;...else return
F999            701      PrintMsgCont:
F999 E8CCFF      702      Call    CharOut        ;...output it
F99C EBF5        703      Jmp     PrintMessage    ;...and loop till 0 found
                704      PrintMessage Endp
                705
                706      ;
                707      ; *****
                708      ; * Beep *
                709      ; *****
F99E            710      Beep    Proc     Near
F99E 50          711      Push   Ax              ;Save Ax
F99F 51          712      Push   Cx              ;...and Cx
F9A0 B0B6        713      Mov     Al,10110110B    ;Setup timer
F9A2 E643        714      Out    PortCTCMode,A1  ;...2
F9A4 B82805      715      Mov     Ax,1320         ;Save freq as IBM
F9A7 E642        716      Out    PortCTCLoadCh2,A1 ;...Lo byte
F9A9 8AC4        717      Mov     Al,Ah          ;...Hi
F9AB E642        718      Out    PortCTCLoadCh2,A1 ;...byte
F9AD E461        719      In     Al,PortPPIPortB ;Get current value of ctl port
F9AF 50          720      Push   Ax              ;...save it
F9B0 0C03        721      Or     Al,00000011B    ;Turn on the speaker
F9B2 E661        722      Out    PortPPIPortB,A1 ;...during the beep
F9B4 33C9        723      Xor    Cx,Cx          ;Long count in Cx
F9B6            724      BeepLoop:
F9B6 E2FE        725      Loop   BeepLoop        ;Wait
F9B8 FECB        726      Dec    Bl              ;...decrement user supplied
F9BA 75FA        727      Jnz    BeepLoop        ;...variable and loop till done
F9BC 58          728      Pop    Ax              ;Restore port value prior to
F9BD E661        729      Out    PortPPIPortB,A1 ;...beep and restore it
F9BF 59          730      Pop    Cx              ;Restore used registers
F9C0 58          731      Pop    Ax              ;...and
F9C1 C3          732      Ret                          ;...return
                733      Beep    Endp
                734
                735      ;
                736      ; *****
                737      ; * Video Default Initialization Procedure *
                738      ; *****
F9C2            739      VideoInit Proc     Near
F9C2 8A260000    E 740      Mov     Ah,Byte Ptr EquipFlag ;Which card to init?
F9C6 80E430      741      And    Ah,30H          ;Isolate display bits
F9C9 B000        742      Mov     Al,0            ;Mode = Monochrome
F9CB 80FC30      743      Cmp    Ah,30H          ;Is monochrome selected?
F9CE 7409        744      Je     InitVidJump      ;...jump if so
F9D0 B001        745      Mov     Al,1            ;Mode = 40x25
F9D2 80FC10      746      Cmp    Ah,10H          ;Is color 40x25?
F9D5 7402        747      Je     InitVidJump      ;...jump if so
F9D7 B003        748      Mov     Al,3            ;Mode = 80x25
                749

```

```

LOC OBJ          LINE      SOURCE
F9D9             750      InitVidJmp:
F9D9 B400        751          Mov     Ah,VidCmdInit      ;Command = initialize
F9DB CD10        752          Int     TrapVideo        ;...A1 = target mode
F9DD C3         753          Ret
F9DE             754      VideoInit  Endp
F9DE             755      ;
F9DE             756      ;          *****
F9DE             757      ;          * Clear Screen *
F9DE             758      ;          *****
F9DE             759      ;
F9DE             760      ClearScreen Proc     Near
F9DE B04F18      761          Mov     Dx,184FH          ;Clear screen
F9E1 33C9        762          Xor     Cx,Cx            ;...
F9E3 B80006      763          Mov     Ax,0600H        ;...
F9E6 B707        764          Mov     Bh,7            ;...
F9E8 CD10        765          Int     TrapVideo        ;...
F9EA B402        766          Mov     Ah,VidCmdCurPos ;Home the cursor
F9EC B00000      767          Mov     Dx,0            ;...
F9EF B700        768          Mov     Bh,0            ;...
F9F1 CD10        769          Int     TrapVideo        ;...
F9F3 C3         770          Ret
F9F3 C3         771          ;Return
F9F3 C3         771      ClearScreen Endp
F9F3 C3         772      ;
F9F3 C3         773      ;          *****
F9F3 C3         774      ;          * Position Cursor *
F9F3 C3         775      ;          *****
F9F3 C3         776      ;
F9F4             777      PositionCursor Proc     Near
F9F4 52          778          Push   Dx                ;Save Dx
F9F5 53          779          Push   Bx                ;...and Bx
F9F6 8BD0        780          Mov     Dx,Ax            ;Position from Ax
F9F8 B402        781          Mov     Ah,VidCmdCurPos ;Load the command
F9FA B700        782          Mov     Bh,0            ;...page 0
F9FC CD10        783          Int     TrapVideo        ;...Do-it
F9FE 5B          784          Pop     Bx                ;Restore
F9FF 5A          785          Pop     Dx                ;...registers
FA00 C3         786          Ret
FA00 C3         786          ;Return
FA00 C3         787      PositionCursor endp
FA00 C3         788      ;
FA00 C3         789      ;          *****
FA00 C3         790      ;          * ROM Checksum Procedure:  Entry 1 = 8K, Entry 2 = (Cx)K *
FA00 C3         791      ;          *****
FA00 C3         792      ;
FA01             793      ROMCheck8K  Proc     Near
FA01 B90020      794          Mov     Cx,8192          ;Length of ROM = 8k
FA04             795      ROMCheckCx:
FA04 B000        796          Mov     Al,0            ;Zero the checksum
FA06             797      ROMCheckLoop:
FA06 0207        798          Add     Al,DS:[Bx]        ;Bx holds the offset
FA08 43          799          Inc     Bx                ;...inc it
FA09 E2FB        800          Loop   ROMCheckLoop      ;...and loop till all added
FA0B 0AC0        801          Or     Al,Al            ;Affect Z flag with Al
FA0D C3         802          Ret
FA0D C3         802          ;...and return, if z=1, rom=ok
FA0D C3         803      ROMCheck8k  Endp
FA0D C3         804      ;
FA0D C3         805      ;          *****
FA0D C3         806      ;          * Memory Test *
FA0D C3         807      ;          -----
FA0D C3         808      ;          * Es:00 = Start Address *
FA0D C3         809      ;          * Returns: *
FA0D C3         810      ;          * If Cy = 0, NO ERROR *
FA0D C3         811      ;          * else Es:Di = addr. *
FA0D C3         812      ;          *****
FA0D C3         813      ;
FA0E             814      MemoryTest  Proc     Near
FA0E B80004      815          Mov     Bx,1024          ;Do a 1k Block
FA11 B055        816          Mov     Al,055H          ;Start with 55H
FA13 33FF        817          Xor     Di,Di            ;Re-load pointer
FA15 8BCB        818          Mov     Cx,Bx            ;Re-load count
FA17 F3          819          Rep   Stosb              ;Write it
FA18 AA          ;
FA19 33FF        820          Xor     Di,Di            ;Re-load pointer
FA1B 8BCB        821          Mov     Cx,Bx            ;Re-load count
FA1D F3          822          Repe  Scasb              ;Test memory with accum
FA1E AE          ;

```

LOC	OBJ	LINE	SOURCE
FA1F	E302	823	Jcxz TestAA
FA21	F9	824	Stc
FA22	C3	825	Ret
FA23		826	TestAA:
FA23	33FF	827	Xor Di,Di
FA25	BBCB	828	Mov Cx,Bx
FA27	F6D0	829	Not Al
FA29	F3	830	Rep Stosb
FA2A	AA		
FA2B	33FF	831	Xor Di,Di
FA2D	BBCB	832	Mov Cx,Bx
FA2F	F3	833	Repe Scasb
FA30	AE		
FA31	E302	834	Jcxz Test01
FA33	F9	835	Stc
FA34	C3	836	Ret
FA35		837	Test01:
FA35	33FF	838	Xor Di,Di
FA37	BBCB	839	Mov Cx,Bx
FA39	B001	840	Mov Al,1
FA3B	F3	841	Rep Stosb
FA3C	AA		
FA3D	33FF	842	Xor Di,Di
FA3F	BBCB	843	Mov Cx,Bx
FA41	F3	844	Repe Scasb
FA42	AE		
FA43	E302	845	Jcxz Test00
FA45	F9	846	Stc
FA46	C3	847	Ret
FA47		848	Test00:
FA47	33FF	849	Xor Di,Di
FA49	BBCB	850	Mov Cx,Bx
FA4B	FECB	851	Dec Al
FA4D	F3	852	Rep Stosb
FA4E	AA		
FA4F	33FF	853	Xor Di,Di
FA51	BBCB	854	Mov Cx,Bx
FA53	F3	855	Repe Scasb
FA54	AE		
FA55	E302	856	Jcxz NoMemError
FA57	F9	857	Stc
FA58	C3	858	Ret
FA59		859	NoMemError:
FA59	BCC0	860	Mov Ax,Es
FA5B	054000	861	Add Ax,1024 / 16
FA5E	BEC0	862	Mov Es,Ax
FA60	C3	863	Ret
		864	MemoryTest Endp
		865	
----		866	Bios Ends
		867	
		868	End

ASSEMBLY COMPLETE, NO ERRORS FOUND

IBM PC/XT 8086/8087/8088 MACRO ASSEMBLER V1.1 ASSEMBLY OF MODULE GRAPHICS
OBJECT MODULE PLACED IN GRAPHICS.OBJ
ASSEMBLER INVOKED BY: ASM86 GRAPHICS.SRC

```
LOC OBJ          LINE    SOURCE
                1 +1    $Title ('DTC/PC BIOS Graphic Character Generator V1.0')
                2 +1    $Pagelength (80) Pagewidth (132) Debug Nogen
                3        Name Graphics
                4
                5
                6        ;    Author:      Don K. Harrison
                7
                8        ;    Start date:  November 25, 1983      Last edit:  November 25, 1983
                9
                10
                11        ;                *****
                12        ;                * Module Description *
                13        ;                *****
                14
                15        ;    This module contains the character generator array used to
                16        ;    produce character data on a graphics screen.
                17
                18
                19
                20
                21
                22        ;                (c) Display Telecommunications Corporation, 1983
                23        ;                All Rights Reserved
                24
                25    $Eject
```

LOC	OBJ	LINE	SOURCE
		26	
		27	
		28	;
		29	;
		30	;
		31	*****
		32	
		33	
		34	
		35	
		36	
		37	
		38	\$Eject

LOC	OBJ	LINE	SOURCE
		39	
		40	;
		41	;
		42	*****
		43	;
		44	Public VideoGraphicsPointer
		45	
		46	
		47	*****
		48	;
		49	*****
---		50	
		51	Bios Segment Common
		52	
		53	Assume Cs:Bios
		54	\$Eject

LOC	OBJ	LINE	SOURCE	Org	Label	Byte	
FA6E		55		Org	0FA6EH		
FA6E		56					
FA6E		57	VideoGraphicsPointer			Label	Byte
FA6E	00	58					
FA6E	00	59		Db	0000000B		;..... ;00H (Ct1 0)
FA6F	00	60		Db	0000000B		;.....
FA70	00	61		Db	0000000B		;.....
FA71	00	62		Db	0000000B		;.....
FA72	00	63		Db	0000000B		;.....
FA73	00	64		Db	0000000B		;.....
FA74	00	65		Db	0000000B		;.....
FA75	00	66		Db	0000000B		;.....
		67					
FA76	7E	68		Db	0111110B		;.000000. ;01H (Ct1 A)
FA77	81	69		Db	1000001B		;0.....0
FA78	A5	70		Db	10100101B		;0.0..0.0
FA79	81	71		Db	1000001B		;0.....0
FA7A	8D	72		Db	10111101B		;0.0000.0
FA7B	99	73		Db	10011001B		;0..00..0
FA7C	81	74		Db	1000001B		;0.....0
FA7D	7E	75		Db	0111110B		;.000000.
		76					
FA7E	7E	77		Db	0111110B		;.000000. ;02H (Ct1 B)
FA7F	FF	78		Db	11111111B		;00000000
FA80	DB	79		Db	11011011B		;00.00.00
FA81	FF	80		Db	11111111B		;00000000
FA82	C3	81		Db	1100011B		;00....00
FA83	E7	82		Db	11100111B		;000..000
FA84	FF	83		Db	11111111B		;00000000
FA85	7E	84		Db	0111110B		;.000000.
		85					
FA86	6C	86		Db	01101100B		;.00.00.. ;03H (Ct1 C)
FA87	FE	87		Db	1111110B		;0000000.
FA88	FE	88		Db	1111110B		;0000000.
FA89	FE	89		Db	1111110B		;0000000.
FA8A	7C	90		Db	01111100B		;.000000..
FA8B	38	91		Db	00111000B		;..000...
FA8C	10	92		Db	00010000B		;...0....
FA8D	00	93		Db	00000000B		;.....
		94					
FA8E	10	95		Db	00010000B		;...0.... ;04H (Ct1 D)
FA8F	38	96		Db	00111000B		;..000...
FA90	7C	97		Db	01111100B		;.000000..
FA91	FE	98		Db	1111110B		;0000000.
FA92	7C	99		Db	01111100B		;.000000..
FA93	38	100		Db	00111000B		;..000...
FA94	10	101		Db	00010000B		;...0....
FA95	00	102		Db	00000000B		;.....
		103					
FA96	38	104		Db	00111000B		;..000... ;05H (Ct1 E)
FA97	7C	105		Db	01111100B		;.000000..
FA98	38	106		Db	00111000B		;..000...
FA99	FE	107		Db	1111110B		;0000000.
FA9A	FE	108		Db	1111110B		;0000000.
FA9B	7C	109		Db	01111100B		;.000000..
FA9C	38	110		Db	00111000B		;..000...
FA9D	7C	111		Db	01111100B		;.000000..
		112					
FA9E	10	113		Db	00010000B		;...0.... ;06H (Ct1 F)
FA9F	10	114		Db	00010000B		;...0....
FAA0	38	115		Db	00111000B		;..000...
FAA1	7C	116		Db	01111100B		;.000000..
FAA2	FE	117		Db	1111110B		;0000000.
FAA3	7C	118		Db	01111100B		;.000000..
FAA4	38	119		Db	00111000B		;..000...
FAA5	7C	120		Db	01111100B		;.000000..
		121					
FAA6	00	122		Db	00000000B		;..... ;07H (Ct1 G)
FAA7	00	123		Db	00000000B		;.....
FAA8	18	124		Db	00011000B		;...00...
FAA9	3C	125		Db	00111100B		;..0000..
FAAA	3C	126		Db	00111100B		;..0000..
FAAB	18	127		Db	00011000B		;...00...
FAAC	00	128		Db	00000000B		;.....
FAAD	00	129		Db	00000000B		;.....

LOC	OBJ	LINE	SOURCE
		130	\$Eject

LOC	OBJ	LINE	SOURCE
FABE	FF	131	Db 1111111B ;0000000 ;08H (Ct1 H)
FABF	FF	132	Db 1111111B ;0000000
FAB0	E7	133	Db 11100111B ;000..000
FAB1	C3	134	Db 11000011B ;00....00
FAB2	C3	135	Db 11000011B ;00....00
FAB3	E7	136	Db 11100111B ;000..000
FAB4	FF	137	Db 1111111B ;0000000
FAB5	FF	138	Db 1111111B ;0000000
		139	
FAB6	00	140	Db 0000000B ;..... ;09H (Ct1 I)
FAB7	3C	141	Db 00111100B ;..0000..
FAB8	66	142	Db 01100110B ;.00..00.
FAB9	42	143	Db 01000010B ;.0....0.
FABA	42	144	Db 01000010B ;.0....0.
FABB	66	145	Db 01100110B ;.00..00.
FABC	3C	146	Db 00111100B ;..0000..
FABD	00	147	Db 0000000B ;.....
		148	
FABE	FF	149	Db 1111111B ;0000000 ;0AH (Ct1 J)
FABF	C3	150	Db 11000011B ;00....00
FAC0	99	151	Db 10011001B ;0..00..0
FAC1	8D	152	Db 10111101B ;0.0000.0
FAC2	8D	153	Db 10111101B ;0.0000.0
FAC3	99	154	Db 10011001B ;0..00..0
FAC4	C3	155	Db 11000011B ;00....00
FAC5	FF	156	Db 1111111B ;0000000
		157	
FAC6	0F	158	Db 00001111B ;....0000 ;0BH (Ct1 K)
FAC7	07	159	Db 00000111B ;.....000
FAC8	0F	160	Db 00001111B ;....0000
FAC9	7D	161	Db 01111101B ;.00000.0
FACA	CC	162	Db 11001100B ;00..00..
FACB	CC	163	Db 11001100B ;00..00..
FACC	CC	164	Db 11001100B ;00..00..
FACD	78	165	Db 01111000B ;.0000...
		166	
FACE	3C	167	Db 00111100B ;..0000.. ;0CH (Ct1 L)
FACF	66	168	Db 01100110B ;.00..00.
FAD0	66	169	Db 01100110B ;.00..00.
FAD1	66	170	Db 01100110B ;.00..00.
FAD2	3C	171	Db 00111100B ;..0000..
FAD3	18	172	Db 00011000B ;...00..
FAD4	7E	173	Db 01111110B ;.000000.
FAD5	18	174	Db 00011000B ;...00..
		175	
FAD6	3F	176	Db 00111111B ;..000000 ;0DH (Ct1 M)
FAD7	33	177	Db 00110011B ;..00..00
FAD8	3F	178	Db 00111111B ;..000000
FAD9	30	179	Db 00110000B ;..00....
FADA	30	180	Db 00110000B ;..00....
FADB	70	181	Db 01110000B ;.000....
FADC	F0	182	Db 11110000B ;0000....
FADD	E0	183	Db 11100000B ;000.....
		184	
FADE	7F	185	Db 01111111B ;.0000000 ;0EH (Ct1 N)
FADF	63	186	Db 01100011B ;.00..00
FAE0	7F	187	Db 01111111B ;.0000000
FAE1	63	188	Db 01100011B ;.00..00
FAE2	63	189	Db 01100011B ;.00..00
FAE3	67	190	Db 01100111B ;.00..000
FAE4	E6	191	Db 11100110B ;000..00.
FAE5	C0	192	Db 11000000B ;00.....
		193	
FAE6	99	194	Db 10011001B ;0..00..0 ;0FH (Ct1 O)
FAE7	5A	195	Db 01011010B ;.0.00.0.
FAE8	3C	196	Db 00111100B ;..0000..
FAE9	E7	197	Db 11100111B ;000..000
FAEA	E7	198	Db 11100111B ;000..000
FAEB	3C	199	Db 00111100B ;..0000..
FAEC	5A	200	Db 01011010B ;.0.00.0.
FAED	99	201	Db 10011001B ;0..00..0
		202	

\$Eject

LOC	OBJ	LINE	SOURCE
FAEE	80	203	Db 10000000B ;D..... ;10H (Ct1 P)
FAEF	E0	204	Db 11100000B ;000.....
FAF0	F8	205	Db 11111000B ;00000...
FAF1	FE	206	Db 11111100B ;0000000.
FAF2	F8	207	Db 11111000B ;00000...
FAF3	E0	208	Db 11100000B ;000.....
FAF4	80	209	Db 10000000B ;D.....
FAF5	00	210	Db 00000000B ;.....
		211	
FAF6	02	212	Db 00000100B ;.....D. ;11H (Ct1 Q)
FAF7	0E	213	Db 00001110B ;....000.
FAF8	3E	214	Db 00111110B ;..00000.
FAF9	FE	215	Db 11111100B ;0000000.
FAFA	3E	216	Db 00111110B ;..00000.
FAFB	0E	217	Db 00001110B ;....000.
FAFC	02	218	Db 00000100B ;.....D.
FAFD	00	219	Db 00000000B ;.....
		220	
FAFE	18	221	Db 00011000B ;...00... ;12H (Ct1 R)
FAFF	3C	222	Db 00111100B ;..0000..
FB00	7E	223	Db 01111100B ;.000000.
FB01	18	224	Db 00011000B ;...00...
FB02	18	225	Db 00011000B ;...00...
FB03	7E	226	Db 01111100B ;.000000.
FB04	3C	227	Db 00111100B ;..0000..
FB05	18	228	Db 00011000B ;...00...
		229	
FB06	66	230	Db 01100110B ;.00..00. ;13H (Ct1 S)
FB07	66	231	Db 01100110B ;.00..00.
FB08	66	232	Db 01100110B ;.00..00.
FB09	66	233	Db 01100110B ;.00..00.
FB0A	66	234	Db 01100110B ;.00..00.
FB0B	00	235	Db 00000000B ;.....
FB0C	66	236	Db 01100110B ;.00..00.
FB0D	00	237	Db 00000000B ;.....
		238	
FB0E	7F	239	Db 01111111B ;.0000000 ;14H (Ct1 T)
FB0F	DB	240	Db 11011011B ;00.00.00
FB10	DB	241	Db 11011011B ;00.00.00
FB11	7B	242	Db 01111011B ;.0000.00
FB12	1B	243	Db 00011011B ;...00.00
FB13	1B	244	Db 00011011B ;...00.00
FB14	1B	245	Db 00011011B ;...00.00
FB15	00	246	Db 00000000B ;.....
		247	
FB16	3E	248	Db 00111110B ;..00000. ;15H (Ct1 U)
FB17	63	249	Db 01100011B ;.00...00
FB18	38	250	Db 00111000B ;..000...
FB19	6C	251	Db 01101100B ;.00.00..
FB1A	6C	252	Db 01101100B ;.00.00..
FB1B	38	253	Db 00111000B ;..000...
FB1C	CC	254	Db 11001100B ;00..00..
FB1D	78	255	Db 01111000B ;.0000...
		256	
FB1E	00	257	Db 00000000B ;..... ;16H (Ct1 V)
FB1F	00	258	Db 00000000B ;.....
FB20	00	259	Db 00000000B ;.....
FB21	00	260	Db 00000000B ;.....
FB22	7E	261	Db 01111100B ;.000000.
FB23	7E	262	Db 01111100B ;.000000.
FB24	7E	263	Db 01111100B ;.000000.
FB25	00	264	Db 00000000B ;.....
		265	
FB26	18	266	Db 00011000B ;...00... ;17H (Ct1 W)
FB27	3C	267	Db 00111100B ;..0000..
FB28	7E	268	Db 01111100B ;.000000.
FB29	18	269	Db 00011000B ;...00...
FB2A	7E	270	Db 01111100B ;.000000.
FB2B	3C	271	Db 00111100B ;..0000..
FB2C	18	272	Db 00011000B ;...00...
FB2D	FF	273	Db 11111111B ;00000000
		274	\$Eject

LOC	OBJ	LINE	SOURCE
FB2E	18	275	Db 00011000B ;...00... ;18H (Ct1 X)
FB2F	3C	276	Db 00111100B ;..0000..
FB30	7E	277	Db 01111110B ;.000000.
FB31	18	278	Db 00011000B ;...00...
FB32	18	279	Db 00011000B ;...00...
FB33	18	280	Db 00011000B ;...00...
FB34	18	281	Db 00011000B ;...00...
FB35	00	282	Db 00000000B ;.....
		283	
FB36	18	284	Db 00011000B ;...00... ;19H (Ct1 Y)
FB37	18	285	Db 00011000B ;...00...
FB38	18	286	Db 00011000B ;...00...
FB39	18	287	Db 00011000B ;...00...
FB3A	7E	288	Db 01111110B ;.000000.
FB3B	3C	289	Db 00111100B ;..0000..
FB3C	18	290	Db 00011000B ;...00...
FB3D	00	291	Db 00000000B ;.....
		292	
FB3E	00	293	Db 00000000B ;..... ;1AH (Ct1 Z)
FB3F	18	294	Db 00011000B ;...00...
FB40	0C	295	Db 00001100B ;....00..
FB41	FE	296	Db 11111110B ;0000000.
FB42	0C	297	Db 00001100B ;....00..
FB43	18	298	Db 00011000B ;...00...
FB44	00	299	Db 00000000B ;.....
FB45	00	300	Db 00000000B ;.....
		301	
FB46	00	302	Db 00000000B ;..... ;1BH (Ct1 [)
FB47	30	303	Db 00110000B ;..00....
FB48	60	304	Db 01100000B ;.00.....
FB49	FE	305	Db 11111110B ;0000000.
FB4A	60	306	Db 01100000B ;.00.....
FB4B	30	307	Db 00110000B ;..00....
FB4C	00	308	Db 00000000B ;.....
FB4D	00	309	Db 00000000B ;.....
		310	
FB4E	00	311	Db 00000000B ;..... ;1CH (Ct1 \)
FB4F	00	312	Db 00000000B ;.....
FB50	C0	313	Db 11000000B ;00.....
FB51	C0	314	Db 11000000B ;00.....
FB52	C0	315	Db 11000000B ;00.....
FB53	FE	316	Db 11111110B ;0000000.
FB54	00	317	Db 00000000B ;.....
FB55	00	318	Db 00000000B ;.....
		319	
FB56	00	320	Db 00000000B ;..... ;1DH (Ct1])
FB57	24	321	Db 00100100B ;..0..0..
FB58	66	322	Db 01100110B ;.00..00.
FB59	FF	323	Db 11111111B ;00000000
FB5A	66	324	Db 01100110B ;.00..00.
FB5B	24	325	Db 00100100B ;..0..0..
FB5C	00	326	Db 00000000B ;.....
FB5D	00	327	Db 00000000B ;.....
		328	
FB5E	00	329	Db 00000000B ;..... ;1EH (Ct1 ^)
FB5F	18	330	Db 00011000B ;...00...
FB60	3C	331	Db 00111100B ;..0000..
FB61	7E	332	Db 01111110B ;.000000.
FB62	FF	333	Db 11111111B ;00000000
FB63	FF	334	Db 11111111B ;00000000
FB64	00	335	Db 00000000B ;.....
FB65	00	336	Db 00000000B ;.....
		337	
FB66	00	338	Db 00000000B ;..... ;1FH (Ct1 _)
FB67	FF	339	Db 11111111B ;00000000
FB68	FF	340	Db 11111111B ;00000000
FB69	7E	341	Db 01111110B ;.000000.
FB6A	3C	342	Db 00111100B ;..0000..
FB6B	18	343	Db 00011000B ;...00...
FB6C	00	344	Db 00000000B ;.....
FB6D	00	345	Db 00000000B ;.....
		346	\$Eject

LOC	OBJ	LINE	SOURCE			
FB6E	00	347	Db	0000000B	; ;20H (Space)
FB6F	00	348	Db	0000000B	;
FB70	00	349	Db	0000000B	;
FB71	00	350	Db	0000000B	;
FB72	00	351	Db	0000000B	;
FB73	00	352	Db	0000000B	;
FB74	00	353	Db	0000000B	;
FB75	00	354	Db	0000000B	;
		355				
FB76	30	356	Db	0011000B	;	.00.... ;21H (!)
FB77	78	357	Db	0111100B	;	.0000...
FB78	78	358	Db	0111100B	;	.0000...
FB79	30	359	Db	0011000B	;	.00....
FB7A	30	360	Db	0011000B	;	.00....
FB7B	00	361	Db	0000000B	;
FB7C	30	362	Db	0011000B	;	.00....
FB7D	00	363	Db	0000000B	;
		364				
FB7E	6C	365	Db	0110110B	;	.00.00.. ;22H (*)
FB7F	6C	366	Db	0110110B	;	.00.00..
FB80	6C	367	Db	0110110B	;	.00.00..
FB81	00	368	Db	0000000B	;
FB82	00	369	Db	0000000B	;
FB83	00	370	Db	0000000B	;
FB84	00	371	Db	0000000B	;
FB85	00	372	Db	0000000B	;
		373				
FB86	6C	374	Db	0110110B	;	.00.00.. ;23H (#)
FB87	6C	375	Db	0110110B	;	.00.00..
FB88	FE	376	Db	1111110B	;	.0000000.
FB89	6C	377	Db	0110110B	;	.00.00..
FB8A	FE	378	Db	1111110B	;	.0000000.
FB8B	6C	379	Db	0110110B	;	.00.00..
FB8C	6C	380	Db	0110110B	;	.00.00..
FB8D	00	381	Db	0000000B	;
		382				
FB8E	30	383	Db	0011000B	;	.00.... ;24H (\$)
FB8F	7C	384	Db	0111100B	;	.000000..
FB90	C0	385	Db	1100000B	;	.00.....
FB91	78	386	Db	0111100B	;	.000000..
FB92	0C	387	Db	0000110B	;00..
FB93	F8	388	Db	1111100B	;	.000000..
FB94	30	389	Db	0011000B	;	.00....
FB95	00	390	Db	0000000B	;
		391				
FB96	00	392	Db	0000000B	; ;25H (Percent)
FB97	C6	393	Db	11000110B	;	.00...00.
FB98	CC	394	Db	1100110B	;	.00..00..
FB99	18	395	Db	0001100B	;	...00...
FB9A	30	396	Db	0011000B	;	.00....
FB9B	66	397	Db	01100110B	;	.00..00.
FB9C	C6	398	Db	11000110B	;	.00..00.
FB9D	00	399	Db	0000000B	;
		400				
FB9E	38	401	Db	0011100B	;	.0000... ;26H (&)
FB9F	6C	402	Db	0110110B	;	.00.00..
FBA0	38	403	Db	0011100B	;	.0000...
FBA1	76	404	Db	0110110B	;	.000.00.
FBA2	DC	405	Db	1101110B	;	.00.000..
FBA3	CC	406	Db	1100110B	;	.00..00..
FBA4	76	407	Db	0110110B	;	.000.00.
FBA5	00	408	Db	0000000B	;
		409				
FBA6	60	410	Db	0110000B	;	.00..... ;27H (')
FBA7	60	411	Db	0110000B	;	.00.....
FBA8	C0	412	Db	1100000B	;	.00.....
FBA9	00	413	Db	0000000B	;
FBAA	00	414	Db	0000000B	;
FBAB	00	415	Db	0000000B	;
FBAC	00	416	Db	0000000B	;
FBAD	00	417	Db	0000000B	;
		418				

\$Eject

LOC	OBJ	LINE	SOURCE
FBAE	18	419	Db 00011000B ;...00... ;28H ()
FBAF	30	420	Db 00110000B ;..00....
FBB0	60	421	Db 01100000B ;.00.....
FBB1	60	422	Db 01100000B ;.00.....
FBB2	60	423	Db 01100000B ;.00.....
FBB3	30	424	Db 00110000B ;..00....
FBB4	18	425	Db 00011000B ;...00...
FBB5	00	426	Db 00000000B ;.....
		427	
FBB6	60	428	Db 01100000B ;.00..... ;29H ()
FBB7	30	429	Db 00110000B ;..00....
FBB8	18	430	Db 00011000B ;...00...
FBB9	18	431	Db 00011000B ;...00...
FBBA	18	432	Db 00011000B ;...00...
FBBB	30	433	Db 00110000B ;..00....
FBBC	60	434	Db 01100000B ;.00.....
FBBD	00	435	Db 00000000B ;.....
		436	
FBBE	00	437	Db 00000000B ;..... ;2AH (+)
FBBF	66	438	Db 01100110B ;.00..00.
FBC0	3C	439	Db 00111100B ;..0000..
FBC1	FF	440	Db 11111111B ;00000000
FBC2	3C	441	Db 00111100B ;..0000..
FBC3	66	442	Db 01100110B ;.00..00.
FBC4	00	443	Db 00000000B ;.....
FBC5	00	444	Db 00000000B ;.....
		445	
FBC6	00	446	Db 00000000B ;..... ;2BH (+)
FBC7	30	447	Db 00110000B ;..00....
FBC8	30	448	Db 00110000B ;..00....
FBC9	FC	449	Db 11111100B ;000000..
FBCA	30	450	Db 00110000B ;..00....
FBCB	30	451	Db 00110000B ;..00....
FBCD	00	452	Db 00000000B ;.....
		453	Db 00000000B ;.....
		454	
FBCE	00	455	Db 00000000B ;..... ;2CH (,)
FBCF	00	456	Db 00000000B ;.....
FBD0	00	457	Db 00000000B ;.....
FBD1	00	458	Db 00000000B ;.....
FBD2	00	459	Db 00000000B ;.....
FBD3	30	460	Db 00110000B ;..00....
FBD4	30	461	Db 00110000B ;..00....
FBD5	60	462	Db 01100000B ;.00.....
		463	
FBD6	00	464	Db 00000000B ;..... ;2DH (-)
FBD7	00	465	Db 00000000B ;.....
FBD8	00	466	Db 00000000B ;.....
FBD9	FC	467	Db 11111100B ;000000..
FBDA	00	468	Db 00000000B ;.....
FBDB	00	469	Db 00000000B ;.....
FBDC	00	470	Db 00000000B ;.....
FBD9	00	471	Db 00000000B ;.....
		472	
FBDE	00	473	Db 00000000B ;..... ;2EH (.)
FBD9	00	474	Db 00000000B ;.....
FBE0	00	475	Db 00000000B ;.....
FBE1	00	476	Db 00000000B ;.....
FBE2	00	477	Db 00000000B ;.....
FBE3	30	478	Db 00110000B ;..00....
FBE4	30	479	Db 00110000B ;..00....
FBE5	00	480	Db 00000000B ;.....
		481	
FBE6	06	482	Db 00000110B ;...00. ;2FH (/)
FBE7	0C	483	Db 00001100B ;...00..
FBE8	18	484	Db 00011000B ;...00...
FBE9	30	485	Db 00110000B ;..00....
FBEA	60	486	Db 01100000B ;.00.....
FBE9	00	487	Db 11000000B ;00.....
FBEC	80	488	Db 10000000B ;0.....
FBED	00	489	Db 00000000B ;.....
		490	\$Eject

LOC	OBJ	LINE	SOURCE
FBEE	7C	491	Db 01111100B ;.00000.. ;30H (0)
FBEF	CE	492	Db 11000110B ;00..00.
FBF0	CE	493	Db 11001110B ;00..000.
FBF1	DE	494	Db 11011110B ;00.0000.
FBF2	F6	495	Db 11110110B ;0000.00.
FBF3	E6	496	Db 11100110B ;000..00.
FBF4	7C	497	Db 01111100B ;.00000..
FBF5	00	498	Db 00000000B ;.....
		499	
FBF6	30	500	Db 00110000B ;.00.... ;31H (1)
FBF7	70	501	Db 01110000B ;.000....
FBF8	30	502	Db 00110000B ;.00....
FBF9	30	503	Db 00110000B ;.00....
FBFA	30	504	Db 00110000B ;.00....
FBFB	30	505	Db 00110000B ;.00....
FBFC	FC	506	Db 11111100B ;000000..
FBFD	00	507	Db 00000000B ;.....
		508	
FBFE	78	509	Db 01111000B ;.0000... ;32H (2)
FBFF	CC	510	Db 11001100B ;00..00.
FC00	0C	511	Db 00001100B ;....00..
FC01	38	512	Db 00111000B ;.0000...
FC02	60	513	Db 01100000B ;.00.....
FC03	CC	514	Db 11001100B ;00..00.
FC04	FC	515	Db 11111100B ;000000..
FC05	00	516	Db 00000000B ;.....
		517	
FC06	78	518	Db 01111000B ;.0000... ;33H (3)
FC07	CC	519	Db 11001100B ;00..00.
FC08	0C	520	Db 00001100B ;....00..
FC09	38	521	Db 00111000B ;.0000...
FC0A	0C	522	Db 00001100B ;....00..
FC0B	CC	523	Db 11001100B ;00..00.
FC0C	78	524	Db 01111000B ;.0000...
FC0D	00	525	Db 00000000B ;.....
		526	
FC0E	1C	527	Db 00011100B ;...000.. ;34H (4)
FC0F	3C	528	Db 00111100B ;.00000..
FC10	6C	529	Db 01101100B ;.00.00..
FC11	CC	530	Db 11001100B ;00..00.
FC12	FE	531	Db 11111110B ;0000000.
FC13	0C	532	Db 00001100B ;....00..
FC14	1E	533	Db 00011110B ;...0000.
FC15	00	534	Db 00000000B ;.....
		535	
FC16	FC	536	Db 11111100B ;000000.. ;35H (5)
FC17	C0	537	Db 11000000B ;00.....
FC18	F8	538	Db 11111000B ;00000..
FC19	0C	539	Db 00001100B ;....00..
FC1A	0C	540	Db 00001100B ;....00..
FC1B	CC	541	Db 11001100B ;00..00.
FC1C	78	542	Db 01111000B ;.0000...
FC1D	00	543	Db 00000000B ;.....
		544	
FC1E	38	545	Db 00111000B ;.0000... ;36H (6)
FC1F	60	546	Db 01100000B ;.00.....
FC20	C0	547	Db 11000000B ;00.....
FC21	F8	548	Db 11111000B ;00000..
FC22	CC	549	Db 11001100B ;00..00.
FC23	CC	550	Db 11001100B ;00..00.
FC24	78	551	Db 01111000B ;.0000...
FC25	00	552	Db 00000000B ;.....
		553	
FC26	FC	554	Db 11111100B ;000000.. ;37H (7)
FC27	CC	555	Db 11001100B ;00..00.
FC28	0C	556	Db 00001100B ;....00..
FC29	18	557	Db 00011000B ;...00...
FC2A	30	558	Db 00110000B ;.00.....
FC2B	30	559	Db 00110000B ;.00.....
FC2C	30	560	Db 00110000B ;.00.....
FC2D	00	561	Db 00000000B ;.....
		562	\$Eject

LOC	OBJ	LINE	SOURCE
FC2E	78	563	Db 01111000B ;.0000... ;38H (8)
FC2F	CC	564	Db 11001100B ;00..00..
FC30	CC	565	Db 11001100B ;00..00..
FC31	78	566	Db 01111000B ;.0000...
FC32	CC	567	Db 11001100B ;00..00..
FC33	CC	568	Db 11001100B ;00..00..
FC34	78	569	Db 01111000B ;.0000...
FC35	00	570	Db 00000000B ;.....
		571	
FC36	78	572	Db 01111000B ;.0000... ;39H (9)
FC37	CC	573	Db 11001100B ;00..00..
FC38	CC	574	Db 11001100B ;00..00..
FC39	7C	575	Db 01111100B ;.000000..
FC3A	0C	576	Db 00001100B ;....00..
FC3B	18	577	Db 00011000B ;...00...
FC3C	70	578	Db 01110000B ;.000....
FC3D	00	579	Db 00000000B ;.....
		580	
FC3E	00	581	Db 00000000B ;..... ;3AH (:)
FC3F	30	582	Db 00110000B ;..00....
FC40	30	583	Db 00110000B ;..00....
FC41	00	584	Db 00000000B ;.....
FC42	00	585	Db 00000000B ;.....
FC43	30	586	Db 00110000B ;..00....
FC44	30	587	Db 00110000B ;..00....
FC45	00	588	Db 00000000B ;.....
		589	
FC46	00	590	Db 00000000B ;..... ;3BH (;)
FC47	30	591	Db 00110000B ;..00....
FC48	30	592	Db 00110000B ;..00....
FC49	00	593	Db 00000000B ;.....
FC4A	00	594	Db 00000000B ;.....
FC4B	30	595	Db 00110000B ;..00....
FC4C	30	596	Db 00110000B ;..00....
FC4D	60	597	Db 01100000B ;.00.....
		598	
FC4E	18	599	Db 00011000B ;...00... ;3CH (')
FC4F	30	600	Db 00110000B ;..00....
FC50	60	601	Db 01100000B ;.00.....
FC51	00	602	Db 11000000B ;00.....
FC52	60	603	Db 01100000B ;.00.....
FC53	30	604	Db 00110000B ;..00....
FC54	18	605	Db 00011000B ;...00...
FC55	00	606	Db 00000000B ;.....
		607	
FC56	00	608	Db 00000000B ;..... ;3DH (=)
FC57	00	609	Db 00000000B ;.....
FC58	FC	610	Db 11111100B ;000000..
FC59	00	611	Db 00000000B ;.....
FC5A	00	612	Db 00000000B ;.....
FC5B	FC	613	Db 11111100B ;000000..
FC5C	00	614	Db 00000000B ;.....
FC5D	00	615	Db 00000000B ;.....
		616	
FC5E	60	617	Db 01100000B ;.00..... ;3EH (')
FC5F	30	618	Db 00110000B ;..00....
FC60	18	619	Db 00011000B ;...00...
FC61	0C	620	Db 00001100B ;....00..
FC62	18	621	Db 00011000B ;...00...
FC63	30	622	Db 00110000B ;..00....
FC64	60	623	Db 01100000B ;.00.....
FC65	00	624	Db 00000000B ;.....
		625	
FC66	78	626	Db 01111000B ;.0000... ;3FH (?)
FC67	CC	627	Db 11001100B ;00..00..
FC68	0C	628	Db 00001100B ;....00..
FC69	18	629	Db 00011000B ;...00...
FC6A	30	630	Db 00110000B ;..00....
FC6B	00	631	Db 00000000B ;.....
FC6C	30	632	Db 00110000B ;..00....
FC6D	00	633	Db 00000000B ;.....
		634	

\$Eject

LOC	OBJ	LINE	SOURCE
FC6E	7C	635	Db 01111100B ;.00000.. ;40H (E)
FC6F	C6	636	Db 11000110B ;.00...00.
FC70	DE	637	Db 11011110B ;.00.0000.
FC71	DE	638	Db 11011110B ;.00.0000.
FC72	DE	639	Db 11011110B ;.00.0000.
FC73	C0	640	Db 11000000B ;.00.....
FC74	78	641	Db 01111000B ;.0000...;
FC75	00	642	Db 00000000B ;.....;
		643	
FC76	30	644	Db 00110000B ;..00.... ;41H (A)
FC77	78	645	Db 01111000B ;.0000...;
FC78	CC	646	Db 11001100B ;.00..00..
FC79	CC	647	Db 11001100B ;.00..00..
FC7A	FC	648	Db 11111100B ;.000000..
FC7B	CC	649	Db 11001100B ;.00..00..
FC7C	CC	650	Db 11001100B ;.00..00..
FC7D	00	651	Db 00000000B ;.....;
		652	
FC7E	FC	653	Db 11111100B ;.000000.. ;42H (B)
FC7F	66	654	Db 01100110B ;.00..00..
FC80	66	655	Db 01100110B ;.00..00..
FC81	7C	656	Db 01111100B ;.000000..
FC82	66	657	Db 01100110B ;.00..00..
FC83	66	658	Db 01100110B ;.00..00..
FC84	FC	659	Db 11111100B ;.000000..
FC85	00	660	Db 00000000B ;.....;
		661	
FC86	3C	662	Db 00111100B ;..0000.. ;43H (C)
FC87	66	663	Db 01100110B ;.00..00..
FC88	C0	664	Db 11000000B ;.00.....
FC89	C0	665	Db 11000000B ;.00.....
FC8A	C0	666	Db 11000000B ;.00.....
FC8B	66	667	Db 01100110B ;.00..00..
FC8C	3C	668	Db 00111100B ;..0000..
FC8D	00	669	Db 00000000B ;.....;
		670	
FC8E	F8	671	Db 11111000B ;.000000.. ;44H (D)
FC8F	6C	672	Db 01101100B ;.00.00..
FC90	66	673	Db 01100110B ;.00..00..
FC91	66	674	Db 01100110B ;.00..00..
FC92	66	675	Db 01100110B ;.00..00..
FC93	6C	676	Db 01101100B ;.00.00..
FC94	F8	677	Db 11111000B ;.000000..
FC95	00	678	Db 00000000B ;.....;
		679	
FC96	FE	680	Db 11111110B ;.0000000. ;45H (E)
FC97	62	681	Db 01100010B ;.00...0.
FC98	68	682	Db 01101000B ;.00.0...;
FC99	78	683	Db 01111000B ;.0000...;
FC9A	68	684	Db 01101000B ;.00.0...;
FC9B	62	685	Db 01100010B ;.00...0.
FC9C	FE	686	Db 11111110B ;.0000000.
FC9D	00	687	Db 00000000B ;.....;
		688	
FC9E	FE	689	Db 11111110B ;.0000000. ;46H (F)
FC9F	62	690	Db 01100010B ;.00...0.
FCA0	68	691	Db 01101000B ;.00.0...;
FCA1	78	692	Db 01111000B ;.0000...;
FCA2	68	693	Db 01101000B ;.00.0...;
FCA3	60	694	Db 01100000B ;.00.....
FCA4	F0	695	Db 11110000B ;.0000...;
FCA5	00	696	Db 00000000B ;.....;
		697	
FCA6	3C	698	Db 00111100B ;..0000.. ;47H (G)
FCA7	66	699	Db 01100110B ;.00..00..
FCA8	C0	700	Db 11000000B ;.00.....
FCA9	C0	701	Db 11000000B ;.00.....
FCAA	CE	702	Db 11001110B ;.00..000.
FCAB	66	703	Db 01100110B ;.00..00..
FCAC	3E	704	Db 00111110B ;..00000.
FCAE	00	705	Db 00000000B ;.....;
		706	

\$Eject

LOC	OBJ	LINE	SOURCE
FCAE	CC	707	Db 11001100B ;00..00. ;48H (H)
FCAF	CC	708	Db 11001100B ;00..00.
FCB0	CC	709	Db 11001100B ;00..00.
FCB1	FC	710	Db 11111100B ;000000.
FCB2	CC	711	Db 11001100B ;00..00.
FCB3	CC	712	Db 11001100B ;00..00.
FCB4	CC	713	Db 11001100B ;00..00.
FCB5	00	714	Db 00000000B ;.....
		715	
FCB6	78	716	Db 01111000B ;.0000... ;49H (I)
FCB7	30	717	Db 00110000B ;..00....
FCB8	30	718	Db 00110000B ;..00....
FCB9	30	719	Db 00110000B ;..00....
FCBA	30	720	Db 00110000B ;..00....
FCBB	30	721	Db 00110000B ;..00....
FCBC	78	722	Db 01111000B ;.0000...
FCBD	00	723	Db 00000000B ;.....
		724	
FCBE	1E	725	Db 00011110B ;...0000. ;4AH (J)
FCBF	0C	726	Db 00001100B ;....00..
FCC0	0C	727	Db 00001100B ;....00..
FCC1	0C	728	Db 00001100B ;....00..
FCC2	CC	729	Db 11001100B ;00..00..
FCC3	CC	730	Db 11001100B ;00..00..
FCC4	78	731	Db 01111000B ;.0000...
FCC5	00	732	Db 00000000B ;.....
		733	
FCC6	E6	734	Db 11100110B ;000..00. ;4BH (K)
FCC7	66	735	Db 01100110B ;.00..00.
FCC8	6C	736	Db 01101100B ;.00.00..
FCC9	78	737	Db 01111000B ;.0000...
FCCA	6C	738	Db 01101100B ;.00.00..
FCCB	66	739	Db 01100110B ;.00..00.
FCCC	E6	740	Db 11100110B ;000..00.
FCCD	00	741	Db 00000000B ;.....
		742	
FCCE	F0	743	Db 11110000B ;0000.... ;4CH (L)
FCCF	60	744	Db 01100000B ;.00.....
FCD0	60	745	Db 01100000B ;.00.....
FCD1	60	746	Db 01100000B ;.00.....
FCD2	62	747	Db 01100010B ;.00...0.
FCD3	66	748	Db 01100110B ;.00..00.
FCD4	FE	749	Db 11111110B ;0000000.
FCD5	00	750	Db 00000000B ;.....
		751	
FCD6	C6	752	Db 11000110B ;00...00. ;4DH (M)
FCD7	EE	753	Db 11101110B ;000.000.
FCD8	FE	754	Db 11111110B ;0000000.
FCD9	FE	755	Db 11111110B ;0000000.
FCDA	D6	756	Db 11010110B ;00.0.00.
FCDB	C6	757	Db 11000110B ;00...00.
FCDC	C6	758	Db 11000110B ;00...00.
FCDD	00	759	Db 00000000B ;.....
		760	
FCDE	C6	761	Db 11000110B ;00...00. ;4EH (N)
FCDF	E6	762	Db 11100110B ;000..00.
FCE0	F6	763	Db 11110110B ;0000.00.
FCE1	DE	764	Db 11011110B ;00.0000.
FCE2	CE	765	Db 11001110B ;00..000.
FCE3	C6	766	Db 11000110B ;00...00.
FCE4	C6	767	Db 11000110B ;00...00.
FCE5	00	768	Db 00000000B ;.....
		769	
FCE6	38	770	Db 00111000B ;..000... ;4FH (O)
FCE7	6C	771	Db 01101100B ;.00.00..
FCE8	C6	772	Db 11000110B ;00...00.
FCE9	C6	773	Db 11000110B ;00...00.
FCEA	C6	774	Db 11000110B ;00...00.
FCEB	6C	775	Db 01101100B ;.00.00..
FCEC	38	776	Db 00111000B ;..000...
FCED	00	777	Db 00000000B ;.....
		778	\$Eject

LOC	OBJ	LINE	SOURCE
FCEE	FC	779	Db 11111100B ;000000.. ;50H (P)
FCEF	66	780	Db 01100110B ;.00..00.
FCF0	66	781	Db 01100110B ;.00..00.
FCF1	7C	782	Db 01111100B ;.000000..
FCF2	60	783	Db 01100000B ;.00.....
FCF3	60	784	Db 01100000B ;.00.....
FCF4	F0	785	Db 11110000B ;0000....
FCF5	00	786	Db 00000000B ;.....
		787	
FCF6	78	788	Db 01111000B ;.0000... ;51H (Q)
FCF7	CC	789	Db 11001100B ;00..00..
FCF8	CC	790	Db 11001100B ;00..00..
FCF9	CC	791	Db 11001100B ;00..00..
FCFA	DC	792	Db 11011100B ;00.000..
FCFB	78	793	Db 01111000B ;.0000...;
FCFC	1C	794	Db 00011100B ;...000..
FCFD	00	795	Db 00000000B ;.....
		796	
FCFE	FC	797	Db 11111100B ;000000.. ;52H (R)
FCFF	66	798	Db 01100110B ;.00..00.
FD00	66	799	Db 01100110B ;.00..00.
FD01	7C	800	Db 01111100B ;.000000..
FD02	6C	801	Db 01101100B ;.00.00..
FD03	66	802	Db 01100110B ;.00..00.
FD04	E6	803	Db 11100110B ;000..00.
FD05	00	804	Db 00000000B ;.....
		805	
FD06	78	806	Db 01111000B ;.0000... ;53H (S)
FD07	CC	807	Db 11001100B ;00..00..
FD08	E0	808	Db 11100000B ;000.....
FD09	70	809	Db 01110000B ;.000....
FD0A	1C	810	Db 00011100B ;...000..
FD0B	CC	811	Db 11001100B ;00..00..
FD0C	78	812	Db 01111000B ;.0000...;
FD0D	00	813	Db 00000000B ;.....
		814	
FD0E	FC	815	Db 11111100B ;000000.. ;54H (T)
FD0F	B4	816	Db 10110100B ;0.00.0.
FD10	30	817	Db 00110000B ;..00....
FD11	30	818	Db 00110000B ;..00....
FD12	30	819	Db 00110000B ;..00....
FD13	30	820	Db 00110000B ;..00....
FD14	78	821	Db 01111000B ;.0000...;
FD15	00	822	Db 00000000B ;.....
		823	
FD16	CC	824	Db 11001100B ;00..00.. ;55H (U)
FD17	CC	825	Db 11001100B ;00..00..
FD18	CC	826	Db 11001100B ;00..00..
FD19	CC	827	Db 11001100B ;00..00..
FD1A	CC	828	Db 11001100B ;00..00..
FD1B	CC	829	Db 11001100B ;00..00..
FD1C	FC	830	Db 11111100B ;000000..
FD1D	00	831	Db 00000000B ;.....
		832	
FD1E	CC	833	Db 11001100B ;00..00.. ;56H (V)
FD1F	CC	834	Db 11001100B ;00..00..
FD20	CC	835	Db 11001100B ;00..00..
FD21	CC	836	Db 11001100B ;00..00..
FD22	CC	837	Db 11001100B ;00..00..
FD23	78	838	Db 01111000B ;.0000...;
FD24	30	839	Db 00110000B ;..00....
FD25	00	840	Db 00000000B ;.....
		841	
FD26	C6	842	Db 11000110B ;00...00. ;57H (W)
FD27	C6	843	Db 11000110B ;00...00.
FD28	C6	844	Db 11000110B ;00...00.
FD29	D6	845	Db 11010110B ;00.0.00.
FD2A	FE	846	Db 11111110B ;00000000.
FD2B	EE	847	Db 11101110B ;000.000.
FD2C	C6	848	Db 11000110B ;00...00.
FD2D	00	849	Db 00000000B ;.....
		850	\$Eject

LOC	OBJ	LINE	SOURCE
FD2E	C6	851	Db 11000110B ;00...00. ;58H (X)
FD2F	C6	852	Db 11000110B ;00...00.
FD30	6C	853	Db 01101100B ;.00.00..
FD31	38	854	Db 00111000B ;.000...
FD32	38	855	Db 00111000B ;.000...
FD33	6C	856	Db 01101100B ;.00.00..
FD34	C6	857	Db 11000110B ;00...00.
FD35	00	858	Db 00000000B ;.....
		859	
FD36	CC	860	Db 11001100B ;00..00. ;59H (Y)
FD37	CC	861	Db 11001100B ;00..00..
FD38	CC	862	Db 11001100B ;00..00..
FD39	78	863	Db 01111000B ;.0000...
FD3A	30	864	Db 00110000B ;.00....
FD3B	30	865	Db 00110000B ;.00....
FD3C	78	866	Db 01111000B ;.0000...
FD3D	00	867	Db 00000000B ;.....
		868	
FD3E	FE	869	Db 11111110B ;0000000. ;5AH (Z)
FD3F	C6	870	Db 11000110B ;00...00.
FD40	8C	871	Db 10001100B ;0...00..
FD41	18	872	Db 00011000B ;...00...
FD42	32	873	Db 00110010B ;..00..0.
FD43	66	874	Db 01100110B ;.00..00.
FD44	FE	875	Db 11111110B ;0000000.
FD45	00	876	Db 00000000B ;.....
		877	
FD46	78	878	Db 01111000B ;.0000... ;5BH (I)
FD47	60	879	Db 01100000B ;.00.....
FD48	60	880	Db 01100000B ;.00.....
FD49	60	881	Db 01100000B ;.00.....
FD4A	60	882	Db 01100000B ;.00.....
FD4B	60	883	Db 01100000B ;.00.....
FD4C	78	884	Db 01111000B ;.0000...
FD4D	00	885	Db 00000000B ;.....
		886	
FD4E	C0	887	Db 11000000B ;00..... ;5CH (J)
FD4F	60	888	Db 01100000B ;.00.....
FD50	30	889	Db 00110000B ;..00....
FD51	18	890	Db 00011000B ;...00...
FD52	0C	891	Db 00001100B ;....00..
FD53	06	892	Db 00000110B ;.....00.
FD54	02	893	Db 00000010B ;.....0.
FD55	00	894	Db 00000000B ;.....
		895	
FD56	78	896	Db 01111000B ;.0000... ;5DH (K)
FD57	18	897	Db 00011000B ;...00...
FD58	18	898	Db 00011000B ;...00...
FD59	18	899	Db 00011000B ;...00...
FD5A	18	900	Db 00011000B ;...00...
FD5B	18	901	Db 00011000B ;...00...
FD5C	78	902	Db 01111000B ;.0000...
FD5D	00	903	Db 00000000B ;.....
		904	
FD5E	10	905	Db 00010000B ;...0... ;5EH (^)
FD5F	38	906	Db 00111000B ;..000...
FD60	6C	907	Db 01101100B ;.00.00..
FD61	C6	908	Db 11000110B ;00...00.
FD62	00	909	Db 00000000B ;.....
FD63	00	910	Db 00000000B ;.....
FD64	00	911	Db 00000000B ;.....
FD65	00	912	Db 00000000B ;.....
		913	
FD66	00	914	Db 00000000B ;..... ;5FH ()
FD67	00	915	Db 00000000B ;.....
FD68	00	916	Db 00000000B ;.....
FD69	00	917	Db 00000000B ;.....
FD6A	00	918	Db 00000000B ;.....
FD6B	00	919	Db 00000000B ;.....
FD6C	00	920	Db 00000000B ;.....
FD6D	FF	921	Db 11111111B ;00000000
		922	\$Eject

LOC	OBJ	LINE	SOURCE
FD6E	30	923	Db 00110000B ;..00.... ;60H (')
FD6F	30	924	Db 00110000B ;..00....
FD70	18	925	Db 00011000B ;...00...
FD71	00	926	Db 00000000B ;.....
FD72	00	927	Db 00000000B ;.....
FD73	00	928	Db 00000000B ;.....
FD74	00	929	Db 00000000B ;.....
FD75	00	930	Db 00000000B ;.....
		931	
FD76	00	932	Db 00000000B ;..... ;61H (a)
FD77	00	933	Db 00000000B ;.....
FD78	78	934	Db 01111000B ;.0000...
FD79	0C	935	Db 00001100B ;...00...
FD7A	7C	936	Db 01111100B ;.00000...
FD7B	CC	937	Db 11001100B ;00..00..
FD7C	76	938	Db 01110110B ;.000.00.
FD7D	00	939	Db 00000000B ;.....
		940	
FD7E	E0	941	Db 11100000B ;000..... ;62H (b)
FD7F	60	942	Db 01100000B ;.00.....
FD80	60	943	Db 01100000B ;.00.....
FD81	7C	944	Db 01111100B ;.00000...
FD82	66	945	Db 01100110B ;.00..00.
FD83	66	946	Db 01100110B ;.00..00.
FD84	DC	947	Db 11011100B ;00.000..
FD85	00	948	Db 00000000B ;.....
		949	
FD86	00	950	Db 00000000B ;..... ;63H (c)
FD87	00	951	Db 00000000B ;.....
FD88	78	952	Db 01111000B ;.0000...
FD89	CC	953	Db 11001100B ;00..00..
FD8A	C0	954	Db 11000000B ;00.....
FD8B	CC	955	Db 11001100B ;00..00..
FD8C	78	956	Db 01111000B ;.0000...
FD8D	00	957	Db 00000000B ;.....
		958	
FD8E	1C	959	Db 00011100B ;...000.. ;64H (d)
FD8F	0C	960	Db 00001100B ;...00...
FD90	0C	961	Db 00001100B ;...00...
FD91	7C	962	Db 01111100B ;.00000...
FD92	CC	963	Db 11001100B ;00..00..
FD93	CC	964	Db 11001100B ;00..00..
FD94	76	965	Db 01110110B ;.000.00.
FD95	00	966	Db 00000000B ;.....
		967	
FD96	00	968	Db 00000000B ;..... ;65H (e)
FD97	00	969	Db 00000000B ;.....
FD98	78	970	Db 01111000B ;.0000...
FD99	CC	971	Db 11001100B ;00..00..
FD9A	FC	972	Db 11111100B ;000000...
FD9B	C0	973	Db 11000000B ;00.....
FD9C	78	974	Db 01111000B ;.0000...
FD9D	00	975	Db 00000000B ;.....
		976	
FD9E	38	977	Db 00111000B ;..000... ;66H (f)
FD9F	6C	978	Db 01101100B ;.00.00..
FDA0	60	979	Db 01100000B ;.00.....
FDA1	F0	980	Db 11110000B ;0000....
FDA2	60	981	Db 01100000B ;.00.....
FDA3	60	982	Db 01100000B ;.00.....
FDA4	F0	983	Db 11110000B ;0000....
FDA5	00	984	Db 00000000B ;.....
		985	
FDA6	00	986	Db 00000000B ;..... ;67H (g)
FDA7	00	987	Db 00000000B ;.....
FDA8	76	988	Db 01110110B ;.000.00.
FDA9	CC	989	Db 11001100B ;00..00..
FDAA	CC	990	Db 11001100B ;00..00..
FDAB	7C	991	Db 01111100B ;.00000...
FDAC	0C	992	Db 00001100B ;...00...
FDAD	F8	993	Db 11111000B ;00000...
		994	\$Eject

LOC	OBJ	LINE	SOURCE		
FDAE	E0	995	Db	11100000B	;000.... ;68H (h)
FDAF	60	996	Db	01100000B	;..00....
FDB0	6C	997	Db	01101100B	;..00.00..
FDB1	76	998	Db	01110110B	;..000.00..
FDB2	66	999	Db	01100110B	;..00..00..
FDB3	6E	1000	Db	01100110B	;..00..00..
FDB4	E6	1001	Db	11100110B	;000..00..
FDB5	00	1002	Db	00000000B	;.....
		1003			
FDB6	30	1004	Db	00110000B	;..00.... ;69H (i)
FDB7	00	1005	Db	00000000B	;.....
FDB8	70	1006	Db	01110000B	;..000....
FDB9	30	1007	Db	00110000B	;..00....
FDBA	30	1008	Db	00110000B	;..00....
FDBB	30	1009	Db	00110000B	;..00....
FDBC	70	1010	Db	01111000B	;..0000...
FDBD	00	1011	Db	00000000B	;.....
		1012			
FDBE	0C	1013	Db	00001100B	;....00.. ;6AH (j)
FDBF	00	1014	Db	00000000B	;.....
FDC0	0C	1015	Db	00001100B	;....00..
FDC1	0C	1016	Db	00001100B	;....00..
FDC2	0C	1017	Db	00001100B	;....00..
FDC3	CC	1018	Db	11001100B	;00..00..
FDC4	CC	1019	Db	11001100B	;00..00..
FDC5	78	1020	Db	01111000B	;..0000...
		1021			
FDC6	E0	1022	Db	11100000B	;000.... ;6BH (k)
FDC7	60	1023	Db	01100000B	;..00....
FDC8	66	1024	Db	01100110B	;..00..00..
FDC9	6C	1025	Db	01101100B	;..00..00..
FDCA	78	1026	Db	01111000B	;..0000...
FDCB	6C	1027	Db	01101100B	;..00..00..
FDCC	E6	1028	Db	11100110B	;000..00..
FDCD	00	1029	Db	00000000B	;.....
		1030			
FDCE	70	1031	Db	01110000B	;..000.... ;6CH (l)
FDCF	30	1032	Db	00110000B	;..00....
FDD0	30	1033	Db	00110000B	;..00....
FDD1	30	1034	Db	00110000B	;..00....
FDD2	30	1035	Db	00110000B	;..00....
FDD3	30	1036	Db	00110000B	;..00....
FDD4	78	1037	Db	01111000B	;..0000...
FDD5	00	1038	Db	00000000B	;.....
		1039			
FDD6	00	1040	Db	00000000B	;..... ;6DH (m)
FDD7	00	1041	Db	00000000B	;.....
FDD8	CC	1042	Db	11001100B	;00..00..
FDD9	FE	1043	Db	11111110B	;00000000.
FDDA	FE	1044	Db	11111110B	;00000000.
FDDB	D6	1045	Db	11010110B	;00.0.00..
FDDC	C6	1046	Db	11000110B	;00..0.00..
FDDD	00	1047	Db	00000000B	;.....
		1048			
FDDE	00	1049	Db	00000000B	;..... ;6EH (n)
FDDF	00	1050	Db	00000000B	;.....
FDE0	F8	1051	Db	11111000B	;000000..
FDE1	CC	1052	Db	11001100B	;00..00..
FDE2	CC	1053	Db	11001100B	;00..00..
FDE3	CC	1054	Db	11001100B	;00..00..
FDE4	CC	1055	Db	11001100B	;00..00..
FDE5	00	1056	Db	00000000B	;.....
		1057			
FDE6	00	1058	Db	00000000B	;..... ;6FH (o)
FDE7	00	1059	Db	00000000B	;.....
FDE8	78	1060	Db	01111000B	;..0000...
FDE9	CC	1061	Db	11001100B	;00..00..
FDEA	CC	1062	Db	11001100B	;00..00..
FDEB	CC	1063	Db	11001100B	;00..00..
FDEC	78	1064	Db	01111000B	;..0000...
FDED	00	1065	Db	00000000B	;.....
		1066			
FDEE	00	1067	Db	00000000B	;..... ;70H (p)
FDEF	00	1068	Db	00000000B	;.....
FDF0	DC	1069	Db	11011100B	;00.000..

LOC	OBJ	LINE	SOURCE		
FDF1	66	1070	Db	01100110B	; .00.00.
FDF2	66	1071	Db	01100110B	; .00.00.
FDF3	7C	1072	Db	01111100B	; .00000..
FDF4	60	1073	Db	01100000B	; .00.....
FDF5	F0	1074	Db	11110000B	; 0000....
		1075			
FDF6	00	1076	Db	00000000B	; ;71H (q)
FDF7	00	1077	Db	00000000B	; ;71H (q)
FDF8	76	1078	Db	01110110B	; .000.00.
FDF9	CC	1079	Db	11001100B	; 00..00..
FDF9	CC	1080	Db	11001100B	; 00..00..
FDFB	7C	1081	Db	01111100B	; .00000..
FDFC	0C	1082	Db	00001100B	; ...00..
FDFD	1E	1083	Db	00011110B	; ...0000.
		1084			
FDFE	00	1085	Db	00000000B	; ;72H (r)
FDFE	00	1086	Db	00000000B	; ;72H (r)
FE00	DC	1087	Db	11011100B	; 00.000..
FE01	76	1088	Db	01110110B	; .000.00.
FE02	66	1089	Db	01100110B	; .00..00.
FE03	60	1090	Db	01100000B	; .00.....
FE04	F0	1091	Db	11110000B	; 0000....
FE05	00	1092	Db	00000000B	; ;72H (r)
		1093			
FE06	00	1094	Db	00000000B	; ;73H (s)
FE07	00	1095	Db	00000000B	; ;73H (s)
FE08	7C	1096	Db	01111100B	; .00000..
FE09	C0	1097	Db	11000000B	; 00.....
FE0A	78	1098	Db	01111000B	; .0000... ;73H (s)
FE0B	0C	1099	Db	00001100B	; ...00..
FE0C	F8	1100	Db	11111000B	; 00000... ;73H (s)
FE0D	00	1101	Db	00000000B	; ;73H (s)
		1102			
FE0E	10	1103	Db	00010000B	; ...0... ;74H (t)
FE0F	30	1104	Db	00110000B	; ..00... ;74H (t)
FE10	7C	1105	Db	01111100B	; .00000..
FE11	30	1106	Db	00110000B	; ..00... ;74H (t)
FE12	30	1107	Db	00110000B	; ..00... ;74H (t)
FE13	34	1108	Db	00110100B	; ..00.0.. ;74H (t)
FE14	18	1109	Db	00011000B	; ...00... ;74H (t)
FE15	00	1110	Db	00000000B	; ;74H (t)
		1111			
FE16	00	1112	Db	00000000B	; ;75H (u)
FE17	00	1113	Db	00000000B	; ;75H (u)
FE18	CC	1114	Db	11001100B	; 00..00.. ;75H (u)
FE19	CC	1115	Db	11001100B	; 00..00.. ;75H (u)
FE1A	CC	1116	Db	11001100B	; 00..00.. ;75H (u)
FE1B	CC	1117	Db	11001100B	; 00..00.. ;75H (u)
FE1C	76	1118	Db	01110110B	; .000.00. ;75H (u)
FE1D	00	1119	Db	00000000B	; ;75H (u)
		1120			
FE1E	00	1121	Db	00000000B	; ;76H (v)
FE1E	00	1122	Db	00000000B	; ;76H (v)
FE20	CC	1123	Db	11001100B	; 00..00.. ;76H (v)
FE21	CC	1124	Db	11001100B	; 00..00.. ;76H (v)
FE22	CC	1125	Db	11001100B	; 00..00.. ;76H (v)
FE23	78	1126	Db	01111000B	; .0000... ;76H (v)
FE24	30	1127	Db	00110000B	; ..00... ;76H (v)
FE25	00	1128	Db	00000000B	; ;76H (v)
		1129			
FE26	00	1130	Db	00000000B	; ;77H (w)
FE27	00	1131	Db	00000000B	; ;77H (w)
FE28	C6	1132	Db	11000110B	; 00...00. ;77H (w)
FE29	D6	1133	Db	11010110B	; 00.0.00. ;77H (w)
FE2A	FE	1134	Db	11111110B	; 0000000. ;77H (w)
FE2B	FE	1135	Db	11111110B	; 0000000. ;77H (w)
FE2C	6C	1136	Db	01101100B	; .00.00.. ;77H (w)
FE2D	00	1137	Db	00000000B	; ;77H (w)
		1138			
FE2E	00	1139	Db	00000000B	; ;78H (x)
FE2F	00	1140	Db	00000000B	; ;78H (x)
FE30	C6	1141	Db	11000110B	; 00...00. ;78H (x)
FE31	6C	1142	Db	01101100B	; .00.00.. ;78H (x)
FE32	30	1143	Db	00111000B	; ..000... ;78H (x)
FE33	6C	1144	Db	01101100B	; .00.00.. ;78H (x)

LOC	OBJ	LINE	SOURCE			
FE34	C6	1145		Db	11000110B	;00...00.
FE35	00	1146		Db	00000000B	;.....
		1147				
FE36	00	1148		Db	00000000B	;..... ;79H (y)
FE37	00	1149		Db	00000000B	;.....
FE38	CC	1150		Db	11001100B	;00..00..
FE39	CC	1151		Db	11001100B	;00..00..
FE3A	CC	1152		Db	11001100B	;00..00..
FE3B	7C	1153		Db	01111100B	;.00000..
FE3C	0C	1154		Db	00001100B	;...00..
FE3D	F8	1155		Db	11111000B	;00000...
		1156				
FE3E	00	1157		Db	00000000B	;..... ;7AH (z)
FE3F	00	1158		Db	00000000B	;.....
FE40	FC	1159		Db	11111100B	;000000..
FE41	98	1160		Db	10011000B	;0..00...
FE42	30	1161		Db	00110000B	;..00....
FE43	64	1162		Db	01100100B	;..00..0..
FE44	FC	1163		Db	11111100B	;000000..
FE45	00	1164		Db	00000000B	;.....
		1165				
FE46	1C	1166		Db	00011100B	;...000.. ;7BH (t)
FE47	30	1167		Db	00110000B	;..00....
FE48	30	1168		Db	00110000B	;..00....
FE49	E0	1169		Db	11100000B	;000.....
FE4A	30	1170		Db	00110000B	;..00....
FE4B	30	1171		Db	00110000B	;..00....
FE4C	1C	1172		Db	00011100B	;...000..
FE4D	00	1173		Db	00000000B	;.....
		1174				
FE4E	18	1175		Db	00011000B	;...00... ;7CH (i)
FE4F	18	1176		Db	00011000B	;...00...
FE50	18	1177		Db	00011000B	;...00...
FE51	00	1178		Db	00000000B	;.....
FE52	18	1179		Db	00011000B	;...00...
FE53	18	1180		Db	00011000B	;...00...
FE54	18	1181		Db	00011000B	;...00...
FE55	00	1182		Db	00000000B	;.....
		1183				
FE56	E0	1184		Db	11100000B	;000..... ;7DH (j)
FE57	30	1185		Db	00110000B	;..00....
FE58	30	1186		Db	00110000B	;..00....
FE59	1C	1187		Db	00011100B	;...000..
FE5A	30	1188		Db	00110000B	;..00....
FE5B	30	1189		Db	00110000B	;..00....
FE5C	E0	1190		Db	11100000B	;000.....
FE5D	00	1191		Db	00000000B	;.....
		1192				
FE5E	76	1193		Db	01110110B	;.000.00. ;7EH (~)
FE5F	DC	1194		Db	11011100B	;00.000..
FE60	00	1195		Db	00000000B	;.....
FE61	00	1196		Db	00000000B	;.....
FE62	00	1197		Db	00000000B	;.....
FE63	00	1198		Db	00000000B	;.....
FE64	00	1199		Db	00000000B	;.....
FE65	00	1200		Db	00000000B	;.....
		1201				
FE66	00	1202		Db	00000000B	;..... ;7FH (Del)
FE67	10	1203		Db	00010000B	;...0....
FE68	38	1204		Db	00111000B	;..000...
FE69	6C	1205		Db	01101100B	;.00.00..
FE6A	C6	1206		Db	11000110B	;00..00.
FE6B	C6	1207		Db	11000110B	;00..00.
FE6C	FE	1208		Db	11111100B	;0000000.
FE6D	00	1209		Db	00000000B	;.....
		1210				
----		1211	Bios	Ends		
		1212	End			

ASSEMBLY COMPLETE, NO ERRORS FOUND

IBM PC/XT 8086/8087/8088 MACRO ASSEMBLER V1.1 ASSEMBLY OF MODULE TIME
OBJECT MODULE PLACED IN TIME.OBJ
ASSEMBLER INVOKED BY: ASM86 TIME.SRC

```
LOC OBJ          LINE    SOURCE
                1 +1  $Title ('DTC/PC BIOS Time of Day and Real Time Clock V1.0')
                2 +1  $Pagelength (80) Pagewidth (132) Debug Nogen
                3      Name Time
                4
                5
                6      ;   Author:      Don K. Harrison
                7
                8      ;   Start date:  December 7, 1983      Last edit:  December 20, 1983
                9
               10
               11      ;           *****
               12      ;           * Module Description *
               13      ;           *****
               14      ;
               15      ;           This module contains the time of day driver routine (Int 26)
               16      ;           and the real time clock hardware interrupt service routine (Int 8)
               17      ;
               18
               19
               20
               21
               22      ;           (c) Display Telecommunications Corporation, 1983
               23      ;           All Rights Reserved
               24
               25  $Eject
```

LOC	OBJ	LINE	SOURCE
		26	
		27	
		28	:
		29	*****
		30	* Revision History *
		31	*****
		32	
		33	
		34	
		35	
		36	
		37	
		38	\$Eject

* Revision History *

```
LOC OBJ          LINE    SOURCE
                 39
                 40      ;          *****
                 41      ;          * Public Symbols *
                 42      ;          *****
                 43
                 44      Public TodDriver, TimerHdwrInt
                 45
                 46
                 47      ;          *****
                 48      ;          * Equates *
                 49      ;          *****
                 50
                 51      ;          All Equates in include file: IbmInc
                 52
                 53      $Include (IbmInc)
=1 54      ;          *****
=1 55      ;          * Global Include File *
=1 56      ;          *****
=1 57      $Nolist
577     $Eject
```

LOC	OBJ	LINE	SOURCE
		578	
		579	;
		580	;
		581	;
		582	*****
		583	
----		584	BiosDataArea Segment Public
		585	Extrn TimerLow:Word, TimerHigh:Word, TimerOverflow:Byte
		586	Extrn MotorStatus:Byte, MotorCount:Byte
----		587	BiosDataArea Ends
		588	%Eject


```

LOC OBJ          LINE    SOURCE
                    589
                    590 ; *****
                    591 ; * Code Segment *
                    592 ; *****
                    593
-----          594 Bios      Segment Common
                    595
                    596          Assume Cs:Bios, Ds:BiosDataArea
                    597
                    598 ; *****
                    599 ; * Time of Day Driver *
                    600 ; *****
                    601
FE6E            602          Org      0FE6EH          ;Align with Pc and Xt
FE6E            603          Proc      Far
FE6E FB         604          Sti          ;Restore interrupts
FE6F 1E         605          Push     Ds          ;Setup
FE70 50         606          Push     Ax          ;...our
FE71 B8----- R         607          Mov      Ax,BiosDataArea ;...data
FE74 8ED8       608          Mov      Ds,Ax          ;...segment
FE76 58         609          Pop      Ax          ;Restore command
FE77 FA         610          Cli          ;Allow no ints
FE78 0AE4       611          Or       Ah,Ah        ;Command = 0 = Read
FE7A 7413       612          Jz       ReadClock    ;...jump if so
FE7C FECC       613          Dec     Ah          ;Command = 1 = Set
FE7E 751A       614          Jnz     TodReturn    ;...jump if not and return
                    615
                    616          ;      Set the clock
                    617
FE80 89160000   E         618          Mov      TimerLow,Dx    ;Low portion of day clock
FE84 890E0000   E         619          Mov      TimerHigh,Cx   ;High portion of day clock
FE88 C606000000 E         620          Mov      TimerOverflow,0 ;Overflow = false
FE8D EB0B       621          Jmp     Short TodReturn ;...Return
                    622
                    623          ;      Read the clock
                    624
FE8F            625          ReadClock:
FE8F 8B0E0000   E         626          Mov      Cx,TimerHigh   ;Return High portion
FE93 8B160000   E         627          Mov      Dx,TimerLow   ;Return Low portion
FE97 A00000     E         628          Mov      Al,TimerOverflow ;Return overflow status
FE9A            629          TodReturn:
FE9A FB         630          Sti          ;Restore interrupts
FE9B 1F         631          Pop      Ds          ;Restore user data seg.
FE9C CF         632          Iret         ;...and return
                    633          TodDriver
                    634          $Eject

```

```

LOC OBJ                LINE    SOURCE
                        635
                        636                ;
                        637                ;
                        638                ;
                        639                ;
FEA5                    640                Org      0FEASH                ;Align with PC and Xt
                        641
FEA5                    642                TimerHdwrInt Proc   Far
FEA5 FB                 643                Sti                ;Restore interrupts
FEA6 1E                 644                Push   Ds           ;Save minimum registers
FEA7 52                 645                Push   Dx           ;...so this runs
FEA8 50                 646                Push   Ax           ;...fast
FEA9 B8-----         R        647                Mov    Ax,BiosDataArea ;Load segment register
FEAC 8ED8              648                Mov    Ds,Ax       ;...with bios segment
                        649
                        650                ;      First, process diskette motor timer
                        651
FEAE FE0E0000          E        652                Dec    MotorCount   ;Reduce timer for motor by 1
FEB2 750B              653                Jnz   MotorsStillRun ;...and return if not 0
FEB4 80250000F0        E        654                And   MotorStatus,11110000B ;Clear motor running bits
FEB9 B00C              655                Mov   A1,00001100B  ;...and turn off motors
FEBB BAF203            656                Mov   Dx,PortFDCAdptMode ;Point at port
FEBE EE                657                Out   Dx,A1         ;...turn them off
FEBF                    658                MotorsStillRun:
                        659
                        660                ;      Now, process clock
                        661
FEBF FF060000          E        662                Inc   TimerLow      ;Inc the low part of timer
FEC3 7504              663                Jnz   NoCarryToHi  ;...and jump if it didn't carry
FEC5 FF060000          E        664                Inc   TimerHigh     ;...else increment High timer
FEC9                    665                NoCarryToHi:
FEC9 833E000018        E        666                Cmp   TimerHigh,10H ;Does the timer = 24 hrs
FECE 7519              667                Jne   NoCarryToOflo ;...jump if not
FED0 813E0000B000      E        668                Cmp   TimerLow,0B0H ;Does the timer = 24 hrs
FED6 7511              669                Jne   NoCarryToOflo ;...jump if not
                        670
                        671                ;      Set overflow
                        672
FED8 C70600000000      E        673                Mov   TimerHigh,0   ;Clear the
FEDE C70600000000      E        674                Mov   TimerLow,0    ;...timer and
FEE4 C606000001        E        675                Mov   TimerOverflow,1 ;...set the overflow flag
                        676
FEE9                    677                NoCarryToOflo:
                        678                ;      Now, process user RTC tick routine
                        679
FEE9 CD1C              680                Int   TrapTimerBreak ;Probably a dummy return
                        681
                        682                ;      Acknowledge interrupt
                        683
FEEB 8020              684                Mov   A1,PicEoi     ;End of interrupt
FEED E620              685                Out   PortPICDCW2,A1 ;...to 8259A
FEF0 5A                686                Pop   Ax            ;Restore
FEF1 1F                687                Pop   Dx            ;...registers
FEF2 CF                688                Pop   Ds            ;...and
                        689                Iret               ;...return
                        690                TimerHdwrInt      Endp
-----                691                Bios              Ends
                        692
                        693                End
                        694

```

ASSEMBLY COMPLETE, NO ERRORS FOUND

IBM PC/XT 8086/8087/8088 MACRO ASSEMBLER V1.1 ASSEMBLY OF MODULE PRINTSCREEN
OBJECT MODULE PLACED IN PRINTSCN.OBJ
ASSEMBLER INVOKED BY: ASMB6 PRINTSCN.SRC

LOC	OBJ	LINE	SOURCE
		1 +1	\$Title ('DTC/PC BIOS Print Screen Driver V1.0')
		2 +1	\$Pagelength (80) Pagewidth (132) Debug Nogen
		3	Name PrintScreen
		4	
		5	
		6	; Author: Don K. Harrison
		7	
		8	; Start date: December 7, 1983 Last edit: December 20, 1983
		9	
		10	
		11	; *****
		12	; * Module Description *
		13	; *****
		14	; This module contains the print screen interrupt routine
		15	; (Int 5).
		16	
		17	
		18	
		19	
		20	
		21	
		22	; (c) Display Telecommunications Corporation, 1983
		23	; All Rights Reserved
		24	
		25	\$Eject

LOC	OBJ	LINE	SOURCE
		26	
		27	
		28	;
		29	;
		30	*****
		31	;
		32	
		33	
		34	
		35	
		36	
		37	
		38	\$Eject

```
LOC OBJ          LINE    SOURCE
                 39
                 40      ;          *****
                 41      ;          * Public Symbols *
                 42      ;          *****
                 43
                 44      Public PrintScreenInt
                 45
                 46
                 47      ;          *****
                 48      ;          * Equates *
                 49      ;          *****
                 50
000A             51      AsciiLineFeed      Equ      0AH      ;Ascii line feed
000D             52      AsciiCarriageReturn Equ      0DH      ;Ascii carriage return
                 53
                 54
                 55      $Include (IbmInc)
=1              56      ;          *****
=1              57      ;          * Global Include File *
=1              58      ;          *****
=1              59      $Nolist
                 579     $Eject
```

LOC	OBJ	LINE	SOURCE
		580	
		581	;
		582	;
		583	;
		584	
----		585	BiosDataArea Segment Public
----		586	Extrn PrintScnStatus:Byte
		587	BiosDataArea Ends
		588	%Eject

```

LOC OBJ          LINE    SOURCE
                    589
                    590      ;          *****
                    591      ;          * Code Segment *
                    592      ;          *****
                    593
----             594      Bios          Segment Common
                    595
FF54             596      Org          0FF54H          ;Align with Pc and Xt
                    597
                    598      Assume   Cs:Bios, Ds:BiosDataArea
                    599
FF54             600      PrintScreenInt Proc   Far
FF54 FB         601      StI          ;Restore interrupts
FF55 1E         602      Push       Ds          ;.....
FF56 50         603      Push       Ax          ;. Save      .
FF57 53         604      Push       Bx          ;.
FF58 51         605      Push       Cx          ;. Registers .
FF59 52         606      Push       Dx          ;.....
FF5A B8----- R   607      Mov        Ax,BiosDataArea ;Load our segment
FF5D 8E08       608      Mov        Ds,Ax        ;...into Ds
FF5F 803E00001 E   609      Cmp        PrintScnStatus,1 ;In progress already?
FF64 7456       610      Je         InProgressEnd ;...jump if so and return
                    611
                    612      ;          Set status = 1 = in progress
                    613
FF66 C60600001 E   614      Mov        PrintScnStatus,1 ;Set status
                    615
                    616      ;          Initialize printer
                    617
FF6B E85400     618      Call      PCrLf         ;Only needs a line feed
                    619
                    620      ;          Get cursor position
                    621
FF6E B40F       622      Mov        Ah,VidCmdInfo ;Get page number and info
FF70 CD10       623      Int       TrapVideo    ;...for use later
FF72 50         624      Push      Ax           ;Save columns
FF73 B403       625      Mov        Ah,VidCmdRdCurPos ;Read cursor position
FF75 CD10       626      Int       TrapVideo    ;...for restoration purposes
FF77 58         627      Pop       Ax          ;Get back columns
FF78 52         628      Push     Dx           ;...and save cursor pos
FF79 B519       629      Mov        Ch,25       ;Always do 25 rows
FF7B 8ACC       630      Mov        Cl,Ah       ;...and (Columns) columns
FF7D 33D2       631      Xor       Dx,Dx       ;Start of screen = 0,0
                    632
                    633      ;          Print the screen
                    634
FF7F             635      PrintScreenLoop:
FF7F B402       636      Mov        Ah,VidCmdCurPos ;Set cursor to next (first)
FF81 CD10       637      Int       TrapVideo    ;...line
FF83 B408       638      Mov        Ah,VidCmdRdCurChAt ;Read the
FF85 CD10       639      Int       TrapVideo    ;...character
FF87 0AC0       640      Or        Al,Al       ;If zero,
FF89 7502       641      Jnz      CharOk       ;...then convert
FF8B B020       642      Mov        Al,' '     ;...to a space
FF8D             643      CharOk:
                    644
                    645      ;          Print the character
                    646
FF8D 52         647      Push     Dx           ;Save cursor position
FF8E 33D2       648      Xor      Dx,Dx       ;Select printer 1
FF90 8AE2       649      Mov      Ah,Dl       ;...print command
FF92 CD17       650      Int     TrapPrintDrive ;Print it
FF94 5A         651      Pop     Dx           ;Restore cursor
                    652
                    653      ;          Test for error
                    654
FF95 F6C425     655      Test     Ah,00100101B ;Error if any bit set
FF98 7407       656      Jz       NoError      ;...jump if no error
FF9A C6060000FF E   657      Mov      PrintScnStatus,0FFH ;Error status
FF9F EB16       658      Jmp      Short PrintScnEnd
                    659
                    660      ;          Increment to next character position
FFA1             661      NoError:
FFA1 FEC2       662      Inc     Dl           ;Bump column
FFA3 3ACA       663      Cmp     Cl,Dl       ;...is it right limit?

```

LOC	OBJ	LINE	SOURCE
FFA5	75D8	664	Jnz PrintScreenLoop ;...if not, loop
FFA7	32D2	665	Xor DI,DI ;Do carriage return on screen
FFA9	E81600	666	Call PCrIf ;Do carriage return on printer
FFAC	FEC6	667	Inc Dh ;Increment row
FFAE	3AF5	668	Cmp Dh,Ch ;...over limit?
FFB0	75CD	669	Jnz PrintScreenLoop ;...jump if not
FFB2	C606000000	670	Mov PrintScnStatus,0 ;Indicate all done
		671	
		672	; Restore cursor and return
		673	
FFB7		674	PrintScnEnd:
FFB7	5A	675	Pop Dx ;Restore original
FFB8	B402	676	Mov Ah,VidCmdCurPos ;...cursor
FFBA	CD10	677	Int TrapVideo ;...to screen
FFBC		678	InProgressEnd:
FFBC	5A	679	Pop Dx ;.....
FFBD	59	680	Pop Cx ;. Restore .
FFBE	5B	681	Pop Bx ;. Registers .
FFBF	58	682	Pop Ax ;. and .
FFC0	1F	683	Pop Ds ;. Return .
FFC1	CF	684	Iret ;.....
		685	
		686	; *****
		687	; * Printer Carriage Return *
		688	; *****
		689	; *****
FFC2		690	PCrIf Proc Near
FFC2	52	691	Push Dx ;Save Dx
FFC3	33D2	692	Xor Dx,Dx ;Select printer 1
FFC5	8AE2	693	Mov Ah,DI ;...and command = print
FFC7	B00A	694	Mov Al,AsciiLineFeed ;Send LF first
FFC9	CD17	695	Int TrapPrintDrive ;...like IBM does
FFCB	32E4	696	Xor Ah,Ah ;Ignore status
FFCD	B00D	697	Mov Al,AsciiCarriageReturn ;Send Cr
FFCF	CD17	698	Int TrapPrintDrive ;...next
FFD1	5A	699	Pop Dx ;Restore Dx
FFD2	C3	700	Ret ;...and return
		701	PCrIf Endp
		702	
		703	PrintScreenInt Endp
		704	
----		705	Bios Ends
		706	
		707	End

ASSEMBLY COMPLETE, NO ERRORS FOUND